MASTER EN SCIENCES MATHÉMATIQUES

Méthodes de moindres carrés linéaires avec contraintes linéaires. Application aux bilans cohérents

Ulrix, Laurence

Award date:
1987

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES N.D. DE LA PAIX
NAMUR
FACULTE DES SCIENCES

Méthodes de moindres carrés linéaires avec contraintes linéaires. Application aux bilans cohérents.

Promoteur: J.-J; STRODIOT
Directeur: P. MANNENBACK

Mémoire présenté pour l'obtention du grade de Licencié en Sciences mathématiques
par

LAURENCE ULRIX

Facultés Universitaires Notre-Dame de la Paix
Faculté des Sciences
rue de Bruxelles 61, B-5000 NAMUR
Tél. 081 - 22.90.61 Télex 59222 facnam-b Téléfax 081 - 23.03.91

Méthodes de moindres carrés linéaires avec contraintes linéaires. Application aux bilans cohérents.

ULRIX Laurence

Résumé.

Plusieurs méthodes de moindres carrés linéaires avec contraintes linéaires ont été étudiées.
L'une d'entre elles a été approfondie et transformée afin de l'appliquer à un problème de bilans cohérents.

Abstract

Some methods of linear least squares with linear constraints have been studied. One of them has been examined thoroughly and modified in order to apply it to a problem of mass balance.

Mémoire de licence en Sciences Mathématiques
Juin 1987
Promoteur: **J.-J. STRODIOT**
Directeur: **P. MANNEBACK**

Ce mémoire est le résultat d'une collaboration entre la Société SOLVAY & Cie et l'Unité d'Optimisation du Département de Mathématique des FNDP.

Je tiens à remercier Monsieur V.H. Nguyen et Monsieur J.-J. Strodiot de l'Unité d'Optimisation pour m'avoir guidée tout au long de ce travail ainsi que Monsieur Vergison et Monsieur P. Manneback de la Direction Technologies, Société SOLVAY & Cie pour leur disponibilité et les nombreux conseils qu'ils m'ont prodigués durant toute cette année.

Je remercie également H. Claes et J. Engels pour leur aide dans la partie informatique.

Mes remerciements vont également à mes parents qui m'ont permis de faire mes études et qui m'ont soutenue tout le long de ces quatre années.

Table des matières

1.INTRODUCTION

Les procédés d'industrie chimique constituent une partie importante des applications pour la détection d'erreurs et pour le rapprochement des données. En effet, ces installations peuvent être modélisées sous la forme d'un graphe. Les bilans s'intéressent aux flux entre les différentes unités des installations. La cohérence de bilans impose l'équation

$$\Sigma \text{ INPUTS} = \Sigma \text{ OUTPUTS}$$

en chaque nœud du graphe, mais cette contrainte est rarement vérifiée par les mesures observées. Nous voudrions donc trouver des valeurs théoriques se rapprochant le plus possible de ces valeurs observées.

Pour ce faire, nous allons étudier différentes méthodes de moindres carrés linéaires avec contraintes linéaires.

La première méthode est une résolution d'un problème de moindres carrés, linéaire avec contraintes linéaires où on élimine les équations afin d'avoir un problème sans contraintes (Lawson and Hanson,1974).

La deuxième méthode est un algorithme pour moindres carrés avec contraintes où on se base sur le problème précédent en pondérant certaines équations (Haskell and Hanson,1981)

La troisième méthode est une résolution d'un problème de moindres carrés linéaire avec bornes et contraintes linéaires, où on pondère également certaines contraintes. (Hanson,1986)

Ensuite, nous allons approfondir une quatrième méthode de factorisation pour résoudre le problème de moindres carrés linéaire avec contraintes (Stoer et Schittkowski,1979), afin de l'appliquer au problème posé ci-dessus.

Nous pourrions également étudier des méthodes détectant les "grosses" erreurs sur les données observées.

2.PARTIE THEORIQUE

2.1 PREREQUIS

2.1.1 Problème de moindres carrés linéaire sans contrainte

Le problème de moindres carrés linéaire sans contrainte revient à trouver un $\mathbf{x} \in IR^n$ de telle façon que $\mathbf{Ax}$ est la "meilleure" approximation du vecteur $\mathbf{b} \in IR^m$, où $\mathbf{A} \in IR^{mxn}$

La "meilleure" solution est définie dans ce cas-ci, par

$$\min \| \mathbf{Ax-b} \|_2 \quad , \mathbf{A} \in IR^{mxn} \quad , \mathbf{b} \in IR^n \tag{2.1}$$

$\mathbf{x}$ est appelée la solution au sens de moindres carrés du système $\mathbf{Ax = b}$

Ce système est noté $\mathbf{Ax} \equiv \mathbf{b}$ quand on doit trouver la solution au sens des moindres carrés.

L'ensemble des solutions du système $\mathbf{Ax = b}$ est défini par

$$X = \{x \in IR^n \text{ t.q. } \| \mathbf{Ax-b} \|_2 = \min \} \tag{2.2}$$

et est caractérisé par le théorème suivant :

Théorème 2.1:

$$\mathbf{x} \in X \iff A^T(\mathbf{b-Ax}) = \mathbf{0}$$

on appelle $\mathbf{r = b-Ax}$ le vecteur résidu de $\mathbf{x}$ .Le théorème 1 affirme que le vecteur résidu de la solution de $\mathbf{Ax-b}$ est orthogonal au sous-espace

$$R(A) = \{ \mathbf{Ax} \text{ t.q. } \mathbf{x} \in IR^n \} \tag{2.3}$$

Le vecteur **b** peut donc être décomposé en 2 parties orthogonales:

$$\mathbf{b}=\mathbf{Ax}+\mathbf{r} \quad , \quad \mathbf{r} \text{ perpendiculaire à } \mathbf{Ax} \tag{2.4}$$

Du théorème 2.1, il suit également, que la solution au sens de moindres carrés satisfait les équations normales

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b} \tag{2.5}$$

La matrice $\mathbf{A}^T\mathbf{A}$ est symétrique et définie non négative.
De plus, si **A** est non singulière, **x** est une solution unique et de (2.5), il suit que

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{2.6}$$

Pour plus de précision voir (Björch 1986, chap.I), (Lawson and Hanson, 1977).

2.1.2 Résolution du problème par la décomposition QR.

Le problème de moindres carrés peut être résolu par décomposition orthogonale de la matrice $\mathbf{A} \in \mathrm{IR}^{mxn}$ de la forme $\mathbf{HRK}^T$ où **H** et **K** sont des matrices orthogonales (Lawson and Hanson, 1974 chap.2)
Par propriété des matrices orthogonales, on a que

$$\| \mathbf{Ax-b} \| = \|\mathbf{Q(Ax-b)}\| = \| \mathbf{QAx-Qb} \| \tag{2.7}$$

pour toute matrice orthogonale **Q** (mxm)

Avec les transformations orthogonales, on peut écrire la solution au sens des moindres carrés sous la forme suivante:

Théorème 2.2 :

Supposons que la matrice $\mathbf{A}$ (mxn) est de rang k et que $\mathbf{A} = \mathbf{HRK}^T$

ou $\mathbf{H}$ est une matrice orthogonale (mxn)

$\mathbf{K}$ est une matrice orthogonale (nxn)

$\mathbf{R}$ est une matrice (mxn) de la forme

$$\begin{pmatrix} \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

où $\mathbf{R}_{11}$ est une matrice (kxk) de rang k

Définissons le vecteur

$$\mathbf{H}^T\mathbf{b} = \mathbf{g} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{pmatrix} \begin{matrix} \}k \\ \}m-k \end{matrix}$$

Introduisons les nouvelles variables

$$\mathbf{K}^T\mathbf{x} = \mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \begin{matrix} \}k \\ \}n-k \end{matrix}$$

Définissons $\mathbf{y}_1^*$ comme étant la solution unique de

$$\mathbf{R}_{11}\mathbf{y}_1 = \mathbf{g}_1$$

Alors 1) Toute solution du problème minimiser $\| \mathbf{Ax-b} \|$ est de la forme

$$\mathbf{x}^* = \mathbf{K}\begin{pmatrix} \mathbf{y}_1^* \\ \mathbf{y}_2 \end{pmatrix}$$

où $\mathbf{y}_2$ est arbitraire

2) Chaque $\mathbf{x}^*$ a un vecteur résidu $\mathbf{r}$ satisfaisant

$$\mathbf{r} = \mathbf{b-Ax}^* = \mathbf{H}\begin{pmatrix} \mathbf{0} \\ \mathbf{g}_2 \end{pmatrix}$$

3) La norme de $\mathbf{r}$ satisfait $\| \mathbf{r} \| = \| \mathbf{b-Ax}^* \| = \| \mathbf{g}_2 \|$

4) La solution unique de longueur minimale est $\mathbf{x}^* = \mathbf{K}\begin{pmatrix} \mathbf{y}_1^* \\ \mathbf{0} \end{pmatrix}$

Deux transformations orthogonales élémentaires sont présentées dans les deux lemmes suivants : (Lawson and Hanson 1976 chap. 3)

Lemme 2.1

Etant donné un vecteur $\mathbf{v}$ de dimension m qui n'est pas le vecteur nul, il existe une matrice orthogonale $\mathbf{Q}$ t q

$\mathbf{Qv} = -\sigma \| \mathbf{v} \|_2$

où $\sigma = \begin{cases} +1 & \text{si } v_1 \geq 0 \\ -1 & \text{si } v_1 < 0 \end{cases}$

La matrice $\mathbf{Q}$ est appelée matrice de transformation de Householder. Cette transformation peut être vue géométriquement comme un réflecteur dans le sous-espace, S, orthogonal au vecteur **u**.

Dans le cas spécial où l'on veut mettre à 0 un seul élément du vecteur **v**, on emploie les transformations de Givens, définies par le lemme suivant :

Comme ces transformations ne modifient que deux composantes du vecteur, on peut discuter les opérations sur un vecteur de dimension 2.

Lemme 2.2

Etant donné un vecteur de dimension 2 $\mathbf{v} = (v_1, v_2)^T$ (avec $v_1 \neq 0$ ou $v_2 \neq 0$), il existe une matrice orthogonale G

$$\mathbf{G} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

avec $c^2+s^2 = 1$

tel que $\mathbf{Gv} = \begin{pmatrix} (v_1^2+v_2^2)^{1/2} \\ 0 \end{pmatrix}$

Théorème 2.3 :

Soit $\mathbf{A}$ une matrice (mxn)

Il existe une matrice orthogonele $\mathbf{Q}$ (mxm),

t.q. $\mathbf{A} = \mathbf{QR}$ et $\mathbf{R}$ est nulle en dessous de la diagonale principale

Cette décomposition comme produit d'une matrice orthogonale et d'une matrice triangulaire supérieure est appelée décomposition QR.

Suivant les formes de **A**, on a différentes formes de **R** .

1°) si m = n et **A** de rang n:

$$\mathbf{A} = \mathbf{QR} \qquad \mathbf{Q}(nxn), \quad \mathbf{R}(nxn)$$

2°) si m = n et **A** de rang k

$$\mathbf{A} = \mathbf{Q}\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \qquad \mathbf{Q}(nxn), \quad \mathbf{R}(kxk)$$

3°) si m < n

$$\mathbf{A} = \mathbf{Q}^T (\mathbf{R} : \mathbf{M}) \qquad \text{où } \mathbf{M} \text{ est } ((n-m)x(n-m))$$

$\mathbf{R}$ est (mxm)

4°) Si m > n

$$\mathbf{A} = \mathbf{Q}^T\begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}$$

Dans les cas 1 et 4, la décomposition, $\mathbf{A} = \mathbf{QRI}_n$ satisfait les conditions du théorème 2.2.

Dans le cas 3, on applique la théorème 2.2 à la décomposition

$$\mathbf{A}^T = \mathbf{I}_m\mathbf{R}^T\mathbf{Q}^T$$

Si le rang k de **A** est déficient, on applique des permutations par la droite (Lawson and Hanson 1974 chap.3)

Voir aussi (Gill, Murray and Wright, 1981)

2.1.3 Résolution du problème par décomposition en valeurs singulières

Théorème 2.4

Soit **A** une matrice (mxn) de rang k

Alors, il existe une matrice orthogonale **U**(mxm), une matrice orthogonale **V**(nxn) et une matrice diagonale **S** t.q.

$\mathbf{U}^T\mathbf{A}\mathbf{V} = \mathbf{S}$ ou $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$

Ici, la diagonale de **S** peut être arrangée en ordre décroissant. Les éléments diagonaux sont tous différents de 0 et k d'entre eux sont strictement positifs.

Les éléments diagonaux de **S** sont appelés les valeurs singulières de **A**
Ce sont les valeurs propres de la matrice $\mathbf{A}^T\mathbf{A}$
La décomposition en valeurs singulières satisfait le théorème 2.2. (Lawson and Hanson, 1974, chap.4)

2.1.4 Problème de programmation quadratique

La forme générale d'un problème de programmation quadratique avec contraintes d'inégalité est :
QP (quadratic Programming)

$$\begin{cases} \min_{\mathbf{x} \in IR^n} & \mathbf{c}^T\mathbf{x}+1/2\mathbf{x}^T\mathbf{G}\mathbf{x} \\ & \mathbf{A}\mathbf{x} \geq \mathbf{b} \end{cases}$$

Pour une matrice constante **G** est un vecteur constant **c**

La plupart des algorithmes pour résoudre ce problème sont des méthodes d'ensemble actif.

Le problème QP est réduit temporairement à un sous-problème où les contraintes sont d'égalité.

On cherche alors une direction de descente en résolvant les équations de Newton. Etant donné la nature de la fonction objectif, il n'y a que deux choix possibles pour la longueur du pas dans cette direction.

Le pas unitaire donne le minimun du problème restreint aux contraintes actives. Si on n'a pas rencontré de contraintres dans cette direction, on peut alors calculer les multiplicateurs de Lagrange afin de déterminer si une contrainte doit être enlevée de l'ensemble actif ou si la solution du sous-problème est la solution du problème QP

Par contre, si on rencontre une contrainte dans la direction de descente, on rajoute cette contrainte à l'ensemble actif. Ensuite, on recommence avec le nouveau sous-probème correspondant au nouvel ensemble actif. On obtient de cette façon une suite de points admissibles décroissante qui converge vers le minimun.

La solution d'une minimisation de fonction quadratique avec contraintes linéaires d'inégalité est caractérisée par les conditions de Kuhn -Tucker suivantes (2.8):

1. $\mathbf{x}^*$ est admissible pour le problème (QP) et le sous-problème i.e. $\mathbf{A}\mathbf{x}^* \geq \mathbf{b}$ et $\mathbf{x}^*$ satisfait les contraintes actives comme égalité $\mathbf{A}'\mathbf{x}^* = \mathbf{b}'$
2. le gradient de f : $\mathbf{c}+\mathbf{G}\mathbf{x}$ est t.q. $\mathbf{c}+\mathbf{G}\mathbf{x}+\mathbf{A}'^{T}\lambda = \mathbf{0}$
   où λ est le vecteur des multiplicateurs de Lagrange associé au sous-problème. Ce qui est équivalent à dire que le gradient réduit $\mathbf{Z}^T(\mathbf{c}+\mathbf{G}\mathbf{x}) = \mathbf{0}$
3. le hessien réduit $\mathbf{Z}^T\mathbf{G}\mathbf{Z}$ est semi défini positif.

Pour plus de précision voir (Gill and Murray, 1981)

2.2 METHODES EXISTANTES

2.2.1 Problème de moindres carrés avec contraintes linéaires (Hanson and Lawson, 1974)

Soit **E** : une matrice $m_2$xn , **f** : un vecteur de dimension m_2, **G** : une matrice (mxn) et **h** un vecteur de dimension n.
Le problème des moindres carrés avec des contraintes d'inégalité linéaires peut être posé comme suit:
Problème LSI (Least Squares with Inequality Constraints)

$$\begin{cases} \min \ \| \mathbf{Ex-f} \| \\ \text{s.c.} \quad \mathbf{Gx} \geq \mathbf{h} \end{cases} \tag{2.9}$$

Les cas spéciaux importants suivants seront traités en premier lieu:
Problème NNLS (Nonnegative Least Squares)

$$\begin{cases} \min \ \| \mathbf{Ex-f} \| \\ \text{s.c.} \quad \mathbf{x} \geq \mathbf{0} \end{cases} \tag{2.10}$$

Problème LDP (Least Distance Programming)

$$\begin{cases} \min \ \| \mathbf{x} \| \\ \text{s.c.} \quad \mathbf{Gx} \geq \mathbf{h} \end{cases} \tag{2.11}$$

Ensuite le problème général LSI sera remplacé par un problème LDP, et nous verrons par après comment rajouter des contraintes d'égalité.

2.2.1.1 Problème NNLS

Le problème NNLS est défini par (2.10);
L'utilisateur donne la matrice **E** ($m_2$xn), les entiers m_2 et n, et le vecteur **f** de dimension m_2.

Les vecteurs **w** et **z** de dimension n sont des tableaux de travail.
Les ensembles d'indices $\mathcal{P}$ et $\mathcal{Z}$ seront définis et modifiés au cours de l'exécution de l'algorithme. Les variables dont l'indice se trouve dans $\mathcal{Z}$ seront mises à 0. Les variables dont l'indice se trouve dans $\mathcal{P}$ seront libres de prendre des valeurs différentes de 0.
Si une telle variable prend une valeur négative, l'algorithme mettra cette variable soit à une valeur positive, soit à la valeur 0. et, dans ce dernier cas, déplacera l'indice correspondant de l'ensemble $\mathcal{P}$ à l'ensemble $\mathcal{Z}$.

A la fin de l'algorithme, **x** sera la solution et **w** sera le vecteur dual.

Algorithme NNLS ($\mathbf{E}$, m_2 , n , $\mathbf{f}$, $\mathbf{x}$, $\mathbf{w}$, $\mathbf{z}$, $\mathcal{P}$, $\mathcal{Z}$)

1. Poser $\mathcal{P} = \varnothing$, $\mathcal{Z} = \{1, 2, \ldots, n\}$ et $\mathbf{x} = \mathbf{0}$
2. Calculer le vecteur $\mathbf{w} = \mathbf{E}^T(\mathbf{f}-\mathbf{E}\mathbf{x})$
3. Si l' ensemble $\mathcal{Z}$ est vide ou si $w_j \leq 0 \quad \forall j \in \mathcal{Z}$ aller en 12
4. Trouver un indice $t \in \mathcal{Z}$ t.q. $w_t = \max \{w_j , j \in \mathcal{Z}\}$
5. Déplacer l'indice t de l'ensemble $\mathcal{Z}$ à l'ensemble $\mathcal{P}$
6. Soit $\mathbf{E}_{\mathcal{P}}$, la matrice ($m_2$xn) définie par

   colonne j de $\mathbf{E}_{\mathcal{P}}$ = colonne j de $\mathbf{E}$ si $j \in \mathcal{P}$

   0 si $j \in \mathcal{Z}$

   Calculer le vecteur $\mathbf{z}$ comme étant la solution du problème des moindres carrés $\mathbf{E}_{\mathcal{P}}\mathbf{z} = \mathbf{f}$

   Remarquons que seules les composantes z_j, $j \in \mathcal{P}$

   sont déterminées par ce problème

   Posons $z_j = 0$, $j \in \mathcal{Z}$
7. Si $z_j > 0 \quad \forall j \in \mathcal{P}$, poser $\mathbf{x} = \mathbf{z}$ et aller en 2
8. Trouver un indice $q \in \mathcal{P}$

   t.q. $x_q/(x_q-z_q) = \min \{x_j/(x_j-z_j), z_j \leq 0, j \in \mathcal{P}\}$
9. Poser $\alpha = x_q/(x_q-z_q)$
10. Poser $\mathbf{x} = \mathbf{x}+\alpha(\mathbf{z}-\mathbf{x})$
11. Déplacer de l'ensemble $\mathcal{P}$ à l'ensemble $\mathcal{Z}$ tous les indices $j \in \mathcal{P}$

    t.q. $x_j = 0$ et aller en 6
12. Commentaire: l'algorithme est terminé.

A la fin de l'algorithme, le vecteur solution $\mathbf{x}$ satifait

$x_j > 0$, $j \in \mathcal{P}$

$x_j = 0$, $j \in \mathcal{Z}$

et est une solution du problème des moindres carrés $\mathbf{E}_{\mathcal{P}}\mathbf{z} = \mathbf{f}$

Le vecteur dual $\mathbf{w}$ satisfait

$$\mathbf{w}_j = 0, \quad j \in \mathcal{P}$$
$$\mathbf{w}_j \leq 0, \quad j \in \mathcal{Z}$$
$$\text{et} \quad \mathbf{w} = \mathbf{E}^T(\mathbf{f}-\mathbf{E}\mathbf{x})$$

Les conditions satisfaites par $\mathbf{x}$ et par $\mathbf{w}$ constituent les conditions du théorème de Khun-Tucker caractérisant une solution pour le problème NNLS. Avant de discuter la convergence de l'algorithme NNLS, il sera commode d'énoncer le lemme suivant:

Lemme 2.3

Soient $\mathbf{A}$ une matrice (mxn) de rang n et

$\mathbf{b}$ un vecteur de dimension m satisfaisant

$$\mathbf{A}^T\mathbf{b} = [\underbrace{0 \ldots 0}_{(n-1)} \ \underbrace{\omega}_{(1)}]^T$$

Si $\mathbf{x}^*$ est la solution au sens des moindres carrés de $\mathbf{Ax} \cong \mathbf{b}$, alors $\mathbf{x}^*_n > 0$ où $\mathbf{x}^*_n$ est la $n^{\text{ième}}$ composante de $\mathbf{x}^*$.

L'algorithme NNLS peut être regardé comme consistant en une boucle principale (a) et une boucle intérieure (b).

(b) consiste en les étapes 6-11 et a un seul point d'entrée à l'étape 6 et une seule sortie à l'étape 7.

(a) consiste en les étapes 2-5 et (b). On entre dans la boucle (a) à l'étape 2 et on en sort à l'étape 3.

A l'étape 2 de (a), l'ensemble $\mathcal{P}$ identifie les composantes du vecteur courant $\mathbf{x}$ qui sont positives. Les composantes de $\mathbf{x}$ dont l'indice est dans $\mathcal{Z}$ sont nulles à ce point.

