وزارة التعليم العالي والبحث العلمي

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université du 8 mai 1945 Guelma

Faculté des Sciences et de la Technologie

جامعة 8 ماي 1945 قالمة

كلية العلوم و التكنولوجيا

Département de Génie Mécanique

Cours Optimisation

Cours destiné aux étudiants de première année Master

Filière : Génie Mécanique

Option : Construction

Elaboré par : Dr BOUCHERIT Septi

2017-2018

Semestre : 2	**Unité d'enseignement : UEM 1.2**
Matière : Optimisation	**VHS : 45h00 (cours: 01h30, TP:1h30)**
Crédits : 4	**Coefficient : 2**

Objectifs de l'enseignement:

Se familiariser avec les modèles de recherche opérationnelle. Apprendre à formuler et à résoudre les problèmes d'optimisation et maitriser les techniques et les algorithmes appropriés.

Connaissances préalables recommandées :

Notions de bases de mathématiques. Algèbre linéaire. Algèbre matricielle.

Contenu de la matière :

Chapitre I : Optimisation linéaire **(3 semaines)**

- Formulation générale d'un programme linéaire
- Exemples de programmes linéaires (Problème de production, Problème de Mélange, Problème de découpage, Problème de transport)
- Résolution du problème par la méthode Simplexe :
  - Bases et solutions de base des programmeslinéaires
  - L'algorithme du simplexe
  - Initialisation de l'algorithme du simplexe (la méthode à deux phases).

Chapitre II : Optimisation non- linéaire sans contraintes **(5 semaines)**

- Positivité, Convexité, Minimum
- Gradient et Hessien
- Conditions nécessaires pour un minimum
- Conditions suffisantes pour un minimum
- Méthodes locales
- Méthodes de recherche unidimensionnelle
- Méthodes du gradient
- Méthodes des directions conjuguées
- Méthode de Newton
- Méthodes quasi-Newton

Chapitre III : Optimisation non-linéaires avec contraintes **(4 semaines)**

- Multiplicateurs de Lagrange
- Conditions de Karush-Kuhn-Tucker
- Méthode des pénalités
- Programmation quadratique séquentielle

Chapitre IV : Méthodes d'optimisation stochastiques **(3 semaines)**

- L'algorithme génétique
- La méthode d'essaim particulaire

Organisation des TP : il est préférable que les TP soient des applications directes dans le domaine de la construction mécanique.

TP 1 : présentation des fonctions références d'optimisation en Matlab

TP 2 : Présentation de l'outil d'optimisation optimtool dans matlab

TP 3 : Définition et traçage des courbes de quelques fonctions test en optimisation

TP 4 : Résolution d'un problème d'optimisation linéaire sans contraintes

TP 5 : Résolution d'un problème d'optimisation linéaire avec contraintes

TP 6 : Minimisation non linéaire sans contraintes

TP 7: Minimisation non linéaire sans contraintes avec gradient et Hessien

TP 8 : Minimisation non linéaire avec contraintes d'égalité

TP 9 : Minimisation non linéaire avec contraintes d'inégalité

TP 10 : Minimisation avec contraintes d'égalité et d'inégalité

TP 11 : Utilisation de l'outil optimtool ou autre pour la résolution d'un problème d'optimisation non linéaire avec contraintes

TP 12 : Minimisation avec contraintes en utilisant la fonction GA

Mode d'évaluation: Contrôle Continu : 40%, **Examen :** 60%.

Sommaire

Chapitre I : Optimisation linéaire

I.1	Introduction	1
I.2	Définition	1
I.3	Exemple de problème de programmation linéaire	1
I.3.1	Problème de production	1
I.3.2	Problème de Mélange	4
I.3.3	Problème de découpe	4
I.3.4	Problème de transport	5
I.4	Résolution du problème par la méthode Simplexe	7
I.4.1	Bases et solutions de base des programmes linéaires	9
I.4.2	Recherche d'une solution optimale	11
I.5	Tableaux du simplexe et procédure de calcul	14
I.6	Recherche d'une solution réalisable de base initiale	19
I.7	Initialisation de l'algorithme du simplexe (la méthode à deux phases)	20

Chapitre 2 : Optimisation non-linéaire

II.1	Introduction	25
II.2	Le Gradient	25
II.3	Le Hessien	25
II.4	Condition nécessaire pour l'existence d'un minimum local	26
II.5	Conditions suffisantes pour l'existence d'un minimum local	27
II.6	Fonctions convexes	27
II.6.1	Fonctions convexes	27
II.6.2	Minimum de fonctions convexes	28
II.7	Méthodes de recherche unidimensionnelle	28
II.7.1	Méthodes du premier ordre (dichotomie)	28
II.7.2	Méthodes du second ordre (Méthode de Newton)	28
II.8	Méthodes du gradient	29
II.9	Méthodes des directions conjuguées	32
II.9	Méthode de Newton	36
II.10	Méthodes quasi-Newton	38

Chapitre III : Optimisation avec contraintes

III.1	Introduction	41
III.2	Problème avec contraintes d'égalité	41
III.2.1	Le théorème de Lagrange	42
III.2.2	Définition du lagrangien	42
III.2.3	Condition nécessaire du second ordre	43
III.2.4	Condition nécessaire du second ordre	43
III.3	Problème avec contraintes d'inégalité (conditions de KKT)	44
III.4	Conditions de KKT pour un problème avec limitations d'égalité et d'inégalité	46
III.5	Méthodes de pénalité	47
III.5.1	Principe	47
III.5.2	Pénalité extérieure	47
III.5.3	Pénalité intérieure	49
III.6	Programmation quadratique séquentielle	50

Chapitre IV : Méthodes d'optimisation stochastiques

IV.1	L'algorithme génétique	53
IV.1.1	Le codage	53
IV.1.2	L'opérateur de sélection	54
IV.1.3	L'opérateur de croisement ou crossover	56

IV.1.4 L'opérateur de mutation 59
IV.1.5 L'opérateur de remplacement 60
IV.2 La méthode d'essaim particulaire 63
IV.2.1 Configuration de la méthode 64
IV.2.1.1 Nombre de particules 64
IV.2.1.2 Topologie du voisinage 64
IV.2.1.3 Coefficients de confiance 65
IV.2.1.4 Vitesse maximale et coefficient de constriction 65
IV.2.1.5 Facteur d'inertie 66
IV.2.1.6 Initialisation de l'essaim 66
IV.2.1.7 Critères d'arrêt 66
IV.2.2 Algorithme de synthèse 67

Chapitre I :

Optimisation linéaire

I-1 Introduction

L'optimisation (programmation) linéaire est une technique mathématique permettant de déterminer la meilleure solution d'un problème dont les données et les inconnues satisfont à une série d'équations et d'inéquations linéaires. La programmation linéaire a été formulée par George Bernard Dantzig en 1947 et connaît un développement rapide par suite de son application directe à la gestion scientifique des entreprises. Le facteur expliquant l'essor de la programmation linéaire est la construction d'ordinateurs puissants qui ont permis de traiter les problèmes concrets de taille très grande. On l'applique surtout en gestion et en économie appliquée. On peut citer les domaines d'application de la programmation linéaire qui sont : les transports, les banques, les industries lourdes et légères, l'agriculture, les chaînes commerciales, la sidérurgie, et même le domaine des applications militaires.

Les méthodes de résolution sont la méthode du simplexe, méthode duale du simplexe, méthodes des potentiels, méthode lexicographique et des méthodes récentes appelées méthodes des points intérieurs.

I.2 Définition

Un programme linéaire (PL) est un modèle mathématique qu'on peut écrire sous la forme :
Maximiser ou minimiser la fonction objectif :

$$\min \ max \ \ z = \sum_{i=1}^{n} c_i x_i \tag{I.1}$$

Sous les contraintes $$a_{i1}x_1 + \cdots + a_{in}x_n \begin{matrix} \leq \\ = \\ \geq \end{matrix} b_i \quad i = 1, \ldots, m \tag{I.2}$$

Et $$x_j \geq 0, \quad j = 1, \ldots, n \tag{I.3}$$

où a_{ij} , b_i *et* c_i sont des constantes connues.

Les contraintes $x_j \geq 0, \quad j = 1, \ldots, n$ sont appelées contraintes de non-négativité.

Définition 1. On appelle *solution* d'un problème de programmation linéaire tout vecteur *x* qui satisfait les contraintes (I.2).
Une solution est appelée *solution réalisable* si elle vérifie les contraintes de non-négativité (I.3). Dans le cas contraire, on dit que la solution n'est *pas réalisable*.

Définition2. Une solution réalisable est une *solution optimale* s'il n'existe pas d'autres solutions réalisables qui fournissent une plus grande valeur de la fonction objectif. À noter que dans un problème possédant des solutions réalisables, il se peut que la valeur optimale de la fonction objectif soit infinie. Dans ce cas, on parle de solution *optimale infinie*.

I-3 Exemple de problème de programmation linéaire

I-3-1 Problème de production

Une entreprise fabrique des chaises et des tables à l'aide de deux machines A et B. Chaque produit passe obligatoirement par les deux machines. Pour produire une chaise, il faut 2 heures de machine A et 1 heure de machine B. Pour produire une table, il faut 1 heure de machine A et 2 heures de machine B.

L'entreprise réalise un bénéfice de 3 DZD.- sur chaque chaise et de 4 DZD.- sur chaque table. Les deux machines A et B sont disponibles 12 heures par jour au maximum.

Le problème consiste à savoir combien de chaises et de tables il faut fabriquer pour maximiser le bénéfice. Le tableau suivant montre :

1. le nombre d'heures nécessaires pour produire chaque table et chaque chaise par machine ;
2. le nombre total des heures disponibles ;
3. le profit pour chaque unité de chaise et de table produite.

Machine	Produit		Disponibilité
	Chaise	Table	
A	2	1	12
B	1	2	12
Bénéfice par unité	3	4	

Il s'agit à présent de formuler mathématiquement le problème.

Notons respectivement par x_1 et x_2 le nombre de chaises et de tables qui doit être produit par jour. Le bénéfice pour une chaise étant de 3 DZD.- et celui pour une table de 4 DZD, le bénéfice journalier est donc donné par la fonction objectif :

$$z = 3x_1 + 4x_2$$

Il faut formuler maintenant les contraintes. Puisque le temps où les machines peuvent fonctionner est limité, on ne peut pas accroître la production indéfiniment. Les machines A et B ne peuvent pas fonctionner plus de 12 heures par jour. On sait que pour produire une chaise, il faut 2 heures de machine A alors que pour une table il n'en faut qu'une.

La contrainte concernant la machine A est donc :

$$2x_1 + x_2 \leq 12$$

De même, une chaise requiert 1 heure de machine B, tandis qu'une table en demande 2. La contrainte concernant la machine B, avec la même durée maximale de 12 heures par jour, est donnée par :

$$x_1 + 2x_2 \leq 12$$

De plus, comme on ne peut pas produire de quantités négatives, il faut ajouter encore deux contraintes de non-négativité :

$$x_1, x_2 \geq 0$$

Le problème consiste donc à trouver les valeurs des variables x_1 et x_2 qui maximisent le bénéfice journalier (fonction objectif) tout en satisfaisant les contraintes.

En résumé, le problème s'écrit sous la forme :

$$\begin{aligned} \text{Maximiser} \quad & z = 3x_1 + 4x_2 \\ \text{Sous contraintes} \quad & 2x_1 + x_2 \leq 12 \end{aligned}$$

$$x_1 + 2x_2 \leq 12$$

Et $$x_1, x_2 \geq 0$$

Représentation graphique du problème

L'exemple décrit ci-dessus peut être représenté graphiquement puisqu'il ne contient que deux variables. Avant de rechercher la solution du problème, représentons sur la figure I.1 la région des points (x_1; x_2) qui satisfont les quatre contraintes simultanément. Les deux contraintes de non-négativité $x_1, x_2 \geq 0$ indiquent que cette région se trouve dans le premier quadrant. En ce qui concerne les deux autres inégalités, on étudie les équations de droite :

$$2x_1 + x_2 = 12$$
$$x_1 + 2x_2 = 12$$

La première est une droite passant par les points (6 ;0) et (0 ;12), tandis que la deuxième passe par (12 ;0) et (0 ;6). L'intersection de ces deux droites est le point (4 ;4). La région des points satisfaisant les quatre inégalités est la région hachurée sur la figure I.1. Elle a pour sommets les points (0 ;0), (0 ;6), (4 ;4) et (6 ;0).

Pour résoudre le problème de programmation linéaire, nous devons trouver le ou les points appartenant à ce polyèdre convexe et maximisant la fonction objectif : $z = 3x_1 + 4x_2$. Pour une valeur donnée de *z*, il s'agit d'une droite. En attribuant différentes valeurs à *z* on obtient autant de droites parallèles (quelle que soit la valeur de *z*, la pente reste inchangée et vaut -3/4).

Puisque le but est de maximiser la fonction objectif, la recherche de la solution optimale se fait en translatant la droite $z = 3x_1 + 4x_2$ de manière à ce que l'ordonnée à l'origine soit la plus grande. De plus, pour satisfaire les contraintes, l'intersection de cette droite avec la région hachurée doit être non vide.

Sur la figure I.1, la fonction objectif est tracée pour trois valeurs de *z*. La première valeur est $z_1 = 18$. Cette valeur n'est pas optimale puisque l'on peut encore déplacer la fonction objectif en gardant des points communs avec la région hachurée. La deuxième valeur $z_2 = 28$. Il s'agit de la valeur optimale puisque l'ordonnée à l'origine est la plus grande possible tout en gardant un point d'intersection avec la région hachurée.

La troisième valeur $z_3 = 38$ ne peut pas être optimale puisqu'il n'y a plus de point en commun entre la droite et la région hachurée.

Ainsi dans cet exemple, la solution optimale se situe à l'intersection des deux droites $2x_1 + x_2 = 12$ et $x_1 + 2x_2 = 12$. Il s'agit d'une solution unique donnée par $x_1 = 4$ et $x_2 = 4$. Cela signifie que le bénéfice est maximal lorsqu'on produit 4 chaises et 4 tables par jour. Ce bénéfice s'élève à $z = 3(4) + 4(4) = 28$ par jour.

À noter que la solution optimale est l'un des sommets du polyèdre (point extrême).

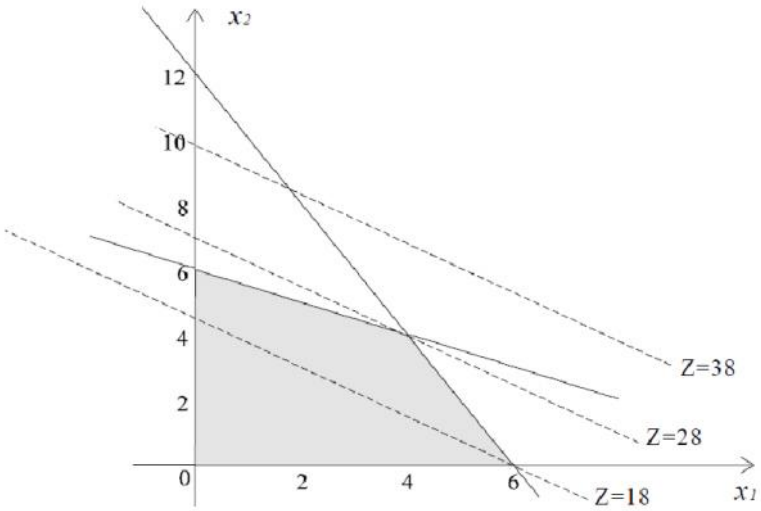


Figure I.1 : Région réalisable et fonction objectif pour $z_1 = 18$, $z_2 = 28$ et $z_3 = 38$

I-3-2 Problème de Mélange. Il faut mélanger trois gaz de telle manière que le gaz mixte soit le plus bon marché que possède un pouvoir calorifique entre plus de 1700 et 2000 k. cal/ m^3 et un taux de sulfure au plus de 2,8 g/ m^3 . Indications sur les trois gaz :

Gaz	Pouvoir calorifique k. cal/ m^3	Taux de sulfure g/ m^3	Prix en DA
1	1000	6	100
2	2000	2	250
3	1500	3	200

Ecrire le modèle mathématique de ce problème.

Formulation du problème

- ✓ **Variables :** x_1, x_2 et x_3 sont les nombres en m^3du 1^{er}, $2^{ème}$ et $3^{ème}$ gaz à fabriquer respectivement.
- ✓ **Fonction objectif à maximiser :** La fonction objective Z correspond au profit total:

$$Z = 100x_1 + 250x_2 + 200x_3$$

On cherche donc

$$MaxZ = 100x_1 + 250x_2 + 200x_3$$

- ✓ **Contraintes :**
  - Pouvoir calorifique $1700 \leq 1000x_1 + 2000x_2 + 1500x_3 \leq 2000$
  - Taux de sulfure $6x_1 + 2x_2 + 3x_3 \leq 2.8$
  - Positivité des variables $x_1 \geq 0,\ x_2 \geq 0\ , x_3 \geq 0$

I-3-3 Problème de découpe. Une usine a reçu des plaques de métal d'une largeur de 200 cm et d'une longueur de 500 cm. Il faut en fabriquer au moins 30 plaques de largeur de 110 cm , 40 plaques de largeur de 75 cm et 15 plaques de largeur de 60 cm. Donner le modèle mathématique pour que les déchets soient les plus petits possibles .

Formulation du problème

Une plaque de 200 cm de largeur peut être découpée de cinq façons :

1. une plaque de 75 cm et deux plaques de 60 cm. Les déchets seront de 05 cm.

2. une plaque de 110 cm et une plaque de 75 cm. Les déchets seront de 15 cm.

3. une plaque de 110 cm et une plaque de 60 cm. Les déchets seront de 30 cm.

4. trois plaques de 60 cm. Les déchets seront de 20 cm.

5. deux plaques de 75 cm. Les déchets seront de 50 cm.

- ✓ **Variables :** x_1, x_2 ,x_3,x_4 et $x_5$sont les nombres de plaques a découper parla 1ere, 2ème,3ème,4ème et la 5ème façons respectivement.
- ✓ **Fonction objectif à minimiser :** La fonction objective Z correspond déchet total:

$$Min\ Z = 5x_1 + 15x_2 + 30x_3 + 20x_4 + 50x_5$$

On cherche donc

$$Min\ Z = 5x_1 + 15x_2 + 30x_3 + 20x_4 + 50x_5$$

- ✓ **Contraintes :**
  - Plaques de largeur 110cm $x_2 + x_3 \geq 30$
  - Plaques de largeur 75cm $x_1 + x_2 + x_5 \geq 40$
  - Plaques de largeur 60cm $2x_1 + x_3 + 3x_4 \geq 15$
  - Positivité des variables $x_1 \geq 0, \dots, x_5 \geq 0$

I-3-4 Problème de transport.

Il s'agit ici d'un problème que l'on peut résoudre par la programmation linéaire, c'est-à-dire un problème de transport. Ce type de problème se définit comme suit.

Connaissant les quantités disponibles de chacune des unités de production, les quantités requises aux points de distribution et le coût de transport d'un bien d'une usine vers un point de vente, il s'agit de déterminer le plan de transport optimal, c'est-à-dire de déterminer les quantités de biens que chaque usine va envoyer vers chacun des points de vente afin que le coût de transport total soit minimum. On suppose qu'il est possible d'expédier des produits de n'importe quelle origine vers n'importe quelle destination.

L'exemple ci-dessous est le cas d'une fabrique de conserves qui expédie des caisses vers ses dépôts. Nous voulons que le plan d'expédition des caisses minimise le coût total de transport des usines aux dépôts. Pour simplifier, supposons qu'il y a deux usines et trois dépôts. L'offre des usines et les demandes des dépôts sont les suivantes (en nombre de caisses) :

Usine 1 : 350 dépôt 1 : 200
Usine 2 : 450 dépôt 2 : 300
dépot 3 : 50

Les coûts de transport de chaque origine vers chaque destination sont donnés dans le tableau ci-dessous (en DA par caisse) :

Usines	Dépôts		
	1	2	3
1	25	17	16
2	24	18	14

Ainsi, le coût pour transporter une caisse de l'usine 1 au dépôt 1 est de 25, le coût pour transporter une caisse de l'usine 2 vers le dépôt 1 est de 24, et ainsi de suite. Ce problème peut se formuler sous la forme d'un problème de programmation linéaire. Notons par c_{ij} le coût de transport d'une caisse de l'origine *i* vers la destination *j*. Nous avons donc, d'après le tableau précédent :

$$c_{11} = 25 \ , c_{12} = 17 \ , c_{13} = 16 \ , c_{21} = 24 \ , c_{22} = 18 \ , c_{23} = 14$$

Soit a_i la quantité de caisses disponibles à l'origine *i* et b_j celle requise à la destination *j*.
Nous pouvons représenter le problème sous forme d'un diagramme (figure I.2). Les lignes qui relient les usines aux dépôts peuvent être considérées comme des routes. On y indique leur coût de transport unitaire respectif.
À noter que le nombre de caisses disponibles doit être supérieur ou égal au nombre de caisses requises :

$$a_1 + a_2 \geq b_1 + b_2 + b_3$$

Dans le cas contraire, le problème n'a pas de solutions réalisables.
Si x_{ij} représente le nombre de caisses expédiées de l'origine *i* vers la destination *j*, le coût total de l'expédition se traduit alors par l'équation :

$$z = \sum_{j=1}^{3} \sum_{i=1}^{2} c_{ij} x_{ij} = c_{11}x_{11} + c_{12}x_{12} + c_{13}x_{13} + c_{21}x_{21} + c_{22}x_{22} + c_{23}x_{23}$$

$$z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23}$$

C'est la fonction objectif à minimiser.

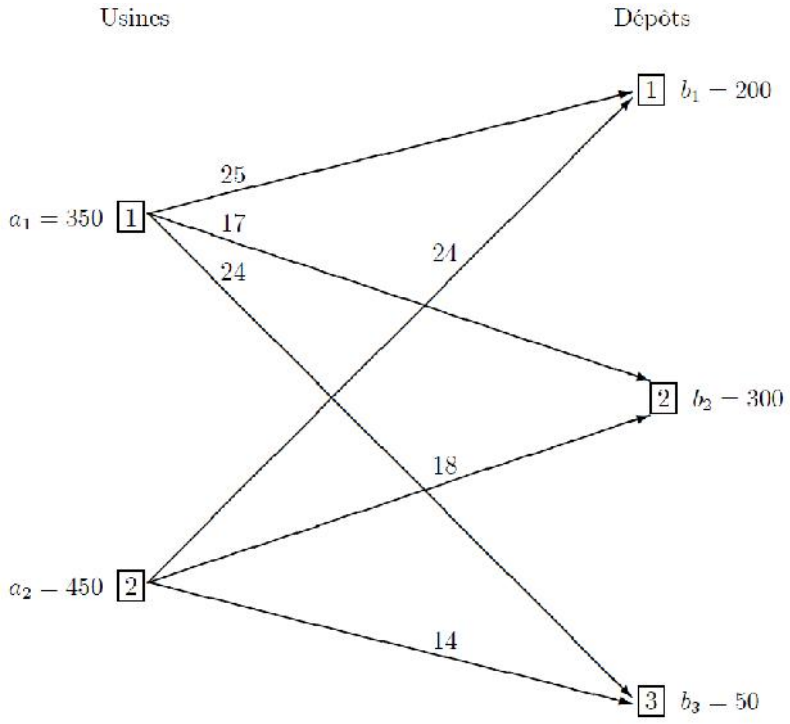


Figure I.2 : Coût de transport unitaire des usines aux dépôts

Comme il est impossible d'expédier plus de caisses d'une origine donnée qu'il n'y en a de disponibles, nous sommes confrontés aux deux contraintes :

$$\sum_{j=1}^{3} x_{1j} = x_{11} + x_{12} + x_{13} \leq 350 \qquad (usine\ 1)$$

$$\sum_{j=1}^{3} x_{2j} = x_{21} + x_{22} + x_{23} \leq 450 \qquad (usine\ 2)$$

De plus, il faut approvisionner chacun des trois dépôts avec la quantité requise, ce qui nous donne trois nouvelles contraintes :

$$\sum_{i=1}^{2} x_{i1} = x_{11} + x_{21} = 200 \qquad (dépot\ 1)$$

$$\sum_{i=1}^{2} x_{i2} = x_{12} + x_{22} = 300 \qquad (dépot\ 2)$$

$$\sum_{i=1}^{2} x_{i3} = x_{13} + x_{23} = 50 \qquad (dépot\ 3)$$

Comme il n'est pas possible d'expédier des quantités négatives, nous avons encore les six contraintes de non-négativité suivantes :

$$x_{ij} \geq 0, \quad i = 1,2 \quad et \quad j = 1,2,3$$

Finalement, le programme linéaire à résoudre est :

$$\begin{aligned} &Minimiser && Z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23} \\ &Sous\ contraintes && x_{11} + x_{12} + x_{13} \leq 350 \\ & && x_{21} + x_{22} + x_{23} \leq 450 \\ & && x_{11} + x_{21} = 200 \\ & && x_{12} + x_{22} = 300 \\ & && x_{13} + x_{23} = 50 \\ & && x_{ij} \geq 0, \quad i = 1,2 \quad et \quad j = 1,2,3 \end{aligned}$$

Comme ce problème présente plus de deux variables, nous ne pouvons pas le résoudre géométriquement.