Dans la boucle (a), l'indice t sélectionné à l'étape 4 donne un coefficient qui n'est pas encore dans $\mathcal{P}$ et qui deviendra positif (par le lemme 2.3) s'il est introduit dans la solution. Ce coefficient est amené dans la tentative de solution à l'étape 6 de la boucle (b).

Si toutes les autres composantes de $\mathbf{z}$ indicées dans l'ensemble $\mathcal{P}$ restent positives, alors, à l'étape 7, l'algorithme pose $\mathbf{x} = \mathbf{z}$ et retourne au début de la boucle (a).

Dans ce procédé, l'ensemble $\mathcal{P}$ est augmenté et l'ensemble $\mathcal{Z}$ est diminué par le transfert de l'indice t.

Dans beaucoup d'exemples, cette séquence est simplement répétée, avec l'addition d'un coefficient supplémentaire à chaque itération de (a) jusqu'à ce que le test de terminaison de l'étape 3 soit finalement satisfait.

Cependant, si un coefficient dont l'indice se trouve dans $\mathcal{P}$ devient nul ou négatif à létape 6, alors l'algoritme reste dans la boucle (b) et fait un mouve - ment qui remplace $\mathbf{x}$ par $\mathbf{x}+\alpha(\mathbf{z}-\mathbf{x})$, $0 < \alpha \leq 1$ où α est choisi le plus grand possible tout en gardant le nouvel $\mathbf{x}$ positif.

La boucle (b) est répétée jusqu'à ce que l'algorithme satisfasse les conditions de sortie de l'étape 7.

Le caractère fini de la boucle (b) peut être prouvé en montrant que toutes les opérations dans la boucle (b) sont bien définies, que au moins un indice, appelé q à ce point, est enlevé de $\mathcal{P}$ chaque fois que l'étape 11 est exécutée, et que $\mathbf{z}_t$ est toujours positif (par le lemme 2.3). Donc, sortir de la boucle (b) à l'étape 7 doit arriver après moins de π-1 itérations, où π dénote le nombres d'indices dans l'ensemble $\mathcal{P}$ au moment d'entrer dans (b).

En pratique, la boucle (b) se termine à l'étape 7 et atteint rarement les étapes 8-11.

Le caractère fini de la boucle (a) peut être prouvé en montrant que le résidu $\| \mathbf{f}-\mathbf{E}\mathbf{x} \|$ a une valeur strictement plus petite chaque fois que l'algorithme passe à l'étape 2, et donc, que le vecteur $\mathbf{x}$ et son ensemble associé $\mathcal{P} = \{i: \mathbf{x}_i > 0\}$ sont distincts de chaque $\mathbf{x}$ et $\mathcal{P}$ trouvés auparavant à cette même étape. Comme $\mathcal{P}$ est un sous-ensemble de $\{1, 2, \ldots, n\}$ et qu'il y a un nombre fini de tels sous-ensembles, la boucle (a) doit se terminer après un nombre fini d'iterations.

2.2.1.2 Problème LDP

Le vecteur solution du problème LDP peut être obtenu par une normalisation appropriée du vecteur résidu dans un problème NNLS s'y rapportant.

L'utilisateur doit donner la matrice $\mathbf{G}$ (mxn), les entiers m et n et le vecteur $\mathbf{h}$ de dimension m.

Si les inégalités $\mathbf{G}\mathbf{x} \geq \mathbf{h}$ sont compatibles, alors l'algorithme mettra la variable logique ϕ à vrai et calculera le vecteur $\mathbf{x}^*$ de norme minimale satisfaisant ces inégalités. Si les inégalités sont incompatibles, l'algorithme mettra ϕ à faux et aucune valeur ne sera assignée à $\mathbf{x}^*$

Les tableaux de travail dont l'algorithme a besoin ne sont pas explicitement indiqués dans la liste de paramètres

Algorithme LDP ($\mathbf{G}$, m, n, $\mathbf{h}$, $\mathbf{x}^*$, ϕ)

1. Définir la matrice $\mathbf{E}(n+1 \times m)$, et le vecteur $\mathbf{f}$ de dimension n+1, comme
   $\mathbf{E} = \begin{pmatrix} \mathbf{G}^T \\ \mathbf{h}^T \end{pmatrix}$ et $\mathbf{f} = (0 \ldots 0, 1)^T$
   Employer l'algorithme NNLS pour calculer un vecteur $\mathbf{u}^*$ de dimension m résolvant le problème NNLS
   $$\begin{cases} \min \| \mathbf{Eu-f} \| \\ \text{s.c.} \quad \mathbf{u} \geq \mathbf{0} \end{cases}$$
2. Calculer le vecteur $\mathbf{r} = \mathbf{Eu}^* - \mathbf{f}$
3. Si $\| \mathbf{r} \| = 0$, poser ϕ = faux et aller en 6
4. Poser ϕ = vrai
5. Pour $j = 1, \ldots, n$, calculer $\mathbf{x}^*_j = -\mathbf{r}_j / \mathbf{r}_{n-1}$
6. Fin de l'algorithme.

Premièrement, considérons le problème NNLS qui est résolu à l'étape 1 de l'algoriththme LDP. Le vecteur gradiant de la fonction objectif $1/2 \| \mathbf{Eu-f} \|^2$ a une solution $\mathbf{u}^*$ qui est $\mathbf{p} = \mathbf{E}^T\mathbf{r}$.
Des conditions de Kuhn-Tucker pour ce problème NNLS, il existe des ensembles d'indices disjoints $\mathcal{E}$ et $\mathcal{S}$ tel que

$$\mathcal{E} \cup \mathcal{S} = \{1, 2, \ldots, m\},$$
$$\mathbf{u}^*_i = 0 \text{ pour } i \in \mathcal{E}, \quad \mathbf{u}^*_i > 0 \text{ pour } i \in \mathcal{S},$$
et $\mathbf{p}_i \geq 0$ pour $i \in \mathcal{E}$, $\mathbf{p}_i = 0$ pour $i \in \mathcal{S}$.

En employant ceci, on obtient

$$\| \mathbf{r} \|^2 = \mathbf{r}^T\mathbf{r} = \mathbf{r}^T(\mathbf{Eu}-\mathbf{f}) = \mathbf{p}^T\mathbf{u}^* - \mathbf{r}_{n+1}$$
$$= -\mathbf{r}_{n+1}$$

Considérons le cas où $\| \mathbf{r} \| > 0$ à l'étape 3. Par ce qui est ci-dessus, cela implique que $\mathbf{r}_{n+1} < 0$, donc on peut diviser par $\mathbf{r}_{n+1}$ à l'étape 5.

Par les équations des étapes 2 et 5, on établit l'admissibilité de $\mathbf{x}^*$ comme suit:

$$\mathbf{0} \leq \mathbf{p} = \mathbf{E}^T\mathbf{r}$$
$$= (\mathbf{G} : \mathbf{h}) \begin{pmatrix} \mathbf{x}^* \\ -1 \end{pmatrix}$$
$$= (\mathbf{Gx}^* - \mathbf{h}) \| \mathbf{r} \|^2$$

$$\Rightarrow \quad \mathbf{Gx}^* \geq \mathbf{h}$$

Les lignes du système d'inégalité $\mathbf{Gx}^* \geq \mathbf{h}$ dont l'indice est dans $\mathcal{S}$ sont satisfaites comme égalité.

Le vecteur gradiant de la fonction objectif $1/2 \| \mathbf{x} \|^2$ du problème LDP est simplement $\mathbf{x}$.

Les conditions de Kuhn-Tucker sur $\mathbf{x}^*$ pour

$$\begin{cases} \min 1/2 \| \mathbf{x} \|^2 \\ \text{s.c. } \mathbf{Gx} \geq \mathbf{h} \end{cases}$$

demandent que le vecteur gradiant $\mathbf{x}^*$ doit être représentable comme une combinaison linéaire non négative des lignes de $\mathbf{G}$ dont l'indice est dans $\mathcal{S}$.

Des étapes 2 et 5 et de l'équation $\| \mathbf{r} \|^2 = - \mathbf{r}_{n+1}$, on a que

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{r}_1 \\ : \\ \mathbf{r}_n \end{bmatrix} (-\mathbf{r}_{n+1})^{-1} = \mathbf{G}^T(-\mathbf{r}_{n+1})^{-1}\mathbf{u}^* = \mathbf{G}^T\mathbf{u}^* \| \mathbf{r} \|^2$$

En remarquant les conditions de signes sur $\mathbf{u}^*$ ($\mathbf{u}^* \geq 0$) la preuve que $\mathbf{x}^*$ est une solution du problème NNLS est complète.

C'est aussi la seule solution car si $\mathbf{x}'$ est une solution différente, on a que $\| \mathbf{x}' \|$ et $\| \mathbf{x}^* \|$ et le vecteur $1/2(\mathbf{x}' + \mathbf{x}^*)$ serait une autre solution admissible ayant une norme strictement plus petite que celle de $\mathbf{x}^*$, ce qui contredit le fait que $\mathbf{x}^*$ est un vecteur admissible de norme minimale.

Maintenant, considérons le cas où $\| \mathbf{r} \| = 0$ à l'étape 3.

On doit montrer que les inégalités $\mathbf{Gx} \geq \mathbf{h}$ sont incompatibles

Supposons le contraire, i.e. qu'il existe un vecteur $\mathbf{x}'$ satisfaisant $\mathbf{Gx}' \geq \mathbf{h}$.

Définissons $\mathbf{q} = \mathbf{Gx}'-\mathbf{h} = (\mathbf{G} : \mathbf{h}) \begin{pmatrix} \mathbf{x}' \\ -1 \end{pmatrix} \geq \mathbf{0}$

Alors $0 = (\mathbf{x}'^T -1)\, \mathbf{r} = (\mathbf{x}'^T -1) \left(\begin{pmatrix} \mathbf{G}^T \\ \mathbf{h}^T \end{pmatrix} \mathbf{u}^* - \mathbf{f} \right) = \mathbf{q}^T\mathbf{u}^* + 1$

Cette dernière expression ne peut pas être nulle parce que $\mathbf{q} \geq \mathbf{0}$ et $\mathbf{u}^* \geq \mathbf{0}$.

De cette contradiction, on conclut que la condition $\| \mathbf{r} \| = 0$ implique l'incompatibilité du système $\mathbf{Gx} \geq \mathbf{h}$

Ceci termine la vérification mathématique de l'algorithme LDP.

2.2.1.3 Conversion du problème LSI. en un problème LDP.

Soit k , le rang de la matrice $\mathbf{E}(m_2 x n)$ du problème LSI.

On peut obtenir une décomposition orthogonale de la matrice $\mathbf{E}$:

$$\mathbf{E} = \mathbf{Q}\begin{pmatrix}\mathbf{R}\ \mathbf{0}\\ \mathbf{0}\ \mathbf{0}\end{pmatrix}\mathbf{K}^T = (\mathbf{Q}_1 : \mathbf{Q}_2)\begin{pmatrix}\mathbf{R}\ \mathbf{0}\\ \mathbf{0}\ \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{K}^T_1\\ \mathbf{K}^T_2\end{pmatrix} \quad \begin{matrix}\} k\\ \} n-k\end{matrix}$$

où $\mathbf{Q}$ est une matrice orthogonale $(m_2 x m_2)$, $\mathbf{K}$ est une matrice orthogonale (nxn) et $\mathbf{R}$ est une matrice non singulière (kxk)

Par après, $\mathbf{R}$ peut être obtenue sous une forme diagonale ou triangulaire.

Introduisons le changement orthogonal des variables : $\mathbf{x} = \mathbf{Ky}$.

La fonction objectif qui doit être minimisée dans le problème LSI peut être écrite comme

$$\phi(\mathbf{x}) = \| \mathbf{f}-\mathbf{Ex} \|^2 = \left\| \begin{pmatrix}\mathbf{Q}^T_1\,\mathbf{f}\\ \mathbf{Q}^T_2\,\mathbf{f}\end{pmatrix} - \begin{pmatrix}\mathbf{Ry}\\ \mathbf{0}\end{pmatrix} \right\|^2$$

$$= \| \mathbf{f}^*_1-\mathbf{Ry} \|^2 + \| \mathbf{f}^*_2 \|^2$$

où $\mathbf{f}^*_i = \mathbf{Q}^T_i\,\mathbf{f}$, i = 1, 2.

Avec un changement supplémentaire des variables :

$$\mathbf{z} = \mathbf{Ry} - \mathbf{f}^*_1$$

on peut écrire

$$\phi(\mathbf{x}) = \| \mathbf{z} \|^2 + \| \mathbf{f}^*_2 \|^2$$

Le problème initial LSI

$$\begin{cases} \min \ \| \mathbf{Ex-f} \| \\ \text{s.c.} \quad \mathbf{Gx} \geq \mathbf{h} \end{cases}$$

est donc équivalent, excepté pour la constante additive $\| \mathbf{f}^*_2 \|^2$ dans la fonction objectif, au problème LDP suivant :

$$\begin{cases} \min \ \| \mathbf{z} \| \\ \text{s.c.} \quad \mathbf{GK}_1\mathbf{R}^{-1}\mathbf{z} \geq \mathbf{h} - \mathbf{GK}_1\mathbf{R}^{-1}\mathbf{zf}^*_1 \end{cases}$$

Si un vecteur $\mathbf{z}^*$ est calculé comme étant une solution du problème LDP, alors une solution $\mathbf{x}^*$ du problème LSI peut être calculée avec les équations

$$\mathbf{z} = \mathbf{Ry} - \mathbf{f}^*_1$$

et $$\mathbf{x} = \mathbf{K}_1\mathbf{y}$$

La norme du vecteur résidu peut être calculée par l'équation

$$\phi(\mathbf{x}) = \| \mathbf{z} \|^2 + \| \mathbf{f}^*_2 \|^2$$

2.2.1.4 Le problème LSI avec des contraintes l'égalités

Considérons le problème LSI auquel on ajoute un système de contraintes d'égalité, soit $\mathbf{Cx} = \mathbf{d}$ où $\mathbf{C}$ est une matrice $(m_1 \times n)$ de rang $m_1 < n$.

Ces contraintes peuvent être éliminées de deux façons différentes :

1°) On peut introduire un changement orthogonal des variables

$$\mathbf{x} = \mathbf{K}\begin{pmatrix} \mathbf{y}_1 \} m_1 \\ \mathbf{y}_2 \} n-m_1 \end{pmatrix} \quad \text{où } \mathbf{K} \text{ triangularise } \mathbf{C} \text{ par la droite}$$

$$\begin{pmatrix} \mathbf{C} \\ \mathbf{E} \\ \mathbf{G} \end{pmatrix} \mathbf{K} = \begin{pmatrix} \mathbf{C}^*_1 & \mathbf{0} \\ \mathbf{E}^*_1 & \mathbf{E}^*_2 \\ \mathbf{G}^*_1 & \mathbf{G}^*_2 \end{pmatrix}$$

Alors $\mathbf{y}^*_1$ est la solution du système triangulaire inférieur

$$\mathbf{C}^*_1\mathbf{y}_1 = \mathbf{d}$$

et $\mathbf{y}^*_2$ est la solution du problème LSI suivant :

$$\begin{cases} \min \ \| \mathbf{E}^*_2\mathbf{y}_2 - (\mathbf{f} - \mathbf{E}^*_1\mathbf{y}^*_1) \| \\ \text{s.c.} \quad \mathbf{G}^*_2\mathbf{y}_2 \geq \mathbf{h}\mathbf{G}^*_1\mathbf{y}^*_1 \end{cases}$$

Par après, la solution $\mathbf{x}^*$ peut être calculée en employant l'équation

$$\mathbf{x} = \mathbf{K}\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix}$$

2°) On peut calculer $\mathbf{Q}_1$, $\mathbf{C}^*_1$, $\mathbf{C}^*_2$, $\mathbf{d}^*$, $\mathbf{E}^*_1$, $\mathbf{E}^*_2$, et $\mathbf{f}^*$ en employant les équations :

$$\mathbf{Q}_1(\mathbf{C}_1 : \mathbf{C}_2 : \mathbf{d}) = (\mathbf{C}^*_1 : \mathbf{C}^*_2 : \mathbf{d}^*)$$

$$\mathbf{E}^*_1\mathbf{C}^*_1 = \mathbf{E}_1$$

$$\mathbf{E}^*_2 = \mathbf{E}_2 - \mathbf{E}^*_1\mathbf{C}^*_2$$

et $\mathbf{f}^* = \mathbf{f} - \mathbf{E}^*_1\mathbf{d}^*$

En plus on résoud le système triangulaire supérieur :

$$\mathbf{G}^*_1\mathbf{C}^*_1 = \mathbf{G}_1$$

Alors $\mathbf{x}^*$ est la solution du problème LSI suivant :

$$\begin{cases} \min \ \| \mathbf{E}^*_2\mathbf{x}_2 - \mathbf{f}^* \| \\ \text{s.c.} \quad (\mathbf{G}_2 - \mathbf{G}^*_1\mathbf{C}^*_2)\mathbf{x}_2 \geq \mathbf{h} - \mathbf{G}^*_1\mathbf{d}^* \end{cases}$$

et $\mathbf{x}^*_1$ est la solution du système triangulaire supérieur

$\mathbf{C}^*_1 \mathbf{x}_1 = \mathbf{d}^* - \mathbf{C}^*_2 \mathbf{x}^*_2$

Et la solution $\mathbf{x}^* = \begin{pmatrix} \mathbf{x}^*_1 \\ \mathbf{x}^*_2 \end{pmatrix}$

On peut donc toujours se ramener à un problème LSI pour résoudre un problème de moindres carrés linéaire avec contraintes d'égalité et d'inégalité linéaires

2.2.2 **Un algorithme pour problèmes de moindres carrés linéaires avec contraintes d'égalité et de non négativité** (Haskell and Hanson,1981)

2.2.2.1 Introduction

La méthode employée pour résoudre le problème est basée sur le fait de combiner les contraintes d'égalités avec des équations de moindres carrés pondérées différentiellement pour former un système de moindres carrés augmenté.

Ce système de moindres carrés pondéré, qui est équivalent à une méthode de fonction de pénalité, est résolu avec des contraintes de non négativité sur certaines variables.

On présente, ici, un algorithme numériquement stable pour résoudre le problème de moindres carrés avec contraintes linéaires suivant :

NNLSE (nonnegative least square equations)

$$(2.12)\quad \begin{cases} \mathbf{Ex} = \mathbf{f} & \text{(équations devant être satisfaites exactement)} \\ \mathbf{Ax} \cong \mathbf{b} & \text{(équations devant être satisfaites au sens de moindres carrés)} \\ x_i \geq 0 & i = l+1, \ldots, n \quad , \; 0 \leq l \leq n \end{cases}$$

Les matrices réelles $\mathbf{E}$ et $\mathbf{A}$ sont de dimension respective $(m_E x n)$ et $(m_A x n)$. Les variables $x_1, \ldots, x_l$ sont libres.

Dans le cas où les équations $\mathbf{Ex} = \mathbf{f}$ sont incompatibles, on prend $\mathbf{x} = \mathbf{x}^+ + \mathbf{y}$ où $\mathbf{x}^+$ est la solution de longueur minimale des équations et $\mathbf{y}$ est solution de $\mathbf{Ey} = \mathbf{0}$.

$\mathbf{y}$ est choisi pour minimiser la longueur du vecteur résidu $\mathbf{b-Ax}$ sous contraintes $x_i \geq 0, \; i > l$.

Le problème qui est fréquemment vu dans la pratique est :
LSIE (least square inequality equations)

$$(2.13)\begin{cases} \mathbf{Ex} = \mathbf{f} \\ \mathbf{Ax} \cong \mathbf{b} \\ \mathbf{Gx} \geq \mathbf{h} \end{cases}$$ (contrainte d'inégalité que la solution doit satisfaire)

où **G** est une matrice réelle ($m_G$xn).

Si le problème (NNLSE) est résolu, alors le problème (LSEI) est résolu ou si NNLSE ne l'est pas, on peut montrer qu'il n'y a pas de solution pour (LSEI)
En effet, on peut réduire le problème (LSEI) de 3 façons différentes au problème (NNLSE).
1°) Il est bien connu qu'en introduisant des variables bidons (2.13) peut être converti en un système de la forme (2.12).
A cette fin, un vecteur **w** de dimension m_G de variables non négatives est introduit dans les contraintes d'inégalité de (2.13).

Avec les matrices et les vecteurs

$$\mathbf{E}^* = \begin{pmatrix} \overbrace{\mathbf{E}}^{n} & \overbrace{\mathbf{0}}^{m_G} \\ \mathbf{G} & -\mathbf{I} \end{pmatrix} \begin{matrix} \} \, m_E \\ \} \, m_G \end{matrix}$$

$$\mathbf{f}^* = \begin{pmatrix} \mathbf{f} \\ \mathbf{h} \end{pmatrix}$$

$$\mathbf{A}^* = \begin{pmatrix} \overbrace{\mathbf{A}}^{n} & \overbrace{\mathbf{0}}^{m_G} \end{pmatrix} \quad \} \, m_A$$

$$\mathbf{x}^* = \begin{pmatrix} \mathbf{x} \\ \mathbf{w} \end{pmatrix} \begin{matrix} \} \, n \\ \} \, m_G \end{matrix}$$

On obtient le système de moindres carrés

$$\begin{cases} \mathbf{E}^*\mathbf{x}^* = \mathbf{f}^* \\ \mathbf{A}^*\mathbf{x}^* \cong \mathbf{b} \\ \mathbf{w} \geq \mathbf{0} \end{cases} \qquad (2.14)$$

Le problème (2.14) est de type (NNLSE) , avec les paramètres (m_E, m_A, n, l) de (2.12) identifiés avec les dimensions (m_E+m_G, m_A, $n+m_G$, n).

Quand cette transformation est conceptuellement directe, les techniques qu'on présente ici sont souvent plus efficientes pour résoudre le problème LSEI car il revient souvent à résoudre un problème de taille plus petite.

2°) Le cas spécial important du problème LSEI avec $m_G \leq n$ inégalités peut être transformé plus efficacement en un problème NNLSE.

Le développement apparemment nouveau résulte en un problème qui a le même nombre de paramètres que le premier.

Le fait le plus important dans cette idée est sans doute que les transformations requises ne demandent pas d'espace de travail supplémrentaire. Comme avant, on introduit m_G variables bidons $\mathbf{0 \leq w = Gx-h.}$

Employant des matrices orthogonales $\mathbf{H}$ (m_Gxm_G) et $\mathbf{K}$ (nxn), on calcule une décomposition orthogonale de la matrice $\mathbf{G}$ ($m_G$xn) tel que

$$\mathbf{H^T G K} = \begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

où $\mathbf{T}$ (gxg) est une matrice triangulaire inférieure non singulière et g est le rang numérique de $\mathbf{G}$

Posons $\mathbf{x = Ky}$ et partitionnons le vecteur et la matrice:

$$\mathbf{y} = \begin{pmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{pmatrix} \quad \begin{matrix} \} \ g \\ \} \ n\text{-}g \end{matrix} \qquad \text{et} \quad \begin{matrix} g & m_G\text{-}g \\ \end{matrix} \quad \mathbf{H} = (\widetilde{\mathbf{H_1}} : \widetilde{\mathbf{H_2}})$$

L'équation $\mathbf{Gx-h = w}$ devient alors

$$\mathbf{T y_1 = H_1^T (w+H)}$$

et $$\mathbf{0 = H_2^T (w+H)}$$

En faisant la substitution, on arrive au problème de moindres carrés avec contraintes pour le vecteur $\mathbf{x}^* = (\mathbf{y_2}^T : \mathbf{w}^T)^T$ de dimension $n+m_G-g$ de la forme

$$\begin{cases} \mathbf{E}^*_1\mathbf{y}_2 + \mathbf{E}^*_2\mathbf{w} = \mathbf{f}^* \\ \mathbf{A}^*_1\mathbf{y}_2 + \mathbf{A}^*_2\mathbf{w} \cong \mathbf{b}^* \\ \mathbf{w} \leq \mathbf{0} \end{cases} \qquad (2.15)$$

Le problème (2.15) est de type (NNLSE) avec les paramètres (m_E, m_A, n, l) de (2.12) identifiés avec les dimensions (m_E+m_G-g, m_A, $n+m_G-g$, $n-g$) de (2.15).

3°) La troisième méthode pour transformer (2.13) en un problème du type NNLSE procède en trois étapes

Première étape: Elimination des contraintes d'égalité.

Pour éliminer les contraintes d'égalité, on calcule une décomposition orthogonale de la matrice des contraintes d'égalité, de rang e

$$\mathbf{H}^T\mathbf{E}\mathbf{K} = \begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

Ici $\mathbf{L}$ est une matrice triangulaire inférieure (exe)
$\mathbf{H}$ et $\mathbf{K}$ sont des matrices orthogonales respectivement (m_Exm_E) et (nxn).
Soient $\mathbf{x} = \mathbf{K}\mathbf{y}$, $\mathbf{g} = \mathbf{H}^T\mathbf{f}$ et les vecteurs partitionnés

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \quad \begin{matrix} \} \text{ e} \\ \} \text{ n-e} \end{matrix}$$

$$\mathbf{g} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{pmatrix} \quad \begin{matrix} \} \text{ e} \\ \} \text{ } m_E\text{-e} \end{matrix}$$

On calcule la solution $\mathbf{y}^*_1$ de l'inverse généralisé des contraintes d'égalité en résolvant $\mathbf{L}\mathbf{y}_1 = \mathbf{g}_1$.

Le vecteur non spécifié $\mathbf{y}_1$ est employé pour résoudre les équations de moindres carrés avec les contraintes d'inégalité restantes

Deuxième étape: Réduction des équations de moindres carrés avec contraintes d'inégalités

Suivant l'élimination des contraintes d'égalité, on obtient un problème de la forme suivante

$$\begin{cases} \mathbf{A}\mathbf{x} \cong \mathbf{b} \\ \mathbf{G}\mathbf{x} \geq \mathbf{h} \end{cases}$$

On calcule une décomposition orthogonale de la matrice des moindres carrés de rang r :

$$\mathbf{H}^T\mathbf{A}\mathbf{K} = \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

Ici, $\mathbf{R}$ est une matrice triangulaire supérieure (rxr).

$\mathbf{H}$ et $\mathbf{K}$ sont des matrices orthogonales respectivement (m_Axm_A) et (nxn)

Elles sont différentes de celles citées ci-dessus

Soient $\mathbf{x} = \mathbf{K}\mathbf{y}$, $\mathbf{g} = \mathbf{H}^T\mathbf{b}$ et les vecteurs partitionnés

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \quad \begin{matrix} \} \ \text{r} \\ \} \ \text{n-r} \end{matrix}$$

$$\mathbf{g} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{pmatrix} \quad \begin{matrix} \} \ \text{r} \\ \} \ m_A\text{-r} \end{matrix}$$

On fait encore un changement de variables

$$\mathbf{z}_1 = \mathbf{R}\mathbf{y}_1 - \mathbf{g}_1$$
$$\mathbf{z}_2 = \mathbf{y}_2$$

Sans contraintes d'inégalité, le choix $\mathbf{z}_1 = \mathbf{0}$, $\mathbf{z}_2$ arbitraire est possible.

Avec $\mathbf{x}^+ = \mathbf{A}^+\mathbf{b}$ (la solution de l'inverse généralisé sans contraintes ou de Moore-Penrose) et

$$\mathbf{G}^* = \mathbf{GK}\begin{pmatrix} \mathbf{R}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = (\mathbf{G}^*_1 : \mathbf{G}^*_2)$$

nous résolvons le problème équivalent suivant (LPDP):

$$\begin{cases} \min \ 1/2 \| \mathbf{z}_1 \|^2 \\ \text{s.c.} \quad \mathbf{G}^*\mathbf{z} \geq \mathbf{h} - \mathbf{G}\mathbf{x}^+ = \mathbf{h}^* \end{cases}$$

Troisième étape: Résoudre le problème LPDP dans sa forme duale.

Le problème LPDP est résolu en deux étapes, chacune d'elles demandant de résoudre un problème du type NNLSE

1°/ On détermine $\mathbf{z}_1$ en résolvant le problème de moindres carrés

$$\begin{cases} \min \begin{pmatrix} \mathbf{G}^*_1{}^T \\ \mathbf{h}^{*T} \end{pmatrix} \mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \text{s.c.} \quad \mathbf{G}^*_2{}^T \mathbf{p} = \mathbf{0} \\ \qquad \mathbf{p} \geq \mathbf{0} \end{cases}$$

pour une solution $\mathbf{p} = \mathbf{p}^*$

Dans le cas où le scalaire $s=1-\mathbf{h}^{*T}\mathbf{p}^* = 0$, les contraintes d'inégalités sont incompatibles.
Sinon, on peut montrer que $s > 0$ et que la solution est $\mathbf{z}^*_1 = \mathbf{G}^*_1{}^T (s^{-1}\mathbf{p}^*)$.

2°/ Nous déterminons la longueur minimale $\mathbf{z}_2$ de telle manière que les contraintes d'inégalités sont satisfaites, en prenant $\mathbf{z}_2$ qui résoud

$$\mathbf{G}_2\mathbf{z}_2 \geq \mathbf{h}-\mathbf{G}_1\mathbf{z}_1.$$

Ce problème est résolu en employant l'algorithme LDP
Les trois étapes de troisième méthode sont la base du programme LSEI ,pour résoudre le problème LSEI (2.13).
Le choix des décompositions orthogonales et la détermination du rang de $\mathbf{E}$ ne sont pas spécifiés.
La méthode qui suit est une modification de l'algorithme de Golub-Businger:
La modification revient à une stratégie de pivotage appliquée aux lignes de $\mathbf{E}$ qui ne dépend pas du scaling sur les lignes.
Juste avant l'étape 1 de la triangularisation, la décomposition est de la forme:

$$\mathbf{P}_{i-1}\ldots\mathbf{P}_1\mathbf{E}\mathbf{K}_1\ldots\mathbf{K}_{i-1} = \begin{pmatrix} \mathbf{L}_{i-1} & \mathbf{0} \\ \mathbf{u}_i & \mathbf{v}_i \\ \mathbf{u}_{mE} & \mathbf{v}_{mE} \end{pmatrix}$$

Les $\mathbf{P}_i$ sont des matrices de permutations élémentaires et les $\mathbf{K}_i$ sont des transformations élémentaires de Householder qui éliminent les éléments $j+1,\ldots, n$ de la ligne j.
La matrice $\mathbf{L}_{i-1}$ est une matrice triangulaire inférieure $(i-1 x i-1)$, non singulière.

Les $\mathbf{u}_i$ et $\mathbf{v}_i$ sont un partitionnement des lignes i,..., m_E.

Il y a i-1 composantes dans chaque $\mathbf{u}_j$ et n-i+1 composantes dans chaque $\mathbf{v}_j$.

Au lieu d'éviter d'obtenir des éléments pivots qui sont petits, on choisit une permutation des lignes de la matrice $\mathbf{P}_i$ qui interchange les lignes i et l tel que $s_l = (\min \|\mathbf{u}_j\| / \|\mathbf{v}_j\| , j \geq i \text{ et } \mathbf{v}_j \neq 0)$

Avant de permuter les lignes, nous voulons voir si tous les $\mathbf{v}_j = 0$ de telle manière que s_l n'est pas défini, ou si $s_l^{-1} \leq t$.

Dans ce cas, on choisit $l = i$

Le paramètre t est une tolérance qui ne peut pas être plus petite que la précision machine.

Le rang approximé de **E** est alors défini comme e=i-1.

Dans le cas où $s_l^{-1} > t$, les lignes i et l sont interchangées et l'élimination est faite.

Quand $e < \min(m_E,n)$, les $\mathbf{v}_j$ restant sont mis à 0.

Ceci perturbe légèrement la matrice **E.**

Finalement, la matrice perturbée

$$\begin{bmatrix} \mathbf{L}_e & \mathbf{0} \\ \mathbf{u}_{e+1} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{u}_{mE} & \mathbf{0} \end{bmatrix}$$

est triangularisée en employant e matrices de Houselhoder.

Un autre détail concerne la compatibilité des équations **Ex=f**

En pratique, ceci revient à dire que le vecteur $\mathbf{g}_2$ est petit.

Ici, on préfère éviter d'analyser la compatibilité des équations, mais on retourne la longueur de $\mathbf{g}_2$ comme paramètre de sortie.

Cette valeur dépend évidemment des scalings sur les lignes.

L'algorithme de Golub-Businger sans modification sera employé à l'étape 2 pour les décompositions orthogonales.

Il y a un risque de sous-estimer le nombre de conditionnement de la matrice **A**.

On doit employer la décomposition en valeur singulière de **A** pour éviter cette possibilité.

2.2.2.2. Contraintes de non-négativité et contraintes d'égalités par poids différentiel

L'approche pour résoudre le problème NNLSE est basée sur la combinaison de deux algorithmes existants.
Le premier est une méthode de fonction de pénalité de Powell et Reid pour résoudre des équations de moindres carrés avec des contraintes d'égalité.
Le second est l'algorithme NNLS pour résoudre les problèmes de moindres carrés où les inconnues sont contraintes d'être non négatives.
Dans cette section, on discute une méthode pour résoudre le problème NNLSE en employant une pondération des contraintes d'égalité et des équations de moindres carrés.
Le lemme 2.7 établit la convergence de l'algorithme vers une solution de NNLSE.
La preuve de ce lemme montrera, par après, que l'algorithme converge aussi dans le cas où le problème NNLSE n'a pas de solution.
Le problème NNLSE a des contraintes de non négativité sur les variables l+1,...,n.
En fait, les variables 1,..., l peuvent être éliminées, en employant des matrices d'élimination orthogonales,de telle manière que les variables restantes sont contraintes d'être non négatives.
Une autre technique familière, est d'écrire chaque variable libre
$x_i = x_i' - x_i''$ où x_i' et x_i'' sont non négatives.
Pour la simplicité de la discussion qui suit, on suppose que $l = 0$.

Spécifiquement, l'approche pour résoudre le problème NNLSE est basée sur la résolution du problème suivant (WNNLS):

$$\begin{cases} \mathbf{A}_\varepsilon \mathbf{x} \cong \mathbf{b}_\varepsilon \\ \text{s.c.} \quad \mathbf{x} \geq \mathbf{0} \end{cases}$$

avec

$$(\mathbf{A}_\varepsilon : \mathbf{b}_\varepsilon) = \begin{pmatrix} \mathbf{E} & \mathbf{f} \\ \varepsilon\mathbf{A} & \varepsilon\mathbf{b} \end{pmatrix} \begin{matrix} \} \, m_E \\ \} \, m_A \end{matrix}$$

où ε est un petit paramètre réel.

On montrera qu'une solution du problème WNNLS a une limite quand $\varepsilon \to 0_+$.

De plus, si l'ensemble $\{\mathbf{x} : \mathbf{Ex} = \mathbf{f}, \mathbf{x} \geq \mathbf{0}\}$ est non vide, le vecteur limite est une solution du problème NNLSE .

Ce développement et sa preuve sont résumés dans le lemme 2.7.

Une autre façon d'expliquer cela est que la fonction à être minimisée dans WNNLS est $\| \mathbf{Ex-f} \|^2 + \varepsilon^2 \| \mathbf{Ax-b} \|^2$, sous contraintes $\mathbf{x} \geq \mathbf{0}$.

Ici, la norme vectorielle emloyée est la norme Euclidienne.

L'approche revient donc à une méthode de minimisation d'une fonction de pénalité qui est implémentée d'une manière numériquement stable.

En principe, le problème WNNLS , pour chaque $\varepsilon > 0$, peut être résolu par l'algorithme NNLS .

En pratique, il faut faire d'autres considérations pour résoudre le problème WNNLS .

On emploie la norme Euclidienne dans les tests pour l'indépendance linéaire des vecteurs colonnes de la matrice générée durant le procédé de décomposition orthogonale.

En fait , il faut employer les valeurs m_E, m_A et 1 pour calculer la norme Euclidienne pondérée dans l'algorithme

La stabilité numérique est maintenue en employant des rotations planes avec un interchangement sélectionné des lignes.
Dans les lemmes 2.4-2.7, on montre que la solution du problème WNNLS est bien définie.

Lemme 2.4 :

La solution de $\begin{cases} \min \mathbf{A}_\varepsilon \mathbf{x} \cong \mathbf{b}_\varepsilon \\ \text{s.c. } \mathbf{x} \geq \mathbf{0} \end{cases}$

obtenue en employant l'algorithme NNLS est un vecteur

$$\mathbf{x} = \mathbf{x}_\varepsilon \geq \mathbf{0} \text{ pour chaque } \varepsilon$$

L'ensemble $\mathcal{Z}_\varepsilon = \{ i, \mathbf{x}_i(\varepsilon) = 0 \}$ est fixé pour $0 < \varepsilon \leq \varepsilon_o$, ε_o suffisamment petit.

Lemme 2.5 :

Le problème de moindres carrés avec contraintes

$$\begin{cases} \mathbf{A}_\varepsilon \mathbf{u} \cong \mathbf{b}_\varepsilon \\ \text{s.c. } \mathbf{u} \geq \mathbf{0} \end{cases}$$

a une solution $\mathbf{u}_\varepsilon$. Cette solution a une limite $\mathbf{u}_o$ quand $\varepsilon \to 0$.

Les lemmes 2.6 et 2.7 concernent le cas où les contraintes d'égalités sont compatibles.
On montre que la solution du problème WNNLS existe et résoud approximativement le problème NNLSE.

Lemme 2.6:

Considérons les deux problèmes de moindres carrés avec contraintes

Problème NNLSE	Problème WNNLS
$\mathbf{Eu} = \mathbf{f}$	$\mathbf{A}_\varepsilon\mathbf{u} \cong \mathbf{b}_\varepsilon$, $0 < \varepsilon \leq \varepsilon_0$
$\mathbf{Au} \cong \mathbf{b}$	$\mathbf{u} \geq \mathbf{0}$
$\mathbf{u} \geq \mathbf{0}$	une solution $\mathbf{u}_\varepsilon \geq \mathbf{0}$ obtenue
une solution $\mathbf{u}' \geq \mathbf{0}$	par l'algorithme NNLS

Supposons que $\mathbf{Eu} = \mathbf{f}$ est compatible pour un $\mathbf{u}$ admissible i.e.

$\exists\ \mathbf{u}^* \geq \mathbf{0}$ t.q. $\mathbf{Eu}^* = \mathbf{f}$

Alors a) $\mathbf{Eu}' = \mathbf{f}$

et $\forall\ \varepsilon > 0$

b) $\| \mathbf{Eu}_\varepsilon - \mathbf{f} \|^2 + \varepsilon^2 \| \mathbf{Au}_\varepsilon - \mathbf{b} \|^2 \leq \| \mathbf{Eu} - \mathbf{f} \|^2 + \varepsilon^2 \| \mathbf{Au} - \mathbf{b} \|^2 \quad \forall\ \mathbf{u} \geq \mathbf{0}$

c) $\| \mathbf{Eu}_\varepsilon - \mathbf{f} \|^2 \leq \varepsilon^2 \| \mathbf{Au}' - \mathbf{b} \|^2$

d) $\| \mathbf{Au}_\varepsilon - \mathbf{b} \|^2 \leq \| \mathbf{Au}' - \mathbf{b} \|^2$

Lemme 2.7:

Supposons qu'on ait les mêmes hypothèses et les mêmes solutions qu'au lemme 2.6

Alors a) $\lim_{\varepsilon \to 0} \mathbf{u}_\varepsilon = \mathbf{u}_0 \geq \mathbf{0}$ existe

b) $\mathbf{Eu}_0 = \mathbf{f}$

c) $\| \mathbf{Au}_0 - \mathbf{b} \| = \| \mathbf{Au}' - \mathbf{b} \|$

Les lemmes 2.5 et 2.7 donnent la base mathématique d'un algorithme pour résoudre le problème WNNLS et donc pour résoudre le problème NNLSE.

Pour calculer une solution du problème NNLSE numériquement, le système

$$\begin{cases} \mathbf{A}_\varepsilon \mathbf{x} \cong \mathbf{b}_\varepsilon \\ \text{s.c. } \mathbf{x} \geq \mathbf{0} \end{cases}$$

est résolu juste une fois, en employant un ε positif suffisament petit.
Cette valeur de ε est choisie par l'algorithme pour calculer la solution avec le plus de précision possible.
Résoudre le problème WNNLS en employant ce procédé de pondération détermine un sous-ensemble de variables qui sont fixées à zéro.
L'admissibilité de cette solution peut être démontrée et un raffinement numérique peut être obtenu en résolvant un problème de moindres carrés avec contraintes d'égalité avec ces composantes mises à 0. Cette étape n'est pas nécessaire dans l'application pratique de la méthode.

2.2.2.3. <u>L'algorithme WNNLS</u>

Dans cette section, on trouve la description de l'algorithme numérique employé pour résoudre le problème WNNLS .
Cette discussion comprendra le choix du petit paramètre ε et des normes employées dans la détermination des rangs.
A cause de la complexité de l'algorithme, seuls les points essentiels sont donnés ici.
Pour faciliter la notation, on remplace le problème WNNLS par le suivant:
La matrice augmentée $(\mathbf{A}_\varepsilon : \mathbf{b}_\varepsilon)$ est écrite comme une nouvelle matrice augmentée $(\mathbf{DA} : \mathbf{Db})$.
Les nouveaux vecteurs et matrices sont donc définis par

$$\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \mathbf{w} \end{pmatrix} \begin{matrix} \} \ 1, & \mathbf{y} \text{ libre} \\ \} \ n\text{-}1, & \mathbf{w} \geq \mathbf{0} \end{matrix}$$

$$\mathbf{D} = \mathbf{diag}(\underbrace{1,\ldots,1}_{<m_E>}, \underbrace{\varepsilon,\ldots,\varepsilon}_{<m_A>})$$

et le problème WNNLS devient le problème suivant :

$$\begin{cases} \mathbf{DAx} \cong \mathbf{Db} & \mathbf{A} \ \ (m\text{x}n) \ \text{ où } m = m_E + m_A \\ \quad \mathbf{w} \geq \mathbf{0} & \end{cases}$$

On partitionne la matrice $\mathbf{DA} = (\underbrace{\mathbf{DA}_1}_{1} : \underbrace{\mathbf{DA}_2}_{n-1})$ correspondant aux dimensions des vecteurs **y** et **w**.
Dans la discussion qui suit, l'algorithme WNNLS sera présenté par une structure "top down".

Algorithme WNNLS

1. Initialiser les variables et faire la première triangularisation
2. Until (done)
   Calculer la direction de recherche et un point admissible
   Si une contrainte est rencontrée
     alors ajouter la contrainte
     sinon - faire le test de Kuhn-Tucher
           - si la solution est optimale aller en 3
           - sinon enlever une contrainte et aller en 2
3. Calculer la solution finale.

Chaque étape va être discutée dans les sections 3.1-3.3

3.1. Initialisation, triangularisation initiale et solution finale

La phase initiale de l'algorithme fait trois fonctions principales.
Premièrement, le poids ε est choisi.
Deuxièmement, la matrice $\mathbf{DA}_1$ est triangularisée .

Troisièmement les contraintes d'égalité dépendantes des autres sont éliminées.
N'importe quel choix de ε dans l'intervalle $(L/\eta)^{1/2} < \varepsilon < \varepsilon_o = \eta^{1/2}$
est satisfaisant. Ici L est le plus petit nombre machine positif et η est la précision relative de la machine.
Un poids d'au moins $\eta^{1/2}$ est nécessaire pour avoir le plus de précision possible dans un problème de moindres carrés avec contraintes d'égalité. La borne inférieure $(L/\eta)^{1/2}$ est là pour prévenir les "underflows".
En employant une décomposition orthogonale, on triangularise la sous-matrice $\mathbf{DA}_1$ (mxl). Ces transformations sont appliquées aux colonnes restantes et au vecteur **Db**.

On triangularise $\mathbf{DA}_1$ en employant, principalement, l'algorithme de Golub-Businger avec les extensions pour la déficience de rang.

Les transformations orthogonales de Givens sont employées pour calculer les triangularisations suivantes.

Comme dans l'algorithme de Golub-Businger, on interchange les colonnes pour maximiser la taille du terme diagonal résultant, à chaque étape importante.

Quand $\mathbf{DA}_1$ est déficient en rang, des transformations orthogonales de Householder sont appliquées par la droite pour obtenir une matrice triangulaire supérieure carrée et non singulière.