I.4 Résolution du problème par la méthode Simplexe :

La méthode de simplexe est l'outil principal de résolution des problèmes de programmation linéaire. Elle consiste à suivre un certain nombre d'étapes avant d'obtenir la solution d'un problème donné. Il s'agit d'une méthode algébrique itérative qui permet de trouver la solution exacte d'un problème de programmation linéaire en un nombre fini d'étapes.

La forme matricielle permet de représenter un problème de programmation linéaire sous une forme plus concise.

$$Max\ Z = cx \tag{I.4}$$

$$\begin{matrix} \leq \\ Ax = b \\ \geq \end{matrix} \qquad x \geq 0 \tag{I.5}$$

Où c est un vecteur-ligne de dimension ($1 \times n$), x un vecteur-colonne de dimension ($n \times 1$), $\boldsymbol{A}$ une matrice de dimension ($m \times n$), b un vecteur-colonne de dimension ($m \times 1$) et 0 le vecteur nul à n composantes.
Un problème de programmation linéaire peut se présenter sous différentes formes. En voici la terminologie.

- **Forme canonique**

Si la fonction objectif doit être maximisée et si toutes les contraintes sont des inéquations du type $\leq$, on dit que le programme linéaire se présente sous une forme canonique. Matriciellement, un problème de programmation linéaire se présente sous sa forme canonique de la manière suivante :

$$Maximiser \quad z = cx \tag{I.6}$$
$$sous\ contraintes \quad Ax \leq b \tag{I.7}$$
$$et \quad x \geq 0 \tag{I.8}$$

À noter que toute contrainte peut être transformé sous forme canonique.

- **Forme standard**

Un problème de programmation linéaire se présente sous sa forme standard si toutes les contraintes sont des équations. La fonction objectif doit également être maximisée. Sous forme matricielle, la forme standard s'écrit :

$$Maximiser \quad z = cx \tag{I.9}$$
$$sous\ contraintes \quad Ax = b \tag{I.10}$$
$$et \quad x \geq 0 \tag{I.11}$$

- **Transformation minimisation-maximisation**

Tout problème de minimisation peut être transformé en un problème équivalent de maximisation. En effet, le problème :

$$Minimiser \quad z = cx$$

est équivalent à :

$$Maximiser \quad -z = -cx$$

- **Variables d'écart**

La procédure que nous allons développer pour trouver la solution d'un problème de programmation linéaire s'appelle la méthode du simplexe. Cette méthode exige que le programme linéaire soit sous forme standard. Pour cette raison, il faut transformer les inégalités rencontrées en égalités. Cette transformation se fait simplement en introduisant des variables non-négatives (qui vérifient les contraintes de non-négativité) appelées variables d'écart.
Si les contraintes sont du type :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

Nous introduisons une nouvelle variable $x_{n+i} \geq 0$ et écrivons :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + x_{n+i} = b_i$$

De même, nous pouvons être contraints de transformer les contraintes de la forme :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i$$

en une égalité en soustrayant cette fois une variable d'écart $x_{n+i} \geq 0$. Nous écrivons alors :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - x_{n+i} = b_i$$

Dans la fonction objectif le réel c_j est souvent appelé le "prix" associé à la variable x_j. En assignant un prix nul à chaque variable d'écart, la transformation des contraintes en un système d'équations linéaires ne change pas la fonction objectif.

Exemple I.1

Résoudre le programme linéaire suivant à l'aide de la méthode algébrique :

Maximiser la fonction objectif : $Z = 7x_1 + 4x_2$
Soumise aux contraintes linéaires suivantes :

$$2x_1 + x_2 \leq 140$$
$$x_1 + x_2 \leq 104$$
$$5x_1 + 3x_2 \leq 360$$

$$\text{Et } x_1 \geq 0, \quad x_2 \geq 0$$

En introduisant les variables d'écart dans chaque contrainte, on obtient la forme standard du modèle :

$$2x_1 + x_2 + x_3 = 140$$
$$x_1 + x_2 + x_4 = 104$$
$$5x_1 + 3x_2 + x_5 = 360$$

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_4 \geq 0, \quad x_5 \geq 0$$

$Z = \max(7x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5)$ Fonction objectif reste inchangée

I.4.1 Bases et solutions de base des programmes linéaires

Pour développer la méthode du simplexe, nous avancerons les hypothèses suivantes :
1. $r(A) = r(A \mid b)$, c'est-à-dire que le système d'équations est compatible,
2. $r(A) = m$, où m est le nombre de contraintes.

La seconde hypothèse permet de former, à partir de ***A***, une (***m*×*m***) sous matrice ***B*** non-singulière. Cette matrice ***B*** peut être formée par n'importe quel ensemble de ***m*** colonnes linéairement indépendantes de ***A***. Les colonnes de ***B*** seront notées $\boldsymbol{b_1, b_2, \ldots, b_m}$ (à ne pas confondre avec le second membre ***b***). La matrice ***B*** est appelée matrice de base puisqu'elle est formée de ***m*** vecteurs linéairement indépendants.
Sans restreindre la généralité, on peut supposer que les colonnes de ***A*** ont été ordonnées de manière à pouvoir écrire ***A*** sous la forme $A = (B, N)$, avec ***B*** de dimension (***m*×*m***) la matrice de base et ***N*** de dimension $(m \times (n - m))$ contenant les colonnes de ***A*** qui n'appartiennent pas à **B**. Le vecteur ***x*** peut être partitionné de façon analogue en posant = $\begin{matrix} x_B \\ x_N \end{matrix}$. Les variables $\boldsymbol{x_B}$ sont appelés variables de base et les variables $\boldsymbol{x_N}$ les variables hors base.
Finalement le vecteur ***c*** peut lui aussi être partitionné de la même manière en $\boldsymbol{c = (c_B, c_N)}$.
Le programme linéaire (I.9)-(I.11) peut donc être reformulé de la manière suivante :

$$\begin{matrix} Maximiser & z = c_B x_B + c_N x_N & \text{(I.12)} \\ sous\ contraintes & Bx_B + Nx_N = b & \text{(I.13)} \end{matrix}$$

$$et \qquad x_B, x_n \geq 0 \qquad \text{(I.14)}$$

La contrainte (I.13) peut s'écrire de manière équivalente :

$$Bx_B = b - Nx_N$$

et puisque ***B*** est non-singulière (inversible) :

$$x_B = B^{-1}b - B^{-1}Nx_N \qquad \text{(I.15)}$$

Lorsque toutes les variables hors base sont nulles, $x_N = 0$, (I.15) devient

$$x_B = B^{-1}b$$

On appelle solution de base la solution :

$$\begin{matrix} x_B \\ x_N \end{matrix} = \begin{matrix} B^{-1}b \\ 0 \end{matrix}$$

Lorsque $x_B = B^{-1}b \geq 0$ et $x_N = 0$, on parle de **solution réalisable de base**.

Exemple I.2 Soit le problème de programmation linéaire suivant :

$$\begin{array}{ll} \text{Maximiser} & z = 3x_1 + 5x_2 + x_3 \\ \text{Sous contraintes} & x_1 + 2x_2 - x_3 \leq 16 \\ & 3x_1 - 4x_2 \leq 20 \\ \text{Et} & x_1, x_2, x_3 \geq 0 \end{array}$$

Ce problème peut se mettre sous forme standard en introduisant les variables d'écart x_4 et x_5 :

$$\begin{array}{ll} \text{Maximiser} & z = 3x_1 + 5x_2 + x_3 + 0x_4 + 0x_5 \\ \text{Sous contraintes} & x_1 + 2x_2 - x_3 + x_4 = 16 \\ & 3x_1 - 4x_2 + x_5 = 20 \\ \text{Et} & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array}$$

Sous forme matricielle, on obtient donc :

$$c = \begin{matrix} 3 & 5 & 1 & 0 & 0 \end{matrix}, \; A = \begin{matrix} 1 & 2 & -1 & 1 & 0 \\ 3 & -4 & 0 & 0 & 1 \end{matrix}, \; x = \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{matrix}, b = \begin{matrix} 16 \\ 20 \end{matrix}$$

Formons à partir de ***A*** une matrice de base ***B*** en prenant par exemple les colonnes 2 et 4, dans cet ordre :

$$B = \begin{matrix} 2 & 1 \\ -4 & 0 \end{matrix}$$

Il s'agit bien d'une base puisque ***B*** est non-singulière ($|B| = 4 \neq 0$).

À noter que dans notre cas, on a :

$b_1 = a_2$ (2^e colonne de ***A***)

$b_2 = a_4$ (4^e colonne de ***A***).

A cette matrice de base correspond une solution de base donnée par :

$$x_B = B^{-1}b$$

Dans notre cas :

$$B^{-1} = \begin{matrix} 0 & -1/4 \\ 1 & 1/2 \end{matrix}$$

$$x_B = \begin{matrix} 0 & -1/4 \\ 1 & 1/2 \end{matrix} \begin{matrix} 16 \\ 20 \end{matrix} = \begin{matrix} -5 \\ 26 \end{matrix}$$

Les autres variables étant nulles, $x_1 = x_3 = x_5 = 0$. Cette solution de base n'est pas réalisable pour la simple raison que $x_{B1} = x_2 = -5$ viole la contrainte de non-négativité.
Dans cet exemple, il est très facile de trouver une base qui fournisse une solution réalisable de base. En effet les colonnes $\boldsymbol{a_4}$ et $\boldsymbol{a_5}$ forment une base qui est l'identité :

$$B = I = \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}$$

Ici, $b_1 = a_4$ et $b_2 = a_5$.
La solution de base est le vecteur $\boldsymbol{b}$ puisque :

$$x_B = Ib = b = \begin{matrix} 16 \\ 20 \end{matrix}$$

Comme $\boldsymbol{b}$ doit être non-négatif lorsque le problème est présenté sous sa forme standard, la solution est une solution réalisable de base.
Z a pour valeur *: z = 0· 16 + 0 · 20 = 0*

I.4.2 Recherche d'une solution optimale

Résultat 4.1 *Tout point extrême est une solution réalisable de base.*
Ainsi pour trouver la solution optimale, il suffit d'examiner les points extrêmes ou de manière équivalente les solutions réalisables de base.
À la solution réalisable de base initiale correspond une valeur de la fonction objectif *z*. Le but est d'améliorer la valeur de la fonction objectif en l'évaluant en un point extrême adjacent. Pour cela, étant donné une base $\boldsymbol{B}$, on trouve un point extrême adjacent (nouvelle solution réalisable de base) en échangeant l'une des colonnes de la base (b_i) contre une colonne a_j qui n'est pas dans la base. Cependant, en faisant cet échange, il faut s'assurer que la solution de base reste réalisable et que la valeur de la fonction objectif augmente (ou du moins ne diminue pas).
Il y a donc deux règles à suivre pour cet échange. La première consiste à déterminer quelle colonne a_j de $\boldsymbol{A}$ (à laquelle correspond une variable x_j) doit entrer dans la base pour améliorer la fonction objectif. La seconde consiste à sélectionner l'une des colonnes b_i qui doit quitter la base de manière à ce que la nouvelle solution de base reste réalisable.
Nous développons ici les critères d'entrée et de sortie de la base pour obtenir une nouvelle solution réalisable de base qui améliore la valeur de la fonction objectif. Il restera ensuite à déterminer si la nouvelle solution réalisable de base est optimale ou non.
Pour développer ces deux critères, nous reformulons le programme linéaire
(I.12)-(I.14) sous la forme suivante :

$$Maximiser \qquad z = c_B x_B + \sum_{j \in J} c_j x_j \tag{I.16}$$

$$sous\ contraintes \quad Bx_B + \sum_{j \in J} x_j a_j = b \tag{I.17}$$

$$et \qquad x_B, x_j \geq 0\ , j \in J \tag{I.18}$$

où $\boldsymbol{J}$ est l'ensemble des indices des variables hors base.
Comme $\boldsymbol{B}$ est une base formée de m colonnes linéairement indépendantes, toute colonne a_j de la matrice $\boldsymbol{A}$ peut s'écrire comme une combinaison linéaire des colonnes de la matrice $\boldsymbol{B}$. On peut donc écrire :

$$a_j = y_{1j}b_1 + y_{2j}b_2 + \cdots + y_{mj}b_m = \sum_{i=1}^{m} y_{ij}b_i = By_j$$

On peut également écrire (puisque $\boldsymbol{B}$ est inversible) :

$$y_i = B^{-1}a_j, \qquad j = 1, \ldots, n \tag{I.19}$$

Avec $y_j = \begin{matrix} y_{1j} \\ y_{2j} \\ \vdots \\ y_{mj} \end{matrix}$

Définissons encore une nouvelle variable z_j par :

$$z_j = y_{1j}c_{B1} + y_{2j}c_{B2} + \cdots + y_{mj}c_{Bm} = \boldsymbol{c_B y_j} \quad \text{(I.20)}$$

En utilisant (I.19), les contraintes (I.17) s'écrivent :

$$\boldsymbol{B x_B} + \sum_{j \in \bar{J}} x_j \boldsymbol{a_j} = \boldsymbol{b}$$

$$\boldsymbol{x_B} = \boldsymbol{B^{-1}}\ b - \sum_{j \in \bar{J}} x_j \boldsymbol{a_j}$$

$$\boldsymbol{x_B} = \boldsymbol{B^{-1}} b - \sum_{j \in \bar{J}} x_j \boldsymbol{B^{-1} a_j}$$

$$\boldsymbol{x_B} = \boldsymbol{B^{-1}} b - \sum_{j \in J} x_j \boldsymbol{y_j} \quad \text{(I.21)}$$

Introduisons (I.21) dans la fonction objectif (I.16) :

$$z = \boldsymbol{c_B}\ \boldsymbol{B^{-1} b} - \sum_{j \in \bar{J}} x_j \boldsymbol{y_j} + \sum_{j \in \bar{J}} c_j x_j$$

Finalement, en utilisant (I.20), la fonction objectif s'écrit :

$$z = \boldsymbol{c_B B^{-1} b} - \sum_{j \in J} x_j \boldsymbol{c_B y_j} + \sum_{j \in J} c_j x_j$$

$$z = \boldsymbol{c_B B^{-1} b} - \sum_{j \in \bar{J}} x_j z_j + \sum_{j \in \bar{J}} c_j x_j$$

D'où

$$z = \boldsymbol{c_B B^{-1} b} - \sum_{j \in J} z_j - c_j\ x_j \quad \text{(I.22)}$$

Lorsqu'on obtient une solution réalisable de base, les valeurs des variables x_j sont nulles pour tout indice $j \in J$. Dans ce cas, la valeur de la fonction objectif est $z = \boldsymbol{c_B B^{-1} b}$.

Le but est d'améliorer au maximum cette valeur de z. Dans l'équation (I.22), les variables x_j sont non-négatives et la sommation est précédée du signe négatif. C'est donc la variable x_j , au plus petit $z_j - c_j < 0$, qui améliorera le plus la valeur de la fonction objectif. Le critère d'entrée dans la base est donc le suivant.

• Critère d'entrée dans la base

Le vecteur a_k qui doit entrer dans la base correspond à la variable hors base x_k pour laquelle $(z_j - c_j)$ a la plus petite valeur négative :

$$z_k - c_k = min_{j \epsilon J}\ z_j - c_j \quad avec \quad z_j - c_j < 0$$

Il reste encore à déterminer quel vecteur de base doit quitter la base. Pour cela, reprenons la solution de base (I.21) :

$$\boldsymbol{x_B} = \boldsymbol{B^{-1} b} - \sum_{j \in \bar{J}} x_j \boldsymbol{y_j}$$

Toutes les variables hors base sont nulles sauf x_k qui devient une variable de base. Comme x_k est une nouvelle variable de base, elle sera notée $\hat{x}_k$. Ainsi la nouvelle solution de base, notée $\hat{x}_B$ s'écrit :

$$\hat{x}_B = \boldsymbol{B^{-1}b} - \hat{x}_j \boldsymbol{y}_j$$

La i[ème] composante de cette solution de base est :

$$\hat{x}_{Bi} = x_{B1} - \hat{x}_k y_{ik} \qquad \text{(I.23)}$$

Pour que cette solution de base soit réalisable, il faut que les variables de base soient non-négatives, c'est-à-dire :

$$\hat{x}_{Bi} = x_{B1} - \hat{x}_k y_{ik} \geq 0, \quad i = 1, \dots, m \qquad \text{(I.24)}$$

Les contraintes de non-négativité (I.24) sont évidemment satisfaites pour les variables qui ont un y_{ik} négatif ou nul. En revanche, en ce qui concerne les variables qui ont y_{ik} positif, il faut que :

$$\hat{x}_k \leq \frac{x_{Bi}}{y_{ik}}$$

Pour ne pas violer les contraintes de non-négativité. Cette dernière inégalité devant être valable pour tout i tel que $y_{ik} > 0$, $\hat{x}_k$ doit être égale au plus petit rapport x_{Bi}/y_{ik}. Supposons que ce plus petit rapport soit associé à la r[ème] variable de base x_{Br}.
Alors en posant :

$$\hat{x}_k = \frac{x_{Br}}{y_{rk}}$$

Toutes les contraintes de non-négativité dans (I.24) sont satisfaites et par (I.23) :

$$\hat{x}_{Br} = x_{Br} - \frac{x_{Br}}{y_{rk}} y_{rk} = 0$$

ce qui permet de sortir x_{Br} de la base. Nous avons ainsi obtenu le critère pour sortir une colonne $\boldsymbol{b}_r$ de la base.

• Critère de sortie de la base

Soit le vecteur a_k qui doit entrer dans la base. Le vecteur b_r associé à la variable x_{Br} qui doit sortir de la base est celui qui satisfait :

$$\frac{x_{Br}}{y_{rk}} = min_i \ \frac{x_{Bi}}{y_{ik}}, y_{ik} > 0$$

Résultat 4.2 *La solution réalisable de base associée à la base* ***B*** *est optimale si :*

$$(z_j - c_j) \geq 0$$

pour chaque colonne a_j *de* ***A*** *qui n'est pas dans la base.*

Remarque 4.1 *Il se peut que la solution optimale d'un problème de programmation linéaire possédant une solution réalisable de base soit infinie.*

• Étapes de la méthode du simplexe

Etape 1 : Rechercher la solution optimale à partir d'une solution réalisable de base :

$$\boldsymbol{x}_B = B^{-1}\boldsymbol{b}$$

Etape 2 : Examiner les (z_j - c_j) pour toutes les colonnes a_j qui ne sont pas dans la base. Si tous les $(z_j - c_j) \geq 0$, passer à l'étape 6, sinon passer à l'étape 3.

Etape 3 : S'il existe un (z_j - c_j) négatif pour lequel il n'y a pas d'éléments positifs dans y_j, arrêter la recherche puisque la solution optimale est infinie.
Sinon, choisir le vecteur (et donc la variable) associée à la valeur la plus négative des (z_j - c_j) pour entrer dans la base :

$$z_k - c_k = min_{j \epsilon J}\ z_j - c_j \quad avec \quad z_j - c_j < 0$$

Etape 4 : Déterminer le vecteur (et donc la variable) qui sort de la base à l'aide du critère :

$$\frac{x_{Br}}{y_{rk}} = min_i \ \frac{x_{Bi}}{y_{ik}}, y_{ik} > 0$$

Etape 5 : Établir la nouvelle base, calculer la nouvelle solution réalisable de base et la nouvelle valeur de la fonction objectif. Retourner à l'étape 2.

Etape 6 : La solution réalisable de base courante est la solution optimale. La solution optimale n'est pas unique s'il existe un (z_j - c_j) nul pour un vecteur a_j qui ne se trouve pas dans la base.