Donc, on calcule $\mathbf{Q}_1$ et $\mathbf{H}_1$ satisfaisant

$$\mathbf{Q}_1(\mathbf{DA}_1)\mathbf{H}_1^T = \begin{pmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ k & l\text{-}k \end{pmatrix} \begin{matrix} \} \ k \\ \} \ m\text{-}k \end{matrix}$$

Ici le k est le rang de $\mathbf{DA}_1$. Les relations suivantes tiennent entre les valeurs entières $1, k, m_E$ et $m : 0 \le k \le l$ et $0 \le m_E \le m$.

Quand $k < m_E$, on emploie de nouveau l'algorithme de Golub-Businger pour la sous-matrice de $\mathbf{Q}_1$ $(\mathbf{DA}_2)$ consistant en les lignes $k+1, \ldots, m_E$ et les colonnes $l+1, \ldots, n$.

Si une contrainte d'égalité est dépendante des autres, elle est enlevée du problème et m_E est réduit au rang de la sous-matrice.

Comme résultat de cette étape, les contraintes d'égalité sont linéairement indépendantes quand on introduit les contraintes de non négativité.

Durant le procédé d'élimination dans toutes les phases de l'algorithme, on veut que les poids des lignes $1, \ldots, m_E$ et des lignes $m_E+1, \ldots, m$ restent inégaux à chaque étape.

On élimine les termes sous-diagonaux de la colonne j comme suit:

Fig.1

	colonne j
ligne 1	x
:	:
ligne m_E	x
----------	----
ligne m_E+1	ε
:	0
:	:
ligne m	0

L'élimination procède avec la ligne i qui est chassée à la ligne i-1, i=m,...m_E+2, en employant des transformations de Givens.
Après, l'élément de la ligne m_E+1 est chassé dans la ligne du plus grand indice dont l'élément est non nul, où $j \le p \le m_E$.

Après, l'élimination des termes sous-diagonaux de la colonne j est complétée en chassant la ligne i à la ligne i-1, i:p,...,j+1.
Dans le cas exceptionnel où les composantes j,...,m_E sont nulles, on interchange les lignes m_E+1 et j et on agrandit la valeur m_E de 1. Ceci définit une nouvelle contrainte d'égalité.
Sans cette étape de pivotage, i.e. si la composante m_E+1 est éliminée dans la composante j , l'information représentée par cette équation serait perdue. Cela conduirait à une instabilité numérique de l'algorithme.
Dans cette phase initiale de l'algorithme, le rang de la matrice $\mathbf{DA}_1$ est déterminée en déclarant la colonne j , $j \le m_E$ comme étant dépendante si la norme Euclidienne des composantes j,...,m est petite comparée à la norme Euclidienne des composantes 1,...,j-1.

Pour $j > m_E$, on emploie le même critère, mais on emploie la norme Euclidienne pondérée $\| \mathbf{D}^{-1}(.) \|$

Une fois que le vecteur $\mathbf{w}^* > 0$ a été calculé en employant les procédures expliquées dans la section suivante, la solution finale est calculée.

On veut résoudre le problème de moindres carrés $\mathbf{DA}_1\mathbf{y} \cong \mathbf{b}\text{-}\mathbf{DA}_2\mathbf{w}^*$ pour $\mathbf{y} = \mathbf{y}^*$.

Employant les décompositions orthogonales $\mathbf{Q}_1$ et $\mathbf{H}_1$, on définit

$\mathbf{g} = \mathbf{Q}_1(\mathbf{b}\text{-}\mathbf{DA}_2\mathbf{w}^*)$

Alors, on calcule,

$$\mathbf{y}^* = \mathbf{H}_1 \begin{pmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}^T \mathbf{g} = \mathbf{H}_1 \begin{pmatrix} \mathbf{R}_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{g}$$

La solution finale de $\begin{cases} \mathbf{DAx} \cong \mathbf{Db} \\ \mathbf{w} \geq \mathbf{0} \end{cases}$

est donnée par $\mathbf{x}^* = (\mathbf{y}^{*T} : \mathbf{w}^{*T})^T$, avec la norme résiduelle correspondante $\| \mathbf{D}^{-1}(0,\ldots, 0, \mathbf{g}_{k+1},\ldots, \mathbf{g}_m)^T \|$

3.2. Distinctions entre l'algorithme WNNLS et l'algorithme NNLS

Dans cette section, on considère le sous-problème restant de la boucle dans l'algorithme WNNLS où toutes les variables sont contraintes d'être non négatives. Ceci revient à résoudre le problème WNNLS où $l = 0$

Mathématiquement, c'est un problème NNLS .

A cause de la matrice de poids $\mathbf{D}$ du problème WNNLS , un nombre de détails supplémentaires sont nécesaires pour implémenter l'algorithme WNNLS avec succès. Comme dans la section 3-1, la plupart de ces considérations sont basées sur le fait de maintenir les poids inégaux à chaque étape de l'algorithme.

Les variations spécifiques de l'implémentation entre l'algorithme NNLS et WNNLS seront indiquées ci-dessous.

L'algorithme commence avec toutes les contraintes actives, $\mathbf{w} = \mathbf{0}$.

A chaque itération, un point admissible $\mathbf{w} \geq \mathbf{0}$ est généré avec une réduction de la norme du vecteur résidu $\mathbf{r} = \mathbf{D}(\mathbf{b}-\mathbf{A}\mathbf{w})$ quand il n'y a plus de réduction possible, "done" est mis à "true".

Durant l'algorithme, quand des contraintes sont enlevées et ajoutées, il est possible que quelques unes des lignes $1,\ldots,m_E$ aient des scalings disparates.

Supposons qu'il y a j composantes positives dans la solution qui correspondent aux j premières colonnes. Avant d'enlever une nouvelle contrainte de non négativité, les lignes $j+1,\ldots,m_E$ sont testées pour leurs scalings disparates. Si une telle ligne est trouvée, elle est reclasifiée comme une équation de moindres carrés et la valeur de m_E est réduite.Ce fait introduit un détail nécessaire dans l'implantation de l'algorithme WNNLS qui ne l'était pas dans l'algorithme NNLS

Chaque variable, qui entre dans l'ensemble solution, doit être positive. Chaque fois qu'une variable entre dans cet ensemble, la décomposition orthogonale doit être ajustée. Ceci revient à triangulariser une matrice qui est déjà triangulaire supérieure excepté pour la dernière colonne, qui est non nulle en dessous de la diagonale. L'ajustement de la forme triangulaire est obtenu en employant exactement la même procédure que celle décrite pour le problème sans contraintes dans la section 3.1;

L'indépendance de la colonne rajoutée, correspondant à la contrainte que l'on vient d'enlever, est décidée par l'équation.

$0 < \mathbf{w}_j < (\| \mathbf{b} \| / \| \mathbf{A} \|)\eta^{1/2}$ et $\mathbf{a}_{jj} \neq 0$.

Ici, l'idée est d'employer des composantes de la solution qui sont positives mais pas "trop grandes".

Ce test n'est employé que dans l'algorithme WNNLS car il vérifie la compatibilité du vecteur du membre de droite par rapport aux scalings disparates des lignes de la matrice **D**.

Un test pour l'indépendance du nouveau vecteur colonne basé seulement sur le nombre de conditionnement des j colonnes de la matrice triangularisée est inapproprié pour l'algorithme WNNLS

A chaque étape importante de l'algorithme, on teste pour voir si aucune composante de la solution est négative. Donc, on voit si une contrainte est touchée ou violée. Si cela a lieu, une interpolation à un point admissible est faite. Si cette étape d'interpolation est nécessaire, toutes les colonnes correspondantes négatives doivent être enlevées de la triangularisation. La décomposition orthogonale des contraintes non actives doit être retriangularisée. Dans l'algorithme NNLS , cette redécomposition est faite simplement en chassant les éléments sous-diagonaux à la diagonale de la matrice Hessenberg résultant de l'enlèvement de la colonne.

Dans l'implémentation de l'algorithme WNNLS , cette retriangularisation est nécessairement plus compliquée. A nouveau, c'est dû aux scalings disparates des lignes de la matrice. La retriangularisation de la matrice Hessenberg est la même que dans l'algorithme NNLS avec l'exception suivante. Le procédé d'élimination dans la colonne est montré à la figure 1. Comme on élimine seulement les éléments non nuls, quand les composantes m_E+1 et m_E sont nulles, il faut interchanger les lignes m_E+1 et m_E et on augmente la valeur de m_E .

Mathématiquement, quand une contrainte est touchée ou violée, l'interpolation causera toujours un mouvement vers un point non négatif. Numériquement, il est possible pour une ou plusieurs composantes de la solution de rester non positive. Donc, quand une contrainte a été ajoutée, d'autres contraintes doivent être rajoutées jusqu'à ce que toutes les composantes soient positives. Chaque telle étape demande une retriangularisation de la matrice Hessenberg comme expliqué ci-dessus.

2.2.3 **Problème de moindres carrés linéaire avec bornes et contraintes linéaires** (Richard J. Hanson,1986)

2.2.3.1. Introduction

L'algorithme BNDCLS résoud

$$\begin{cases} \mathbf{Ax} \cong \mathbf{b} & \mathbf{A}\ (m \times n) \\ \text{s.c. } \alpha_j \le \mathbf{x}_j \le \beta_j & j = 1,\ldots, n \\ \mathbf{Cx} = \mathbf{y} & \mathbf{C}\ (m \times n) \\ \alpha'_i \le \mathbf{x}_i \le \beta'_i & i = 1,\ldots, m \end{cases}$$

C comprend les contraintes $\mathbf{C}_i^T\mathbf{x} = \mathbf{f}_i$, $\mathbf{C}_i^T\mathbf{x} \ge \mathbf{g}_i$, $\mathbf{C}_i^T\mathbf{x} \le \mathbf{h}_i$

On pourrait mettre les contraintes ensemble dans une matrice **C'** mais on ne le fait pas pour des raisons d'efficacité et d'économie de stockage.

Cet algorithme traite le problème par une méthode de pénalité.

Cet algorithme ne demande pas que la matrice des moindres carrés **A** soit de rang plein au contraire de LCLSQ .

Cette version de BNDCLS résoud deux problèmes, un numérique et un conceptuel.

Le problème implique la translation des bornes.

A priori, il semble intéressant de translater les contraintes de telle manière que la borne inférieure soit 0. Numériquement, ça peut devenir un désastre. Par exemple, si on a une équation x = a et des bornes $-b \le x \le b$. Supposons que la quantité b est très grande par rapport à a et que a+b est calculé comme étant b . Si on translate en posant y-b = x , l'équation deviendra y = b qui ne dépend plus du tout de a .

L'algorithme évitera donc de translater les bornes. Ce fait complique l'algorithme parce qu'il faudra distinguer les contraintes actives et celles qui ne le sont pas.

Le problème conceptuel vient du fait qu'il semble important de noter que les contraintes linéaires sont incompatibles. L'algorithme, au lieu de donner un message d'erreur, perturbe les bornes afin de rendre les contraintes compatibles dans certains cas.

2.2.3.2 Réduction du problème avec contraintes linéaires à un problème de moindres carrés avec bornes.

Dans ce paragraphe, nous considérons le cas général où le nombre de contraintes linéaires, MCON , est positif.

La méthode présentée ici est basée sur une approche en deux phases.

Phase 1: Résoudre les équations des contraintes $\mathbf{Cx-y = 0}$ dans un sens de moindres carrés avec les contraintes sur $\mathbf{x}$ et $\mathbf{y}$.

Dans le cas où les contraintes sont incompatibles, on les rend admissibles en perturbant les valeurs de α'_i et β'_i .

Pratiquement, on calcule $\mathbf{y}^* = \mathbf{Cx}^*$ où $\mathbf{x}^*$ est le vecteur des NCOLS premières composantes du vecteur solution $(\mathbf{x}^T : \mathbf{y}^T)^T$.

Après, on élargit les bornes sur $\mathbf{y}_i$:

$$\alpha''_i = \min(\alpha'_i , \mathbf{y}^*_i)$$

$$\beta''_i = \min(\beta'_i , \mathbf{y}^*_i)$$

Remarquons que $\alpha''_i = \alpha'_i$ et $\beta''_i = \beta'_i$, i=1,..., MCON si les contraintes sont admissibles.

Dans le cas où les contraintes ne sont pas admissibles, l'algorithme donne une solution particulière au problème en élargissant les bornes sur $\mathbf{y}$ de telle manière que la solution des équations $\mathbf{Cx-y = 0}$ existe.

On peut résoudre ce système de moindres carrés pour sa seule solution de longueur minimale et après, élargir les bornes sur $\mathbf{y}$ comme décrit ci-dessus. Cette solution de longueur minimale peut être obtenue par la méthode précédente (Haskell-Hanson,1981)

La méthode simple expliquée ci-dessus est souvent préférée parce que le calcul de la solution de longueur minimale coûte relativement cher.

Il semble aussi que la solution approximée des contraintes linéaires peut être motivée comme les solutions approximées des équations algébriques linéaires.

Phase 2 : Résoudre le problème de moindres carrés pondéré

$$\begin{pmatrix} \mathbf{C} & \mathbf{-I} \\ \tau\mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \tau\mathbf{b} \end{pmatrix}$$

avec bornes

$$\alpha_j \leq \mathbf{x}_j \leq \beta_j \qquad j = 1,\ldots, \text{NCOLS}$$
$$\alpha'_i \leq \mathbf{x}_i \leq \beta'_i \qquad i = 1,\ldots, \text{MCON}$$

Le paramètre de poids τ est choisi de telle manière que le scaling du problème résultant est balancé:

$$\tau = \begin{cases} \eta \| \mathbf{C} \| / \| \mathbf{A} \| & \text{si } \mathbf{C} \neq \mathbf{0} \text{ et } \mathbf{A} \neq \mathbf{0} \\ \eta(1/\| \mathbf{A} \|) & \text{si } \mathbf{C} = \mathbf{0} \text{ et } \mathbf{A} \neq \mathbf{0} \\ 1 & \text{si } \mathbf{A} = \mathbf{0} \end{cases}$$

où η est la précision arithmétique relative i.e. le plus grand nombre machine t.q. $1+\eta = 1$

L'emploi de η dans la définition de τ est arbitraire. En fait, τ n'a pas de borne inférieure positive. Une borne supérieure est $\eta^{1/2}$.

La fonction de pénalité est employée pour deux raisons.

La premiére raison implique le fait que le problème avec bornes mais sans contraintes linéaires générales est direct. Avec des bornes simples, la compatibilité des contraintes est vérifiée simplement en regardant si $\alpha_j \leq \beta_j$.

Avec des contraintes linéaires générales, il est plus difficile de voir que les contraintes sont compatibles. l'auteur a choisi de passer au dessus de cette question en faisant la première phase expliquée ci-dessus.

La seconde raison pour employer la fonction de pénalité est qu'on sait depuis quelque temps que les méthodes de pénalité sont efficaces pour les contraintes d'égalités linéaires (sans simples bornes).

Les algorithmes pour les problèmes de moindres carrés existants peuvent être employés aussi longtemps que le pivotage de lignes et de colonnes est employé. Mais on doit éviter de mélanger les lignes pondérées et celles qui ne le sont pas pour ne pas perdre les informations essentielles du problème. Comme les méthodes de pénalité marchent bien dans ce contexte et que des algorithmes existants ont été développés pour le cas de bornes simples, l'auteur fut conduit à investiguer l'approche présentée ci-dessous pour résoudre le problème BNDCLS

L'idée de base revient à un choix systématique de sous-ensemble de colonnes de **A**.

Employant ces colonnes, une solution d'un problème de moindres carrés sans contraintes peut être calculée en triangularisant la matrice par une décomposition orthogonale.

Les variables qui ne sont pas dans le sous-ensemble sont soit à leur borne, soit à la valeur 0.

Le choix du sous-ensemble, appellé $\mathcal{B}$ dans l'algorithme, est fait de telle façon que la longueur du vecteur résidu du problème de moindres carrés sans contraintes correspondant est décroissante à chaque étape.(l'examen systématique de tous les sous-ensembles est hors de question sauf peut-être pour les petits problèmes).

Pour être robuste, l'algorithme doit tester l'indépendance linéaire des colonnes du sous-ensemble.

Le test suggéré emploie une norme qui enlève le poids appliqué aux lignes MCON+1,..., MCON+MROWS de la matrice $\begin{pmatrix} \mathbf{C} & \mathbf{-I} \\ \tau\mathbf{A} & \mathbf{0} \end{pmatrix}$

La norme pondérée est définie par

$$\|.\| = ((.)^2+\ldots+(.)^2+\tau^{-2}(.)^2+\ldots+\tau^{-2}(.)^2)^{1/2}$$

<-- MCON -> <--- MROWS --->

Les colonnes de **A** correspondant aux indices de l'ensemble $\mathcal{B}$ sont triangularisées.

Les variables de $\mathcal{B}$ sont choisies de telle manière que les colonnes sont linéairement indépendantes et le problème de moindres carrés sans contraintes (avec ces colonnes) donne une solution qui satisfait les bornes dans un sens strict.

Dans le choix d'un nouveau sous-ensemble, une colonne candidate est adjointe au triangle supérieur pour former une nouvelle matrice partitionnée.

$$\begin{pmatrix} \mathbf{T} & \mathbf{u} \\ \mathbf{0} & \mathbf{v} \end{pmatrix}$$

où **T** est une matrice trangulaire supérieure (kxk), **u** est un vecteur de dimension k et **v** est un vecteur de dimension m-k .

Employant la norme pondérée, on définit la nouvelle colonne comme étant linéairement dépendante quant à un paramètre ε_1 si $\|\mathbf{v}\| \leq \varepsilon_1 \|\mathbf{u}\|$ (la valeur du paramètre ε_1 peut être changée par l'utilisateur ; $\varepsilon_1 = \eta^{1/2}$ est la valeur par défaut).

Si cette condition n'est pas vérifiée, la colonne candidate $(\mathbf{u}^T : \mathbf{v}^T)^T$ est linéairement indépendante. Mathématiquement, le test pour la dépendance linéaire de la colonne candidate est "$\|\mathbf{v}\| = 0$".

Dans l'algorithme, on remplace ce test par "|| **v** || est petite" . Un choix particulier est celui donné ci-dessus.

Remarquons que ce test est indépendant du scaling de la colonne de la matrice de départ. Pour le test, on doit employer la norme pondérée car une norme simple impliquerait toujours que || **v** || soit petite quand $k \geq$ MCON .Ceci est dû au fait que le poids τ appliqué aux équations de moindres carrés persiste tout le long de l'algorithme.

On complète la triangularisation de la nouvelle colonne par des rotations planes pour éliminer les entrées en dessous de la diagonale principale.

Un deuxième détail d'implémentation important associé au poids τ implique un type de pivotage pour éliminer les composantes de la colonne k+1 à partir de k+2 jusque MCON+MROWS .

On doit éviter une situation numériquement instable où une élimination est faite sur deux entrées $(\mathbf{u}_i : \mathbf{v}_j)^T$ où $\mathbf{u}_i$ est petit (relativement à la norme de la colonne) et $\mathbf{v}_j$ correspond à une ligne pondérée par τ .

Cela donnerait une rotation plane qui mélange les lignes pondérées et celles qui ne le sont pas et il en résulterait une perte d'informations essentielles sur le problème.

Le pivotage est accompli, dans notre algorithme, en éliminant les entrées dans l'ordre MCON+MROWS,...,MCON+1 puis MCON,...,k+2 et finalement, en éliminant les plans entre MCON+1 et n+1.

A la place de garantir que l'entrée k+1 n'est pas petite, on interchange les colonnes au préalable.

La table ci-dessous peut aider le lecteur à suivre la séquence d'élimination. pour $k \geq$ MCON , seulement la première partie est faite. Pour $k <$ MCON , toutes les parties doivent être faites.

ligne k+1	x	x	x	x
	x	x	x	0
	:	:	:	:
	x	x	0	0
	x	x	0	0
ligne MCON+1	x	x	x	x
	x	x	0	0
	x	0	0	0
	:	:	:	:
ligne MCON+MROWS	x	0	0	0
	depart	1°partie	2°partie	fin

2.2.3.3 Algorithme pour un problème de moindres carrés pondéré avec bornes simples.

Dans cette section, on présente une méthode pour résoudre le problème BNDCLS expliqué dans les paragraphes précédents.

L'algorithme est, dès lors, un problème de moindres carrés linéaire avec bornes simples sur les variables, $\alpha_j \le \mathbf{x}_j \le \beta_j$ j=1,...,NCOLS .

L'utilisateur doit spécifier les bornes sur chaque variable selon l'un des 4 cas suivants:

cas 1:	$\alpha_j \le \mathbf{x}_j$	$\alpha_j > -\infty$
cas 2:	$\mathbf{x}_j \le \beta_j$	$\beta_j < +\infty$
cas 3:	$\alpha_j \le \mathbf{x}_j \le \beta_j$	$-\infty < \alpha_j \le \beta_j < +\infty$
cas 4:	$\mathbf{x}_j$ est libre.	

En ajoutant les bornes manquantes dans les cas 1, 2 et 4, on normalise le problème de telle manière que pour chaque j, $-b \leq \alpha_j \leq \mathbf{x}_j \leq \beta_j \leq b$, où b est le plus grand nombre réel de la machine.

Par après, on normalise le problème de telle façon que si $\alpha_j < u < \beta_j$ alors $|\alpha_j| \leq \beta_j$ où $0 < \alpha_j \leq \beta_j$. Ceci peut demander un changement de signe pour $\mathbf{x}_j$.

Ces conditions partitionnent les variables en deux ensembles disjoints $\mathcal{L}$ et $\mathcal{T}$ $j \in \mathcal{T}$ si $0 \leq \alpha_j \leq \beta_j$ sinon $j \in \mathcal{L}$.

Les ensembles $\mathcal{L}$ et $\mathcal{T}$ vont changer durant l'algorithme.

Les éléments de $\mathcal{L}$ vont dans $\mathcal{T}$ quand une variable va jusqu'à sa borne. On peut penser que $\mathcal{T}$ désigne les variables qui ont déjà touché leur borne. Les variables de $\mathcal{L}$ sont celles qui sont restées entre leurs deux bornes.

Pour les variables de $\mathcal{T}$, on translate la borne inférieure pour qu'elle devienne 0, $\mathbf{x}'_j = \mathbf{x}_j - \alpha_j$.

Remarquons que le scaling de la colonne $\mathbf{x}_j = \mathbf{d}_j \mathbf{w}_j$, $\mathbf{d}_j > 0$ ne change pas le partitionnement de $\{\mathcal{T}, \mathcal{L}\}$. Il affecte le choix de la variable qui est choisie pour devenir libre dans les étapes 4 et 5 de l'algorithme NNLSB.

Il peut aussi affecter la solution particulière choisie par l'algoritme quand le rang de la matrice $\mathbf{A}$ est plus petit que NCOLS.

On effectue un scaling sur les colonnes de telle manière que la norme de grandeur maximale des colonnes non nulles ait la valeur 1. Il y a une option pour que l'utilisateur puisse choisir un autre scaling s'il le veut.

L'algorithme commence avec le point admissible $\mathbf{x} = \mathbf{0}$.

Dans la boucle principale, les éléments de $\mathcal{B}$ sont enlevés et ajoutés de telle manière que la longueur du vecteur résidu $\| \mathbf{b-Ax} \|$ est décroissante.

La preuve de convergence finie de l'algorithme consiste à remarquer que le nombre d'itérations entre une stricte décroissance de la longueur du vecteur résidu et la terminaison de l'algorithme est au plus NCOLS.

Les entiers $I = \{1,\ldots,\text{NCOLS}\}$ sont partitionnés par l'algorithme en deux paires d'ensembles disjoints $\{\mathcal{Z}, \mathcal{B}\}$ et $\{\mathcal{T}, \mathcal{L}\}$.Le partitionnement $\{\mathcal{Z}, \mathcal{B}\}$.

est déterminé par le fait qu'une variable est à la valeur 0 ou non.

Le tableau IP contient le nombre de fois qu'une variable a touché sa base supérieure. Cette information est employée dans l'étape 3 pour déterminer si permettre une variable de décroître de sa borne supérieure entrainerait une baisse de la norme résiduelle. Elle est aussi requise pour réfléchir les variables qui sont à leur borne supérieure pour calculer la solution dans les étapes de 17 à 19.

En résumé, l'algorithme est une méthode de direction de descente admissible. En fait, les étapes de 10 à 12 reviennent à choisir une longueur de pas et définir un nouveau point admissible dans cette direction de descente et les étapes 14 et 15 ajustent l'ensemble $\mathcal{B}$ de telle manière que le problème de moindres carrés sans contraintes correspondant a une solution qui satisfait les bornes mais qui n'est pas à une borne.

Algorithme NNLSB .

1. Poser $\mathcal{Z} = \{1,\ldots,\text{NCOLS}\}$, $\mathcal{B} = \emptyset$ $\mathbf{x} = \mathbf{0}$, $IP(j) = 1$ $j=1,\ldots,\text{NCOLS}$
2. Calculer le vecteur $\mathbf{w} = \mathbf{A}^T(\mathbf{b}-\mathbf{Ax})$ de dimension NCOLS
3. Pour $j=1,\ldots,\text{NCOLS}$, poser $\mathbf{w}_j = -\mathbf{w}_j$ si $IP(J) = 0 \pmod 2$
4. Trouver un index $t \in \mathcal{Z}$ t.q. $\mathbf{w}_t = \max\{\mathbf{w}_j , j \in \mathcal{Z}$ et $\mathcal{T}, \mathbf{w}_j > 0, \mathbf{x}_j$ limité$\}$
5. Trouver un index $s \in \mathcal{Z}$ t.q. $|\mathbf{w}_s| = \max\{|\mathbf{w}_j|, j \in \mathcal{Z}$ et $\mathcal{L}\}$
6. Trouver le plus maximum entre $\mathbf{w}_t$ et $|\mathbf{w}_s|$ et le u correspondant (colonne s ou t)
   Si aucun t ni s n'est définit dans les étapes 4 et 5, aller à l'étape 17.
7. Si la colonne u est linéairement indépendante des colonnes de $\mathcal{B}$ alors déplacer l'indice u de l'ensemble $\mathcal{Z}$ à l'ensemble $\mathcal{B}$.
   Sinon poser $\mathbf{w}_u = 0$ et aller à l'étape 4.
8. Poser $\mathbf{A}_{\mathcal{B}} = (\mathbf{a}^*_1,\ldots,\mathbf{a}^*_{\text{NCOLS}})$ où $\mathbf{a}^*_j = \begin{cases} \text{colonne } j \text{ de } \mathbf{A} & \text{si } j \in \mathcal{B} \\ 0 & \text{sinon} \end{cases}$
9. Calculer le vecteur $\mathbf{z}$ (NCOLS) comme étant la solution au sens de moindres carrés du système $\mathbf{A}_{\mathcal{B}}\mathbf{z} \cong \mathbf{b}$ (seul les $\mathbf{z}_i$, $j \in \mathcal{B}$ sont définis; pour $j \in \mathcal{Z}$, $\mathbf{z}_j = 0$).
10. Poser $\alpha = 1$, $\beta = 1$
11. Pour chaque j , définir $\gamma_j = 0$ si $j \in \mathcal{T}$,sinon $\gamma_j = \alpha_j$.
    si $\mathbf{z}_j \le \gamma_j$, pour $j \in \mathcal{B}$, trouver un indice $q \in \mathcal{B}$ t.q.
    $\alpha = (\mathbf{x}_q-\gamma_q)/(\mathbf{x}_q-\mathbf{z}_q) = \min((\mathbf{x}_j-\gamma_j)/(\mathbf{x}_j-\mathbf{z}_j), j \in \mathcal{B}, \mathbf{z}_j \le \gamma_j)$
12. Si $\mathbf{z}_j \ge \beta_j$ pour un $j \in \mathcal{B}$, trouver un indice $p \in \mathcal{B}$ t.q.
    $\beta = (\beta_p-\mathbf{x}_p)/(\mathbf{z}_p-\mathbf{x}_p) = \min((\beta_j-\mathbf{x}_j)/(\mathbf{z}_j-\mathbf{x}_j), j \in \mathcal{B}, \mathbf{z}_j \ge \beta_j)$
13. $\mathbf{x} = \mathbf{x}+\min(\alpha,\beta)(\mathbf{z}-\mathbf{x})$
14. Déplacer de l'ensemble $\mathcal{B}$ à l'ensemble $\mathcal{Z}$ tous les indices j t.q.$\mathbf{x}_j \le \gamma_j$
    Si ,$j \in \mathcal{L}$ ajuster la borne $\beta_j = \beta_j-\alpha_j$ et déplacer j de l'ensemble $\mathcal{L}$ à l'ensemble $\mathcal{T}$
15. Déplacer de l'ensemble $\mathcal{B}$ à l'ensemble $\mathcal{Z}$ tous les indices j t.q.$\mathbf{x}_j \ge \beta_j$
    Si $j \in \mathcal{T}$ ajuster $\mathbf{x}_j = \beta_j-\mathbf{x}_j$ et ajuster la polarité de ce x_j : $IP(j) = IP(j)+1$
    Sinon, changer le signe de $\mathbf{x}_j$, $\beta_{j'} = \beta_j-\alpha_j$, $\alpha_j = -\beta_j$ et déplacer j de $\mathcal{L}$ à $\mathcal{T}$
16. Aller en 2
17. Pour j = 1,..,NCOLS, poser $\mathbf{x}_j = \beta_j-\mathbf{x}_j$ si $IP(j) = 0 \pmod 2$

18. Pour j = 1,..,NCOLS, poser $x_j = x_j + \alpha_j$ si $j \in \mathcal{T}$
19. Pour j = 1,..,NCOLS, poser $x_j = -x_j$ pour chaque j correspondant à une variable dont le signe a été changé pendant l'algorithme

Remarque sur l'algorithme NNLSB

Cet algorithme converge. Ceci est prouvé en remarquant que la norme résiduelle $\| \mathbf{b-Ax} \|$ décroît monotonement, par rapport au nombre d'itérations de la boucle de l'étape 2 à l'étape 16.

Comme dans la discussion de l'algorithme NNLS , la norme décroît strictement ou l'ensemble $\mathcal{B}$ perd des indices à chaque itération. Supposons que l'ensemble $\mathcal{B}$ commence en étant vide. A ce point, si une colonne bouge de l'ensemble $\mathcal{Z}$ à l'ensemble $\mathcal{B}$, dans les étapes 4 et 5, alors la norme résiduelle décroît strictement parce que $\min(\alpha,\beta) > 0$ à l'étape 13 ou $\min(\alpha,\beta) = 0$.

De l'étape 11 et du fait que $\alpha > 0$, il suit que $\min(\alpha,\beta) = 0$ donc $\beta = 0$

Dans ce cas, le $w_j > 0$ correspondant choisi à l'étape 4 deviendra négatif et donc ne sera plus choisi à l'étape 4 quand $\beta = 0$.

Donc, dans ce cas extrême, le nombre de w_j positif décroît.

Ceci termine la preuve de convergence finie de l'algorithme.

Aux étapes 14 et 15, les variables doivent satisfaire les bornes $x_j \geq \gamma_j$ et $x_j \leq \beta_j$. respectivement.

A cause de la précision finie, il peut y avoir un x_j qui viole les bornes et il est essentiel qu'un tel indice bouge de l'ensemble $\mathcal{B}$ à l'ensemble $\mathcal{Z}$.

Il reste à examiner si les variables satisfaisant $x_j > \gamma_j$ ou $x_j < \beta_j$. sont mal classifiées. Elles doivent être classifiées si $x_j = \alpha_j$ ou $x_j = \beta_j$. Heureusement, ceci n'est pas sérieux car les éléments de l'ensemble $\mathcal{B}$ satisfont $\alpha_j < x_j < \beta_j$ à l'étape 2 à cause de l'action faite à l'étape 14 et 15

Donc $\mathbf{x}_j$-$\mathbf{z}_j > 0$ à l'étape 11 et $\mathbf{z}_j$-$\mathbf{x}_j > 0$ à l'étape 12 à une itération ultérieure de l'algorithme . Ceci implique que les variables mal classifiées iront, en fin de compte, de l'ensemble $\mathcal{B}$ à l'ensemble $\mathcal{Z}$ si c'est nécessaire.

Le test pour l'indépendance linéaire à l'étape 7 joue un rôle décisif dans l'algorithme. Ce fait a été considéré au paragraphe 2.

Le problème de moindres carrés de l'étape 9 est résolu en appliquant des rotations planes (transformations de Givens) à la matrice $(\mathbf{A} : \mathbf{b})$. Ces transformations sont appliquées dans un certain ordre discuté au paragraphe 2 Les colonnes correspondants aux indices de $\mathcal{B}$ sont déplacées vers la droite. Donc quand une nouvelle colonne est ajoutée à $\mathcal{B}$, la matrice résultante $\mathbf{A}_{\mathcal{B}}$ est triangulaire supérieure avec une pointe dans la dernière colonne.
Les rotations planes sont alors appliquées pour réduire cette matrice à une forme triangulaire supérieure. Quand des indices sont enlevés de $\mathcal{B}$ à l'étape 14 et 15, la matrice résultante $\mathbf{A}_{\mathcal{B}}$ est Hessenberg supérieure. Cette matrice est à nouveau réduite à une forme triangulaire supérieure en employant des rotations planes.
Lors de l'élimination des entrées entre les lignes MCON et MCON+1 ,un pivotage de colonnes peut être requis pour éviter de mélanger les lignes pondérées et celles qui ne le sont pas. Ce pivotage peut impliquer la perte de la forme Hessenberg supérieur, de sorte que la réduction du système à une forme triangulaire demande la triangularisation d'une matrice rectangulaire pleine.
Tout ce travail est le prix pour une stabilité numérique mais ça n'arrive pas souvent.

2.3. UNE METHODE DE FACTORISATION POUR RESOUDRE LE PROBLEME DE MOINDRES CARRES LINEAIRE AVEC CONTRAINTES.

(Stoer and Schittkowski , 1979)

2.3.1 Position du problème.

La forme générale du problème de moindres carrés linéaire avec contraintes linéaires que nous traitons est

$$\begin{cases} \min \| \mathbf{Cx-d} \|_2 \\ \text{s.c. } \mathbf{Ax}\ \rho\ \mathbf{b} \end{cases} \qquad (P1)$$

où **C** est une matrice pxn

x est un vecteur de dimension n

b est un vecteur de dimension p

A est une matrice mxn, non dégénérée i.e. $\forall$ k, $1 \leq k \leq \min(m,n)$, chaque ensemble de k ligne de **A** sont linéairement indépendantes.

ρ est un vecteur relationel de dimension m où la $i^{ème}$ composante de ρ est "$\leq$", "$\geq$" ou "=".

b est un vecteur de dimension m

On suppose que $p \geq n$ et que les colonnes de **C** sont linéairement indépendantes.

2.3.2 Définitions et propriétés

Définition 2.1

On appelle ensemble actif en$\mathbf{x}^*$,l'ensemble de tous les indices de contraintes satisfaites comme egalite en $\mathbf{x}^*$ i.e.

$\mathbf{J(x^*)} = \{j \text{ t.q. } \mathbf{a}_j\mathbf{x}^* = \mathbf{b}_j\} = \{j_1,....,j_\mu\}$

Définition 2.2

P est l'ensemble de toutes les solutions admissibles i.e.

$\mathbf{P} = \{\mathbf{x} \text{ t.q. } \mathbf{Ax}\ \rho\ \mathbf{b}\}$

On note $\mathbf{A}_J$ la matrice active formée par les contraintes actives en $\mathbf{x}^*$

Si **A** est de la forme $\mathbf{A}\begin{pmatrix}\mathbf{a}_1\\ \\ \mathbf{a}_m\end{pmatrix}$ où $\mathbf{a}_i$ est la i^{eme} ligne de **A**,

$\mathbf{A}_J$ est de la forme $\begin{pmatrix}\mathbf{a}_{j1}\\ \\ \mathbf{a}_{j\mu}\end{pmatrix}$ $,j_1,\ldots,j_\mu \in \mathbf{J}(\mathbf{x}^*)$

Factorisation de $\mathbf{A}_J$ et **C**

Les matrices $\mathbf{A}_J$ et **C** peuvent être factorisées de la façon suivante.

Par des transformations de Householder, on trouve une matrice orthogonale **Q**(nxn) et une matrice triangulaire inférieure **L**(μxμ) t.q.

$$\mathbf{A}_J\mathbf{Q} = (\mathbf{L} : \mathbf{0}) \tag{2.16}$$

On détermine ensuite une matrice **E**(pxμ) et une matrice **F**(pxn-μ)t.q.

$$\mathbf{CQ} = (\mathbf{E} : \mathbf{F}) \tag{2.17}$$

Ensuite on factorise **F** par la procédure de Gram-Schmidt (Daniel,Gragg,Kaufman and Stewart, 1976) t.q.

$$\mathbf{F} = \mathbf{VR} \tag{2.18}$$

où **V**(pxn-μ) est une matrice dont les colonnes sont orthogonales et **R**(n-μxn-μ) est une matrice triangulaire supérieure.

Supposons que l'ensemble actif est déterminé, le problème (P1) est réduit momentanément au sous-problème suivant:

$$\begin{cases} \min \| \mathbf{Cx-d} \|_2 \\ \text{s.c. } \mathbf{A_J x = b_J} \end{cases} \qquad \text{(P2)}$$

Ce sous problème (P2) peut être résolu en une étape.

Résolution du sous problème.

Posons $\mathbf{x = x^*+s}$ où $\mathbf{x^*}$ est admissible pour le sous problème (P2) Ce problème est alors équivalent à

$$\begin{cases} \min \| \mathbf{Cx^*+Cs-d} \| \\ \text{s.c. } \mathbf{A_J x^*+A_J s = b_J} \end{cases}$$

Comme $\mathbf{x^*}$ est admissible, $\mathbf{A_J x^* = b_J}$ et en posant $\mathbf{g = d-Cx}$, on obtient

$$\begin{cases} \min \| \mathbf{Cs-g} \| \\ \text{s.c. } \mathbf{A_J s = 0} \end{cases}$$

En posant $\mathbf{s = Qw}$ où $\mathbf{w} = \begin{pmatrix} \mathbf{w_1} \} \mu \\ \mathbf{w_2} \} n-\mu \end{pmatrix}$ on obtient (2.19)

$$\begin{cases} \min \| \mathbf{CQw-g} \| \\ \text{s.c. } \mathbf{A_J Qw = 0} \end{cases}$$

En appliquant les équations (2.16) et (2.17), on a

$$\begin{cases} \min \| (\mathbf{E} : \mathbf{F}) \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix} - \mathbf{g} \| \\ \text{s.c. } (\mathbf{L} : \mathbf{0}) \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{pmatrix} = \mathbf{0} \end{cases}$$

Ce qui est équivalent à

$$\begin{cases} \min \| \mathbf{E}\mathbf{w}_1 + \mathbf{F}\mathbf{w}_2 - \mathbf{g} \| \\ \text{s.c. } \mathbf{L}\mathbf{w}_1 = \mathbf{0} \end{cases}$$

L est non singulière car, par hypothèse, **A** est non dégénérée
Pour que $\mathbf{L}\mathbf{w}_1 = \mathbf{0}$,il faut donc que $\mathbf{w}_1 = \mathbf{0}$, par caractérisation de non singularité.
On obtient alors, en posant $\mathbf{z} = \mathbf{w}_2$

$$\{ \min \| \mathbf{F}\mathbf{z} - \mathbf{g} \| \qquad \text{(P3)}$$

Le problème (P2) est donc équivalent au problème (P3)
La solution du problème (P3) est caractérisée par

$$\mathbf{F}^T(\mathbf{F}\mathbf{z} - \mathbf{g}) = \mathbf{0} \qquad (2.20)$$

$$<=> \mathbf{F}^T\mathbf{F}\mathbf{z} = \mathbf{F}^T\mathbf{g}$$

$$<=> \mathbf{R}^T\mathbf{V}^T\mathbf{V}\mathbf{R}\mathbf{z} = \mathbf{R}^T\mathbf{V}^T\mathbf{g} \quad \text{par(2.18)}$$

$$<=> \mathbf{R}^T(\mathbf{V}^T\mathbf{V}\mathbf{R}\mathbf{z} - \mathbf{V}^T\mathbf{g}) = \mathbf{0}$$

$$<=> \mathbf{R}\mathbf{z} = \mathbf{V}^T\mathbf{g} \qquad (2.21)$$

car $\mathbf{R}^T$ est non singulière car **C** est non singulière par hypothèse. $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ car **V** a ses colonnes orthogonales.

$$<=> \mathbf{z} = \mathbf{R}^{-1}\mathbf{V}^T\mathbf{g} \qquad (2.22)$$

En posant $\mathbf{h}^* = \mathbf{V}^T\mathbf{g}$, on a que (2.23)

$$\mathbf{z} = \mathbf{R}^{-1}\mathbf{h}^* \tag{2.24}$$

est la solution du problème (P3)
Comme $\mathbf{x} = \mathbf{x}^*+\mathbf{s}$ et par (2.19), on a que

$$\mathbf{x} = \mathbf{x}^*+\mathbf{Q}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix} \tag{2.25}$$

est la solution du problème (P2)

Condition nécessaire et suffisante pour que $\mathbf{x}$ soit solution de (P1)

On a une condition nécessaire et suffisante pour que la solution $\mathbf{x}$ du sous-problème (P2) soit solution du problème (P1)

Théorème 2.5

$\mathbf{x}^*$ est optimale <=> le vecteur $\mathbf{v} = (\mathbf{L}^T)^{-1}\mathbf{E}^T\mathbf{h}$ satisfait $\mathbf{S}_j\mathbf{v} \geq \mathbf{0}$

où $\mathbf{h} = (\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}$

$\mathbf{S}_j = \mathrm{diag}(\rho_{j1}\ldots\rho_{j\mu})$

Démonstration

Par les conditions d'optimalité d'un problème quadratique (2.8)
$\mathbf{x}^*$ est optimale <=> $\exists$ $\mathbf{v}$ avec $\begin{cases}\mathbf{C}^T(\mathbf{C}\mathbf{x}^*-\mathbf{d})+\mathbf{A}_J^T\mathbf{v} = \mathbf{0}\\ \mathbf{S}_j\mathbf{v} \geq \mathbf{0}\end{cases}$

On a l'équivalence car les contraintes sont linéaires.

Par (2.19) on a que

$$\mathbf{Cx^*-d = Cs-g = CQ}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix}\mathbf{-g}$$

$$\mathbf{= (E : F)}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix}\mathbf{-g} \qquad \text{par(2.17)}$$

$$\mathbf{= Fz-g}$$

$$\mathbf{= VRR^{-1}V^Tg-g} \qquad \text{par(2.18) et (2.22)}$$

$$\mathbf{= (VV^T-I)g}$$

$$\mathbf{= -h} \qquad \text{par hypothèse}$$

$$\Rightarrow \mathbf{C^T(Cx^*-d) = -C^Th}$$

$$\mathbf{= -QQ^TC^Th} \qquad \mathbf{QQ^T = I}$$

$$\mathbf{= -Q(E : F)^Th} \qquad \text{par(2.17)}$$

$$\mathbf{= -Q(E : VR)^Th} \qquad \text{par(2.18)}$$

$$\mathbf{= -Q}\begin{pmatrix}\mathbf{E^Th}\\ \mathbf{0}\end{pmatrix}$$

car $\mathbf{R^TV^Th = R^TV^T(I-VV^T)g}$

$$\mathbf{= (R^TV^T-R^TV^TVV^T)g} \qquad \mathbf{VV^T = I}$$

$$\mathbf{= (R^TV^T-R^TV^T)g}$$

$$\mathbf{= 0}$$

$$\Rightarrow \mathbf{C^T(Cx^*-d)+A_J^Tv = -Q}\begin{pmatrix}\mathbf{E^Th}\\ \mathbf{0}\end{pmatrix}\mathbf{+Q}\begin{pmatrix}\mathbf{L^T\ v}\\ \mathbf{0}\end{pmatrix} \qquad \text{par(2.16)}$$

$$\mathbf{= Q}\begin{pmatrix}\mathbf{L^Tv-E^Th}\\ \mathbf{0}\end{pmatrix}$$

$$\Rightarrow \mathbf{C^T(Cx^*-d)+A_J^Tv = 0}$$

$$\Leftrightarrow \mathbf{Q}\begin{pmatrix}\mathbf{L^Tv-E^Th}\\ \mathbf{0}\end{pmatrix}\mathbf{= 0}$$

$$\Leftrightarrow \mathbf{L^Tv-E^Th = 0} \qquad \text{car } \mathbf{Q} \text{ est non singulière}$$

$$\Leftrightarrow \mathbf{L^Tv = E^Th}$$

$$\Leftrightarrow \mathbf{v = (L^T)^{-1}E^Th}$$

Définition 2.3

Un tableau $\mathcal{M} = \{\mathbf{x},\mathbf{J},\mathbf{Q},\mathbf{L},\mathbf{V},\mathbf{R},\mathbf{E},\mathbf{h},\mathbf{h}^*\}$ est dit admissible s'il satisfait les conditions suivantes:

1) $\mathbf{x} \in \mathbf{P}$ i.e. $\mathbf{x}$ est admissible.