I.5 Tableaux du simplexe et procédure de calcul

Lorsque l'on calcule la méthode du simplexe à la main, il est préférable de travailler avec un tableau qui contient toutes les données nécessaires. À chaque itération correspond un nouveau tableau prenant la forme suivante :

			c_1		c_j		c_n
c_B	vecteurs de base	x_B	a_1	...	a_j	...	a_n
c_{B1}	b_1	$x_{B1} = y_{10}$	y_{11}	...	y_{1j}	...	y_{1n}
c_{B2}	b_2	$x_{B2} = y_{20}$	y_{21}	...	y_{2j}	...	y_{2n}
⋮	⋮	⋮	⋮	⋱	⋮	⋱	⋮
c_{Bm}	b_m	$x_{Bm} = y_{m0}$	y_{m1}		y_{mj}		y_{mn}
		$z - y_{m+1,0}$	$z_1 - c_1 - y_{m+1,1}$	...	$z_j - c_j - y_{m+1,j}$	...	$z_n - c_n - y_{m+1,n}$

Tableau 1 : Tableau initial du simplexe

La première colonne indique les prix c_B correspondant aux vecteurs de base. La deuxième colonne indique quels vecteurs se trouvent dans la base. Les vecteurs b_i, $i = 1, \ldots, m$ représentent les m colonnes de la matrice de base *B*. Si la matrice de base est formée des colonnes a_1, a_3 et a_4 par exemple, on écrira a_1 à la place de b_1, a_3 à la place de b_2 et a_4 à la place de b_3. La troisième colonne du tableau fournit la solution réalisable de base x_B sauf à la dernière ligne où l'on trouve la valeur de *z* pour la solution réalisable de base. Les colonnes suivantes indiquent les valeurs des y_j pour tous les vecteurs de ***A***. La première ligne donne les prix associés aux variables et la dernière ligne donne les (z_j - c_j).
Voyons comment utiliser les informations données par le tableau pour effectuer les différentes étapes de la méthode du simplexe.
En premier lieu, examinons les éléments de la dernière ligne (z_j - c_j) correspondant aux vecteurs hors base a_j. Si tous les $z_j - c_j \geq 0$, la solution est optimale. S'il existe un $z_j - c_j < 0$ pour lequel il n'existe pas d'éléments positifs dans y_j (la colonne au-dessus de (zj □ cj)), la solution est infinie. Si tel n'est pas le cas, nous choisissons le plus petit $z_j - c_j$. Appelons la colonne correspondante *k*. Ainsi a_k entre dans la base. Le vecteur qui doit sortir de la base est choisi par le critère :

$$\frac{x_{Br}}{y_{rk}} = min_i \ \frac{x_{Bi}}{y_{ik}}, y_{ik} > 0$$

Cela signifie que le vecteur de la $r^{ème}$ ligne dans la colonne "vecteurs de base" est remplacé par a_k. Les valeurs nécessaires au calcul de ce critère sont faciles à trouver, puisque les valeurs x_{Bi} se trouvent en

regard des valeurs y_{ik}. Il faut à présent tracer un nouveau tableau. A l'intersection de la colonne a_k qui entre dans la base et du vecteur b_r qui en sort, se trouve le terme y_{rk} qui est appelé le **pivot** de la transformation. La colonne a_k est appelée **colonne-pivot** et le vecteur b_r **ligne-pivot**. Il est utile d'encercler dans le tableau le pivot ainsi que la colonne-pivot et la ligne-pivot. Établissons les formules de transformation en ajoutant le symbole (ˆ) au-dessus des nouvelles valeurs afin de les distinguer des anciennes. Les nouvelles valeurs des variables de base se calculent facilement à partir des anciennes.
En effet, nous avons vu que la nouvelle variable qui entre dans la base est :

$$\hat{x}_{Br} = \hat{x}_k = \frac{x_{Br}}{y_{rk}} \tag{I.25}$$

En substituant (I.25) dans (I.23), nous obtenons :

$$\hat{x}_{Bi} = \hat{x}_{Bi} - \frac{x_{Br}}{y_{rk}} y_{ik} \qquad i \neq r \tag{I.26}$$

Développons à présent les formules de transformation afin de calculer les termes $\hat{y}_{ij}$. Toute colonne a_j de $\boldsymbol{A}$ peut s'exprimer comme une combinaison linéaire des anciens vecteurs de base :

$$\boldsymbol{a}_j = y_{1j}\boldsymbol{b}_1 + \cdots + y_{mj}\boldsymbol{b}_m = \sum_{i=1}^{m} y_{ij}\boldsymbol{b}_i \tag{I.27}$$

Or le vecteur a_k a remplacé le vecteur b_r dans la base. De (I.27), en posant $j = k$, on déduit que :

$$\boldsymbol{b}_r = -\sum_{i=1\ i\neq r}^{m} \frac{y_{ik}}{y_{rk}}\boldsymbol{b}_i + \frac{1}{y_{rk}}\boldsymbol{a}_k \tag{I.28}$$

Substituons (I.28) dans (I.27) :

$$\boldsymbol{a}_j = \frac{y_{rj}}{y_{rk}}\boldsymbol{a}_k + \sum_{i=1\ i\neq r}^{m} \left(y_{ij} - y_{rj}\frac{y_{ik}}{y_{rk}}\right)\boldsymbol{b}_i = \sum_{i=1}^{m} \hat{y}_{ij}\hat{b}_i \tag{I.29}$$

où $\hat{b}_i = b_i,\ i \neq r\ et\ \hat{b}_r = a_k$. Si l'on compare les deux égalités dans (I.29), on trouve :

$$\hat{y}_{ij} = y_{ij} - \frac{y_{rj}}{y_{rk}} y_{ik}, \quad i \neq r \tag{I.30}$$

$$\hat{y}_{rj} = \frac{y_{rj}}{y_{rk}} \tag{I.31}$$

Calculons les nouvelles valeurs $\hat{z}_j - c_j$ en utilisant la définition :

$$\hat{z}_j - c_j = \hat{c}_B\hat{y}_j - c_j = \sum_{i=1}^{m} \hat{c}_{Bi}\hat{y}_{ij} - c_j \tag{I.32}$$

avec $\hat{c}_{Bi} = c_{Bi}$, $i \neq r$ et $\hat{c}_{Br} = c_k$. En utilisant les résultats (I.30) et (I.31), l'équation (I.32) devient :

$$\hat{z}_j - c_j = \sum_{i=1\ i\neq r}^{m} c_{Bi}\left(y_{ij} - y_{rj}\frac{y_{ik}}{y_{rk}}\right) + c_k\frac{y_{rj}}{y_{rk}} - c_j \tag{I.33}$$

À noter que la sommation peut se faire sans la restriction $i \neq r$ puisque le terme correspondant à cet indice est nul :

$$c_{Br}\left(y_{rj} - y_{rj}\frac{y_{rk}}{y_{rk}}\right) = 0$$

Ainsi, l'équation (I.33) peut s'écrire :

$$\hat{z}_j - c_j = \sum_{i=1}^{m} c_{Bi}y_{ij} - c_j - \frac{y_{rj}}{y_{rk}}\sum_{i=1}^{m} c_{Bi}y_{ik} - c_k$$

ou de manière équivalente :

$$\hat{z}_j - c_j = z_j - c_j - \frac{y_{rj}}{y_{rk}}\left(z_k - c_k\right) \tag{I.34}$$

Enfin, déterminons $\hat{z}$, la nouvelle valeur de la fonction objectif :

$$\hat{z} = \hat{c}_B\hat{x}_B = \sum_{i=1}^{m} \hat{c}_{Bi}\hat{x}_{Bi} \tag{I.35}$$

Avec (I.25), (I.26) et le fait que $\hat{c}_{Bi} = c_{Bi}$, $i \neq r$, $\hat{c}_{Br} = c_k$, on obtient :

$$\hat{z} = \sum_{i=1\ i\neq r}^{m} c_{Bi}\left(x_{Bi} - \frac{x_{Br}}{y_{rk}}y_{ik}\right) + c_k\frac{x_{Br}}{y_{rk}}$$

Dans la sommation, le terme pour $i = r$ étant nul on peut omettre la restriction $i \neq r$. On obtient alors :

$$\hat{z} = \sum_{i=1}^{m} c_{Bi}x_{Bi} - \sum_{i=1}^{m} c_{Bi}y_{ik}\frac{x_{Br}}{y_{rk}} + c_k\frac{x_{Br}}{y_{rk}}$$

Puisque $z_k = \sum_{i=1}^{m} c_{Bi}y_{ik}$ et z= $\sum_{i=1}^{m} c_{Bi}x_{Bi}$, on a finalement :

$$\hat{z} = z - \frac{x_{Br}}{y_{rk}}\, z_k - c_k \qquad \text{(I.36)}$$

Exemple I.3 Soit l'exemple suivant

$$Maximiser\ z = 3x_1 + 4x_2$$
$$sous\ contraintes\ 2x_1 + x_2 \leq 12$$
$$x_1 + 2x_2 \leq 12$$
$$et\ x_1, x_2 \geq 0$$

Tout d'abord, passons ce programme linéaire sous forme standard.

$$Maximiser\ z = 3x_1 + 4x_2 + 0.x_3 + 0.x_4$$
$$sous\ contraintes\ 2x_1 + x_2 + x_3 = 12$$
$$x_1 + 2x_2 + x_4 = 12$$
$$et\ x_1, x_2, x_3, x_4 \geq 0$$

Appliquons les différentes étapes de la méthode du simplexe

Etape 1 : Le tableau initial du simplexe est donné dans le tableau 1.

	c_j		3	4	0	0
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4
0	a_3	12	2	1	1	0
0	a_4	12	1	2	0	1
	$(z_j - c_j)$	$z = 0$	−3	−4	0	0

Tableau 2 : Tableau initial du simplexe

La base B est formée par les vecteurs a_3 et a_4 et correspond à la matrice identité. La solution réalisable de base correspondante est donnée par

$$\boldsymbol{x}_B = B^{-1}\boldsymbol{b} = \boldsymbol{I}\boldsymbol{b} = \begin{matrix} 12 \\ 12 \end{matrix}$$

Les termes (z_j -c_j) correspondant aux vecteurs de base sont nuls et ceux correspondant aux vecteurs hors base s'obtiennent en calculant :

$$z_1 - c_1 = c_{B1}y_{11} + c_{B2}y_{21} - c_1 = 0\ 2 + 0\ 1 - 3 = -3$$
$$z_2 - c_2 = c_{B1}y_{12} + c_{B2}y_{22} - c_2 = 0\ 1 + 0\ 2 - 4 = -4$$

Enfin, la fonction objectif a pour valeur :

$$z = c_{B1}x_{B1} + c_{B2}x_{B2} = 0(12) + 0(12) = 0$$

Etape 2 : Dans le tableau 2, nous voyons qu'il reste des (z_j - c_j) négatifs. Nous passons donc à l'étape 3.

Etape 3 : Les vecteurs y_j situés au-dessus des (z_j -c_j) négatifs n'ont que des éléments positifs. Une amélioration de la fonction objectif est donc possible et nous cherchons le vecteur qui doit entrer dans la base.

La plus petite valeur négative des (z_j -c_j) est (-4) obtenue pour (z_2 - c_2). Ainsi, $k = 2$ et le vecteur a_2 entre dans la base. La colonne correspondante est la colonne-pivot qui a été encadrée dans le tableau 2. Pour connaître le vecteur qui sort de la base, nous passons à l'étape 4.

Etape 4 : Examinons les rapports $\frac{x_{Bi}}{y_{i2}}$ avec $y_{i2} > 0$:

$$\frac{x_{B1}}{y_{12}} = \frac{12}{1} = 12$$

et $$\frac{x_{B2}}{y_{22}} = \frac{12}{2} = 6.$$

Le plus petit rapport est 6 et correspond à $\frac{x_{B2}}{y_{22}}$. Ainsi, $r = 2$ et le vecteur a_4 (correspondant à la variable $x_{B2} = x_4$) sort de la base. La ligne correspondante est la **ligne-pivot** qui a été encadrée dans le tableau 2. Il reste à établir le nouveau tableau à l'étape 5.

Etape 5 : Puisque a_2 remplace a_4 dans la base, on remplace $c_{B2} = c_4 = 0$ par $c_2 = 4$ dans la colonne c_B. Tous les autres éléments du nouveau tableau se calculent à partir des formules (I.30) et (I.31). Commençons par calculer les éléments de la ligne-pivot à l'aide de (I.31)

$$\hat{y}_{rj} = \frac{y_{rj}}{y_{rk}}, \quad j = 0, 1, \ldots, n.$$

$$\hat{y}_{20} = x_{B2} = \frac{y_{20}}{y_{22}} = \frac{12}{2} = 6$$

$$\hat{y}_{21} = \frac{y_{21}}{y_{22}} = \frac{1}{2}$$

$$\hat{y}_{22} = \frac{y_{22}}{y_{22}} = 1$$

$$\hat{y}_{23} = \frac{y_{23}}{y_{22}} = 0$$

$$\hat{y}_{24} = \frac{y_{24}}{y_{22}} = \frac{1}{2}$$

Calculons à présent la première ligne à l'aide de (5.30) : $\hat{y}_{1j} = y_{1j} - \frac{y_{rj}}{y_{rk}} y_{1k}$. À noter que $\frac{y_{1k}}{y_{rk}} = \frac{y_{12}}{y_{22}} = \frac{1}{2}$ est une constante pour cette ligne.

$$\hat{y}_{10} = \hat{x}_{B1} = y_{10} - \frac{1}{2}\ y_{20} = 12 - \frac{1}{2}(12) = 6$$

$$\hat{y}_{11} = y_{11} - \frac{1}{2}\ y_{21} = 2 - \frac{1}{2}(1) = \frac{3}{2}$$

$$\hat{y}_{12} = y_{12} - \frac{1}{2}\ y_{22} = 1 - \frac{1}{2}(2) = 0$$

$$\hat{y}_{13} = y_{13} - \frac{1}{2}\ y_{23} = 1 - \frac{1}{2}(0) = 1$$

$$\hat{y}_{14} = y_{14} - \frac{1}{2}\ y_{24} = 0 - \frac{1}{2}\ 1 = -\frac{1}{2}$$

Il ne reste plus qu'à calculer les éléments de la dernière ligne :

$$\hat{y}_{3j} = y_{3j} - \frac{y_{rj}}{y_{rk}} y_{3k}. \text{ Ici, } \frac{y_{3k}}{y_{rk}} = \frac{y_{32}}{y_{22}} = \frac{-4}{2} = -2$$

$$\hat{y}_{30} = \hat{z} = y_{30} - (-2)y_{20} = 0 + 2(12) = 24$$

$$\hat{y}_{31} = \hat{z}_1 - c_1 = y_{31} + 2\ y_{21} = -3 + 2\ 1 = -1$$

$$\hat{y}_{32} = \hat{z}_2 - c_2 = y_{32} + 2\ y_{22} = -4 + 2(2) = 0$$

$$\hat{y}_{33} = (\hat{z}_3 - c_3) = y_{33} + 2(y_{23}) = 0 + 2(0) = 0$$

$$\hat{y}_{34} = (\hat{z}_4 - c_4) = y_{34} + 2(y_{24}) = 0 + 2(1) = 2$$

Le tableau 3 représente le nouveau tableau du simplexe. Avant de retourner à l'étape 2, mentionnons que certains calculs effectués lors de l'étape 5 ne sont pas nécessaires. En effet les (z_j - c_j) correspondant aux vecteurs dans la base sont forcément nuls. Le vecteur qui entre dans la base devient un vecteur unitaire avec pour composantes : 1 à la place de l'élément pivot et *0* ailleurs. Le vecteur qui

reste dans la base n'est pas modifié. De plus, la ligne-pivot est très facilement calculée puisque chaque élément de cette ligne est divisé par l'élément pivot. Nous appliquerons ces remarques dans les calculs suivants.

	c_j		3	4	0	0
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4
0	a_3	6	3/2	0	1	−1/2
4	a_2	6	1/2	1	0	1/2
	$(z_j - c_j)$	24	−1	0	0	2

Tableau 3 : Tableau du simplexe obtenu après une itération

Etape 2 : Dans le tableau 3, il reste un (z_j - c_j) négatif. Par conséquent, nous passons à l'étape 3.

Etape 3 : Seul $z_1 - c_1 = -1$ est négatif et tous les éléments de y_1 sont positifs. Par conséquent $k = 1$ et le vecteur a_1 entre dans la base ; passons à ,l'étape 4.

Etape 4 : Calculons les rapports $\frac{x_{Bi}}{y_{i1}}, y_{i1} > 0$:

$$x_{B1}y_{11} = \frac{6}{3/2} = 4$$

$$x_{B2}y_{21} = \frac{6}{1/2} = 12$$

Le plus petit rapport correspond à $x_{B1}/y_{11} = 4$, d'où $r = 1$. Le vecteur a_3 sort donc de la base. Calculons le nouveau tableau à l'étape 5.

Etape 5 : Puisque a_1 entre dans la base pour se substituer à a_3, on remplace $c_{B1} = 0$ par $c_1 = 3$. L'élément pivot étant $y_{11} = 3/2$, tous les éléments de la ligne-pivot sont divisés par 3/2. Le vecteur a_1 qui entre dans la base devient le vecteur unitaire avec pour composantes : 1 à la place de l'élément pivot et 0 ailleurs. Le vecteur a_2 qui reste dans la base n'est pas modifié. De plus $\hat{z}_1 - c_1 = 0$ et $\hat{z}_2 - c_2 = 0$, puisque a_1 et a_2 se trouvent dans la base. Les autres éléments se calculent à l'aide de (I.30) comme suit :

$$\hat{y}_{20} = \hat{x}_{B2} = y_{20} - \frac{1}{3}\, y_{10} = 6 - \frac{1}{3}(6) = 4$$

$$\hat{y}_{23} = y_{23} - \frac{1}{3}\, y_{13} = 0 - \frac{1}{3}\, 1 = -\frac{1}{3}$$

$$y_{24} = y_{24} - \frac{1}{3}\, y_{14} = \frac{1}{2} - \frac{1}{3}\left(-\frac{1}{2}\right) = \frac{2}{3}$$

$$y_{30} = \hat{z} = y_{30} + \frac{2}{3}(y_{10}) = 24 + \frac{2}{3}(6) = 28$$

$$y_{33} = \hat{z}_3 - c_3 = y_{33} + \frac{2}{3}(y_{13}) = 0 + \frac{2}{3}(1) = \frac{2}{3}$$

$$y_{34} = z_4 - c_4 = y_{34} + \frac{2}{3}\, y_{14} = 2 + \frac{2}{3} - \left(-\frac{1}{2}\right) = \frac{5}{3}$$

	c_j		3	4	0	0
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4
3	a_1	4	1	0	2/3	−1/3
4	a_2	4	0	1	−1/3	2/3
	$(z_j - c_j)$	28	0	0	2/3	5/3

Tableau 4 : Tableau du simplexe obtenu après deux itérations

Retournons à l'étape 2.

Etape 2 : Tous les (z_j -c_j) correspondant aux vecteurs hors base sont strictement positifs, nous passons donc à l'étape 6.

Etape 6 : Puisque $(z_j - c_j) > 0, \forall j \in J$, la solution optimale est donnée par :

$$x_{B1} = x_1 = 4$$
$$x_{B2} = x_2 = 4$$

Les variables hors base sont nulles : $x_3 = x_4 = 0$.

La valeur de la fonction objectif est $z = 28$.

Observons les solutions x_1 et x_2 obtenues à chaque itération. Dans le tableau initial, $x_1 = 0$ et $x_2 = 0$. Après la première itération, $x_1 = 0$ et $x_2 = 6$. Finalement, à la seconde et dernière itération, $x_1 = 4$ et $x_2 = 4$.

Jusqu'à présent, nous avons toujours étudié des cas où une solution réalisable de base initiale était connue. Cependant, cette situation (idéale) ne se rencontre pas dans tous les problèmes de programmation linéaire. Nous verrons donc dans le paragraphe suivant comment trouver une solution réalisable de base initiale.

I.6 Recherche d'une solution réalisable de base initiale

Jusqu'ici nous avons vu comment trouver une solution réalisable de base initiale lorsque toutes les contraintes sont sous forme d'inégalités du type "≤". Dans ce cas, en ajoutant une variable d'écart à chaque contrainte pour la transformer en une égalité, la matrice ***A*** qui correspond à l'ensemble des contraintes $Ax = b$ prend la forme :

$$A = (R, I)$$

où *I* est la matrice identité d'ordre ($m \times m$), puisqu'il y a m contraintes.

L'intérêt d'obtenir au départ une matrice identité comme sous-matrice de *A* est évident. En effet, comme $B = I$ est une matrice de base, une solution réalisable de base est immédiatement trouvée sous la forme : $x_B = B^{-1}b$. De plus, $y_j = B^{-1}a_j = aj, j = 1, \ldots, n$ et $c_B = 0$ puisque les prix associés aux variables d'écart sont nuls.

À noter que cette matrice identité peut apparaître sans que les variables d'écart aient été rajoutées. Dans ce cas, la méthode du simplexe s'applique également, à la différence près que les prix associés aux variables de base ne sont pas nuls.

Cependant, il n'existe pas, le plus souvent, de sous-matrice identité dans la matrice ***A***. C'est le cas par exemple lorsque des contraintes n'ont pas besoin de l'adjonction de variables d'écart (contraintes déjà sous forme d'égalités).

On peut néanmoins obtenir une matrice identité comme matrice de base initiale en considérant le nouveau système de contraintes :

$$Ax + Ix_a = (A, I) \begin{matrix} x \\ x_a \end{matrix} = b \qquad \text{(I.37)}$$

dans lequel nous avons ajouté m variables xai, i = 1, . . . ,m, dont les colonnes correspondantes sont les vecteurs $\boldsymbol{e_i}$ (vecteurs unitaires). Ces nouvelles variables sont appelées **variables artificielles** car elles n'ont aucune signification pour le système original de contraintes. Les vecteurs artificiels $\boldsymbol{e_i}$ qui correspondent aux variables artificielles seront désignés par q_i de manière à les distinguer des vecteurs a_j de la matrice ***A***. Nous avons fait apparaître une matrice identité comme matrice de base initiale. Nous disposons donc immédiatement d'une solution réalisable de base pour (I.37) qui est $x_a = b$ et $x = 0$. Nous noterons toutefois qu'il ne s'agit pas d'une solution réalisable du système original. Pour qu'une solution de (I.37) soit aussi une solution du système original, il faut que $x_a = 0$, c'est-à-dire que toutes les variables artificielles sortent de la base. Nous allons donc donner à ces variables artificielles des

prix très défavorables, de façon à ce que la fonction objectif puisse être améliorée tant qu'une de ces variables reste dans la base. Si la fonction objectif *z* doit être maximisée, en donnant un prix négatif très grand à chaque variable artificielle, on peut s'attendre à ce que *z* s'améliore aussi longtemps qu'une variable artificielle se trouve dans la base avec une valeur positive.

I.7 Initialisation de l'algorithme du simplexe (la méthode à deux phases).

Comme son nom l'indique, cette méthode consiste à résoudre un problème de programmation linéaire en deux parties. La première partie, appelée phase 1, consiste à annuler toutes les variables artificielles en utilisant une fonction objectif artificielle. Si l'on ne peut pas annuler toutes les variables, alors le problème n'est pas réalisable.

La phase II consiste à remplacer la fonction objectif artificielle de la phase 1 par la vraie fonction objectif à maximiser. On utilise alors la solution réalisable de base obtenue à la fin de la phase 1. Deux cas peuvent se présenter en ce qui concerne cette solution réalisable de base.

Premier cas La solution réalisable de base obtenue à la fin de la phase 1 ne contient plus de variables artificielles. Dans ce cas, on utilise simplement l'algorithme du simplexe décrit précédemment pour obtenir la solution optimale.

Deuxième cas Une ou plusieurs variables artificielles (nulles) font partie de cette solution réalisable de base. L'algorithme du simplexe peut également être utilisé mais il faut s'assurer que ces variables artificielles ne deviennent jamais positives. Pour éviter ce problème, le critère de sortie de la base doit être modifié en conséquence. Après avoir déterminé le vecteur a_k à faire entrer dans la base, il faut examiner les valeurs y_{ik} qui correspondent aux vecteurs artificiels.

Si $y_{ik} \leq 0$pour tous les indices i correspondant aux vecteurs artificiels et $y_{ik} > 0$ pour au moins un indice *i*, le critère usuel de sortie de la base ne sélectionnerait aucun vecteur artificiel. Ce serait donc un vrai vecteur b_r qui serait éliminé. Si $x_{Br}>0$, alors les valeurs des variables artificielles avec $y_{ik} < 0$ deviendraient positives puisque :

$$\hat{x}_{Bi} = x_{Bi} - \frac{y_{ik}}{y_{rk}} x_{Br} = -\frac{y_{ik}}{y_{rk}} x_{Br} > 0$$

Dans une situation de ce type, au lieu d'utiliser le critère usuel de sortie de la base, on fait sortir un vecteur artificiel avec un $y_{ik}<0$. Dans ce cas, les nouvelles valeurs de la solution réalisable de base restent inchangées puisque $x_{Br}=0$. La nouvelle valeur de la fonction objectif n'est pas strictement améliorée mais reste constante, $\hat{Z} = Z$, pour cette itération.

La méthode des deux phases et les étapes de chaque phase sont décrites ci-dessous.

- Phase I

Cette phase consiste à construire une fonction objectif artificielle en attribuant à chaque variable artificielle un prix de -1etun prix nul à toutes les autres variables. Il s'agit donc de maximiser la fonction objectif suivante :

$$Maximiser \quad z^a = -x_{a1} - x_{a2} - \ldots - x_{as}$$

où l'indices indique le nombre de variables artificielles qui ont été rajoutées dans les contraintes. Puisque les variables artificielles x_{ai} sont non-négatives, la fonction objectif artificielle est toujours non-positive et atteint son maximum *0* lorsque chaque variable artificielle est nulle.

Les étapes de la **phase I** sont les suivantes:

Etape 1 : Transformer au besoin le problème de programmation linéaire sous forme standard en ajoutant les variables d'écart et les variables artificielles nécessaires.

Etape 2 : Construire la fonction objectif artificielle z^a en changeant les coefficients de la fonction objectif originale de la manière suivante :

(a) les coefficients des variables artificielles valent -1
(b) les coefficients des autres variables sont nuls.

Etape 3 : Résoudre le problème établi dans les deux étapes précédentes avec la méthode usuelle du simplexe. On s'arrête à **la phase I** dans deux cas :

(a) La valeur de z^a vaut 0 (même s'il reste certains $(z_j\text{-}c_j)$ négatifs).On passe alors à **l'étape 1** de **la phase II**.
(b) Le critère d'optimalité $z_j - c_j \geq 0, \forall j \in J$, est satisfait mais il reste dans la base des variables artificielles avec des valeurs positives (ainsi $z^a<0$). Dans ce cas, le problème original n'a pas de solution réalisable et l'on s'arrête.

Voyons à présent les étapes de la **phase II** dont le but est de trouver une solution optimale au problème original.

- **Phase II**

Etape 1 : Remplacer la fonction objectif artificielle par la fonction objectif originale, y compris les variables d'écart en donnant leur prix réel aux vraies variables et un prix zéro à toute variable artificielle qui peut apparaître dans la base à un niveau zéro. Les colonnes des vecteurs artificiels qui ne sont pas dans la base peuvent être éliminées du tableau car elles ne seront plus candidates pour y entrer.