2) $J = (j_1, \ldots, j_\mu)$ est une base i.e. J contient toutes les contraintes actives en $\mathbf{x}$.

3) La matrice orthogonale $\mathbf{Q}(nxn)$ et la matrice triangulaire inférieure $\mathbf{L}(\mu x\mu)$ sont t.q. $\mathbf{A}_J\mathbf{Q} = (\mathbf{L} : \mathbf{0})$

4) La matrice $\mathbf{V}(pxn\text{-}\mu)$ avec ses colonnes orthonormées et la matrice triangulaire supérieure $\mathbf{R}(n\text{-}\mu xn\text{-}\mu)$ sont t.q. $\mathbf{F} = \mathbf{VR}$

où $\mathbf{F}$ est définit par $\mathbf{CQ} = (\mathbf{E} : \mathbf{F})$

5) $\mathbf{h} = (\mathbf{I}\text{-}\mathbf{V}\mathbf{V}^T)\mathbf{g}$ où $\mathbf{g} = \mathbf{d}\text{-}\mathbf{Cx}$

6) $\mathbf{h}^* = \mathbf{V}^T\mathbf{g}$

2.3.3 Présentation de la méthode

Cette partie décrit la méthode de Stoer et Schittkowski pour résoudre les problèmes de moindres carrés linéaires avec contraintes linéaires (P1).

Supposons que l'ensemble actif est déterminé en un point admissible $\mathbf{x}^*$. La recherche du minimum est temporairement restreint à un sous-espace définit par les contraintes actives.

On obtient alors un sous-problème (P2) dont la solution est $\mathbf{x} = \mathbf{x}^* + \mathbf{s}$ où $\mathbf{s} = \mathbf{Q}\begin{pmatrix} \mathbf{z}^* \\ \mathbf{0} \end{pmatrix}$

On peut observer qu'en bougant en $\mathbf{x}^*+\mathbf{s}$, certaines contraintes inactives peuvent être violées.

On considérera donc $\mathbf{s}$ comme une direction de recherche et on se déplacera dans cette direction jusqu'à ce qu'on rencontre une contrainte ou jusqu'à ce qu'on arrive en $\mathbf{x}^*+\mathbf{s}$.

On se déplacera donc jusqu'en $\mathbf{x}^*+\alpha\mathbf{s}$

où $\alpha = \max\{\alpha \text{ t.q.} \mathbf{x}^*+\alpha\mathbf{s} \text{ est admissible}, \alpha \leq 1\}$

Si une contrainte est rencontrée, on se déplace jusqu'en ce point, on ajoute cette contrainte à l'ensemble actif et on recommence avec le nouveau sous-problème (P2) correspondant au nouvel ensemble actif.

Si aucune contrainte n'est rencontrée dans la direction $\mathbf{s}$, on se déplace jusqu'en $\mathbf{x}^*+\mathbf{s}$ qui est la solution du sous-problème (P2).

Si cette solution satisfait les conditions nécessaire et suffisante d'optimalité (théorème 2.5), elle est la solution du problème (P1).

Si cette condition est violée pour un i, on peut enlever la contrainte i de l'ensemble actif en étant assuré que la valeur de la fonction objectif diminuera en recommençant avec le nouveau sous-problème correspondant au nouvel ensemble actif.

2.3.4. Présentation de l'algorithme.

L'algorithme pour résoudre le problème (P_1) consiste en deux phases:

Si un point admissible n'est pas donné par l'utilisateur, la phase 1 en génère un.

S'il n'existe pas de point admissible, la phase 1 sort de l'algorithme.

Une fois qu'un point admissible est déterminé, la phase 2 trouve une série de points admissibles juqu'à ce que la solution optimale soit trouvée.

Dans l'algorithme, les matrices $\mathbf{Q},\mathbf{L},\mathbf{V},\mathbf{R},\mathbf{E},\mathbf{h},\mathbf{h}^*$ changent quand l'ensemble actif est modifié.

Il n'est pas nécessaire de faire la factorisation des matrices à chaque changement, on peut les réajuster par des transformations de Givens après avoir fait une seule factorisation initiale. Une fois que le minimun sera trouvé, une seconde factorisation sera faite afin de vérifier la justesse de la solution.

Algorithme LCLSQ

Phase 1: Générer un point admissible, $\mathbf{x}_O$

Poser i = 0

Phase 2: 1. Déterminer un tableau admissible $\mathcal{M}\{\mathbf{x},\mathbf{J},\mathbf{Q},\mathbf{L},\mathbf{V},\mathbf{R},\mathbf{E},\mathbf{h},\mathbf{h}^*\}$

2. Déterminer la direction de recherche **s**

3. Déterminer le longueur de pas α

Bouger en $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha\mathbf{s}$

si une contrainte est rencontrée

alors aller en 4

sinon aller en 5

4. Ajouter la contrainte rencontrée à l'ensemble actif

Ajuster **Q,L,V,R,E,h**, et $\mathbf{h}^*$ par des rotations de Givens.

Poser i = i+1

Aller en 2

5. Calculer les multiplicateurs de Lagrange **v**

Si la condition est violée pour i,

enlever la contrainte i de l'ensemble actif

ajuster **Q,L,V,R,E,h**,et $\mathbf{h}^*$

aller en 2

sinon aller en 6.

6. Si une contrainte a été ajoutée ou enlevée,

redécomposér les matrices $\mathbf{A}_J$ et **C** pour obtenir

Q,L,V,R,E,h,et $\mathbf{h}^*$

aller en 2

Sinon sortir.

Détail de l'algorithme

Phase 1: Générer un point admissible

L'algorithme pour générer un point admissible est donné par (Schittkowski,1969). Il consiste en deux parties.

Dans la première partie, on définit un ensemble initial

K = { indices des contraintes satisfaites comme égalité }.

S'il n'y a pas de contraintes d'égalité, on prend K = {1}.

On trouve ensuite un **x** qui satisfait les contraintes qui sont un dans K comme égalité et donc l'ensemble actif J = K.

En employant des transformations de Householder on détermine **Q** et **L** satisfaisant l'équation (2.16)

Un premier point admissible pour l'ensemble K est alors donné par

$$\mathbf{x} = \mathbf{Q}\begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} \tag{2.26}$$

où $\mathbf{Ly} = \mathbf{b}_J$ (2.27)

La deuxième partie de l'algorithme ajoute les contraintes qu'on n'a pas encore traitées à l'ensemble K. Supposons qu'on ajoute la contrainte i.

Si le **x** courant, noté **x***, satisfait cette contrainte son indice i est ajouté à K.

Si la contrainte est inactive en **x***, on ajoute la contrainte à J et **Q** et **L** doivent être ajustées.

Si **x*** ne satisfait pas la contrainte, on cherche un nouveau point qui satisfait les contraintes de K aussi bien que la contrainte i.

On fait ceci en résolvant le problème linéaire suivant

$$\begin{cases} \min \pm \mathbf{a}_i\mathbf{x} \\ \text{s;c;} \quad \mathbf{A}_K\mathbf{x}\ \rho_K\ \mathbf{b}_K \end{cases} \tag{2.28}$$

où le signe de la fonction objectif est + si ρ_i est "$\leq$" et - si ρ_i est "$\geq$".

La solution du problème (2.28) est un point qui satisfait la contrainte i aussi bien que les contraintes de K.

On remarque que $\mathbf{x}^*$ est admissible pour les contraintes de (2.28).

De la théorie de programmation linéaire (Zimmermann, 1977), la solution de (2.28) est un point fini $\mathbf{x}^*$, où il y a un point $\mathbf{x}^*$ et une direction $\mathbf{s}$ pour lesquels $\mathbf{x}^*+\beta\mathbf{s}$ est admissible $\forall\, \beta \geq 0$ et pour lequel la fonction objectif de (2.28) peut décroître jusqu'à $-\infty$ quand β croît.

Si le dernier cas arrive, il est possible de choisir un β pour lequel la contrainte est satisfaite comme égalité.

Dans ce cas, on peut ajouter 1 à K et à l'ensemble actif J et les matrices $\mathbf{Q}$ et $\mathbf{L}$ doivent être ajustées. On peut alors considérer une nouvelle contrainte qui n'est pas encore dans K.

Si la solution de (2.28) est un point fini $\mathbf{x}^*$, il y a deux cas:

Si $\mathbf{x}^*$ ne satisfait pas la i^e contrainte, il n'y a pas de $\mathbf{x}$ admissible pour les contraintes de KU{i} et donc pas de point admissible pour (P_1)

Dans le cas contraire l'algorithme peut continuer.

Si la contrainte i est inactive en $\mathbf{x}^*$, on ajoute i à K et on considère une autre contrainte .

Si la contrainte i est active en $\mathbf{x}^*$, on ajoute i à K et à J , puis on ajuste les matrices $\mathbf{Q}$ et $\mathbf{L}$ et on considère une autre contrainte.

L'algorithme pour trouver un point admissible de (P1) peut être résumé comme suit:

(2.1)

1epartie: Déterminer K={indices des contraintes d'égalité}

S'il n'y a pas de contrainte d'égalité K={1}

Poser J = K

Déterminer les matrices **Q** et **L**

Déterminer **y** et $\mathbf{x}_o$ par les équations (2.26) et (2.27)

2epartie: Boucler sur toutes les contraintes i n'appartenant pas à K

1. Si la contrainte i est admissible en $\mathbf{x}_0$ mais inactive

   ajouter i à K

   considérer la contrainte suivante
2. Si la contrainte i est admissible et active en $\mathbf{x}_o$

   aller en 4
3. Si la contrainte i est inadmissible en $\mathbf{x}_o$,

   résoudre le problème (2.28) (change $\mathbf{x}_o$,J,**Q** et **L**)

   considère la solution $\mathbf{x}_o$

   a. Si le problème (2.28) a une solution infinie le long du vecteur **s**, bouger sur une distance de $(\mathbf{b}_i-\mathbf{a}_i\mathbf{x}_0)/\mathbf{a}_i\mathbf{s}$ le long du vecteur **s**.

      aller en 4

   b. Si le problème (2.28) a une solution finie

   1°) Si la contrainte i est inadmissible en $\mathbf{x}_0$,

      sortir de l'algorithme

   2°) Si la contrainte i est satisfaite en $\mathbf{x}_0$ mais inactive,

      ajouter i à K

      considérer une autre contrainte

   3°) Si la contrainte i est admissible et active en $\mathbf{x}_0$,aller en 4
4. Ajouter i à K et à J

   Ajuster les matrices **Q** et **L** par des transformations de Givens

   Considérer une autre contrainte.

Phase 2.1 Déterminer un tableau admissible initial

L'algorithme pour déterminer un premier tableau admissible peut être résumé de la façon suivante. (2.2)

1. Calculer une base $J=\{j_1,\ldots,j_\mu\}$ contenant toutes les contraintes actives en $\mathbf{x}_0$
2. En employant des transformations de Householder, déterminer une matrice orthogonale $\mathbf{Q}$ (nxn) et une matrice triangulaire inférieure $\mathbf{L}$ (μxμ) avec
   $$\mathbf{A}_J\mathbf{Q}=(\mathbf{L} : \mathbf{0})$$
3. Poser $\mathbf{g} = \mathbf{d}-\mathbf{Cx}$ et $\mathbf{CQ} = (\mathbf{E} : \mathbf{F})$
4. Employant la procédure de Gram-Schmidt, calculer une matrice $\mathbf{V}$ (pxn-μ) et une matrice triangulaire supérieure $\mathbf{R}$ (n-μ)x(n-μ) avec
   $$\mathbf{F}=\mathbf{VR} \qquad \mathbf{V}^T\mathbf{V}=\mathbf{I}$$
5. Employer la même procédure de Gram-Schmidt pour calculer $\mathbf{h}'$, $\mathbf{h}$ et λ t.q.
   $$(\mathbf{V} : \mathbf{g}) = (\mathbf{V} : \mathbf{h}')\begin{pmatrix} \mathbf{I} & \mathbf{h}^* \\ 0 & \lambda \end{pmatrix}; \quad \mathbf{V}^T\mathbf{h}' = \mathbf{0}; \quad \mathbf{h} = \lambda\mathbf{h}'$$

Nous avons alors, avec l'étape 5

$$\mathbf{h}^* = \mathbf{V}^T\mathbf{g}$$

$$\mathbf{h}' = 1/\lambda\,(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}$$

$$\lambda = \| (\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g} \|$$

Pour la procédure de Gram-Schimdt, voir (Daniel, Gragg, Kaufman et Stewart, 1976)

Phase 2.4. Ajouter une contrainte et ajuster le tableau $\mathcal{M}$

Supposons, sans perte de généralité, qu'une seule nouvelle contrainte devient active.

On a alors

$$\mathbf{x}'' = \mathbf{x}+\alpha\mathbf{s} \qquad \text{avec } 0 \le \alpha \le 1 \ , \quad \mathbf{s} = \mathbf{Q}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix} \ , \quad \mathbf{Rz} = \mathbf{h}^*$$

$$J'' = \{j_1,\ldots,j_\mu,k\}$$

$$\mathbf{A}_{J''}\mathbf{Q} = \begin{pmatrix}\mathbf{A}_J\\ \mathbf{a}_k\end{pmatrix}\mathbf{Q} = \begin{pmatrix}\mathbf{L} & \mathbf{0}\\ \mathbf{a}'_k & \mathbf{a}^*_k\end{pmatrix}$$

Les formules pour réajuster le tableau sont (2.3):

1. Calculer les matrices de Givens $\mathbf{H}_{\mu+1,j}$, $j=\mu+2,..,n$ t.q.

$$\mathbf{H}^T\begin{pmatrix}\mathbf{L} & \mathbf{0}\\ \mathbf{a}'_k & \mathbf{a}^*_k\end{pmatrix}^T = \mathbf{H}_{\mu+1,n} \ldots \mathbf{H}_{\mu+1,\mu+2}\begin{pmatrix}\mathbf{L} & \mathbf{0}\\ \mathbf{a}'_k & \mathbf{a}^*_k\end{pmatrix}^T$$

$$= (\mathbf{L}'' : \mathbf{0})^T$$

où $\mathbf{L}^*$ est une matrice triangulaire inférieure $(\mu+1)x(\mu+1)$

Poser $\mathbf{Q}'' = \mathbf{QH}$

2. Considérer la matrice orthogonale $(n-\mu)x(n-\mu)$ $\mathbf{H}''$ définie par

$$\mathbf{H} = \begin{pmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{H}^*\end{pmatrix}$$

et déterminer un vecteur $\mathbf{u}^*$ de dimension $n-\mu$ et une matrice $\mathbf{R}^*$ $(n-\mu)x(n-\mu-1)$par

$$\mathbf{RH}'' = (\mathbf{u}^* : \mathbf{R}^*)$$

3. Calculer les matrices de Givens $\mathbf{H}'_{j,j+1}$, $j=1,...,n-\mu-1$ avec

$$\mathbf{H}'\mathbf{R}^* = \mathbf{H}'_{n-\mu-1,n-\mu}\ldots\mathbf{H}'_{1,2}\mathbf{R}^* = \begin{pmatrix}\mathbf{R}''\\ \mathbf{0}\end{pmatrix}$$

où $\mathbf{R}''$ est une matrice triangulaire supérieure$(n-\mu-1)x(n-\mu-1)$

4. Poser $\mathbf{E}'' = (\mathbf{E} : \mathbf{Vu}^*)$ et $(\mathbf{V}'' : \mathbf{v}'') = \mathbf{VH}'^T$ avec une matrice $\mathbf{V}''$ $px(n-\mu-1)$ et un vecteur $\mathbf{v}''$ de dimension p

5. Déterminer un vecteur $\mathbf{h}^{*''}$ de dimension $n-\mu-1$ et un réel λ avec

$$\mathbf{h}^{*''} = (1-\alpha)\mathbf{H}'\mathbf{h}^*$$

6. Poser $\mathbf{h}'' = \mathbf{h}+\lambda\mathbf{v}''$

On montrera dans la section suivante que le tableau généré de cette manière est admissible

Phase 2.5 Enlever la contrainte et ajuster le tableau $\mathcal{M}$

On peut supposer qu'on enlève qu'une contrainte à la fois

On a alors :

$$\mathbf{x''} = \mathbf{x}+\mathbf{s} \quad , \quad \mathbf{s} = \mathbf{Q}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix} \quad , \quad \mathbf{Rz} = \mathbf{h}^*$$

$$J'' = \{j_1,\ldots,j_{r-1},j_{r+1},\ldots,j_\mu\}$$

$$\mathbf{A}_{J''}\mathbf{Q} = (\mathbf{a}_{j1}{}^T..\mathbf{a}_{jr-1}{}^T\mathbf{a}_{jr+1}{}^T...\mathbf{a}_{j\mu}{}^T)^T\mathbf{Q}$$

Les formules pour réajuster le tableau sont (2.4):

1. Calculer les matrices de Givens $\mathbf{H}_{j,j+1}$, j=r, ,μ-1 avec

   $\mathbf{H}\,(\mathbf{A}_J\mathbf{Q})^T = \mathbf{H}_{\mu-1,\mu}...\mathbf{H}_{r,r+1}(\mathbf{A}_J\mathbf{Q})^T = (\mathbf{L''} : \mathbf{0})^T$

   où **L''** est une matrice triangulaire inférieure (μ-1)x(μ-1)

2. Définir la matrice orthogonale **H'** par

   $$\mathbf{H} = \begin{pmatrix}\mathbf{H'} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}$$

   Noter la dernière ligne de **H'**, $\mathbf{h'}^T$

   Calculer $\mathbf{EH'}^T = (\mathbf{E''} : \mathbf{Eh'})$

3. Soit **P**, la matrice de permutation interchangant les colonnes μ et n

   Poser $\mathbf{Q''} = \mathbf{QH}^T\mathbf{P}$

4. Appliquer une procédure de Gram-Schmidt pour calculer **v**,**r** et σ avec

   $$(\mathbf{V} : \mathbf{Eh'}) = (\mathbf{V} : \mathbf{v})\begin{pmatrix}\mathbf{I} & \mathbf{r}\\ 0 & \sigma\end{pmatrix} \qquad \mathbf{V}^T\mathbf{v} = \mathbf{0}$$

   Poser $\mathbf{R''} = \begin{pmatrix}\mathbf{I} & \mathbf{r}\\ 0 & \sigma\end{pmatrix}$

   $\mathbf{V''} = (\mathbf{V} : \mathbf{v})$

5. Noter $\mathbf{h''}^* = \begin{pmatrix}\mathbf{0}\\ \mathbf{v}^T\mathbf{h}\end{pmatrix}$

6. Calculer $\mathbf{h''} = \mathbf{h}-\mathbf{v}^T\mathbf{hv}$

La procédure de Gram-Schmidt est expliquée dans (Daniel, Gragg, Kaufman, Stewart,1976).

On montrera dans la section suivante que le tableau généré par cet algorithme est admissible.

2.3.5 Admissibilité des tableaux ajustés et convergence de l'algorithme

Théorème 2.6

Le tableau $\mathcal{M}'' = \{\mathbf{x}'', \mathbf{J}'', \mathbf{Q}'', \mathbf{L}'', \mathbf{V}'', \mathbf{R}'', \mathbf{E}'', \mathbf{h}'', \mathbf{h}^{*''}\}$, calculé par l'algorithme (2.3) est admissible dans le sens de la définition (2.3)

Démonstration

De la première étape, on a que

$$\mathbf{A_J Q''} = \mathbf{A_J QH} = (\mathbf{L''} : \mathbf{0})$$

De plus,on a que

$$\mathbf{CQ''} = \mathbf{CQH}$$

$$= (\mathbf{E} : \mathbf{VR}) \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}^* \end{pmatrix} \qquad \text{par l'étape 2}$$

$$= (\mathbf{E} : \mathbf{VRH}^*)$$

$$= (\mathbf{E} : \mathbf{V}(\mathbf{u}^* : \mathbf{R}^*)) \qquad \text{par l'étape 2}$$

$$= (\mathbf{E} : \mathbf{Vu}^* : \mathbf{VR}^*)$$

de l'étape 3 on a que $\mathbf{H'R}^* = \begin{pmatrix} \mathbf{R''} \\ \mathbf{0} \end{pmatrix}$

comme $\mathbf{H}'$ est orthogonale, $\mathbf{H'}^{-1} = \mathbf{H'}^{T}$

$$\Rightarrow \mathbf{R}^* = \mathbf{H'}^{T} \begin{pmatrix} \mathbf{R''} \\ \mathbf{0} \end{pmatrix}$$

$$= \mathbf{E} : \mathbf{Vu}^* : \mathbf{VH'}^{T} \begin{pmatrix} \mathbf{R''} \\ \mathbf{0} \end{pmatrix}$$

$$= \mathbf{E''} : (\mathbf{V''} : \mathbf{v''}) \begin{pmatrix} \mathbf{R''} \\ \mathbf{0} \end{pmatrix} \qquad \text{par l'étape 4}$$

$$= (\mathbf{E''} : \mathbf{V''R''})$$

avec $\mathbf{I} = \mathbf{H'V^TVH'^T}$

$= (\mathbf{V''} : \mathbf{v''})(\mathbf{V''} : \mathbf{v''})$

$= \begin{matrix} \mathbf{V''^TV''} & \mathbf{V''^Tv''} \\ \mathbf{v''^TV''} & \mathbf{v''^Tv''} \end{matrix}$

$\Rightarrow \mathbf{V''^TV''} = \mathbf{I}$

Et comme on a

$(\mathbf{V''} : \mathbf{v''})^T\mathbf{g''} = \mathbf{H'V^T(d-Cx'')}$ par l'étape 4 et par définition de $\mathbf{g''}$

$= \mathbf{H'V^T(d-Cx-\alpha Cs)}$ par définition de $\mathbf{x} = \mathbf{x''}+\alpha\mathbf{s}$

$= \mathbf{H'h^*-\alpha H'V^TCs}$ par définition de $\mathbf{h^*} = \mathbf{V^T(d-Cx)}$

$= \mathbf{H'h^*-\alpha H'V^TCQ}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix}$ par définition de $\mathbf{s} = \mathbf{Q}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix}$

$= \mathbf{H'h^*-\alpha H'V^TVRz}$ car $\mathbf{CQ} = (\mathbf{E} : \mathbf{VR})$

$= \mathbf{H'h^*-\alpha H'Rz}$ car $\mathbf{V^TV} = \mathbf{I}$

$= \mathbf{H'h^*-\alpha H'h^*}$ car $\mathbf{Rz} = \mathbf{V^Tg} = \mathbf{h^*}$

$= (1-\alpha)\mathbf{H'h^*}$

$= \mathbf{h^{*''}}$ par l'étape 5

$= \lambda$

On obtient

$\mathbf{V''^Tg''} = \mathbf{h''^*}$

Et $\mathbf{g''} = \mathbf{d-Cx''}$ par définition de $\mathbf{g}$

$= \mathbf{d-Cx-\alpha Cs}$ par définition de $\mathbf{x} = \mathbf{x}+\alpha\mathbf{s}$

$= \mathbf{g-\alpha Cs}$ par définition de $\mathbf{g}$

$= \mathbf{g-\alpha CQ}\begin{pmatrix}\mathbf{0}\\ \mathbf{z}\end{pmatrix}$ par définition de $\mathbf{s}$

$= \mathbf{g-\alpha Fz}$ car $\mathbf{CQ} = (\mathbf{E} : \mathbf{F})$

$= \mathbf{g-\alpha VRz}$ car $\mathbf{F} = \mathbf{VR}$

$= \mathbf{g-\alpha VV^Tg}$ car $\mathbf{Rz} = \mathbf{V^Tg}$

$= \alpha(\mathbf{I-VV^T})\mathbf{g}+(1-\alpha)\mathbf{g}$

$= \alpha\mathbf{h}+(1-\alpha)\mathbf{g}$ par définition de $\mathbf{h}$

ainsi que

$\mathbf{V} = (\mathbf{V}'' : \mathbf{v}'')\mathbf{H}'$ par l'étape 4

$\mathbf{V}\mathbf{V}^T = (\mathbf{V}'' : \mathbf{v}'')\mathbf{H}'\mathbf{H}'^T(\mathbf{V}'' : \mathbf{v}'')$

$= (\mathbf{V}'' : \mathbf{v}'')(\mathbf{V}'' : \mathbf{v}'')^T$ $\mathbf{H}'\mathbf{H}'^T = \mathbf{H}'\mathbf{H}'^{-1} = \mathbf{I}$ car $\mathbf{H}$ orthogonale

$= \mathbf{V}''\mathbf{V}''^T + \mathbf{v}''\mathbf{v}''^T$

On a également

$\mathbf{v}''^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T) = \mathbf{h}'\mathbf{V}^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T)$ où $\mathbf{h}'$ est la dernière colonne de $\mathbf{H}'$

$= \mathbf{h}'(\mathbf{V}^T-\mathbf{V}^T\mathbf{V}\mathbf{V}^T) = 0$ (*)

On en conclut que

$(\mathbf{I}-\mathbf{V}''\mathbf{V}''^T)\mathbf{g}'' = (\mathbf{I}-\mathbf{V}\mathbf{V}^T+\mathbf{v}''\mathbf{v}''^T)(\alpha\mathbf{h}+(1-\alpha)\mathbf{g})$

$= \alpha(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{h}+\alpha\mathbf{v}''\mathbf{v}''^T\mathbf{h}+(1-\alpha)(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}+(1-\alpha)\mathbf{v}''\mathbf{v}''^T\mathbf{g}$

$= \alpha(\mathbf{I}-\mathbf{V}\mathbf{V}^T)^2\mathbf{g}+\alpha\mathbf{v}''\mathbf{v}''^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}+(1-\alpha)(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}+$

$(1-\alpha)\mathbf{v}''\mathbf{v}''^T\mathbf{g}$ $\mathbf{h} = (\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}$

$= \alpha(\mathbf{I}-\mathbf{V}\mathbf{V}^T)^2\mathbf{g}+(1-\alpha)(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}+(1-\alpha)\mathbf{v}''\mathbf{v}''^T\mathbf{g}$ par (*)

$= \alpha(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}+(1-\alpha)\mathbf{h}+(1-\alpha)\mathbf{v}''\mathbf{v}''^T\mathbf{g}$

$= \alpha\mathbf{h}+(1-\alpha)\mathbf{h}+(1-\alpha)\mathbf{v}''\mathbf{v}''^T\mathbf{g}$

$= \mathbf{h}+(1-\alpha)\mathbf{v}''\mathbf{h}'\mathbf{V}^T\mathbf{g}$ par l'étape 4

$= \mathbf{h}+(1-\alpha)\mathbf{v}''\mathbf{h}'\mathbf{V}^T(\mathbf{V}\mathbf{h}^*+\mathbf{h}'\alpha)$ par l'étape5 de (2.2)

$= \mathbf{h}+(1-\alpha)\mathbf{v}''\mathbf{h}'\mathbf{V}^T\mathbf{V}\mathbf{h}^*+\alpha(1-\alpha)\mathbf{v}''\mathbf{h}'\mathbf{V}^T\mathbf{h}'$

$= \mathbf{h}+(1-\alpha)\mathbf{v}''\mathbf{h}'\mathbf{h}^*$ par l'étape 5 de (2.2)

$= \mathbf{h}+\lambda\mathbf{v}''$ par l'étape 6

$= \mathbf{h}''$

Théorème 2.7

Le tableau $\mathcal{M} = \{\mathbf{x}'', \mathbf{J}'', \mathbf{Q}'', \mathbf{L}'', \mathbf{V}'', \mathbf{R}'', \mathbf{E}'', \mathbf{h}'', \mathbf{h}^{*''}\}$ calculé par l'algorithme (2.4) est admissible dans le sens de la définition (2.3)

Démonstration

$\mathbf{Q}''$ est une matrice orthogonale nxn avec

$$\mathbf{A_J Q}'' = \mathbf{A_J Q H^T P} \qquad \text{par l'étape 3}$$

$$= (\mathbf{L}'' : \mathbf{0})\mathbf{P} \qquad \text{par l'étape 1}$$

$$= (\mathbf{L}'' : \mathbf{0}) \qquad \text{car } \mathbf{P} \text{ interchange les colonnes } \mu \text{ et n qui ne sont pas dans } \mathbf{L}''$$

La propriété 4 de la définition vient de

$$\mathbf{CQ}'' = \mathbf{CQH^T P} \qquad \text{par l'étape 3}$$

$$= (\mathbf{E} : \mathbf{VR})\mathbf{H^T P} \qquad \text{par (2.17) et (2.18)}$$

$$= (\mathbf{E}'' : \mathbf{Eh}' : \mathbf{VR})\mathbf{P} \qquad \text{par l'étape 2}$$

$$= (\mathbf{E}'' : \mathbf{VR} : \mathbf{Eh}')$$

et

$$(\mathbf{VR} : \mathbf{Eh}') = (\mathbf{V} : \mathbf{Eh}')\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ 0 & 1 \end{pmatrix}$$

$$= (\mathbf{V} : \mathbf{v})\begin{pmatrix} \mathbf{I} & \mathbf{r} \\ 0 & \sigma \end{pmatrix}\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ 0 & 1 \end{pmatrix} \qquad \text{par l'étape 4}$$

$$= (\mathbf{V} : \mathbf{v})\begin{pmatrix} \mathbf{R} & \mathbf{r} \\ 0 & \sigma \end{pmatrix}$$

$$= \mathbf{V}''\mathbf{R}'' \qquad \text{par l'étape 4}$$

$$\Rightarrow \mathbf{CQ}'' = (\mathbf{E}'' : \mathbf{V}''\mathbf{R}'')$$

De plus

$$\mathbf{V''^T V''} = (\mathbf{V} : \mathbf{v})^T(\mathbf{V} : \mathbf{v}) \qquad \text{par l'étape 4}$$

$$= \begin{pmatrix} \mathbf{V}^T\mathbf{V} & \mathbf{V}^T\mathbf{v} \\ \mathbf{v}^T\mathbf{V} & \mathbf{v}^T\mathbf{v} \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ 0 & 1 \end{pmatrix} \qquad \text{par l'étape 4}$$

$$= \mathbf{I}$$

La condition 6 de la définition est vérifiée par la condition suivante:

$$\mathbf{V''^T g''} = \mathbf{V''^T(d - Cx'')} \qquad \text{par définition de } \mathbf{g''}$$

$$= \mathbf{V''^T(d - Cx - Cs)} \qquad \text{par définition de } \mathbf{x} = \mathbf{x''} + \mathbf{s}$$

$$= \mathbf{V''^T(g - Cs)} \qquad \text{par définition de } \mathbf{g}$$

$$= \mathbf{V''^T}\ \mathbf{g - CQ}\begin{pmatrix} \mathbf{0} \\ \mathbf{z} \end{pmatrix} \qquad \text{par définition de } \mathbf{s}$$

$$= \mathbf{V''^T}\ \mathbf{g - (E : VR)}\begin{pmatrix} \mathbf{0} \\ \mathbf{z} \end{pmatrix} \qquad \text{par (2.17) et (2.18)}$$

(2.29) $$= \mathbf{V''^T(g - VRz)}$$

$$= \mathbf{V''^T(g - VV^Tg)} \qquad \text{par (2.21)}$$

$$= (\mathbf{V} : \mathbf{v})^T(\mathbf{g - VV^Tg}) \qquad \text{par l'étape 4}$$

$$= (\mathbf{V} : \mathbf{v})^T(\mathbf{I - VV^T})\mathbf{g}$$

$$= (\mathbf{V} : \mathbf{v})^T\mathbf{h}$$

$$= \begin{pmatrix} \mathbf{V}^T\mathbf{h} \\ \mathbf{v}^T\mathbf{h} \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{0} \\ \mathbf{v}^T\mathbf{h} \end{pmatrix} \qquad \text{car } \mathbf{V}^T\mathbf{h} = \mathbf{0} \text{ par l'étape 5 de (2.2)}$$

$$= \mathbf{h''^*} \qquad \text{par l'étape 5}$$

Le point 5 de la définition est vérifié par

$$
\begin{aligned}
(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}'' &= (\mathbf{I}-(\mathbf{V}:\mathbf{v})(\mathbf{V}:\mathbf{v})^T)\mathbf{g}'' && \text{par l'étape 4}\\
&= (\mathbf{I}-(\mathbf{V}:\mathbf{v})(\mathbf{V}:\mathbf{v})^T)(\mathbf{g}-\mathbf{V}\mathbf{R}\mathbf{z}) && \text{par (2.29)}\\
&= (\mathbf{I}-(\mathbf{V}:\mathbf{v})(\mathbf{V}:\mathbf{v})^T)(\mathbf{g}-\mathbf{V}\mathbf{h}^*) && \text{par (2.21) et (2.23)}\\
&= (\mathbf{I}-\mathbf{V}\mathbf{V}^T-\mathbf{v}\mathbf{v}^T)(\mathbf{g}-\mathbf{V}\mathbf{h}^*)\\
&= (\mathbf{I}-\mathbf{V}\mathbf{V}^T)(\mathbf{g}-\mathbf{V}\mathbf{h}^*)-\mathbf{v}\mathbf{v}^T(\mathbf{g}-\mathbf{V}\mathbf{h}^*)\\
&= (\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}-(\mathbf{V}-\mathbf{V}\mathbf{V}^T\mathbf{V})\mathbf{h}^*-\mathbf{v}\mathbf{v}^T(\mathbf{g}-\mathbf{V}\mathbf{h}^*)\\
&= \mathbf{h}-\mathbf{v}\mathbf{v}^T(\mathbf{g}-\mathbf{V}\mathbf{h}^*)\\
&= \mathbf{h}-\mathbf{v}\mathbf{v}^T\mathbf{g}+\mathbf{v}\mathbf{v}^T\mathbf{V}\mathbf{h}^*\\
&= \mathbf{h}-\mathbf{v}\mathbf{v}^T\mathbf{g} && \text{car } \mathbf{v}^T\mathbf{V} = \mathbf{0} \text{ par l'étape 4}\\
&= \mathbf{h}-\mathbf{v}^T\mathbf{g}\mathbf{v} && \text{car } \mathbf{v}^T\mathbf{g} \text{ est un scalaire}\\
&= \mathbf{h}-\mathbf{v}^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\mathbf{g}\mathbf{v} && \mathbf{v}^T\mathbf{V} = \mathbf{0} \Rightarrow \text{on ne rajoute rien}\\
&= \mathbf{h}-\mathbf{v}^T\mathbf{h}\mathbf{v} && \text{par l'étape 6}
\end{aligned}
$$

Théorème de convergence (2.8)

Si α n'est pas égal à 0, à chaque itération; l'algorithme converge en un nombre fini d'itérations.

Démonstration

La preuve est faite par les remarques suivantes:

1. La fonction objectif $\| \mathbf{Cx-d} \|$ est réduite à chaque itération car la fonction est strictement convexe et α est non nul.

2. A chaque itération, les vecteurs $\mathbf{a}_i^T$, $i \in J(x^*)$ sont linéairement indépendants, par construction. En effet, on a montré qu'on gardait toujours un tableau admissible, et pour que le tableau soit admissible, il faut que J soit une base.

3. Après un nombre fini d'itérations, on a $\alpha = 1$. Ce qui est normal car on ne peut rajouter qu'un nombre fini de contraintes.

On arrive donc à un minimum **x** de (P2)

4. Ce **x** est l'unique minimum global de (P2)

5. On ne peut pas revenir sur un (P2) déjà traité car la fonction décroit strictement et il n'y a qu'un seul minimum global pour chaque (P2).

6. Il n'y a qu'un nombre fini de problème (P2).

3. IMPLEMENTATION DE L'ALGORITHME LCLSQ

Dans cette partie, nous allons d'abord examiner en détail les principales procédures de l'algorithme ainsi que les procédures étant modifiées dans la suite.
Par après, nous verrons les quelques changements qui ont été faits, ainsi que ceux que l'on pourrait faire afin de rendre l'algorithme plus général. Pour plus de détails voir (Crane, Garbow, Hillstrom, Minkoff, 1977).

3.1. LCLSQ

Cette procédure résoud le problème (P_1) de la façon expliquée dans la section 2.3.. Elle permet à l'utilisateur de donner un point admissible et vérifiera l'admissibilité de ce point.
Si l'utilisateur ne donne pas de point admissible ou si le point donné ne l'est pas, elle en génère un.
Elle contrôle le procédé pour arriver à la solution et redécomposer les matrices si c'est nécessaire.
Elle appelle les sous-routines FEACHK , FEASOL , FEATAB , LSQU .

Détail:

-Initialiser le compteur du nombre d'itérations

-Vérifier la consistance des paramètres d'entrée

-Si un point admissible est donné, vérifier s'il est admissible.

Le nombre de contraintes actives $\mu = 0$

si m est non nul, appeller FEACHK (si $m = 0$, on ne vérifie rien)

si le point est admissible, aller en 10

sinon prévenir l'utilisateur et continuer

-Essayer de générer un point admissible.

.si $m = 0$ $x(i) = 0 \quad \forall i$ est solution

sinon appeller FEASOL

.si il n'y a pas de solution admissible, sortir

10 -Copier C et D dans les tableaux de travail EF et H (afin de ne pas les changer)

-Générer un tableau admissible en appelant FEATAB

s'il n'y a pas de tableau admissible, sortir

-Calculer la solution de (P1) en appelant LSQU

-Si on ne sort pas normalement de LSQU, ou qu'une redécomposition n'est plus nécessaire (i.e. que la solution est trouvée) sortir

-Sinon aller en 10 pour redécomposer les matrices et améliorer la solution si c'est nécessaire.

Cette routine requiert MxN+IPxN+2xM+IP+N+3xNx(N+1)+IPx(N+3)-1 pour stocker les matrices et les vecteurs d'entrée A, C, B, IRHO, D, et X ainsi que les matrices de décompositions et de travail.

Et requiert un tableau d'entiers de longueur 2xN (contenant l'ensemble actif et un tableau de travail)

Elle requiert donc $3xN^2+4xN+2xIPx(N+2)+Mx(N+2)-1$ doubles précisions et 2xN entiers.

3.2. FEASOL

Cette sous-routine essaie de construire un point admissible pour un système de contraintes linéaires d'égalité et d'inégalité. Elle appelle la sous-routine LINPRG

Détail

-Calculer un premier point admissible pour toutes les contraintes d'égalité:

Boucle pour i=1,m

-mettre μ et l'ensemble actif à jour

-S'il n'y a pas de contraintes d'égalité, forcer la première contrainte à être active

-Calculer la décomposition orthogonale $A_J Q = (AR^T : 0)$

-Si $\exists$ j t.q. $AR(j,j) = 0$ (i.e. s'il existe des contraintes redondantes), on signale l'erreur et on sort

-Résoudre le système linéaire $AR^T y = b_J$

-Calculer le premier le premier point admissible $x = Q\begin{pmatrix} y \\ 0 \end{pmatrix}$

-Ajouter successivement les contraintes d'inégalités et vérifier l'admissibilité:

Boucle sur toutes les contraintes (i=1,m)

-Si i est une contrainte d'égalité passer à l'itération suivante

-Si i est une contrainte d'inégalité

-calculer $AX = a_i x - b_i$

-si la contrainte est satisfaite comme égalité (i.e. $AX = 0$),

-ajuster μ et l'ensemble actif

-ajuster les matrices AR et Q

-passer à l'itération suivante

-si la contrainte est satisfaite comme inégalité (i.e. $AX \rho 0$)

-passer à l'itération suivante

-sinon, (x n'est pas admissible pour la contrainte rajoutée)

-appeller LINPRG (résoud le problème linéaire afin de trouver un nouveau x admissible pour toutes les contraintes déjà traitées).

-vérifier si on est bien sorti de LINPRG , sinon signaler l'erreur et sortir

-si la contrainte est satisfaite comme égalité (si pas de solution finie pour LINPRG),

-ajuster μ et l'ensemble actif

-ajuster les matrices AR et Q

-sinon (la solution de LINPRG est finie), on ne fait rien

-passer à l'itération suivante

Fin de boucle.

3.3 FEATAB

Cette sous-routine calcule un tableau admissible de départ en employant les procédés de Householder et de Gram-Schmidt.Elle appelle les sous-routines ORTHOG et VRFACT

Détail

-Calculer $g = d-Cx$ et le mettre dans d

-Si il n'y a pas de contraintes actives ($\mu = 0$)

$Q = I$

$AR, E = 0$

$F = C$

-Sinon ($\mu \neq 0$)

-calculer la décomposition orthogonale $A_J Q = (AR^T : 0)$

-si $\exists$ j t.q. $AR(j,j)=0$ signaler l'erreur et sortir

-calculer $CQ = (E : F)$ et le mettre dans C

-Si $\mu = n$ (n contraintes actives)

$V = 0$

$h^* = 0$

$h = g$

-Sinon ($\mu < n$)

-calculer $F = VR$ en appelant VRFACT ,

V se trouve alors dans les $n-\mu$ colonnes de C

si la décomposition de Gram-Schmidt ne réussit pas, signaler l'erreur et sortir

-calculer h et h^* en appelant ORTHOG

si la décomposition de Gram-Schmidt ne réussit pas, signaler l'erreur et sortir

3.4 LSQU

Etant donné un tableau admissible de départ, cette procédure résoud le problème de moindres carrés linéaire avec contraintes linéaires. Elle appelle les procédures STPLNG , TABDN , TABUP .

Détail

Boucle: tant qu'on n'a pas atteint le nombre limite d'itérations.

-Si $\mu = n$

$s = 0$ (on doit enlever une contrainte)

-Sinon $(\mu < n)$

-Résoudre le système linéaire $Rz = h^*$

si R est singulière, signaler l'erreur et sortir

-Calculer la direction de recherche $s = Qz$

-Calculer le pas α en appelant STPLNG

si on trouve un point inadmissible dans STPLNG , signaler l'erreur et sortir

-Calculer $x_{i+1} = x_i + \alpha s$

-Si une contrainte est rencontrée dans la direction s, ajuster le tableau admissible et calculer la direction de recherche suivante en appelant TABUP

-Si aucune contrainte n'est rencontrée

-Calculer les multiplicateurs de Lagrange v

-calculer $E^T h$

-résoudre $ARv = E^T h$

-si AR est singulière, le signaler et sortir

-Si tous les multiplicateurs de Lagrange satisfont $\rho_i v_i \geq 0$, x est optimale et on sort.

-Sinon (il y a un multiplicateur de Lagrange le plus négatif, correspondant à une contrainte i)

-Ajuster le tableau admissible et calculer la direction de recherche suivant en appelant TABDN

-Si il y a une erreur dans l'ajustement du tableau, signaler l'erreur et sortir

-Passer à l'itération suivante.

Fin de boucle

3.5 CHANGEMENTS FAITS

Pour généraliser cette procédure, j'ai essayé d'éliminer l'hypothèse de non-dégénescence de la matrice des contraintes A.

Ceci n'a pu être fait que partiellement. En effet, nous pouvons aisément détecter les contraintes redondantes.

Dans le cas des contraintes d'égalité, nous pouvons éliminer les contraintes redondantes car elles sont combinaisons linéaires d'autres contraintes et donc, si ces dernières sont vérifiées, les premières le seront aussi.

Les contraintes redondantes sont éliminées de la façon suivante:

Dans la décomposition orthogonale de A_J, éliminer le passage où on sort après avoir examiner si $\exists j$ t.q. $AR(j,j) = 0$

Ce passage devient:

-Retenir tous les indices j pour lesquels $AR(j,j) = 0$ et $IRHO(j) = 0$

-Pour ces indices,

-Eliminer la colonne j de AR

-Eliminer la ligne JB(j) de A

-Eliminer l'indice j de JB

-Retourner pour faire la décomposition du nouveau A

Cette modification peut également être faite dans la sous-routine FEATAB dans le cas où l'utilisateur donne un point de départ admissible.
En effet, dans ce cas, l'algorithme ne passe pas par la sous-routine FEASOL et les contraintes redondantes n'y sont donc pas détectées.
Dans le cas où l'utilisateur ne donne pas de point de départ, les contraintes redondantes seront détectées dans la sous-routine FEASOL et enlevées à ce moment quand l'algorithme passera dans la sous-routine FEATAB , il n'y aura donc plus de contraintes redondantes.