Etape 2 : Le premier tableau de la phase II et le dernier tableau de la phase I sont identiques à l'exception des coefficients de la fonction objectif et des valeurs $(z_j\text{-}c_j)$ qui doivent être recalculées.

Etape 3 : S'il n'y a plus de variables artificielles dans la base à la fin de la phase I, on applique la méthode usuelle du simplexe. Sinon, on passe à l'étape 4.

Etape 4 : Pour éviter que les variables artificielles de la base ne deviennent positives, il faut examiner les valeurs y_{ik} (la colonne correspondant au vecteur a_k qui entre dans la base) pour chaque variable artificielle. Si ces valeurs sont telles que $y_{ik} \leq 0$ pour tous les indices *i* correspondant aux vecteurs artificiels et $y_{ik}>0$ pour au moins un indice *i*, on fait sortir de la base un vecteur artificiel avec un $y_{ik}<0$. Sinon on utilise le critère usuel de sortie.

Les exemples suivants illustrent respectivement les trois cas qui peuvent se présenter à la fin de la phase I, c'est-à-dire :

1. $z^a=0$ et il ne reste plus de vecteurs artificiels dans la base.
2. $z^a=0$ et il reste un ou plusieurs vecteurs artificiels dans la base.

3. $z^a<0$; dans ce cas, il n'existe pas de solution réalisable.

Exemple : Soit le problème suivant à résoudre avec la méthode des deux phases :

$$\begin{gathered} Minimiser \quad z = -2x_1 + 3x_2 - 5x_3 \\ sous\ contraintes \quad x_1 + x_2 + x_3 = 6 \\ -x_1 + x_2 + 2x_3 = 4 \\ 2x_2 + 3x_3 = 10 \\ x_3 \leq 2 \\ et\ x_1 + x_2 + x_3 = 6 \end{gathered}$$

• Phase I

Etape 1 : Comme il s'agit d'un problème de minimisation, il faut le transformer en un problème de maximisation en inversant le signe des coefficients de la fonction objectif. Cette opération n'intervient que dans la phase II. En revanche, il faut introduire une variable d'écart dans la quatrième contrainte et ajouter trois variables artificielles pour qu'une matrice identité apparaisse dans le tableau 1 de la phase I. Les contraintes s'écrivent alors :

$$\begin{gathered} x_1 + x_2 + x_3 + x_{a1} = 6 \\ -x_1 + x_2 + 2x_3 + x_{a2} = 4 \\ 2x_2 + 3x_3 + x_{a3} = 10 \\ x3 + x4 = 2 \\ et\ x_1, x_2, x_3, x_4, x_{a1}, x_{a2}, x_{a3} \geq 0 \end{gathered}$$

Etape 2 : La fonction objectif artificielle s'écrit :

$$Maximiser\ z^a = -x_{a1} - x_{a2} - x_{a3}$$

Etape 3 : Nous construisons le tableau 1 de la phase I et appliquons l'algorithme usuel du simplexe.

	c_j		0	0	0	0	−1	−1	−1
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4	q_1	q_2	q_3
−1	q_1	6	1	1	1	0	1	0	0
−1	q_2	4	−1	1	2	0	0	1	0
−1	q_3	10	0	2	3	0	0	0	1
0	a_4	2	0	0	1	1	0	0	0
	$(z_j - c_j)$	−20	0	−4	−6	0	0	0	0

Tableau 1

Le critère d'entrée indique que le vecteur a_3 entre dans la base. Le critère de sortie indique deux possibilités pour sortir un vecteur de la base : q_2 ou a_4 puisqu'ils ont tous deux le plus petit rapport (*4/2 = 2/1 = 2*). Le but de la phase I étant de faire sortir les vecteurs artificiels de la base, notre choix portera sur q_2. Nous construisons le tableau 2 de la phase I.

	c_j		0	0	0	0	−1	−1	−1
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4	q_1	q_2	q_3
−1	q_1	4	3/2	1/2	0	0	1	−1/2	0
0	a_3	2	−1/2	1/2	1	0	0	1/2	0
−1	q_3	4	3/2	1/2	0	0	0	−3/2	1
0	a_4	0	1/2	−1/2	0	1	0	−1/2	0
	$(z_j - c_j)$	−8	−3	−1	0	0	0	3	0

Tableau 2

Le tableau 2 de la phase I indique que $z^a = -8$. On peut faire entrer a_1 dans la base. Le vecteur a_4 sort de la base puisqu'il correspond au plus petit rapport $\frac{x_{Bi}}{y_{i1}}, y_{i1} > 0$, c'est-à-dire $\frac{0}{1/2} = 0$.

À noter que puisque $x_{B4} = x_4 = 0$, la valeur de la fonction objectif dans le tableau suivant reste constante, comme l'indique le tableau 3 de la phase I.

	c_j		0	0	0	0	−1	−1	−1
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4	q_1	q_2	q_3
−1	q_1	4	0	2	0	−3	1	1	0
0	a_3	2	0	0	1	1	0	0	0
−1	q_3	4	0	2	0	−3	0	0	1
0	a_1	0	1	−1	0	2	0	−1	0
	$(z_j - c_j)$	−8	0	−4	0	6	0	0	0

Tableau 3

Dans le tableau 3 de la phase I, nous avons $z^a = -8$. Le vecteur qui entre dans la base est le vecteur a_2. Il y a deux vecteurs candidats pour sortir de la base : q_1 et q_3. Nous choisissons arbitrairement de sortir q_1 de la base et calculons un nouveau tableau.

	c_j		0	0	0	0	−1	−1	−1
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4	q_1	q_2	q_3
0	a_2	2	0	1	0	−3/2	1/2	1/2	0
0	a_3	2	0	0	1	1	0	0	0
−1	q_3	0	0	0	0	0	−1	−1	1
0	a_1	2	1	0	0	1/2	1/2	−1/2	0
	$(z_j - c_j)$	0	0	0	0	0	2	2	0

Tableau 4

Puisque dans le tableau 4 de la phase I.$z^a=0$, la phase I est terminée. Notons qu'il reste un vecteur artificiel (q_3) dans la base à un niveau 0. Il faudra donc s'assurer dans la phase II que la variable correspondante ne devienne pas positive. Nous pouvons passer à l'étape 1 de la phase II.

- **Phase II**

Etape 1 : Puisqu'il s'agit d'un problème de minimisation, il faut le transformer en un problème de maximisation. La fonction objectif à maximiser est donc :

$$Maximiser\ (-z) = 2x_1 - 3x_2 + 5x_3$$

Les coefficients de la variable d'écart x_4 et de la variable artificielle x_{a3} dans la base sont nuls. De plus, nous pouvons éliminer du tableau les vecteurs artificiels q_1 et q_2 qui ne sont plus dans la base.

Etape 2 : Établissons le tableau 1 de la phase II.

	c_j		2	-3	5	0	0
c_B	vecteurs de base	x_B	a_1	a_2	a_3	a_4	q_3
-3	a_2	2	0	1	0	$-3/2$	0
5	a_3	2	0	0	1	1	0
0	q_3	0	0	0	0	0	1
2	a_1	2	1	0	0	$1/2$	0
	$(z_j - c_j)$	8	0	0	0	$21/2$	0

Tableau 1

Le tableau 1 de la phase II fournit la solution optimale puisque la seule valeur de (z_j-c_j) pour un vecteur hors base est strictement positive. À noter que dans ce cas, à la fin de la phase II, il reste un vecteur artificiel dans la base à un niveau zéro.

La solution optimale est donnée par $x_1=2$, $x_2=2$ et $x_3=2$. La valeur de la fonction objectif vaut $z=-8$ (car il s'agit d'une minimisation).

Chapitre 2 :

Optimisation non-linéaire sans contraintes

II.1 Introduction

Nous allons étudier le problème d'optimisation sans contraintes ou on effectue la minimisation de la fonction $f: \mathbb{R}^n \to \mathbb{R}$ sur tout l'espace. Nous considérons donc le problème formulé de la façon suivante :

$$\min_{x \in \mathbb{R}^n} f(x)$$

Ou *f* est une fonction de $\mathbb{R}^n$ $vers$ $\mathbb{R}$ $+\infty$

II.2 Le Gradient

La dérivée partielle au premier ordre de la fonction *f* en *a* dans la direction ***u*** :

$$\lim_{t \to 0} \frac{f(a + tu) - f(a)}{t}$$

Dans le cas particulier ou *u* décrit une base $e_1, \dots, e_n$ de $\mathbb{R}^n$, on note

$$\frac{\partial f}{\partial x_1}(a), \dots, \frac{\partial f}{\partial x_j}(a), \dots, \frac{\partial f}{\partial x_n}(a)$$

Les dérivées partielles dans ces directions.

Définition 1: Soit $f: \mathbb{R}^n \to \mathbb{R}$ admettant des dérivées partielles d'ordre 1 en a ; on appelle gradient de *f* en *a* le vecteur $\nabla f(a) = \frac{\partial f}{\partial x_1}(a), \dots, \frac{\partial f}{\partial x_j}(a), \dots, \frac{\partial f}{\partial x_n}(a)^T$

Lorsque $f = f_1, \dots, f_1 : \mathbb{R}^n \to \mathbb{R}^p$, le gradient est une matrice $n \times p$dont les colonnes sont les vecteurs $\nabla f_j(a)$

Fonction de classe C^1

Définition 2: *f*est continument différentiable si et seulement si toutes ces dérivées partielles d'ordre $1\frac{\partial f}{\partial x_i}$ existent et sont continues.

II.3 Le Hessien

Si les dérivées partielles de *f* possèdent à leur tour des dérivées partielles : on dit que *f* possède des dérivées d'ordre 2. La dérivée partielle dans la direction e_i de la dérivée partielle $\frac{\partial f}{\partial x_i}$ est notée : $\frac{\partial^2}{\partial x_i \partial x_j} f(a)$

Définition1 : Soit $f: \mathbb{R}^n \to \mathbb{R}$ admettant des dérivées partielles d'ordre 2 en a ; on appelle gradient de *f* en *a* on appelle Hessien de *f* en a la matrice $n \times n$ donnée par

$$\square^2 f\ a = \begin{matrix} \frac{\partial^2 f}{\partial x_1^2}(a) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(a) \\ \dots & \frac{\partial^2 f}{\partial x_i \partial x_j}(a) & \dots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(a) & \dots & \frac{\partial^2 f}{\partial x_n^2}(a) \end{matrix} = \square\ \square f^{\,T}(a)$$

Exemple II.1: soit $f\ x_1, x_2, x_3 = e^{x_1} + x_1^2 x_3 - x_1 x_2 x_3$ L'hessienne de *f* est donné par

$$H\ x = \begin{matrix} e^{x_1} + 2x_3 & -x_3 & 2x_1 - x_2 \\ -x_3 & 0 & -x_1 \\ 2x_1 - x_2 & -x_1 & 0 \end{matrix}$$

Fonction de classe C^2

Définition2 :*f* est deux fois différentiable si et si seulement toutes ces dérivées partielles d'ordre deux $\frac{\partial^2 f}{\partial x_i \partial x_j}$ existent et sont continues.

Hessien de fonctions C^2

Proposition : Lorsque *f* est C^2 , $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$: le Hessien est une matrice symétrique

Toute matrice symétrique réelle *H* est diagonalisable dans le groupe orthogonal : il existe Δ diagonal, *U* orthogonale telle que

$$H = U\Delta U^T; \quad \Delta \leftrightarrow valeurs\ propres\ de\ H$$

- $\square^2 f\ a$ est positive si et seulementsi $\Delta_{ii} \geq 0$
$$\Leftrightarrow \square^T \square^2 f\ a\ \square \geq 0, \quad \square \square \square^n$$
- $\square^2 f\ a$ est définie positive si et seulement si $\Delta_{ii} \geq 0$

$$\Leftrightarrow \square^T \square^2 f\ a\ \square \geq 0, \quad \square \square \square 0_{\square^n}$$

Définition3 : (minimum global). Soit $f: \square^n \rightarrow \square$ une fonction. $f(x^\square)$est le minimumglobal de *f* si et seulement si :$\square x\ \square\ \square, f(x) \geq f(x^\square)$
$x^\square$est un minimiseur global de *f*.

Définition 4 : (minimum local). Soit $f: \square^n \rightarrow \square$ une fonction. $f(x^\square)$est le minimumlocal de *f* si et seulement si : $\square\varepsilon, \square x\ \square]x^\square - \varepsilon; x^\square + \varepsilon[,\ f(x) \geq f(x^\square)$
$x^\square$est un minimiseur local de *f*.

Définition 5 : x_0 est un point stationnaire si et seulement si $\square f\ x_0 = 0$.

Dans les points stationnaires, il y a les minima et les maxima (locaux et globaux) et il y a aussi d'autres points.

Définition 6 **:** Un point stationnaire qui n'est ni un minimum ni un maximum est un point singulier.

II.4 Condition nécessaire pour l'existence d'un minimum local

Définition 1 : *a* est un minimum local de *f* si et seulement si il existe au voisinage v_a de a tel que pour tout $x \in v_a, f(a) \leq f(x)$

Théorème : Soit $f: \mathbb{R}^n \to \mathbb{R}$ de classe C^2 si *f* possède un minimum local en *a* alors :

1- $\nabla f\ a\ = 0_{\mathbb{R}^n}$
2- $\nabla^2 f\ a\ est\ une\ matrice\ positive.$

II.5 Conditions suffisantes pour l'existence d'un minimum local

Théorème : Soit $f: \mathbb{R}^n \to \mathbb{R}$ de classe C^2.Si

1- $\nabla f\ a\ = 0_{\mathbb{R}^n}$
2- $\nabla^2 f\ a\ est\ une\ matrice\ définie\ positive.$ alors *f* possède un minimum local en *a*.

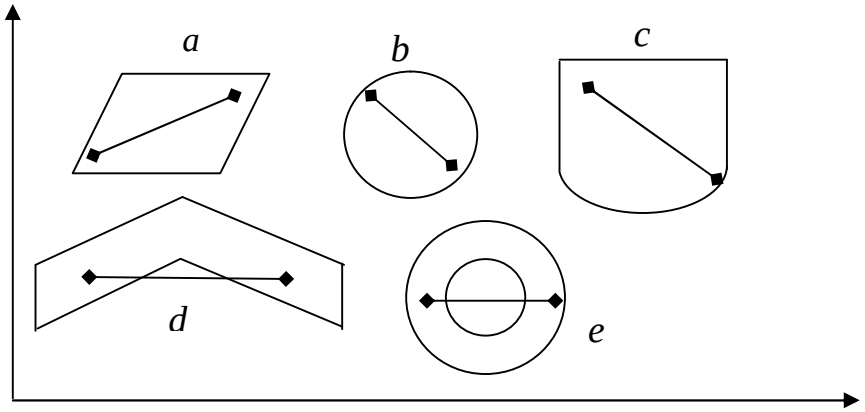
II.6 Fonctions convexes

II.6.1 Fonctions convexes

La convexité joue un rôle extrêmement important en optimisation puisqu'elle permet d'étudier en général l'unicité des solutions d'un problème d'optimisation.

Définition :

1- Un ensemble *D* est dit convexe si, pour tous point *x* et *y* de *D*, le segment *[a,b]* est inclus dans *D*, i.e. Quel que soit tout $t \in [0,1]$, le point $ta + \ 1 - t\ b$ appartient a *D*.
exemple : les ensembles a , b , c sont des ensembles convexes ; d , e sont des ensembles non convexes. Un ensemble convexe ne peut avoir des parties entrantes comme d ou de trous comme e .



2- $f: \mathbb{R}^n \to \mathbb{R}$ définie sur un ensemble convexe *D* est dite convexe si pour tout points $a, b \in \mathbb{R}^n f$ et tout $t \in [0,1]$

$$f\ ta + \ 1 - t\ b\ \leq tf\ a\ + \ 1 - t\ f\ b\ .$$

La fonction est dite strictement convexe si

$$\forall\ a, b\ \in D^2, a \neq b, \forall t \in]0,1[, f\ ta + \ 1 - t\ b\ \leq tf\ a\ + \ 1 - t\ f\ b\ .$$

Théorème

$f: \mathbb{R}^n \to \mathbb{R}$ est de classe C^2, il est équivalent :

1- *f* est convexe.
2- Pour tout $x, a \in \mathbb{R}^n, f(x) \geq f(a) + (x-a)^T \nabla f(x)$
3- Pour tout $x \in \mathbb{R}^n, \nabla^2 f(x)$ *est positive.*

II.6.2 Minimum de fonctions convexes

Théorème : Soit $f: \mathbb{R}^n \to \mathbb{R}$ une fonction convexe possédant des dérivées partielles. *f* admet un minimum globale en *a* si et seulement si $\nabla f(a) = 0_{\mathbb{R}^n}$.

Corollaire : Si *f* est convexe, tout minimum local est un minimum global.

II.7 Méthodes de recherche unidimensionnelle

Soit *f* une application de $\mathbb{R}$ dans $\mathbb{R}$. Les méthodes d'optimisation de telles fonctions ont d'autant plus d'importance qu'elles servent d'outils pour l'optimisation de fonctions de plusieurs variables.

II.7.1 Méthodes du premier ordre (dichotomie)

L'idée la plus simple est de trouver un zéro de la dérivée de la fonction objectif a l'aide d'une recherche dichotomique.

Définition : On dit qu'une fonction est unimodale s'il existe un réel x^* pour lequel lafonction est strictement décroissante sur $]-\infty, x^*]$ et strictement croissante sur $[x^*, +\infty[$.
Le point x^* est alors minimum global de *f*.

Algorithme 1: Algorithme de recherche dichotomique
Données *: f une fonction unimodale de classe* C^1 *sur [a ;b] telle que* $f'(a) < 0 < f'(b)$
1- Initialisation
choix [a ;b] telle que $f'(a) < 0 < f'(b)$
2- Critère d'arrêt
Si $\|b-a\| < \epsilon, STOP$
3- Si non, On pose $m = \frac{a+b}{2}$
4- Si $f'(m) \geq 0$ *alors* $b = m$
Sinon $a = m$
On retourne à 2

II.7.2Méthodes du second ordre (Méthode de Newton)

A partir d'un point x_k, on approche *f* par :

$$g(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2$$

On remarque que $g'(x) = f'(x_k) + f''(x_k)(x - x_k)$
Si $f''(x_k) > 0$ (cas ou f est convexe autour de x_k) , on pose

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

qui est le point où g atteint son minimum$g'(x_{k+1}) = 0$si $f''(x_k) \leq 0$, la méthode échoue

Algorithme 2: Algorithme de Newton unidimensionnel
Données *: f une fonction de classe* C^2 *et* x_0 *un point dans le voisinage d'un minimiseur de f*
1- Initialisation

k=0 : choix de x_0 □ □ dans un voisinage de $x^{□}$.

2- Itération k

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

3- Critère d'arrêt

Si $\|x_{k+1} - x_k\| < \epsilon, STOP$

Si non, On pose k=k+1 et on retourne à 2.

D'un point de vue pratique, cette méthode souffre de nombreux inconvénients :

- la méthode peut diverger si le point de départ est trop éloigné de la solution.
- la méthode n'est pas définie si $f''(x_k) = 0$.
- la méthode peut converger indifféremment vers un minimum, un maximum ou un point de selle.

Cette méthode est donc peu utilisée. Son intérêt majeur est d'illustrer dans □ lefonctionnement des algorithmes de minimisation multidimensionnelle du second ordre

II.8 Méthodes du gradient

Il s'agit d'une famille de méthodes itératives qui s'appliquent à des fonctions dérivables et qui utilisent l'idée ci-dessous.

On veut minimiser une fonction *f*. Pour cela on se donne un point de départ arbitraire x_0. Pour construire l'itéré suivant $x_1$il faut penser qu'on veut se rapprocher du minimum de *f*; on veut donc que $(x_1) < f(x_0)$. On cherche alors x_1 sous la forme $x_1 = x_0 + \rho_1 d_1$ où d_1 est un vecteur non nul de $□^n$ et $\rho_1$est un réel strictement positif. En pratique donc, on cherche d_1 et ρ_1 pour que $f(x_0 + \rho_1 d_1) < f(x_0)$. On ne peut pas toujours trouver d_1. Quand d_1 existe on dit que c'est une **direction de descente** et ρ_1 est **le pas de descente**. La direction et le pas de descente peuvent être fixes ou changer à chaque itération. Le schéma général d'une méthode de descente est le suivant :

$$x_0 □ □^n \quad donnée$$
$$x_{k+1} = x_k + \rho_k d_k \quad , d_k □ □^n - 0 \, , \rho_k □ □^{+□}$$

Ou ρ_k et d_k sont choisis de telle sorte que $f(x_k + \rho_k d_k) < f(x_k)$

De tels algorithmes sont souvent appelés méthodes de descente. Essentiellement, la différence entre ces algorithmes réside dans le choix de la direction de descente d_k.

Une idée naturelle pour trouver une direction de descente est de faire un développement de Taylor à l'ordre 2 de la fonction *f* entre deux itérations x_k et $x_{k+1} = x_k + \rho_k d_k$

$$f(x_k + \rho_k d_k) = f(x_k) + \rho_k □f(x_k, d_k) + O(\rho_k d_k)$$

Comme on veut $f(x_k + \rho_k d_k) < f(x_k)$, on peut choisir en première approximation $d_k = -□f(x_k)$. La méthode ainsi obtenue s'appelle l'algorithme du Gradient. Le pas ρ_k est choisi constant ou variable.

Alghorithme 3 :Algorithme du Gradient

1- Initialisation

k=0 choix de x_0 et de $\rho_0 > 0$

2- Itération k

$$x_{k+1} = x_k + \rho_k \nabla f(x_k)$$

3- Critère d'arrêt

Si $\|x_{k+1} - x_k\| < \epsilon, STOP$
Si non, On pose k=k+1 et on retourne à 2.

ϵ est un réel positif (petit) donné qui représente la précision désirée.
Cette méthode a pour avantage d'être très facile à mettre en œuvre. Malheureusement, les conditions de convergence sont assez lourdes (c'est essentiellement de la stricte convexité) et la méthode est en général assez lente.

Méthode du gradient à pas constant

On utilise le plus souvent la méthode du gradient à pas constant (ρ_k = constant).Toutefois, on peut faire varier le pas à chaque itération : on obtient alors la méthode dugradient à pas variable.

Exemple II.2

Soit la fonction $f\ x\ = x_1^2 + \frac{1}{2}x_2^2 - 3(x_1 + x_2)$ trouver le minimum de cette fonction en partant du point de cordonnée $x_0 = (-2,1.5)$; critère d'arrêt est $\|x_{k+1} - x_k\| < 10^{-3}$

Solution

1- $x_0 = (-2,1.5) \rho = 0.45$

2- Calcul de la dérivée par rapport à x_1

$$\frac{\partial f}{x_1} = 2x_1 - 3$$

- Calcul de la dérivée par rapport à x_2

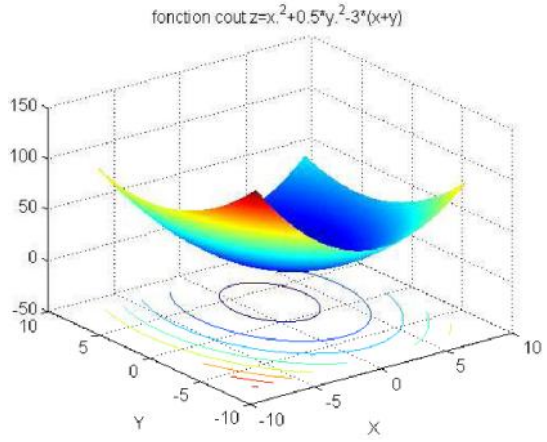
$$\frac{\partial f}{x_1} = x_2 - 3$$

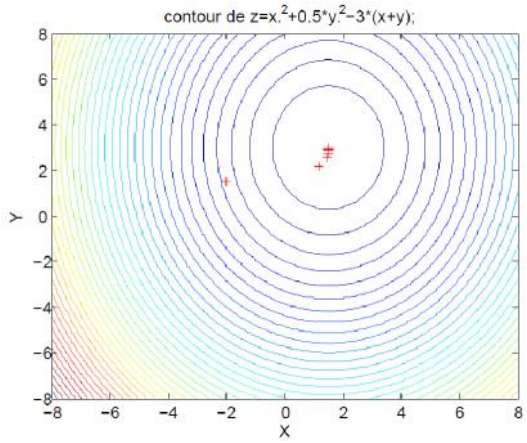
- 1ère itération

$$x_{1\,1} = x_{1\,0} + \rho\ \nabla f(x_1)$$
$$x_{2\,1} = x_{2\,0} + \rho\ \nabla f(x_1)$$

3- Test d'arrêt *Si* $x_{1\,1} - x_{1\,0} < 10^{-3}, STOP$
$x_{2\,1} - x_{2\,0} < 10^{-3}, STOP$

Si non on pose k=2 et on retourne à 2





Les itérations

Itérations	x_1	x_2	$f(x)$
1	-2	1,5000	6,6250
2	1,1500	2,1750	6,6250

3	1,4650	2,5462	-6,2872
4	1,4965	2,7504	-6,6458
5	1,4996	2,8627	6,7188
6	1,4999	2,9245	-6,7406
7	1,4999	2,9585	-6,7471

Méthode du gradient à pas optimal

La méthode du gradient à pas optimal propose un choix du pas qui rend la fonction coût minimale le long de la direction de descente choisie. Plus précisément, l'étape 2devient :

2'- Itération k

$$x_{k+1} = x_k + \rho_k \nabla f(x_k)$$

Où ρ_k réalise le minimum sur $\mathbb{R}^+$ de la fonction $\varphi_k$définie par

$$\varphi_k(\rho) = f\ x_k - \rho_k \nabla f(x_k)$$

En pratique, on ne cherche pas le minimum de φ_k et on détermine ρ_k en effectuant une recherche linéaire de pas optimal suivant une règle de la forme suivante par exemple :

Règle de recherche linéaire de Wolfe

1- Initialisation$\rho = 1$ par exemple, $\rho_- = \rho_+ = 0$. On se donne $0 < \beta_1 < \beta_2 < 1$.

2- Si $\varphi_k\ \rho \leq \varphi_k\ 0 + \beta_1 \rho \varphi_k'(0)$ et $\varphi_k\ \rho \geq \beta_2 \varphi_k'(0)$, STOP : $\rho_k = \rho$.

3- Sinon

- Si $\varphi_k\ \rho > \varphi_k\ 0 + \beta_1 \rho \varphi_k'(0)$, on pose $\rho_+ = \rho$
- Si $\varphi_k\ \rho \leq \varphi_k\ 0 + \beta_1 \rho \varphi_k'(0)$, et $\varphi_k\ \rho < \beta_2 \varphi_k'\ 0$, on pose $\rho_- = \rho$ et on va a 4.

4- Choix du nouveau ρ

- Si $\rho_+ = 0$, on cherche $\rho > \rho_-$ par exemple $\rho = 2\rho_-$
- Si $\rho_+ > 0$, on cherche $\rho \in]\rho_-, \rho_+[$ par exemple $\rho = \frac{\rho_- + \rho_+}{2}$

Retour à 2.

La règle apparaissant à l'étape 2. est connue sous le nom générique de règle d'**Armijo**.

Il existe beaucoup d'autres règles de recherche linéaire.

$$\varphi_k(\rho) = f\ x_k - \rho_k \nabla f(x_k)$$

Cette fonction est dérivable de dérivée :

$$\varphi'\ \rho = -\sum_{j=1}^{n} \frac{\partial f}{\partial x_j}\ x - \rho \nabla f\ x\ \ \nabla f_j\ x = -g^T\ x - \rho g\ x\ \ - g\ x$$

Avec $g\ x = \nabla f(x)$

Notons en particulier que :

$$\varphi'\ 0 == -g^T\ x\ .g\ x = -\|g\ x\|^2$$

Exemple II.3 : Même exemple II.2

Solution

Etape 1

$\rho = 1$ par exemple, On se donne $0 < \beta_1 = 0.1 < \beta_2 = 0.5 < 1$

$$f\ x_0 = 6.6250$$

On calcul le gradient de $f\nabla f\ x_0 = \begin{matrix} \frac{\partial f}{x_1} = 2x_1 - 3 = -7 \\ \frac{\partial f}{x_2} = x_2 - 3 = -1.5 \end{matrix}$

La direction $d_k = -\nabla f\ x_0 = \begin{matrix} 7 \\ 1.5 \end{matrix}$

$$\varphi_k\ \rho = f\ x_k - \rho_k \nabla f\ x_k = 5.5$$

$$= (-2 - 1 \times (-7))^2 + 0.5 \times (1.5 - 1 \times (-1.5))^2 - 3 \times -2 - 1 \times (-7) + (1.5 - 1 \times (-1.5))$$

$$= 5.5$$

$$\varphi'_k\ 0 = -\|g\ x\ \|^2 = 7.17$$

$$\beta_1 \rho \varphi'_k\ 0 = 0.1 \times 1 \times 7.17 = 0.717$$

$$\beta_2 \varphi'_k\ 0 = 0.5 \times 1 \times 7.17 = 3.585$$

Etape 2

On fait le test **Si** $\varphi_k\ 1 \leq \varphi_k\ 0 + \beta_1 \rho \varphi'_k\ 0$ et $\varphi_k\ 1 \geq \beta_2 \varphi'_k(0)$, STOP : $\rho_k = 1$.

II.9 Méthodes des directions conjuguées

Les méthodes du gradient conjugué sont utilisées pour résoudre les problèmes d'optimisation non linéaires sans contraintes spécialement les problèmes de grandes tailles. On l'utilise aussi pour résoudre les grands systèmes linéaires.

Elles reposent sur le concept des directions conjuguées parce que les gradients successifs sont orthogonaux entre eux et aux directions précédentes.

L'idée initiale était de trouver une suite de directions de descente permettant de résoudre le problème. Soit f une fonction quadratique :

$$\min_{x \square\square^n} f(x)$$

Où *f* est régulière (continûment différentiable)

Avant d'accéder à la méthode, on va d'abord donner le principe général d'une méthode à directions conjuguées.

Définition 9.1 Soit A une matrice symétrique *n x n*, définie positive. On dit que deux vecteurs *x* et *y* de $\square^n$ sont A-conjugués (ou conjugués par rapport à A) s'ils vérifient :

$$x^T A y = 0$$

Soit $d_0, d_1, \dots, d_n$ une famille de vecteurs A-conjugués. On appelle alors méthode dedirections conjuguées toute méthode itérative appliquée à une fonction quadratique strictement convexe de *n* variables : $q\ x = \frac{1}{2} x^T A x + b^T x + c$ avec $x \square \square^n$ et A une matrice symétrique et définie positive de *n x n*éléments. $b \square \square^n$et$c \square \square$ conduisant à l'optimum en *n* étapes au plus. Cette méthode est de la forme suivante :

$$x_0\ donnée$$

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

Où α_k est optimal et $d_0, d_1, \dots, d_n$ possédant la propriété d'être mutuellement conjuguées par rapport à la fonction quadratique. Si l'on note$g_k = \square q(x_k)$, la méthode se construit comme suit :

Calcul de α_kComme α_k minimise q dans la direction d_k; on a, $\square k$:

$$q'\ \alpha_k\ =\ d_k^T \square q\ x_{k+1}\ =\ 0$$
$$d_k^T \square q\ x_{k+1}\ =\ d_k^T\ Ax_{k+1} + b\ =\ 0$$

Soit

$$d_k^T A\ x_k + \alpha_k d_k\ +\ d_k^T b = 0$$

D'où l'on tire : $$\alpha_k = \frac{-d_k^T(Ax_k+b)}{d_k^T A d_k}$$

Comment construire les directions A-conjuguées ?

Des directions A-conjuguées $d_0, d_1, \dots, d_n$ peuvent être générées à partir d'un ensemble de vecteurs linéairement indépendants $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_k$en utilisant la procédure dite de GramSchmidt, de telle sorte que pour tout *i* entre *0* et *k*, le sous-espace généré par $d_0, d_1, \dots, d_n$ soit
égale au sous-espace généré par $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_i$.

Alors d_{i+1} est construite comme suit :

$$d_{i+1} = \varepsilon_{i+1} + \sum_{m=0}^{i} \varphi_{i+1\ m} d_m$$

Nous pouvons noter que si d_{i+1} est construite d'une telle manière, elle est effectivement linéairement indépendante avec $d_0, d_1, \dots, d_i$.

En effet, le sous-espace généré par les directions $d_0, d_1, \dots, d_i$ est le même que le sous-espace généré par les directions $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_i$, et ε_{i+1} est linéairement indépendant de $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_i$:

ε_{i+1} ne fait donc pas partie du sous-espace généré par les combinaisons linéaires de laforme$\sum_{m=0}^{i} \varphi_{i+1\ m} d_m$, de sorte que d_{i+1} n'en fait pas partie non plus et est donc linéairement indépendante des $d_0, d_1, \dots, d_i$.

Les coefficients $\varphi_{i+1\ m}$, eux sont choisis de manière à assurer la A-conjugaison des $d_0, d_1, \dots, d_{i+1}$.

Méthode de gradient conjugué dans le cas quadratique

La méthode du gradient conjugué quadratique est obtenue en appliquant la procédure de Gram-Schmidt aux gradients $\square q\ x_0\ , \dots, \square q(x_{n-1})$c'est-à-dire en posant$\varepsilon_0 = \square q\ x_0\ , \dots, \varepsilon_{n-1} = \square q(x_{n-1})$

En outre, nous avons :

$$\square q\ x\ =\ Ax + b$$

Et $$\square^2 q\ x\ =\ A$$

Notons que la méthode se termine si $\square q\ x\ = 0$.

La particularité intéressante de la méthode du gradient conjugué est que le membre de droite de l'équation donnant la valeur de d_{k+1} dans la procédure de Gram-Schmidt peut être grandement simplifié.

Algorithme de La méthode du gradient conjugué pour les fonctions quadratiques

On suppose ici que la fonction à minimiser est quadratique sous la forme :

$$q\ x\ =\ \frac{1}{2} x^T Ax + b^T x + c$$

Si l'on note$g_k = \square f(x_k)$, l'algorithme prend la forme suivante.

Cet algorithme consiste à générer une suite d'itérés x_k sous la forme :

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

L'idée de la méthode est :

1- construire itérativement des directions $d_0, d_1, \dots, d_k$ mutuellementconjuguées :

A chaque étape *k* la direction d_k est obtenue comme combinaison linéaire du gradient

En x_k et de la direction précédente d_{k-1} c'est-à-dire

$$d_{k+1} = -\nabla q\ x_{k+1} + \beta_{k+1} d_k$$

les coefficients β_{k+1}étant choisis de telle manière que d_k soit conjuguée avec toutes les directions précédentes. Autrement dit :

$$d_{k+1}^T A d_k = 0$$

$$d_{k+1}^T A d_k = 0 \Leftrightarrow -\nabla q\ x_{k+1} + \beta_{k+1} d_k\ ^T A d_k = 0$$

$$\Leftrightarrow -\nabla^T q\ x_{k+1}\ A d_k + \beta_{k+1} d_k^T A d_k = 0$$

$$\Leftrightarrow \beta_{k+1} = \frac{\nabla^T q\ x_{k+1}\ A d_k}{d_k^T A d_k} = \frac{g_{k+1}^T A d_k}{d_k^T A d_k}$$

2-déterminer le pas α_k :

En particulier, une façon de choisir α_k consiste à résoudre le problème d'optimisation unidimensionnelle suivant :

$$\alpha_k = min f\ x_k + \alpha d_k\ ,\quad \alpha > 0$$

On en deduit $\alpha_k = \frac{-d_k^T g_k}{d_k^T A d_k}$

Le pas α_k obtenu ainsi s'appelle le pas optimal.

Algorithme 3.1 Algorithme du gradient conjugué "quadratique"

Etape 0 : (initialisation)

Soit x_0 le point de départ, $g_0 = \nabla q\ x_0 = Ax_0 + b$, poser $d_0 = -g_0$;

Poser *k = 0* et aller à l'étape 1:

Etape 1 :

si $g_k = 0$: STOP ($x^* = x_k$)."Test d'arrêt"

si non aller à l'étape 2.

Etape 2 :

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

$$\alpha_k = \frac{-d_k^T g_k}{d_k^T A d_k}$$

$$d_{k+1} = -\nabla q\ x_{k+1} + \beta_{k+1} d_k$$

$$\beta_{k+1} = \frac{g_{k+1}^T A d_k}{d_k^T A d_k}$$

Poser *k=k+1* et aller a l'étape 1.

Exemple II.4 : Même exemple II.2

Solution

Etape 0 : (initialisation)

Soit x_0(-2.1.5) le point de départ, $g_0 = \nabla f\ x_0 = \begin{matrix} \frac{\partial f}{x_1} = 2x_1 - 3 = -7 \\ \frac{\partial f}{x_2} = x_2 - 3 = -1.5 \end{matrix}$

Sous la forme suivante$q\ x_0 = Ax_0 + b$, Matrice $A = \begin{matrix} 2 & 0 \\ 0 & 1 \end{matrix}$ et $b = \begin{matrix} -3 \\ -3 \end{matrix}$ on pose$d_0 = -\nabla f\ x_0 = \begin{matrix} 7 \\ 1.5 \end{matrix}$

Poser *k = 0* et aller à l'étape 1:

Etape 1 :

si $g_0 = 0$: STOP ($x^* = x_k$)."Test d'arrêt" Test non vérifier donc aller a l'étape suivante.

Etape 2

$$x_{1_{0+1}} = x_{1_0} + \alpha_0 d_0$$
$$x_{2_{0+1}} = x_{2_0} + \alpha_0 d_0$$
$$\alpha_0 = \frac{-d_0^T g_0}{d_0^T A d_0} = \frac{-\begin{matrix} 7 & 1.5 \end{matrix} \times \begin{matrix} -7 \\ -1.5 \end{matrix}}{\begin{matrix} 7 & 1.5 \end{matrix} \times \begin{matrix} 2 & 0 \\ 0 & 1 \end{matrix} \begin{matrix} 7 \\ 1.5 \end{matrix}} = 0.5112$$
$$x_{1_1} = -2 + 0.5112 \times 7 = 1.5786$$
$$x_{2_1} = 1.5 + 0.5112 \times 1.5 = 2.2668$$

Test d'arrêt

$$g_1 = \nabla f\ x_1 = \begin{matrix} \frac{\partial f}{x_1} = 2x_{1_1} - 3 = 0.1571 \\ \frac{\partial f}{x_2} = x_{2_1} - 3 = -0.7332 \end{matrix} \neq 0$$

on pose k=k+1

$$x_{1_2} = x_{1_1} + \alpha_1 d_{1_1}$$
$$x_{2_2} = x_{2_1} + \alpha_1 d_{1_2}$$
$$d_1 = -q\ x_1 + \beta_1 d_0$$
$$\beta_1 = \frac{g_1^T A d_0}{d_0^T A d_0}$$
$$g_1 = \nabla f\ x_1 = \begin{matrix} 0.1571 \\ -0.7332 \end{matrix}$$
$$\beta_1 = \frac{0.1571 \quad -0.7332 \times \begin{matrix} 2 & 0 \\ 0 & 1 \end{matrix} \times \begin{matrix} 7 \\ 1.5 \end{matrix}}{\begin{matrix} 7 & 1.5 \end{matrix} \times \begin{matrix} 2 & 0 \\ 0 & 1 \end{matrix} \begin{matrix} 7 \\ 1.5 \end{matrix}} = 0.0110$$
$$d_{1_1} = -q\ x_1 + \beta_1 d_{0_1} = 0.1571 + 0.0110 \times 7 = -0.0803$$
$$d_{1_2} = -q\ x_1 + \beta_1 d_{0_2} = -0.7332 + 0.0110 \times 1.5 = 0.7496$$
$$\alpha_1 = \frac{-d_1^T g_1}{d_1^T A d_1} = \frac{-\begin{matrix} -0.0803 & -0.7496 \end{matrix} \times \begin{matrix} 0.1571 \\ -0.7332 \end{matrix}}{\begin{matrix} -0.0803 & 0.7496 \end{matrix} \times \begin{matrix} 2 & 0 \\ 0 & 1 \end{matrix} \begin{matrix} -0.0803 \\ 0.7496 \end{matrix}} = 0.9780$$
$$x_{1_2} = 1.5786 + 0.9780 \times (-0.0803) = 1.5000$$
$$x_{1_2} = 2.2668 + 0.9780 \times (0.7496) = 3.0000$$

Test d'arrêt $g_2 = \nabla f\ x_1 = \begin{matrix} 0.888 \times 10^{-15} \\ 0 \end{matrix} \approx 0$ arrêt.

Méthode du gradient conjugué dans le cas non quadratique

On s'intéresse dans cette section à la minimisation d'une fonction $f: \square^n \rightarrow \square$, nonnécessairement quadratique :

$$\min f\ x\ ,\ x \square \square^n$$

Les méthodes du gradient conjugué génèrent des suites $x_k\ _{k=0,1,..,n}$de la forme suivante :

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

Le pas $\alpha_k \square \square$ étant déterminé par une recherche linéaire. La direction d_k est définie par la formule de récurrence suivante ($\beta_k \square \square$)

$$d_k = \begin{matrix} -g_k & si\ \ k = 1 \\ -g_k + \beta_k d_{k-1} & si\ \ k \geq 2 \end{matrix}$$

Ces méthodes sont des extensions de la méthode du gradient conjugué linéaire du cas quadratique, si β_k prend l'une des valeurs

$$\beta_k^{PRP} = \frac{g_k^T y_k}{\|g_{k-1}\|^2}$$

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}$$

$$\beta_k^{CD} = \frac{\|g_k\|^2}{-d_{k-1}^T g_{k-1}}$$

Ou $y_{k-1} = g_k - g_{k-1}$

Algorithme 3.2 Algorithme de La méthode du gradient conjugué pour les fonctions quelconques

Etape0 : (initialisation)

Soit x0 le point de départ ,$g_0 = \square f\ x_0$, poser d_0=-g_0

Poser *k=0* et aller à l'étape 1.

Etape 1 :

si $g_k = 0$: STOP ($x^* = x_k$)."Test d'arrêt"

si non aller à l'étape 2.

Etape 2 :

Définir $x_{k+1} = x_k + \alpha_k d_k$ *avec* α_k *:calculer par la recherche linéaire*

$$d_{k+1} = -\ g_{k+1}\ + \beta_{k+1} d_k$$

Ou

$$\beta_{k+1} : d\acute{e}fini\ selon\ la\ m\acute{e}t\square ode$$

Poser *k=k+1* et aller a l'étape 1.

II.9 Méthode de Newton

La méthode de Newton pour les fonctions multivariables est identique à celle des fonctions univariées. Comme précédemment, une estimation $g\ x$ de f au point x_k est donnée par le développement de Taylor du second ordre, qui s'écrit pour une fonction multivariée :

$$g\ x\ = f\ x_k\ +\ x - x_k\ ^T \square f\ x_k\ + \frac{1}{2}(x - x_k)^T\ \square^2 f\ x_k\ \ \ (x - x_k)$$

Si la matrice Hessienne est inversible, on choisit x_{k+1} qui minimise g, soit :

$$x_{k+1} = x_k - \ \square^2 f\ x_k\ ^{-1} \square f(x_k)$$

En pratique, la direction de descente $D = -\ \square^2 f\ x_k\ ^{-1} \square f(x_k)$ est calculée sans inverser $\square^2 f\ x_k$ mais en résolvant :

$$\square^2 f\ x_k\ D = -\square f(x_k)$$

L'intérêt de cette suite est sa convergence quadratique vers un minimiseur local à la condition que x_0 soit assez proche d'un minimiseur. Néanmoins d'un point de vue pratique, cette Méthode comporte les mêmes inconvénients que dans le cas monovariable :

- ✓ la méthode peut diverger si le point de départ est trop éloigné de la solution,
- ✓ la méthode n'est pas définie si la matrice Hessienne n'est pas inversible,
- ✓ la méthode peut converger indifféremment vers un minimum, un maximum ou un point de selle.

Algorithme 4. Algorithme de la méthode de Newton Dans $\square^n$

Données : *f* une fonction différentiable et x_0 un point initial

1- Initialisation

k=0 : choix de x_0 □ □ dans un voisinage de $x^{\square}$.

2- Itération *k*

$$x_{k+1} = x_k + D$$

3- Critère d'arrêt

Si $\| x_{k+1} - x_k \| < \epsilon, STOP$

Si non, On pose *k=k+1* et on retourne à 2.

L'étape 2. de la méthode revient à résoudre le système linéaire suivant:

$$\square^2 f\ x_k\ D = -\square f(x_k)$$

Puis à poser $x_{k+1} = x_k + D$

Actuellement on développe surtout des méthodes dites quasi-Newton qui gardent la rapidité de la méthode de Newton, et sont plus robustes par rapport au du point de départ.

Exemple II.5 : Même exemple II.2

Solution

1- $x_0 = (-2, 1.5)$

2- Calcul de la dérivée par rapport à x_1

$$\frac{\partial f}{x_1} = 2x_1 - 3 = -7\ au\ point\ x_{1\,0}$$

- Calcul de la dérivée par rapport à x_2

$$\frac{\partial f}{x_1} = x_2 - 3 = -1.5\ au\ point\ x_{2\,0}$$

- Calcul de la dérivée deuxième par rapport à x_1

$$\frac{\partial^2 f}{{x_1}^2} = 2$$

- Calcul de la dérivée deuxième par rapport à x_2

$$\frac{\partial^2 f}{{x_1}^2} = 1$$

- Calcule de *D* au point x_0

$$D_{x1} = -\ \square^2 f\ x_k\ ^{-1} \square f\ x_k\ = \frac{2x_1 - 3}{2} = 3.5\ au\ point\ de\ cordonné\ x_{1\,0}, x_{2\,0}$$

$$D_{x2} = -\ \square^2 f\ x_k\ ^{-1} \square f\ x_k\ = \frac{x_2 - 3}{1} = 5\ au\ point\ de\ cordonné\ x_{1\,0}, x_{2\,0}$$

- 1ère itération

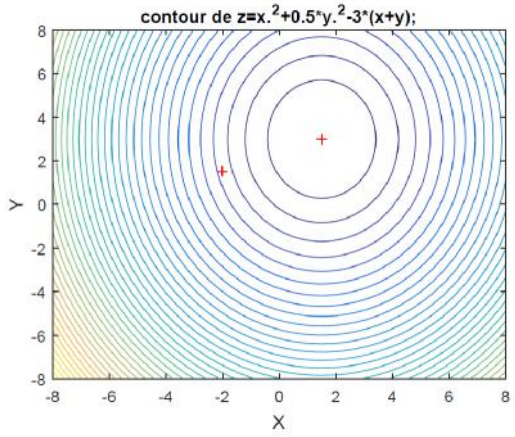
$$x_{1\,1} = x_{1\,0} + D = 1.5000$$

$$x_{2\,1} = x_{2\,0} + D = 3$$

3- Test d'arrêt *Si* $x_{1\,1} - x_{1\,0} < 10^{-3}, STOP$

$x_{2\,1} - x_{2\,0} < 10^{-3}, STOP$

Si non on pose k=2 et on retourne à 2



Les itérations

Itérations	x_1	x_2	*f(x)*
1	-2	1,50000000000000	6,62500000000000
2	1,5000000000000	3	-6.7500

II.11.Méthodes quasi-Newton

Pour des problèmes de grandes dimensions, le calcul du Hessien est trop couteux. On peut alors utiliser des algorithmes, dits quasi-Newton, qui calculent donc une approximationB_k de $\nabla^2 f(x_k)^{-1}$ en fonction de $B_{k-1}, \nabla f(x_k), \nabla f(x_{k-1}), x_k \text{ et } x_{k-1}$.
On trouve notamment deux méthodes de calcul :

- ✓ la formule de *Davidon-Fletcher-Powell* (DFP) :

$$B_k = B_{k-1} - \frac{B_{k-1} Y_k Y_k^T B_{k-1}}{Y_k^T B_{k-1} Y_k} + \frac{\overline{D}_{k-1} \overline{D}_{k-1}^T}{\overline{D}_{k-1}^T Y_k}$$

- ✓ la formule de Broyden-Fletcher-Goldfarb-Shanno (BFGS):

$$B_k = B_{k-1} + \left(1 + \frac{Y_k^T B_{k-1} Y_k}{Y_k^T \overline{D}_{k-1}}\right) \frac{\overline{D}_{k-1} \overline{D}_{k-1}^T}{Y_k^T \overline{D}_{k-1}} - \frac{B_{k-1} Y_k \overline{D}_{k-1}^T + \overline{D}_{k-1} Y_k B_{k-1}}{Y_k^T \overline{D}_{k-1}}$$

Avec

$$Y_k = \nabla f(x_k) - \nabla f(x_{k-1})$$

Ces formules donnent toujours des matrices définies positives. A l'aide de ces estimations, on établit un algorithme à convergence très efficace et particulièrement robuste.
Son unique inconvénient est la nécessité du stockage en mémoire d'une matrice de taille $n \times n$. Dans la pratique, on utilise en général la méthode BFGS qui est plus efficace.

Algorithme 5 Algorithme de quasi-Newton avec mise à jour BFGS ou DFP
Données : *f* une fonction différentiable et x_0 un point initial
Etape 1 :soit $\varepsilon > 0$ critère d'arrêt, choisir β_1 défini positive quelconque (par exemple $\beta_1 = I$)
Poser k=1 et aller à l'étape suivante
Etape 2 : Si $\| \nabla f(x_k) \| < \varepsilon$ STOP ; Si non, poser $d_k = -\beta_k \times g_k$ et déterminer le pas optimale α_k solution du problème linéaire $\alpha_k = min f(x_k + \alpha d_k), \ \alpha \geq 0$(par une méthode de recherche linéaire

par exemple dichotomie ou Wolfe) et poser

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

Etape 3 : construire β_{k+1} comme suit :

$$B_{k+1} = B_k - \frac{B_k Y_{k+1} Y_{k+1}^T B_k}{Y_{k+1}^T B_k Y_{k+1}} + \frac{\bar{D}_k \bar{D}_k^T}{\bar{D}_k^T Y_{k+1}}$$ Selon **DFP** Ou

$$B_{k+1} = B_k + \left(1 + \frac{Y_{k+1}^T B_k Y_{k+1}}{Y_{k+1}^T \bar{D}_k}\right) \frac{\bar{D}_k \bar{D}_k^T}{Y_{k+1}^T \bar{D}_k} - \frac{B_k Y_k \bar{D}_k^T + \bar{D}_k Y_{k+1} B_k}{Y_{k+1}^T \bar{D}_k}$$ Selon **BFGS**

Avec

$$Y_{k+1} = \nabla f(x_{k+1}) - \nabla f(x_k)$$
$$D_k = \alpha_k d_k$$

Remplacer *k* par *k+1* et aller à l'étape 1

Exemple II.6 : Même exemple II.2

Solution

Etape 1 :soit $\varepsilon = 0.001$ critère d'arrêt, choisir β_0 défini positive quelconque (par exemple $\beta_1 = I$)

Poser k=0 et aller à l'étape suivante

Etape 2 : Si $\|\nabla f(x_k)\| < \varepsilon$ STOP ; Test non vérifier,

poser $d_0 = -\beta_0 \times g_0 = \begin{matrix} -1 \times -7.0000 \\ -1 \times -1.5000 \end{matrix}$

et déterminer le pas optimale α_1 solution du problème linéaire $\alpha_0 = minf(x_1 + \alpha d_1), \alpha \geq 0$

$$\alpha_0 = \frac{-d_1^T g_1}{d_1^T A d_1} = \frac{-\begin{matrix}7 & 1.5\end{matrix} \times \begin{matrix}-7 \\ -1.5\end{matrix}}{\begin{matrix}7 & 1.5\end{matrix} \times \begin{matrix}2 & 0 \\ 0 & 1\end{matrix} \begin{matrix}7 \\ 1.5\end{matrix}} = 0.5112$$

poser

$$x_{1_1} = x_{1_0} + \alpha_0 d_0 = 1.5786$$

$$x_{2_1} = x_{1_0} + \alpha_0 d_0 = 2.2668$$

Etape 3 : construire β_1 comme suit :

$$B_1 = B_0 - \frac{B_0 Y_1 Y_1^T B_0}{Y_1^T B_0 Y_1} + \frac{\bar{D}_0 \bar{D}_0^T}{\bar{D}_0^T Y_1}$$ Selon **DFP**

Avec

$$Y_1 = \nabla f(x_1) - \nabla f(x_0)$$
$$D_0 = \alpha_0 d_0$$

Remplacer *k* par *k+1* et aller à l'étape 1

Les itération sont résumé dans le tableau ci-dessous

Itérations		cordonnées	gradient	F(x)	α_k
1	x_1	-2.0000	-7.0000	6.62500	0.5112
	x_2	1.5000	-1.5000		

2	x_1	1.5786	0.1571	-6.4751	0.9888
	x_2	2.2668	-0.7332		
3	x_1	1.5000	0	-6.7500	
	x_2	3.0000	0		

Chapitre III

Optimisation avec contraintes

III.1 Introduction

On s'intéresse maintenant à des problèmes d'optimisation de la forme

$$min_{x \in \mathbb{R}^n} f(x)$$

Sous les contraintes $\begin{matrix} h(x) = 0 \\ g(x) \leq 0 \end{matrix}$

Où les fonctions *f*, *h* et *g* sont différentiables au moins une fois, et *f* est typiquement non-linéaire. Cependant nous étudierons le cas où *h* et *g* sont linéaires avec un intérêt tout particulier. Dans ce chapitre nous allons nous efforcer d'obtenir les conditions d'optimalité associées au problème d'optimisation avec contraintes. Les paragraphes suivants mettront ensuite l'accent sur les méthodes numériques permettant de le résoudre.

III.2 Problème avec contraintes d'égalité

On va tout d'abord s'intéresser au problème suivant, dit problème d'optimisation avec contraintes d'égalité seulement :

$$min_{x \in \mathbb{R}^n} f(x) \qquad \text{(III.1)}$$

Sous les contraintes $h(x) = 0$

Où $h: \mathbb{R}^n \rightarrow \mathbb{R}^p$ est différentiable. On note $S = \{x \in \mathbb{R}^n, h(x) = 0\}$

Considérons une courbe $Q(t)$ définie pour $t \leq 0$ vérifiant

$$Q(t) \in S, \forall t \in]-\alpha, \alpha[, \alpha > 0$$
$$Q(0) = \hat{x}$$

Puisque $Q(t) \subset S$ on a $h_i(Q(t)) = 0$ pour $1 \leq i \leq p$ et on peut écrire que

$$\frac{d}{dt} h_i(Q(t)) = \nabla h_i(Q(t))^T \dot{Q}(t) = 0, \qquad 1 \leq i \leq p.$$

Si on note $y = \dot{Q}(0)$ le vecteur tangent à la courbe $Q(t)$en $t = 0$, on a donc

$$\nabla h_i(\hat{x})^T y = 0, \qquad 1 \leq i \leq p.$$

Cela conduit à la définition suivante :

Définition III.2.1*On dit que $y \in \mathbb{R}^n$est une direction admissible en $\hat{x} \in S$ s'il existe $\alpha > 0$ et une courbe $Q(t)$ vérifiant*

$$Q(t) \in S, \forall t \in [-\alpha, \alpha]$$
$$Q(0) = \hat{x}$$
$$\dot{x}(0) = y$$

On notera alors $T(\hat{x})$.

L'ensemble $T(\hat{x})$ définit le plan tangent à S en $\hat{x}$. L'analyse faite précédemment montre que l'on a l'implication

$$y \in T(\hat{x}) \Rightarrow \nabla h_i(\hat{x})^T y = 0, \qquad 1 \leq i \leq p.$$

qui sera insuffisante pour montrer la condition nécessaire d'optimalité. Nous allons donc maintenant nous attacher à montrer sous quelles conditions cette dernière relation est une condition suffisante d'appartenance à$T(\hat{x})$.

Définition III.2.2*On dit que $\hat{x}$est un point régulier pour la contrainte h(x) = 0si*

- $\hat{x} = 0$
- *Les vecteurs $\nabla h_i(\hat{x})$ sont linéairement indépendants.*

Si on note $\nabla h_i(\hat{x})$ la matrice *n* × *p*

$$\nabla h(\hat{x}) = [\nabla h_1(\hat{x}) \dots \nabla h_p(\hat{x})]$$

la condition d'indépendance linéaire des $\nabla h_i(\hat{x})$ peut s'écrire

$$Rang\,\nabla h_i(\hat{x}) = p.$$

et on a donc $\nabla h_i(\hat{x})^T \dot{x}(0) = 0$pour toute courbe admissible$Q\ t$.

On a la proposition suivante :

Proposition III.2 *Si $\hat{x}$est un point régulier pour la contrainte h(x) = 0, alors*

$$\nabla h(\hat{x})^T y = 0 \Leftrightarrow y \in T(\hat{x})$$

III.2.1 Le théorème de Lagrange

Théorème III.2.1. *Soit$\hat{x} \in S = \{x \in \mathbb{R}^n, h(x) = 0$ un point régulier solution du problème (III.1), vérifiant donc*

$$f(\hat{x}) \leq f(x), \forall x \in S$$

Alors il existe nécessairement un vecteur $\lambda \in \mathbb{R}^p$unique vérifiant

$$\nabla f(\hat{x}) + \nabla h(\hat{x})\lambda = 0$$

soit encore

$$\nabla f(\hat{x}) + \sum_{i=1}^{p} \lambda_i \nabla h_i(\hat{x}) = 0$$

Les composantes du vecteur λ sont appelées multiplicateurs de Lagrange.

III.2.2. Définition du lagrangien

Considérons le problème avec contraintes d'égalité

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{(III.1)}$$

Sous les contraintes $h(x) = 0$

Où $h: \mathbb{R}^n \to \mathbb{R}^p$

Définition III.2.2.1 On appelle lagrangien associé au problème avec contraintes d'égalité la fonction $L: \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ définie par

$$L(x, \lambda) = f(x) + \sum_{i=1}^{p} \lambda_i h_i(x)$$

Les conditions de Lagrange peuvent se reformuler à l'aide du lagrangien: soit $\hat{x}$ solution de problème avec contraintes d'égalité. Alors il existe $\hat{\lambda}$ telque

$$\square_x L\ \hat{x}, \hat{\lambda}$$

Où on a noté $_x$ le gradient partiel par rapport à la variable *x*. Dans la suite nous ferons l'hypothèse que *h* et *f* sont deux fois continûment différentiables.

III.2.3 Condition nécessaire du second ordre

Théorème III.2.3.1 Soit $\hat{x}$ un point régulier solution de problème avec contraintes d'égalité. Alors il existe $\hat{\lambda}$ tel que

$$\square_x L\ x, \lambda\ = 0$$

Et de plus pour tout $y \square T\ \hat{x}\ , y \neq 0$ on a

$$y^T \square^2_{zz} L\ \hat{x}, \hat{\lambda}\ y \geq 0$$

Le résultat suivant est une généralisation du théorème précédent dont la démonstration sera admise.

Théorème III.2.3.2 soit $\hat{x} \square \square^n$ et $\hat{\lambda} \square \square^p$ vérifiant les conditions

$$\square\ \hat{x}\ = 0$$

$$\square f\ \hat{x}\ + \sum_{i=1}^{p} \hat{\lambda}_i \square \square_i\ \hat{x}\ = 0$$

$$y^T \square^2_{zz} L\ \hat{x}, \hat{\lambda}\ y \geq 0\ , \square y \square T\ \hat{x}\ , y \neq 0$$

alors $\hat{x}$ est un minimum local du problème avec contraintes d'égalité.

III.2.4 Condition nécessaire du second ordre

Théorème III.3.2 soit $\hat{x} \square \square^n$ et $\hat{\lambda} \square \square^p$ vérifiant les conditions

$$g(\hat{x}) \leq 0$$

$$\square f\ \hat{x}\ + \sum_{i=1}^{m} \lambda_i \square g_i\ \hat{x}\ = 0.$$

$$\lambda_i \geq 0, i = 1 \ldots m$$

$$\lambda_i \square g_i\ \hat{x}\ = 0, i = 1 \ldots m.$$

$$y^T \square^2_{zz} L\ \hat{x}, \hat{\lambda}\ y \geq 0\ , \square y \square T^+\ \hat{x}\ , y \neq 0$$

Ou on a noté $T^+\ \hat{x}$ le plan tangent en $\hat{x}$ à la sureface

$$S^+ = \ x \square \square^n, g_i\ \hat{x}\ = 0, i \square I\ \hat{x}\ \ et\ \lambda_i \geq 0$$

alors $\hat{x}$ est un minimum local du problème avec contraintes d'égalité.

Exemple 1 :

Minimiser $2x_1 + x_2$

Avec $h(x) = x_1^2 + x_2^2 - 1 = 0$

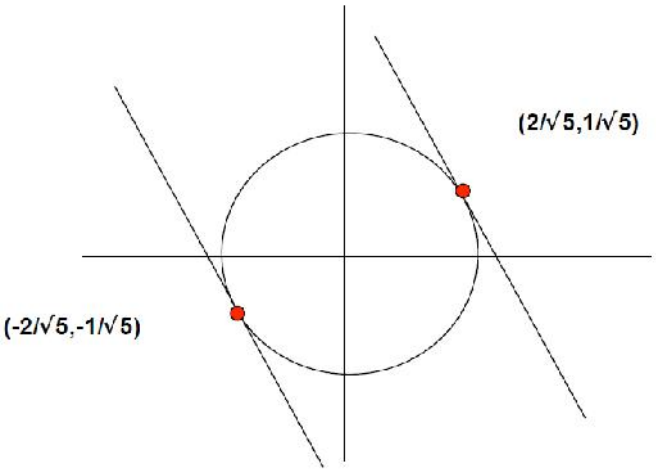
Le Lagrangien est

$$L(x, \lambda) = 2x_1 + x_2 + \lambda(x_1^2 + x_2^2 - 1)$$

$\frac{\partial L}{\partial x_i} = 0$ ce qui donne $\begin{matrix} \frac{\partial f}{\partial x_1} + \lambda\frac{\partial h}{\partial x_1} = 0 \\ \frac{\partial f}{\partial x_2} + \lambda\frac{\partial h}{\partial x_2} = 0 \end{matrix} \Rightarrow \begin{matrix} 2 + 2\lambda x_1 = 0 \\ 1 + 2\lambda x_2 = 0 \end{matrix} \Rightarrow \begin{matrix} x_1^* = \frac{-1}{\lambda} \\ x_2^* = \frac{-1}{2\lambda} \end{matrix}$

$$\Rightarrow x = 0 \Rightarrow \left(\frac{1}{\lambda}\right)^2 + \left(\frac{1}{2\lambda}\right)^2 = 0 \Rightarrow \lambda^* = \pm\frac{\sqrt{5}}{2}$$

2 points vérifient les conditions de Lagrange, mais un seul est minimum : ces conditions sont nécessaires, mais pas suffisantes pour qu'un point soit optimum.



Exemple 2

Minimiser $f(x, y) = x^2 + y^2$

Sous les contraintes $h(x, y) = y - x + 1 = 0$

$$\begin{matrix} \frac{\partial f}{\partial x} + \lambda\frac{\partial h}{\partial x} = 0 \\ \frac{\partial f}{\partial y} + \lambda\frac{\partial h}{\partial y} = 0 \end{matrix} \Rightarrow \begin{matrix} 2x - \lambda = 0 \\ 2y + \lambda = 0 \end{matrix} \Rightarrow \begin{matrix} x^* = \frac{1}{2} \\ y^* = \frac{-1}{2} \\ \lambda^* = 1 \end{matrix}$$

III.3. Problème avec contraintes d'inégalité - Conditions nécessaires d'optimalité (KKT,

Karush, Kuhn, Tucker)

On s'intéresse maintenant au problème suivant, dit problème d'optimisation avec contraintes d'inégalité seulement:

$$min_{x \in \mathbb{R}^n} f(x) \qquad \text{(III.2)}$$

Sous les contraintes $g(x) \leq 0$

Où $g: \square^n \rightarrow \square^m$ est différentiable (il n'y a ici aucune condition sur m). On notera K l'ensemble des points admissibles, c'est-à-dire $K = \{x \in \square^n, g(x) \leq 0\}$
Au point solution de Problème avec contraintes d'inégalité il va de soi que les contraintes effectivement actives vérifieront $g(\hat{x}) = 0$. Cependant, puisque l'on ne sait pas apriori quelles sont ces contraintes, le passage de Problème avec contraintes d'inégalité a un problème du type Problème avec contraintes d'égalité n'est pas direct.

Définition III.3.1 *On appelle contraintes saturées en $\hat{x}$ l'ensemble des indices i tel que $g_i(\hat{x}) = 0$, et on note*

$$I(\hat{x}) = \{i | g_i(\hat{x}) = 0\}$$

Le concept de direction admissible se définit comme suit:

Définition III.3.2. *On dit que $y \in \square^n$ est une direction admissible en $\hat{x} \in K$ s'il existe $\alpha > 0$ et une courbe $Q(t)$ vérifiant*

$$Q(t) \in K, \forall t \in [-\alpha, \alpha]$$
$$Q(0) = \hat{x}$$
$$\dot{Q}(0) = y$$

On notera alors $y \in C(\hat{x})$.

Lemme III.3.1. Soit $y \in \square^n$ une direction admissible en $\hat{x} \in K$, alors on a nécessairement

$$\nabla g_i(\hat{x})^T y \leq 0, i \in I(\hat{x})$$

Comme dans le cas des contraintes d'égalité, on doit définir la notion de point régulier qui est nécessaire pour que la condition précédente soit suffisante:

Définition III.3.3. *On dit que $\hat{x}$ est un point régulier pour la contrainte $g(x) \leq 0$ si*

- $g(\hat{x}) \leq 0$,
- *Les vecteurs $\{\nabla g_i(\hat{x})\}_{i \in I(\hat{x})}$ sont linéairement indépendants.*

Sous l'hypothèse de régularité de $\hat{x}$ on aura, comme dans le cas des contraintes d'égalité

$$\nabla g_i(\hat{x})^T y \leq 0, i \in I(\hat{x}) \quad \forall y \in C(\hat{x})$$

La proposition suivante permet d'effectuer le premier pas vers les conditions de **Kuhnet Tucker**.

Proposition III.3.1. Soit $\hat{x}$ la solution du problème avec contraintes d'inégalité. Il existe $\mu > 0$ telque

$$\forall x \in B(\hat{x}, \mu), g_i(x) < 0, i \in I(\hat{x})$$

Où on a noté $B(\hat{x}, \mu)$ la boule de centre $\hat{x}$ et de rayon μ. Alors $\hat{x}$ est la solution du problème

$$\min_{x \in B(\hat{x}, \mu)} f(x)$$
$$g_i(x) = 0, i \in I(\hat{x})$$

Ce résultat est uniquement dû à la continuité de *g*, et montre que l'on est localement ramené à un problème avec contraintes d'égalité. On peut donc maintenant énoncer le résultat principal:

Théorème III.3.1. Soit $\hat{x} \in K$ un point régulier solution du problème avec contraintes d'inégalité. Alors il existe un unique vecteur $\lambda \in \mathbb{R}^n$ telque

$$\nabla f(\hat{x}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\hat{x}) = 0.$$

$$\lambda_i = 0, i = 1 \dots m$$

$$\lambda_i \nabla g_i(\hat{x}) = 0, i = 1 \dots m.$$

Exemple

Minimiser $f(x) = x_1^2 + x_2$

Sous les contraintes $\begin{matrix} g_1(x) = x_1^2 + x_2^2 - 9 \leq 0 \\ g_2(x) = x_1 + x_2 - 1 \leq 0 \end{matrix}$

On va montrer que la résolution des conditions de KKT donne les valeurs du minimum

x et des multiplicateurs λ associés :

Les conditions de KKT s'écrivent :

$$\nabla f(x) + \lambda_1 \nabla g_1(x) + \lambda_2 \nabla g_2(x) = \begin{matrix} 2x_1 \\ 1 \end{matrix} + \lambda_1 \begin{matrix} 2x_1 \\ 2x_2 \end{matrix} + \lambda_2 \begin{matrix} 1 \\ 1 \end{matrix} = 0 \qquad (1)$$

$$\begin{matrix} \lambda_1 g_1(x) = \lambda_1 (x_1^2 + x_2^2 - 9) \\ \lambda_2 g_2(x) = \lambda_2 (x_1 + x_2 - 1) \end{matrix} \qquad (2)$$

$$\lambda_1, \lambda_2 \geq 0 \qquad (3)$$

Pour résoudre, on étudie les différents cas possibles (ce n'est pas une méthode générale).

Si $\lambda_1 = \lambda_2 = 0$, (2) est vérifiée, mais pas (1)

Si $\lambda_1 \neq 0$ *et* $\lambda_2 = 0$, (1) $\Rightarrow 2x_1 + 2x_1\lambda_1 = 0$

$\lambda_1 = -1$ne vérifie pas (3), donc $x_1 = 0$

(2) s'écrit $x_1^2 - 9 = 0$, et comme $\lambda_1 \neq 0, x_2 = \pm 3$

Si $x_2 = 3$, (1) $\Rightarrow 1 + 6\lambda_1 = 0$, Impossible avec (3)

Donc $x_2 = -3$ et (1) $\Rightarrow 1 - 6\lambda_1 = 0$, d'où $\lambda_1 = \frac{1}{6}$

La résolution des conditions de KKT donne donc

$$x_1 = 0 \qquad x_2 = -3 \qquad \lambda_1 = \frac{1}{6} \qquad \lambda_1 = 0$$

III.4 Conditions de KKT pour un problème avec limitations d'égalité et d'inégalité

On suppose que : *f, g_j, j=1, ..., m , h_l, l =1, ..., p* sont continues et différentiables *l*

Alors, si x_0 est un optimum local, il existe des nombres $\lambda_j \geq 0$ et μ de signe quelconque tels que:

$$\nabla f(x_0) + \sum_{j=1}^{m} \lambda_j \nabla g_j(x_0) + \sum_{l=1}^{p} \mu \nabla h_l(x_0) = 0$$

$$\lambda_j g_j(x_0) = 0 \quad j = 1 \dots m$$

Remarque :

- On suppose $x_0 \in A$, donc on a $g_j(x_0) \leq 0 \quad h_l(x_0) = 0$
- Si on a des limitations d'égalité seulement, on retrouve les conditions de Lagrange

III.5 Méthodes de pénalité

III.5.1 Principe

Remplacer le problème contraint :

$$Minimiser\ f(x)$$
$$Avec \quad g_i(x) \leq 0; j = 1, \dots, m$$

par une suite de problèmes sans contraintes

Si l'on définit la fonction h telle que :

$$h(y) = \begin{cases} 0 \ si \ y \leq 0 \\ +\infty \ si \ y > 0 \end{cases}$$

la résolution du problème d'optimisation revient à résoudre le problème de minimisation sans contrainte suivant :

$$Min\ \phi(x) = f(x) + H(x)$$

Ou $$H(x) = \sum_{j=1}^{m} \mu_j(g_j(x))$$

La fonction *H* étant discontinue et 'infinie' en dehors du domaine admissible, on ne peut utiliser cette méthode telle quelle. Une méthode de pénalité consiste à minimiser une fonction (pénalisée) de la forme :

$$Min_x\ P(x) = f(x) + \psi(x, \alpha)$$

où la fonction ψ est telle :

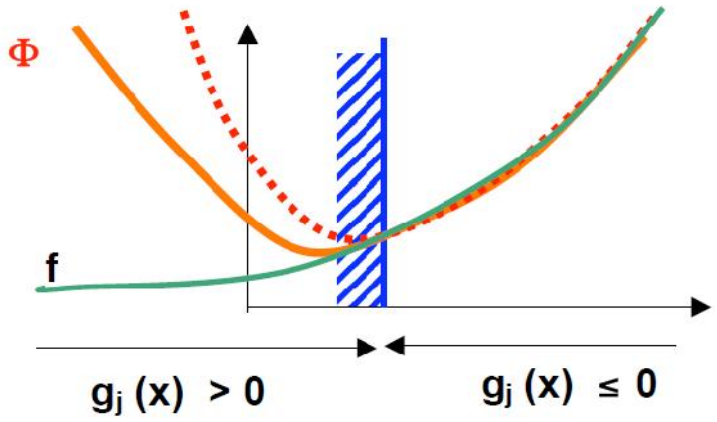
- Qu'elle force l'optimum à saturer approximativement les limitations qui doivent l'être.
- Qu' elle n'altère pas trop la solution approchée obtenue.

Le paramètre de pénalisation α est à caler pour que la solution approchée soit précise sans que le système à minimiser soit trop mal conditionné.

III.5.2 Pénalité extérieure

Modifier la fonction à minimiser pour qu'elle prenne de « grandes » valeurs en dehors du domaine admissible.

Exemple avec 1 variable, 1 limitation :



Pour cela, on définit :

$$< g_j\ x\ >^{+} = \begin{cases} 0 \text{ pour } g_j(x) \leq 0 \\ g_j\ x \text{ pour } g_j(x) > 0 \end{cases}$$

et on considère la fonction obtenue en ajoutant à la fonction f(x) à minimiser les termes :

$$\epsilon < g_j\ x\ >^{+2}$$

où ϵ est un scalaire > 0 (facteur de pénalité), ceci pour chaque contrainte g_j .
On obtient la fonction :

$$\phi\ x, \epsilon\ = f\ x\ + \epsilon \sum_{j=1}^{m} < g_j\ x\ >^{+2}$$

et on considère le problème :

$$minimiser\ \phi\ x, \epsilon\ = f\ x\ + \epsilon \sum_{j=1}^{m} < g_j\ x\ >^{+2} \quad (P')$$

Soit (x^*) la solution de (P'). Si on a choisi ϵ « assez grand », les termes $< g_j\ x\ >^{+2}$ seront « petits », ce qui signifie que x^* sera « presque » admissible.
Mais en pratique, si ϵ est très grand, ϕ est mal conditionnée. C'est pourquoi on procède itérativement, en augmentant ϵ progressivement, et en initialisant chaque recherche par la solution précédente.
Algorithme général :
1- k=1, choix de x_1, ϵ_i (proportion coût/pénalité)
2- Résoudre (par une méthode sans contraintes) :

$$minimiser\ \phi\ x, \epsilon_k\ = f\ x\ + \epsilon_k \sum_{j=1}^{m} < g_j\ x\ >^{+2} \quad (P_k')$$

avec pour point de départ x_k
→ solution x_k^*
3- Tester si les contraintes sont « suffisamment » satisfaites :

$$Max_{j=1,\dots m}(g_j\ x\ ) \leq \varepsilon$$

Si oui, fin, $x^* = x_k$
Si non,
4- augmenter le facteur de pénalité :
$\epsilon_k = \epsilon_k \times c$ par ex $1 < c < 10$
mettre à jour le point de départ : $x_{k+1} = x_k^*$
$k=k+1$
aller en (2)

♦ facile à implanter, mais pas très performant (conditionnement), solution toujours légèrement non admissible.

Remarque : Dans le cas de contraintes d'égalité $h_i(x) = 0 \quad i = 1, \dots, p$), la fonction pénalisée à considérer est :

$$\phi(x, \epsilon) = f(x) + \epsilon \sum_{j=1}^{p} (h_j(x))^{+2}$$

III.5.3 Pénalité intérieure

Même principe que la pénalité extérieure, mais pour obtenir un minimum approché par l'intérieur du domaine (donc toujours admissible). On définit :

$$\phi(x, \epsilon) = f(x) - \epsilon \sum_{j=1}^{m} \frac{1}{g_j(x)}$$

Ou bien

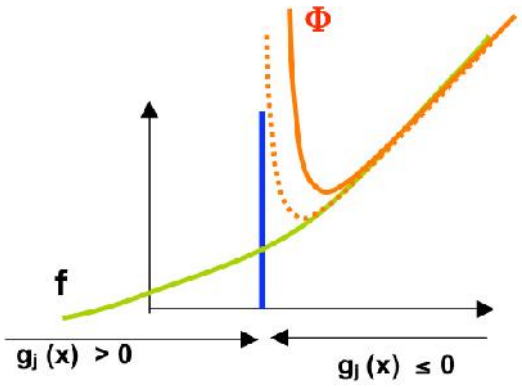
$$\phi(x, \epsilon) = f(x) - \epsilon \sum_{j=1}^{m} \log(-g_j(x))$$

Ou le facteur de pénalité r doit cette fois tendre vers 0 pour que x s'approche de la frontière du domaine admissible. On résout donc une succession de problèmes sans contraintes :

$$\phi(x, \epsilon_k) = f(x) - \epsilon_k \sum_{j=1}^{m} \frac{1}{g_j(x)} \qquad k = 1, \dots$$

Avec

$$\epsilon_{k+1} = \epsilon_k \times c \quad (par\ ex\ 0.1 < c < 1)$$



Il faut connaître un point de départ admissible

♦ Cette pénalisation n'est pas valable sur la frontière ni à l'extérieur du domaine admissible.

Exemple

Soit à minimiser la fonction $f(x) = 0.5\,x$

sous la contrainte $g(x) = 4 - x \leq 0$

par les différentes méthodes de pénalité.

1. Pénalités extérieures.

La fonction pénalisée s'écrit :

$$P_{ext}\ x, \epsilon\ = 0.5x + \epsilon\ 4 - x\ ^{+2}$$

Sa minimisation induit les solutions approchées dépendant de ϵ :

$$x_\epsilon = 4 - \frac{0.5}{2\epsilon}$$

2. Pénalités intérieures

La fonction pénalisée s'écrit :

$$P_{int}\ x, \epsilon\ = 0.5x - \frac{\epsilon}{4 - x}$$

La solution approchée associée est :

$$x_\epsilon = 4 + \frac{\overline{\epsilon}}{0.5}$$

III.6 Programmation quadratique séquentielle

Nous allons nous attaquer maintenant à une dernière classe de méthodes appelées SQP pour Sequential Quadratic Programming.

Intéressons-nous dans un premier temps au cas de contraintes en égalité. Considérons le problème

$$\min f(x)$$
$$x \in C$$

$$C = \{\ x \in \mathbb{R}^n,\ h_i\ x\ = 0,\quad i = 1, \dots, p\}$$

L'idée essentielle consiste à résoudre une succession de problèmes quadratiques avec contraintes linéaires (ces problèmes sont relativement simples à résoudre) qui sont des approximations du problème de départ.

Etant donné x_k, on cherche $x_{k+1} = x_k + \rho_k d_k$ ou $d_k \in \mathbb{R}^n$ est une direction de descente et ρ_k le pas.

Effectuons une approximation des contraintes *h* à l'aide de la formule de Taylor du premier ordre :

$$h_i\ x_k + d\ = h_i\ x_k\ + \nabla h_i\ x_k\ .d + o(\|d\|^2)$$

Si on néglige les termes d'ordre supérieur ou égal à 2, on définit la direction d_k avec la direction permettant d'assumer $h_i(x_k + d) \approx 0$

On pose donc $h_i\ x_k\ + \nabla h_i\ x_k\ .d^k = 0, \forall i = 1, \dots, q$

$$\nabla h_i\ x_k\ .d^k = -h_i\ x_k$$

Ou $\nabla c_i(x_k)$ est la matrice jacobienne de c en x_k. Cette relation correspond à une linéarisation des contraintes au voisinage de x_k : c'est un système linéaire.

Par ailleurs, il faudrait que x_{k+1} diminue la valeur du Lagrangien (puisque c'est le Lagrangien qui joue le rôle de la fonction objectif quand on a des contraintes). On va faire une approximation du Lagrangien.

$$L(x_k + d, \mu) = L(x_k, \mu) + \nabla_x L(x_k, \mu), d + \frac{1}{2} \nabla^2_{xx} L(x_k, \mu)\, d, d + o(\|d\|^3)$$

Si on néglige les termes d'ordre supérieur ou égal à 3, on voit qu'il faut minimiser

$$\nabla_x L(x_k, \mu), d + \frac{1}{2} \nabla^2_{xx} L(x_k, \mu)\, d, d$$

Pour espérer minimiser le Lagrangien. On cherche donc, en fin de compte x_k solution du problème

$$\begin{aligned} &\min \nabla f(x_k), d + (\nabla^2_{xx}(x_k, \mu).d, d) \\ &\nabla c(x_k)\, d_k + c(x_k) = 0 \end{aligned}$$

En effet, nous avons :

$$(\nabla_x L(x_k, \mu), d_k) = \nabla f(x_k)\, d_k + \mu^t \nabla f(x_k) d_k$$

$$= \nabla f(x_k), d_k + \underbrace{\mu^T f(x_k) d_k}_{constant}$$

Le dernier terme étant constant. Il reste ensuite à déterminer le pas ρ_k et le multiplicateur μ_k à chaque itération. Il y a bien sur, beaucoup de possibilités qui génèrent autant de variantes de la méthode.

Nous présentons ici, la méthode qui est basée sur le choix : $\rho_k = 1$

Algorithme : Méthode SQP pour des contraintes en égalité

Initialisation

K=1, choix de $x_0 \in \Omega, \mu_0 \in \mathbb{R}^p$

Itération k : tant que le critère d'arrêt n'est pas satisfait

1- Résoudre le sous problème quadratique

$$\begin{aligned} &\min \nabla f(x_k), d + \frac{1}{2}(\nabla^2_{xx} L(x_k, \mu_k).d, d) \\ &\nabla c(x_k)\, d_k + c(x_k) = 0 \end{aligned} \quad (1)$$

2- On pose alors $\mu_{k+1} \in \mathbb{R}^p$ le multiplicateur associe a la contrainte (en égalité) de (1) et $x_{k+1} = x_k + d_k$

K=k+1

Intéressons-nous maintenant au cas de contraintes générales en égalité et inégalité ; globalement, le principe est le même, seule la fonction Lagrangienne est modifiée.

Pour le problème :

$$\min f(x)$$
$$x \in C$$

ou

$$C = \{x \in \mathbb{R}^n, \quad h(x) = 0, g(x) \leq 0\}$$

Elle vaut $L(x, \mu, \lambda) = f(x) + \mu^T f(x) + \lambda^T g(x)$

Ou $\mu \in \mathbb{R}^p et \lambda \in \mathbb{R}^m$

La méthode SQP s'´ecrit de la façon suivante ou linéarise les contrainte et on fait une approximation quadratique de L. On obtient alors l'algorithme suivant.

Il existe finalement pour ce type d'algorithme, des résultats de convergence de l'algorithme.

Algorithme : Algorithme SQP pour des contraintes générales

Initialisation

K=1, choix de $x_0 \in \Omega, \mu_0 \in \mathbb{R}^p et \lambda_0 \in \mathbb{R}^m$

Itération *k* : tant que le critère d'arrêt n'est pas satisfait

1- Résoudre le sous problème quadratique d'inconnu *d*

$$\begin{aligned} \min \ & \nabla f(x_k) d + \frac{1}{2}(\nabla^2_{xx} L(x_k, \mu_k) . d, d) \\ & \nabla h(x_k) d + h(x_k) = 0 \\ & \nabla g(x_k) d + g(x_k) \leq 0 \end{aligned} \qquad (2)$$

2- On pose alors $\mu_{k+1} \in \mathbb{R}^p$ le multiplicateur associe a la contrainte (en égalité) de (1) et $\lambda_{k+1} \in \mathbb{R}^m$ le multiplicateur (positif) associé a la contrainte en inégalité $x_{k+1} = x_k + \lambda_k$

K=k+1

Chapitre IV :

Méthodes d'optimisation stochastiques

IV.1 L'algorithme génétique

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature : croisements, mutations, sélections, etc.... Ils appartiennent à la classe des algorithmes évolutionnaires.

IV.1.1 Le codage

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée).

Un chromosome est une suite de gène, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position : son locus sur le chromosome en question.

Chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus.

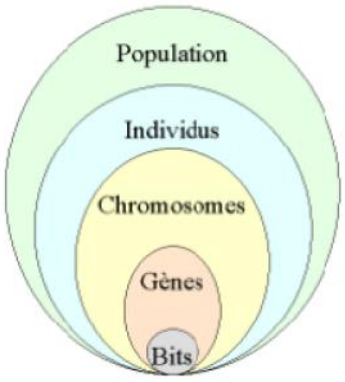


Figure 1: les cinq niveaux d'organisation d'un algorithme génétique

Il y a trois principaux types de codage utilisables, et on peut passer de l'un à l'autre relativement facilement :

• **le codage binaire :** c'est le plus utilisé.

Chaque gène dispose du même alphabet binaire {0, 1}. Un gène est alors représenté par un entier long (32 bits), les chromosomes qui sont des suites de gènes sont représentés par des tableaux de gènes et les individus de notre espace de recherche sont représentés par des tableaux de chromosomes.

Ce cas peut être généralisé à tout alphabet allélique n-aire permettant un codage plus intuitif, par exemple pour le problème du voyageur de commerce on peut préférer utiliser l'alphabet

allélique $\{c_1, c_2, c_3, ..., c_n\}$ où c_i représente la ville de numéro i.

• **le codage réel :** cela peut-être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle.

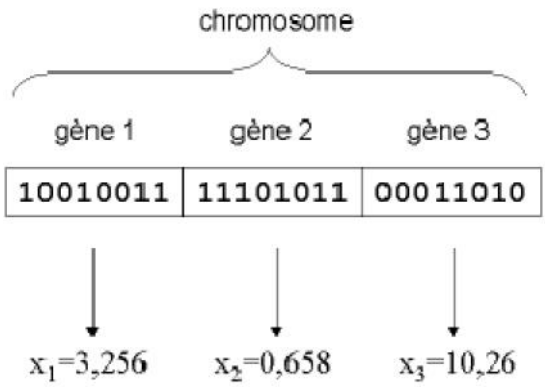


Figure 2 : illustration schématique du codage des variables réelles

• **le codage de Gray :** dans le cas d'un codage binaire on utilise souvent la "distance de Hamming" comme mesure de la dissimilarité entre deux éléments de population, cette mesure compte les différences de bits de même rang de ces deux séquences.

Et c'est la que le codage binaire commence à montrer ses limites. En effet, deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un "codage de Gray" : le codage de Gray est un codage qui a comme propriété que : entre un élément n et un élément n + 1, donc voisin dans l'espace de recherche, un seul bit diffère.

IV.1.2 L'opérateur de sélection

Cet opérateur est chargé de définir quels seront les individus de P qui vont être dupliqués dans la nouvelle population P' et vont servir de parents (application de l'opérateur de croisement).

Soit *n* le nombre d'individus de P, on doit en sélectionner *n/2* (l'opérateur de croisement nous permet de repasser à *n* individus).

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.

On trouve essentiellement quatre types de méthodes de sélection différentes :

• La méthode de la "loterie biaisée" (roulette wheel) de GoldBerg,

• La méthode "élitiste",

• La sélection par tournois,

• La sélection universelle stochastique.

a) La loterie biaisée ou roulette wheel :

Cette méthode est la plus connue et la plus utilisée.

Avec cette méthode chaque individu a une chance d'être sélectionné proportionnelle à sa performance, donc plus les individus sont adaptés au problème, plus ils ont de chances d'être sélectionnés.
Pour utiliser l'image de la "roue du forain", chaque individu se voit attribué un secteur dont l'angle est proportionnel à son adaptation, sa "fitness".
On fait tourner la roue et quand elle cesse de tourner on sélectionne l'individu correspondant au secteur désigné par une sorte de "curseur", curseur qui pointe sur un secteur particulier de celle-ci après qu'elle se soit arrêté de tourner.

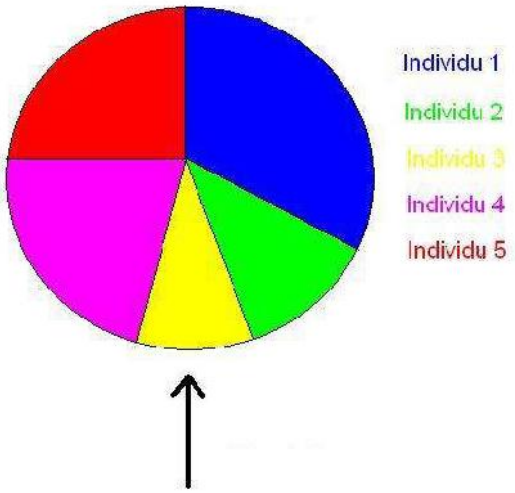


Figure 3: la méthode de sélection de la loterie biaisée

Cette méthode, bien que largement répandue, a pas mal d'inconvénients :
• En effet, elle a une forte variance. Il n'est pas impossible que sur n sélections successives destinées à désigner les parents de la nouvelle génération P', la quasi-totalité, voire pire la totalité des n individus sélectionnés soient des individus ayant une fitness vraiment mauvaise et donc que pratiquement aucun individu voire aucun individu a forte fitness ne fasse partie des parents de la nouvelle génération. Ce phénomène est bien sûr très dommageable car cela va complètement à l'encontre du principe des algorithmes génétiques qui veut que les meilleurs individus soient sélectionnés de manière à converger vers une solution la plus optimale possible.
• A l'inverse, on peut arriver à une domination écrasante d'un individu "localement supérieur". Ceci entrainant une grave perte de diversité. Imaginons par exemple qu'on ait un individu ayant une fitness très élévée par rapport au reste de la population, disons dix fois supérieure, il n'est pas impossible qu'après quelques générations successives on se retrouve avec une population ne contenant que des copies de cet individu. Le problème est que cet individu avait une fitness très élevée, mais que cette fitness était toute relative, elle était très élevée mais seulement en comparaison des autres individus. On se retrouve donc face à problème connu sous le nom de "convergence prématurée; l'évolution se met donc à stagner et on atteindra alors jamais l'optimum, on restera bloqué sur un optimum local.

Il existe certaines techniques pour essayer de limiter ce phénomène, comme par exemple le "scaling", qui consiste à effectuer un changement d'échelle de manière à augmenter ou diminuer de manière forcée la fitness d'un individu par rapport à un autre selon leur écart de fitness.
Malgré tout, il est conseillé d'opter plutôt pour une autre méthode de sélection.

b) La méthode élitiste.

Cette méthode consiste à sélectionner les n individus dont on a besoin pour la nouvelle génération P' en prenant les n meilleurs individus de la population P après l'avoir triée de manière décroissante selon la fitness de ses individus.
Il est inutile de préciser que cette méthode est encore pire que celle de la loterie biaisée dans le sens où elle amènera à une convergence prématurée encore plus rapidement et surtout de manière encore plus sûre que la méthode de sélection de la loterie biaisée ; en effet, la pression de la sélection est trop forte, la variance nulle et la diversité inexistante, du moins le peu de diversité qu'il pourrait y avoir ne résultera pas de la sélection mais plutôt du croisement et des mutations. Là aussi il faut opter pour une autre méthode de sélection.

c) La sélection par tournois :

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants. Le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de P, et on les fait "combattre". Celui qui a la fitness la plus élevée l'emporte avec une probabilité p comprise entre 0.5 et 1. On répète ce processus n fois de manière à obtenir les n individus de P' qui serviront de parents.
La variance de cette méthode est élevée et le fait d'augmenter ou de diminuer la valeur de *p* permet respectivement de diminuer ou d'augmenter la pression de la sélection.

d) La sélection universelle stochastique :

Cette méthode semble être très peu utilisée et qui plus est possède une variance faible, donc introduit peu de diversité, nous n'entrerons donc pas dans les détails, on se contentera d'exposer sa mise en œuvre : On prend l'image d'un segment découpé en autant de sous-segments qu'il y a d'individus. Les individus sélectionnés sont désignés par un ensemble de points équidistants.

IV.1.3 L'opérateur de croisement ou crossover

Le crossover utilisé par les algorithmes génétiques est la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents. Son rôle fondamental est de permettre la *recombinaison* des informations présentes dans le patrimoine génétique de la population.
Cet opérateur est appliqué après avoir appliqué l'opérateur de sélection sur la population P; on se retrouve donc avec une population P' de *n/2* individus et on doit doubler ce nombre pour que notre nouvelle génération soit complète.

On va donc créer de manière aléatoire *n/4* couples et on les fait se "reproduire". Les chromosomes (ensembles de paramètres) des parents sont alors copiés et recombinés de façon à former deux descendants possédant des caractéristiques issues des deux parents.

Détaillons ce qui se passe pour chaque couple au niveau de chacun de leurs chromosomes :

Un, deux, voire jusqu'à *lg - 1* (où lg est la longueur du chromosome) points de croisements (loci) sont tirés au hasard, chaque chromosome se retrouve donc séparé en "segments". Puis chaque segment du parent 1 est échangé avec son "homologue" du parent 2 selon une probabilité de croisement *pc* . De ce processus résulte 2 fils pour chaque couple et notre population P' contient donc bien maintenant *n* individus.

On peut noter que le nombre de points de croisements ainsi que la probabilité de croisement *pc* permettent d'introduire plus ou moins de diversité. En effet, plus le nombre de points de croisements sera grand et plus la probabilité de croisement sera élevée plus il y aura d'échange de segments, donc d'échange de paramètres, d'information, et plus le nombre de points de croisements sera petit et plus la probabilité de croisement sera faible, moins le croisement apportera de diversité.

Ci-dessous, un schéma illustrant un croisement en un point, un autre pour un croisement en deux points, et enfin un schéma représentant un croisement avec *lg - 1* points de croisements (on notera d'ailleurs sur ce schéma que l'échange d'un segment avec son homologue ne se fait pas toujours) :

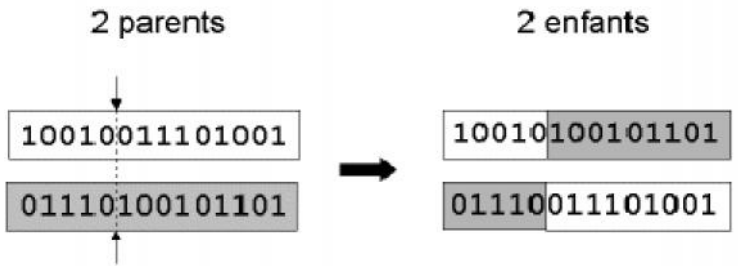


Figure 4: croisement avec un point de crossover

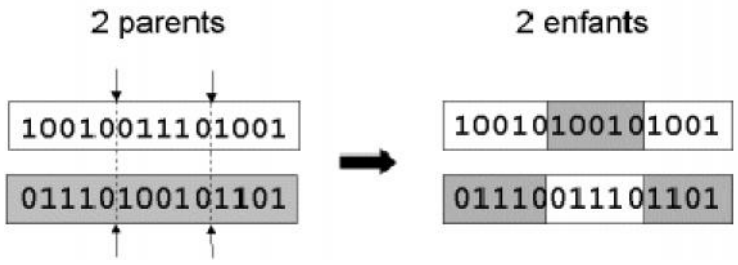


Figure 5: croisement avec 2 points de crossover

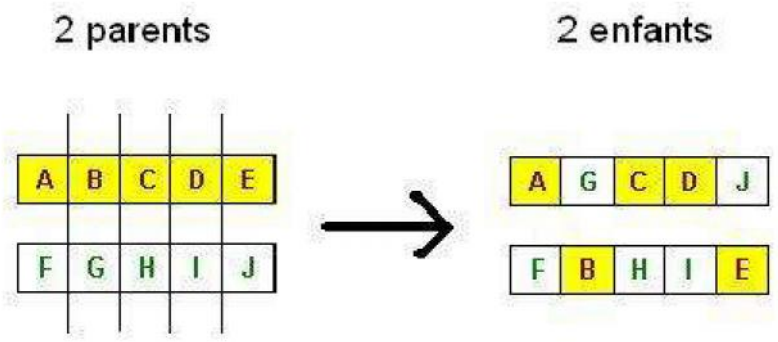


Figure 6: croisement uniforme

On peut citer aussi une autre méthode très utilisée dans le cas des problèmes modélisés par un codage binaire, il s'agit du ***croisement uniforme***. La mise en œuvre de ce procédé est fort simple, elle consiste à définir de manière aléatoire un "masque", c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus, de quel parent le premier fils devra hériter du gène s'y trouvant; si face à un locus le masque présente un 0, le fils héritera le gène s'y trouvant du parent n° 1, si il présente un 1 il en héritera du parent n° 2. La création du fils n° 2 se fait de manière symétrique : si pour un gène donné le masque indique que le fils n° 1 devra recevoir celui-ci du parent n° 1 alors le fils n° 2 le recevra du parent n°2, et si le fils n° 1 le reçoit du parent n° 2 alors le fils 2 le recevra du parent n° 1.

L'opérateur de croisement favorise l'exploration de l'espace de recherche. En effet, considérons deux gènes A et B pouvant être améliorés par mutation. Il est peu probable que les deux gènes améliorés A' et B' apparaissent par mutation dans un même individu. Mais si un parent porte le gène mutant A' et l'autre le gène mutant B', l'opérateur de croisement permettra de combiner rapidement A' et B' et donc de créer un nouvel individu possédant cette combinaison, combinaison grâce à laquelle il est possible qu'il soit encore plus adapté que ses parents. L'opérateur de croisement assure donc le brassage du matériel génétique et l'accumulation des mutations favorables. En termes plus concrets, cet opérateur permet de créer de nouvelles combinaisons des paramètres des composants. Malgré tout, il est possible que l'action conjointe de la sélection et du croisement ne permette pas de converger vers la solution optimale du problème. En effet, imaginons que nous avons une population d'individus possédant un seul chromosome. Considérons un gène particulier de ce chromosome, on l'appellera G, gène ayant 2 allèles possibles : 0 et 1; si aucun individu de la population initiale ne possède l'allèle 1 pour ce gène, aucun croisement possible ne permettra d'introduire cet allèle pour notre gène G. Si la solution optimale au problème est telle que notre gène G possède l'allèle 1, il nous sera impossible d'atteindre cette solution optimale simplement par sélection et croisement. C'est pour remédier entre autre à ce problème que l'opérateur de mutation est utilisé.

IV.1.4 L'opérateur de mutation

Cet opérateur consiste à changer la valeur allélique d'un gène avec une probabilité *pm* très faible, généralement comprise entre 0.01 et 0.001. On peut aussi prendre *pm = 1 / lg* où l*g* est la longueur de la chaîne de bits codant notre chromosome. Une mutation consiste simplement en l'inversion d'un bit (ou de plusieurs bits, mais vu la probabilité de mutation c'est extrêmement rare) se trouvant en un locus bien particulier et lui aussi déterminé de manière aléatoire; on peut donc résumer la mutation de la façon suivante :

On utilise une fonction censée nous retourner *true* avec une probabilité *pm* .

Pour chaque locus **faire**

Faire appel à la fonction

Si cette fonction nous renvoie *true* **alors**

on inverse le bit se trouvant à ce locus

Fin Si

Fin Pour

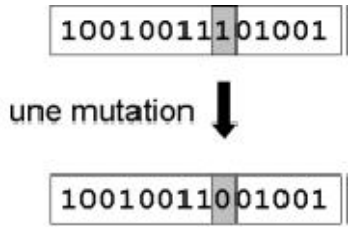


Figure 7 : une mutation

L'opérateur de mutation modifie donc de manière complètement aléatoire les caractéristiques d'une solution, ce qui permet d'introduire et de maintenir la diversité au sein de notre population de solutions. Cet opérateur joue le rôle d'un "élément perturbateur", il introduit du "bruit" au sein de la population.

Cet opérateur dispose de 4 grands avantages :

• Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques.

• Il permet d'éviter un phénomène connu sous le nom de ***dérive génétique***. On parle de dérive génétique quand certains gènes favorisés par le hasard se répandent au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. Le fait que l'opérateur de mutation puisse entraîner de manière aléatoire des changements au niveau de n'importe quel locus permet d'éviter l'installation de cette situation défavorable.

• Il permet de limiter les risques d'une convergence prématurée causée par exemple par une méthode de sélection élitiste imposant à la population une pression sélective trop forte. En effet, dans le cas d'une convergence prématurée on se retrouve avec une population dont tous les individus sont identiques mais ne sont que des optimums locaux. Tous les individus étant identiques, le croisement ne changera rien à la situation. En effet, l'échange d'informations par crossover entre des individus strictement identiques est bien sûr totalement sans conséquences;

on aura beau choisir la méthode de croisement qu'on veut on se retrouvera toujours à échanger des portions de chromosomes identiques et la population n'évoluera pas. L'évolution se retrouvant bloquée on n'attendra jamais l'optimum global.

La mutation entrainant des inversions de bits de manière aléatoire permet de réintroduire des différences entre les individus et donc de nous extirper de cette situation.

Il est quand même utile de garder à l'esprit que ceci n'est pas une solution "miracle" et qu'il est bien entendu plus intelligent de ne pas utiliser de méthodes de sélection connues pour entrainer ce type de problème.

• La mutation permet d'atteindre la propriété d'***ergodicité***.

L'ergodicité est une propriété garantissant que chaque point de l'espace de recherche puisse être atteint.

En effet, une mutation pouvant intervenir de manière aléatoire au niveau de n'importe quel locus, on a la certitude mathématique que n'importe quel permutation de notre chaîne de bits peut apparaître au sein de la population et donc que tout point de l'espace de recherche peut être atteint. Grâce à cette propriété on est donc sûr de pouvoir atteindre l'optimum global. On notera que la mutation règle donc le problème exposé à la fin du Section sur le croisement.

IV.1.5 L'opérateur de remplacement

Cet opérateur est le plus simple, son travail consiste à réintroduire les descendants obtenus par application successive des opérateurs de sélection, de croisement et de mutation (la population P') dans la population de leurs parents (la population P). Ce faisant il vont remplacer une certaine proportion de ceux-ci, proportion pouvant bien sûr être choisie. Le rapport entre le nombre d'individus nouveaux allant être introduits dans la population P et le nombre d'individus de cette population est connu sous le nom de ***génération gap***.

On trouve essentiellement deux méthodes de remplacement différentes :

• Le ***remplacement stationnaire :*** dans ce cas, les enfants remplacent automatiquement les parents sans tenir compte de leurs performances respectives, et le nombre d'individus de la population ne varie pas tout au long du cycle d'évolution simulé, ce qui implique donc d'initialiser la population initiale avec un nombre suffisant d'individus. Cette méthode peut être mise en œuvre de deux façons différentes :

• La première se contente de remplacer la totalité de la population P par la population P', cette méthode est connue sous le nom de ***remplacement générationnel*** et on a donc une génération gap qui vaut 1.

• La deuxième méthode consiste à choisir une certaine proportion d'individus de P' qui remplaceront leurs parents dans P (proportion égale à 100 % dans le cas du remplacement générationnel. Ce type de remplacement engendre une population ayant une grande variation et de se fait favorise la dérive génétique qui se manifeste d'autant plus que la population est de petite taille.

De plus dans bien des cas, étant donné que même un enfant ayant une faible performance remplace forcement un parent, on n'atteint pas la meilleure solution mais on s'en approche seulement.

• Le ***remplacement élitiste*** : dans ce cas, on garde au moins l'individu possédant les meilleures performances d'une génération à la suivante. En général, on peut partir du principe qu'un nouvel individu (enfant) prend place au sein de la population que s'il remplit le critère d'être plus performant que le moins performant des individus de la population précédente. Donc les enfants d'une génération ne remplaceront pas nécessairement leurs parents comme dans le remplacement stationnaire et par la même la taille de la population n'est pas figée au cours du temps.

Ce type de stratégie améliore les performances des algorithmes évolutionnaire dans certains cas. Mais présente aussi un désavantage en augmentant le taux de convergence prématuré.

Néanmoins, des implémentations plus fines procèdent de manière différente. Dans ce cas là, le taux de remplacement n'est pas de 100 %, la taille de la population augmente donc au cours des générations successives, on dit qu'il y a ***overcrowding.*** Il faut donc trouver un moyen pour sélectionner les parents qui seront supprimés, qui vont mourir. De Jong a proposé la solution suivante : imaginons qu'on veuille remplacer 30 % des parents, soit *np* le nombre de parents correspondants à ce pourcentage, on remplacera les *np* parents les plus proches de leurs descendants de P'. Cette méthode permet donc premièrement de maintenir la diversité et deuxièmement d'améliorer la fitness globale de la population.

Exemple :trouver le maximum de la fonction $f(x) = x$ sur l'intervalle $[0, 31]$ où x est un entier naturel.

On a 32 valeurs possibles pour *x* on choisit donc un codage discret sur 5 bits : on obtient donc la séquence 0,1,1,0,1 pour 13, la séquence 1,1,0,0,1 pour 25, etc...

On initialise la population initiale de manière aléatoire et on fixe sa taille à 4 individus. On définit simplement la fitness comme étant la valeur de *x*, vu qu'on en cherche la valeur maximum sur l'intervalle [0, 31] plus la valeur de *x* sera élevée plus on se rapprochera du maximum de la fonction identité et donc plus la fitness sera grande. Soit la population initiale suivante :

Individu	Séquence	Fitness	% du total
1	01011	11	18.6
2	10011	19	32.2
3	00101	5	8.5
4	11000	24	40.7
Total		59	100

Figure 8: la population initiale

On opte pour une sélection par la méthode la loterie biaisée :

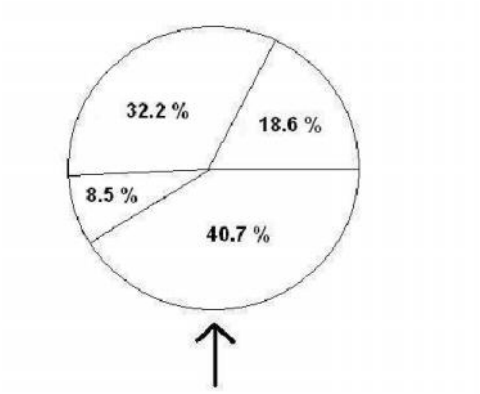


Figure 9: application de la méthode de sélection de la loterie biaisée sur la population

On fait tourner la roue 4 fois de suite, en général on fait tourner *n / 2* fois, soit 2 fois dans ce cas, mais le nombre 2 étant trop petit on décide de la faire tourner 4 fois. On obtient la nouvelle population :

Individu	Séquence
1	11000
2	00101
3	11000
4	01011

Figure 10: individus sélectionnés par la méthode de la loterie biaisée

On applique l'opérateur de croisement en utilisant un seul point de crossover. Normalement chaque couple donne 2 enfants qu'on ajoute à notre nouvelle population P' la faisant passer de *n / 2* individus à n individu, mais vu que dans le cas de notre exemple nous avons déjà atteint nos *n* individus, les 2 enfants du couple remplaceront leurs parents.

Deux couples sont formés de manière aléatoire :

• couple 1 : l'individu 2 avec l'individu 3

• couple 2 : l'individu 1 avec l'individu 4.

Les points de crossover sont eux aussi tiré au hasard. On obtient le résultat suivant :

Parents	Enfants
00101 01011	01011 00101
11000 01011	01011 11000

Figure 11: résultat de l'application de l'opérateur de croisement avec un point de crossover sur les individus sélectionnés par la loterie biaisée

On applique l'opérateur de mutation qui choisit de manière aléatoire si on doit faire une mutation et sur quel locus la faire :

Chromosome avant mutation	Chromosome après mutation
01011	11011
00101	00101
01011	01111
11000	11000

Figure 12: résultat de l'application de l'opérateur de mutation sur les individus engendrés par croisement

Puis on applique l'opérateur de remplacement qui décide de remplacer 100% de la population P, la population P est donc entièrement remplacée par P' et sa taille reste fixe :

Individu	Séquence	Fitness	% du total
1	11011	27	38
2	00101	5	7
3	01111	15	21.1
4	11000	24	33.8
Total		71	100

Figure 13: la nouvelle population après application des différents opérateurs

En une seule génération le maximum est passé de 24 à 27, et la fitness globale de la population a relativement augmentée pour passer de 59 à 71. On s'arrête ici pour cet exemple mais dans la réalité on continuera à engendrer des générations successives jusqu'à obtenir le maximum global : 31.

Le seul intérêt de cet exemple était de montrer une trace simple, sur une génération, de l'application successive des différents opérateurs. Il est bien sûr évident qu'un tel "problème" ne nécessitait pas l'utilisation d'algorithmes génétiques.

Dans la partie suivante nous allons exposer différents problèmes, problèmes considérés comme difficiles, nécessitant un temps de calcul important avec une approche algorithmique classique, et exposer des mises en œuvre possibles à l'aide d'algorithmes génétiques. On montrera en quoi les algorithmes génétiques améliorent la rapidité de résolution, ou carrément permettent une résolution qui n'aurait pas été possible autrement vu la taille des entrées du problème et sa complexité en temps.

IV.2 La méthode d'essaim particulaire

L'optimisation par essaim de particules repose sur un ensemble d'individus originellement disposés de façon aléatoire et homogène, que nous appellerons dès lors des particules, qui se déplacent dans l'hyper-espace de recherche et constituent, chacune, une solution potentielle. Chaque particule dispose d'une mémoire concernant sa meilleure solution visitée ainsi que la capacité de communiquer avec les particules constituant son entourage. À partir de ces informations, la particule va suivre une tendance faite, d'une part, de sa volonté à retourner vers sa solution optimale, et d'autre part, de son mimétisme

par rapport aux solutions trouvées dans son voisinage. A partir d'optimums locaux et empiriques, l'ensemble des particules va, normalement, converger vers la solution optimale globale du problème traité.

a) Formalisation Un essaim de particule est caractérisé par :

a) le nombre de particules de l'essaim, noté *nb* (3.3.1) ;

b) la vitesse maximale d'une particule, notée $\overline{v}_{max}$ (3.3.4) ;

c) la topologie et la taille du voisinage d'une particule qui définissent son réseau social (3.3.2).

d) l'inertie d'une particule, notée Ψ (3.3.5) ;

e) les coefficients de confiance, notés $\rho 1$ et $\rho 2$, qui pondèrent le comportement conservateur (ie. la tendance à retourner vers la meilleure solution visitée) et le panurgisme (ie. la tendance à suivre le voisinage) (3.3.3).

Une particule est caractérisée, à l'instant t, par :

• $\vec{x}_i(t)$: Sa position dans l'espace de recherche ;

• $\vec{v}_i(t)$: Sa vitesse ;

• $\vec{x}_{pbest_i}$: La position de la meilleure solution par laquelle elle est passée ;

• $\vec{x}_{vbest_i}$: La position de la meilleure solution connue de son voisinage ;

• $pbest_i$: La valeur de fitness de sa meilleure solution ;

• $vbest_i$: La valeur de fitness de la meilleure solution connu du voisinage ;

IV.2 .1 Configuration de la méthode

IV.2.1.1 Nombre de particules

La quantité de particules allouées à la résolution du problème dépend essentiellement de deux paramètres : la taille de l'espace de recherche et le rapport entre les capacités de calcul de la machine et le temps maximum de recherche. Il n'y a pas de règle pour déterminer ce paramètre, faire de nombreux essais permet de se doter de l'expérience nécessaire à l'appréhension de ce paramètre.

IV.2.1.2 Topologie du voisinage

La topologie du voisinage défini avec qui chacune des particules va pouvoir communiquer. Il existe de nombreuses combinaisons dont les suivantes sont les plus utilisées :

a) topologie en étoile : chaque particule est reliée à toutes les autres, ie. L'optimum du voisinage est l'optimum global ;

b) topologie en anneau : chaque particule est reliée à n particules (en général, n = 3), c'est la topologie la plus utilisée ;

c) topologie en rayon : les particules ne communiquent qu'avec une seule particule centrale ;

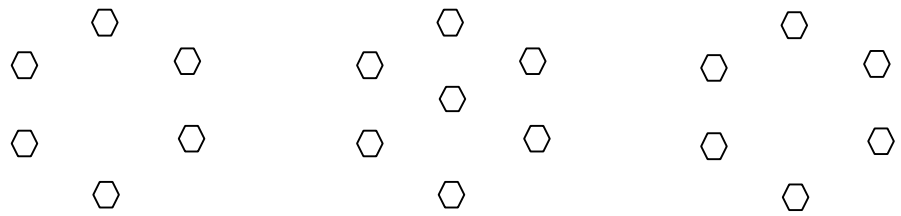


Figure 14 – (a) anneau (avec n = 2), (b) rayon, (c) étoile.

Le voisinage géographique auquel nous sommes amenés à penser en premier lieu n'est pas nécessairement pertinent car, d'une part, il s'agirait d'un voisinage trop local, et d'autre part car la sociabilisation des particules tend à rendre tout voisinage social en voisinage géographique. Enfin, c'est un voisinage très lourd en terme de calculs car nécessitant de recalculer le voisinage de chaque particule à chaque itération.

IV.2.1.3 Coefficients de confiance

Les variables de confiance pondèrent les tendances de la particule à vouloir suivre son instinct de conservation ou son panurgisme. Les variables aléatoires r1 et r2 peuvent être définis de la façon suivante :

$$\rho 1 = r1.c1$$
$$\rho 2 = r2.c2$$

où r1 et r2 suivent une loi uniforme sur [0.1] et c1 et c2 sont des constantes positives déterminées de façon empirique et suivant la relation $c1 + c2 \leq 4$.

IV.2.1.4 Vitesse maximale et coefficient de constriction

Afin d'éviter que les particules ne se déplacent trop rapidement dans l'espace de recherche, passant éventuellement à côté de l'optimum, il peut être nécessaire de fixer une vitesse maximale (notée $\vec{V}\,max$) pour améliorer la convergence de l'algorithme. Cependant, on peut s'en passer si on utilise un coefficient de constriction k « introduit par Maurice CLERC» et qui permet de resserrer l'hyper-espace de recherche.

L'équation de la vitesse devient alors :

$$k = 1 - \frac{1}{\rho} + \frac{\sqrt{|\rho^2 - 4\rho|}}{2}$$

avec $\rho = \rho_1 + \rho_2 > 4$.

$$\vec{v}_i\ t\ = k.\vec{v}_i\ t-1\ + \rho_1\ \vec{x}_{pbest_i} - \vec{x}_i\ t\ + \rho_2\ \vec{x}_{vbest_i} - \vec{x}_i\ t$$

Les études de SHI et EBERHART indiquent que l'utilisation d'un coefficient de constriction donne généralement un meilleur taux de convergence sans avoir à fixer de vitesse maximale. Cependant, dans certains cas, le coefficient de constriction seul ne permet pas la convergence vers la solution optimale pour un nombre d'itérations donné. Pour résoudre ce problème, il peut être intéressant de fixer $\vec{v}_{max} = \vec{x}_{max}$ en plus du coefficient de constriction, ce qui, selon les études de SHI et EBERHART, permet d'améliorer les performances globales de l'algorithme.

IV.2.1.5 Facteur d'inertie

Le facteur d'intertie ψ introduit par SHI et EBERHART permet de définir la capacité d'exploration de chaque particule en vue d'améliorer la converge de la méthode.

Une grande valeur de ψ (> 1) est synonyme d'une grande amplitude de mouvement et donc, in fine, d'exploration globale. A contrario, une faible valeur de ψ (< 1) est synonyme de faible amplitude de mouvement et donc, d'exploration locale. Fixer ce facteur, revient donc à trouver un compromis entre l'exploration locale et l'exploration globale.

Le calcul de la vitesse est alors défini par :

$$\vec{v}_i\ t\ = \psi.\vec{v}_i\ t-1\ + \rho_1\ \vec{x}_{pbest_i} - \vec{x}_i\ t\ + \rho_2\ \vec{x}_{vbest_i} - \vec{x}_i\ t$$

La taille du facteur d'inertie influence directement la taille de l'hyper-espace exploré et aucune valeur de ψ ne peut garantir la convergence vers la solution optimale.

Les études menées par SHI et EBERHART indiquent une meilleure convergences pour ψ □ [0.8,1.2]. Au delà de 1.2, l'algorithme tend à avoir certaines difficultés à converger.

Enfin, il est également possible de faire diminuer le facteur d'inertie au cours du temps, un peu à la manière de la température dans un algorithme de recuit simulé (Simulated Annealing). De bons résultats ont été trouvés pour une valeur décroissant linéairement de 0.9 à 0.4. Pour de plus amples informations sur le réglage de ce paramètre, veuillez vous référer à la thèse de VAN DEN BERGH.

IV.2.1.6 Initialisation de l'essaim

La position des particules ainsi que leur vitesse initiale doivent être initialisés aléatoirement selon une loi uniforme sur [0,1]. Cependant, en ce qui concerne la position des particules, il est préférable d'utiliser un générateur de séquence de SOBOL qui est plus pertinent dans la disposition homogène des particules dans un espace de dimension n.

IV.2.1.7 Critères d'arrêt

Comme indiqué précédemment, la convergence vers la solution optimale globale n'est pas garantie dans tous les cas de figure même si les expériences dénotent la grande performance de la méthode. De ce fait, il est fortement conseillé de doté l'algorithme d'une porte de sortie en définissant un nombre maximum d'itération (que nous noterons $nbIter_{max}$).

L'algorithme doit alors s'exécuter tant que l'un des critères de convergence suivant n'a pas été atteint :

• $nbIter_{max}$ a été atteint ;

• la variation de la vitesse est proche de 0 ;

• le *fitness* de la solution est suffisant.

IV.2.2 Algorithme de synthèse

Algorithme 2 version simpliste (avec voisinage)

Répéter

pour *i = 1* **jusqu'à** *nb* **faire**

si $F(\vec{x}_i) > pbest_i$ **alors**

$pbesti = F(\vec{x}_i)$

$\vec{x}_{pbest_i} = \vec{x}_i$

fin si

si $F(\vec{x}_i) > vbest_i$ **alors**

$vbest_i = F(\vec{x}_i)$

$\vec{x}_{vbest_i} = \vec{x}_i$

fin si

fin pour

pour *i = 1* **to** *nb* **faire**

$\vec{v}_i = k(\vec{v}_i + \rho_1(\vec{x}_{pbest_i} - \vec{x}_i) + \rho_2(\vec{x}_{vbest_i} - \vec{x}_i)$

$\vec{x}_i = \vec{x}_i + \vec{v}_i$

fin pour

jusqu'à (un des critères de convergence est atteint)

Références bibliographiques:

1. *E. Aarts& J. Korst, Simulated annealing and Boltzmann machines : A stochastic approach to combinatorial optimization and neural computing. John Wiley & Sons, New-York, 1997.*
2. *D. Bertsekas, Nonlinear programming. Athena Scientific, Belmont, MA, 1999.*
3. *M. Bierlaire, Introduction à l'optimisation différentiable. Presses polytechniques et universitaires romandes, Lausanne, 2006.*
4. *F. Bonnans, Optimisation continue : cours et problèmes corrigés. Dunod, Paris, 2006.*
5. *F. Bonnans, J. C. Gilbert, C. Lemaréchal et C. Sagastizàbal, Optimisation numérique : aspects théoriques et pratiques. Springer, Berlin, 1997.*
6. *P. G. Ciarlet, Introduction à l'analyse numérique matricielle et à l'optimisation. Masson, Paris, 1994.*
7. *E. Chong et S. Zak, An introduction to optimisation. John Wiley & Sons, New-York, 1995.*
8. *Y. Colette et P. Siarry, Optimisation multiobjectif. Eyrolles, Paris, 2002.*
9. *J. C. Culioli, Introduction à l'optimisation. Ellipses, Paris, 1994.*
10. *J. Dennis & R. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations. Prentice Hall, Englewood Cliffs, NJ, 1983.*
11. *R. Fletcher, Practical methods of optimization. John Wiley & Sons, New-York, 1987.*
12. *P. Gill, W. Murray, & M. Wright, Practical optimization. Academic Press, New-York, 1987.*