4. APPLICATION AU PROBLEME DE COHERENCE DE BILANS

4.1 POSITION DU PROBLEME SIMPLIFIE DE BILANS COHERENTS

Nous examinerons dans cette partie la forme générale avec des contraintes linéaires d'un problème de cohérance de bilans.
La cohérence de bilans intervient dans certaines installations industrielles.
Ces installations peuvent être modélisées sous la forme d'un graphe ou "flowsheet". Ce graphe est composé de nœuds représentants des unités de l'installation telles des réacteurs chimiques, des colonnes de distillation, et d'arcs reliants les unités entre elles (voir figure1, en fin de chapitre).
Les bilans s'intéressent aux flux entre les différentes unités (débits, compositions,...). Ces flux sont mesurés sur chaque arc du graphe.
La cohérence de bilans impose l'équation

$$\Sigma \text{ INPUTS} = \Sigma \text{ OUTPUTS} \qquad (2.30)$$

en chaque nœud du graphe.
Malheureusement, cette contrainte de cohérence est rarement vérifiée par les données mesurées. D'autre part, certaines mesures peuvent être tout à fait erronées, par exemple, à la suite d'un instrument défectueux ou n'ayant pas bien fonctionné. Ces erreurs sont appelées erreurs aberrantes.
D'autre part, les mesures sont tachées d'erreurs aléatoires, par exemple, à cause de la faible précision de la machine. Il importe donc de corriger les valeurs mesurées par des valeurs théoriques cohérentes. C'est là le but des méthodes de calculs proposées dans le chapitre 2.

4.2. RESOLUTION DU PROBLEME SIMPLIFIE DE BILANS COHERENTS

Dans un premier temps, nous n'examinerons que les bilans affectés d'erreurs aléatoires.

Nous pouvons alors exprimer le problème sous la forme d'un problème de moindres carrés linéaire avec contraintes linéaires (P1).

Soit un graphe avec n arcs et m nœuds. Notons que dans ce graphe, certains arcs sont incidents à un seul nœud (voir figure 2, en fin de chapitre, arcs n° 1,8,9,10,12,16,18,21,25).

Nous pouvons associer une matrice A à ce graphe, appelée matrice des contraintes à m lignes et n colonnes, dont les éléments

$$a_{ij} = \begin{cases} 1 & \text{si l'arc } j \text{ a le nœud } i \text{ comme extrémité} \\ -1 & \text{si l'arc } i \text{ a le nœud } i \text{ comme origine} \\ 0 & \text{sinon} \end{cases}$$

$$A = (a_{ij}) \quad \begin{matrix} i=1,..,m \\ j=1,..,n \end{matrix} \qquad (2.31)$$

Des exemples de matrices A sont donnés dans les figures 3, 5 et 7.

Notons x_i , les valeurs théoriques correspondant aux n mesures d_i effectuées sur les différents arcs.

Les contraintes de cohérence (2.30) s'exprimeront sous forme matricielle.

Il faut donc trouver un vecteur de valeurs théoriques x "aussi proche que possible" du vecteur des valeurs mesurées d et satisfaisant les contraintes de cohérence (2.30).

Ceci introduit la notion de distance. Nous choisissons la distance Euclidienne, car elle satisfait le critère de maximun de vraisemblance pour des erreurs aléatoires, indépendantes et normalement distribuées.

Chaque observation peut être pondérée par une valeur w qui devrait être proportionnelle à l'inverse de l'écart-type σ_i estimé sur la mesure i , soit

$$w_i = \lambda / \sigma_i \qquad , i=1,\ldots,n \tag{2.32}$$

Notons W la matrice de pondération

$$W = \begin{bmatrix} w_1 & & 0 \\ & w_2 & \\ & \ldots & \\ 0 & & w_n \end{bmatrix} \tag{2.33}$$

Le problème de cohérence s'écrit alors :

$$\begin{cases} \underset{x}{\text{Min}} \; \| W(x-d) \|_2 \\ \text{s.c.} \quad Ax = 0 \end{cases} \tag{2.34}$$

Notons que les contraintes additionnelles de flux positifs ($x \geq 0$) ne sont pas imposées dans cette formulation.

Pour résoudre le problème (2.34), nous pouvons employer l'algorithme LCLSQ. Si c'est nécessaire, les contraintes de positivité des flux seront rajoutées au cours de l'algorithme de la façon suivante:

-Appliquer la routine LCLSQ au problème (2.34)

-Si, après la sous-routine LSQU , un x_i s'avère négatif, poser une contrainte supplémentaire $x_i \geq 0$ i.e. rajouter une ligne à A.

Retenir toutes les variables $x_i < 0$

Boucler sur ces variables x_i

augmenter m de 1

mettre un 1 dans la ligne m et la colonne i de A

mettre la m^{ieme} composante de ρ à -1

mettre la m^{ieme} composante de b à 0

Si il y avait un $x_i < 0$, retourner au début de la routine LCLSQ, afin de rechercher un nouveau point admissible pour ces contraintes et continuer.

La sous-routine pourrait être transformée pour le cas des bilans cohérents, étant donné la forme de C, de b et de A.

En effet:

-C est une matrice diagonale, il n'est donc pas nécessaire de stocker une matrice pxn , on pourrait simplement retenir les éléments diagonaux dans un vecteur de dimension p.

-b est un vecteur nul. Il n'est donc pas nécessaire de garder un vecteur de dimension n pour le stocker.

-A est une matrice creuse. On pourait donc faire des changements afin que la routine tienne compte de la forme creuse de la matrice, et diminuer la place mémoire nécessaire.

Ceci serait efficace pour des problèmes de grandes dimensions.

Dans un deuxième temps, nous pouvons examiner les bilans affectés de variables aberrantes, en plus des erreurs aléatoires sur chacune des variables. La détection de variables aberrantes n'est pas très difficile, et nous pouvons donc les supprimer de la façon suivante:

-Calculer les vecteurs résidus

-Pour chaque variable, i:

si elle n'est pas aberrante, ne rien faire

si elle est aberrante; l'éliminer par agrégation de nœud

Elimination d'une variable abérrante par agrégation de nœud: (voir figure 4)

Dans le graphe, si nous enlevons un arc, et si nous considérons que les deux nœuds incidents à cet arc ne forment plus qu'un nœud, la matrice A , associée à ce graphe, est modifiée.

Par exemple, supposons que la variable 19 est aberrante dans la figure 2.

Les nœuds 7 et 8 ne forment alors plus qu'un nœud (voir figure 4)

La matrice devient celle de la figure 5. Nous pouvons voir que cette matrice vient de la matrice du graphe d'origine où les lignes 7 et 8 sont additionnées et mises dans une ligne (la ligne 7 de la nouvelle matrice).

La colonne 19 devient alors nulle et nous pouvons l'élimier.

Nous pourrions encore simplifier le graphe en ne mettant qu'un arc à la place des arcs ayant même sommet comme origine et extrémité (voir figure 4, les arcs 13 et 14, 18 et 24, 21 et 19).

Ce fait est beaucoup plus difficile à voir sur la matrice. En effet, il faut trouver une autre ligne ayant 2 unités sur deux colonnes communes avec celles de la ligne 7.

Par exemple, sur la figure 5, nous voyons que dans les colonnes 13 et 14, on trouve deux 1 à la ligne 7 et deux -1 à la ligne 5. Il y a donc 2 arcs partant du nœud 5 et arrivant au nœud 7.

Nous pouvons alors additionner les débits 13 et 14 et supprimer une colonne.

Dans le cas général, pour enlever une variable aberrante i , on procède comme suit:

-Voir dans la colonne i où se trouvent le 1 et le -1 et retenir leur ligne

-Additonner les 2 lignes retenues ci-dessus dans la première ligne

-Eliminer la seconde ligne

-Eliminer la colonne i.

Des méthodes de cette forme sont expliquées avec plus de précision dans les articles (R.W.Serth and W.A.Heenan, May 1986) et (W.A.Heenan and R.W.Serth, November 1986).

Ici, pour les exemples trouvés dans ces 2 articles et dont les matrices sont données dans les figures 3 et 7, la détection d'erreurs aberrantes n'a pas été appliquée. L'algorithme LCLSQ donne alors les résultats suivants:

1°) Pour l'exemple de l'article (W.A.Heeman and R.W.Serth, November 1986) , dont le graphe est donné dans la figure 2, nous obtenons le tableau suivant:

DEBIT OBSERVE	DEBIT THEORIQUE	ERREUR EN %
289.94	360.65	24
85.41	0.0	100
113.86	133.45	17
120.81	131.54	9
241.81	360.65	49
248.39	1.80	99
245.53	0.0	100
1.79	1.80	0.7
4.55	4.72	4
1.89	1.91	1
83.09	27.69	66
71.30	48.08	32
67.90	76.01	12
203.55	208.87	2
202.19	159.23	21
90.07	49.64	45
12.31	13.01	6
104.31	34.38	67
35.52	29.12	18
50.48	25.52	49
274.32	154.74	43
37.49	40.29	7
298.31	139.97	53
8.33	0.0	100
193.42	197.70	2

2°) L'exemple de l'article (R.W.Serth and W.A.Heeman, May 1986) dont le graphe est dans la figure 6, donne le tableau suivant:

DEBIT OBSERVE	DEBIT THEORIQUE	ERREUR EN %
0.86	0.85	0.3
1.00	0.99	1
111.82	108.39	3
109.95	106.54	8
53.27	57.61	8
112.27	90.41	19
2.32	2.36	1
160.05	145.59	11
0.86	0.21	74
52.41	56.75	8
14.86	11.31	24
67.27	45.44	32
111.27	89.42	19
91.86	77.03	16
60.00	56.53	6
23.64	23.56	0.3
32.73	29.38	10
16.23	16.09	0.8
7.95	9.96	25
10.50	11.17	6
87.27	69.64	20
5.45	5.34	2
2.59	4.48	73
46.64	41.31	11
85.45	63.65	25
81.32	14.43	82
70.77	0.00	100
72.23	120.28	66

Nous pouvons voir, grâce à la dernière colonne quelles sont les variables aberrantes. Nous pourrions réécrire les matrices en enlevant les variables aberrantes comme expliqué ci-dessus et recommencer l'algorithme LCLSQ avec ces nouvelles matrices.

Nous obtiendrions alors de meilleurs résultats.

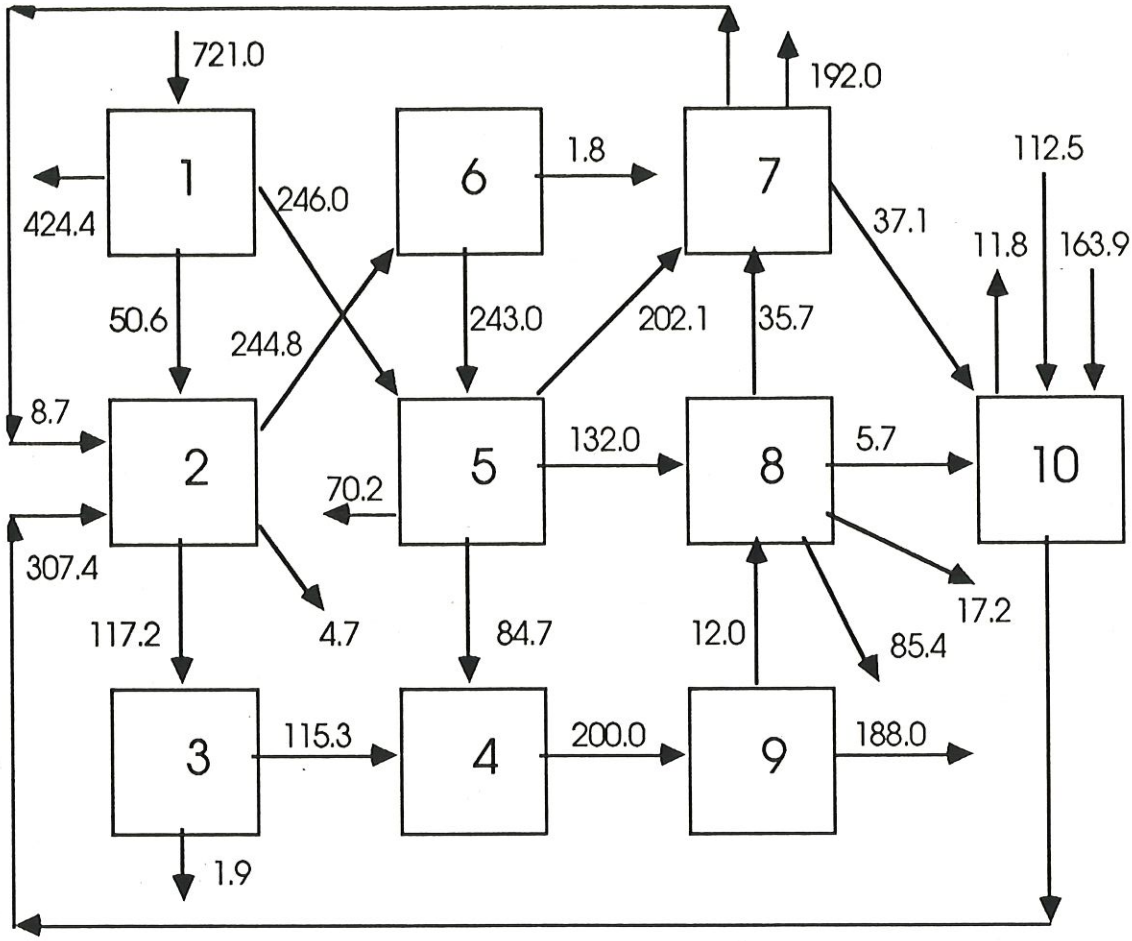

Figure 1. Schéma des flots pour un système typique à vapeurs.

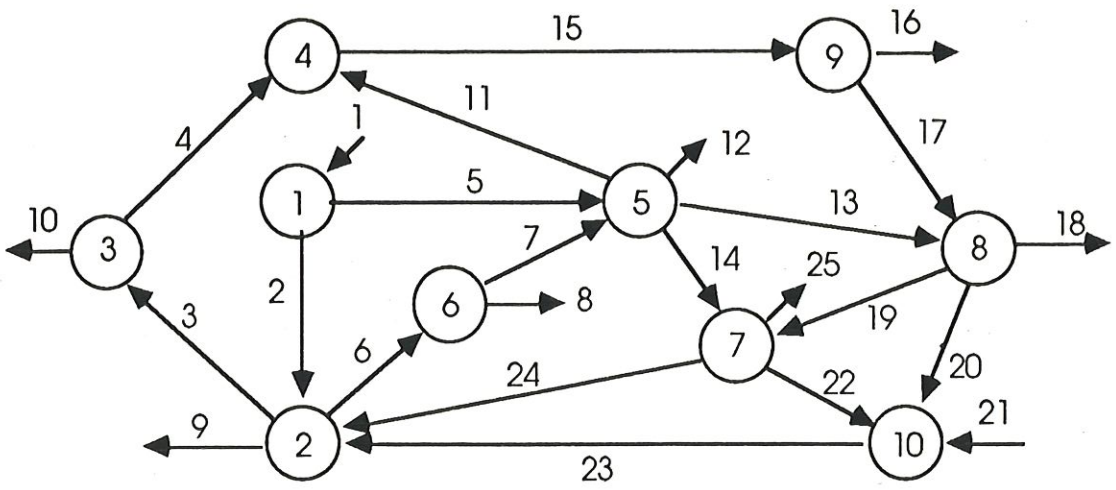

Figure 2. Graphe se rapportant à la figure 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	-1			-1																				
2		1	-1			-1			-1														1	1	
3			1	-1						-1															
4				1							1				-1										
5					1		1				-1	-1	-1	-1											
6						1	-1	-1																	
7														1					1			-1		-1	-1
8													1				1	-1	-1	-1					
9															1	-1	-1								
10																				1	1	1	-1		

Figure 3. Matrice se rapportant au graphe de la figure 2

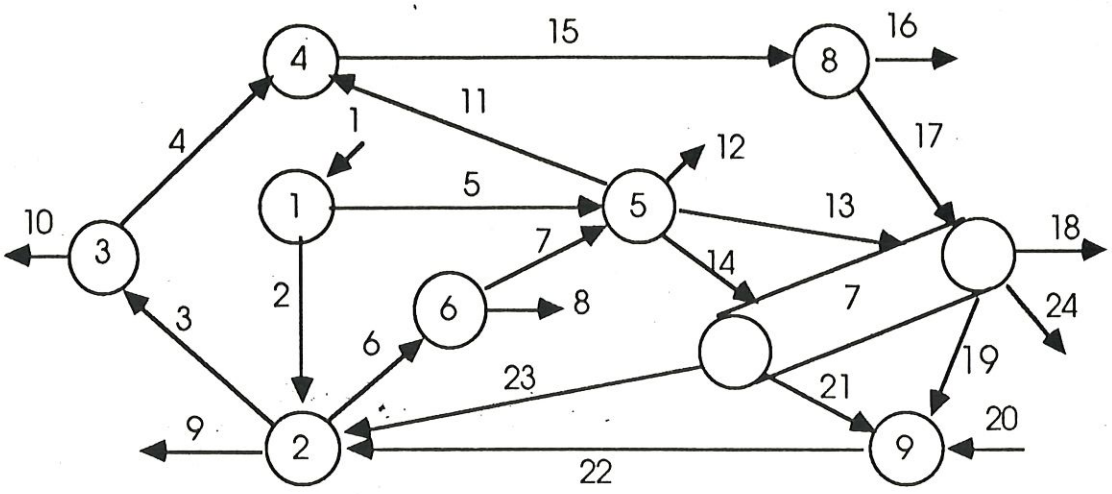

Figure 4. Graphe de la figure 2 où on a éliminé la variable aberrante.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	-1			-1																			
2		1	-1			-1			-1													1	1	
3			1	-1						-1														
4				1							1				-1									
5					1		1				-1	-1	-1	-1										
6						1	-1	-1																
7													1	1			1	-1	-1		-1		-1	-1
8															1	-1	-1							
9																			1	1	1	-1		

Figure 5. Matrice se rapportant à la figure 4.

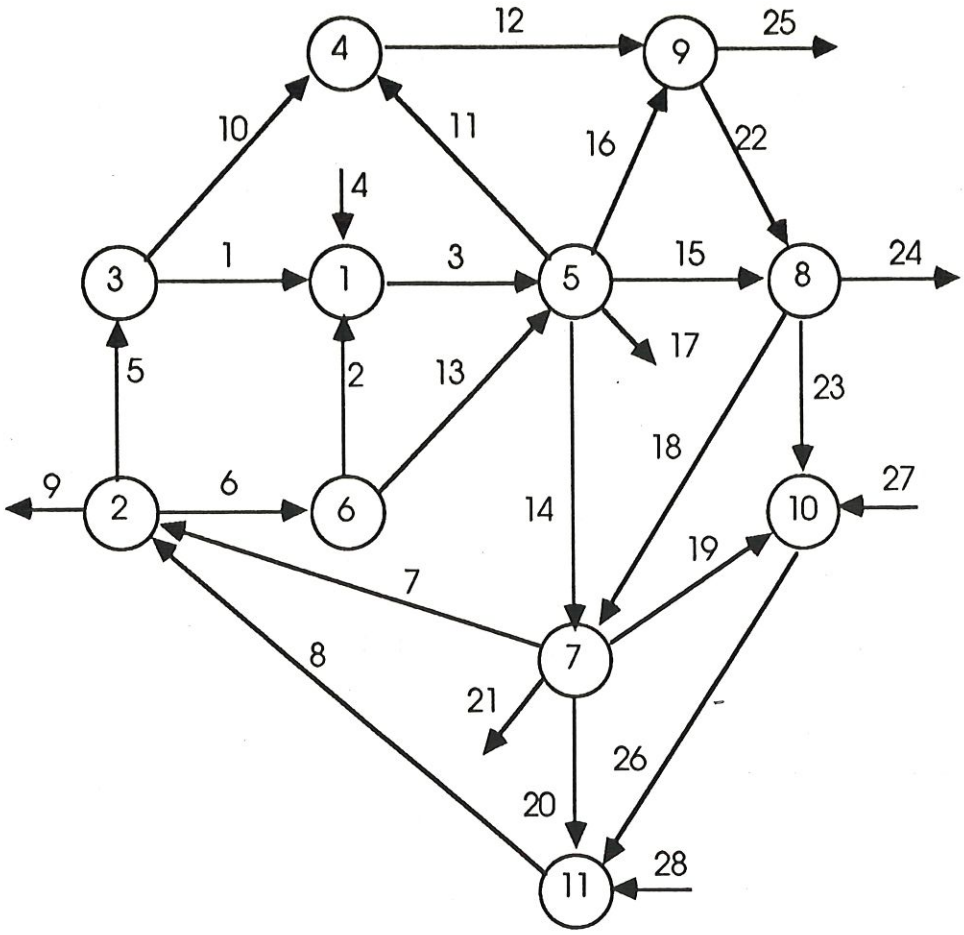

Figure 6. Graphe d'un autre système à vapeurs.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	1	1	-1	1																								
2					-1	-1	1	1	-1																			
3	-1				1					-1																		
4										1	1	-1																
5			1								-1		1	-1	-1	-1	-1											
6		-1				1							-1															
7							-1							1				1	-1	-1	-1							
8															1			-1				1	-1	-1				
9												1				1						-1			-1			1
10																			1				1			-1	1	
11								-1												1						1		

Figure 7. Matrice se rapportant au graphe de la figure 6.

5. CONCLUSION

Nous avons étudier différentes méthodes de moindres carrés linéaires avec contraintes linéaires.

Nous avons modifié la dernière méthode étudiée afin de l'appliquer aux cohérences de bilans.

Les modifications sont les suivantes:

-les contraintes redondantes sont éliminées

-les contraintes de positivité des flux sont rajoutées au cours de l'algorithme si c'est nécessaire.

Cette méthode aurait pu être encore simplifiée à cause de la forme creuse des matrices d'entrées.

Nous pouvons voir en comparant les résultats du chapitre 4 et les articles de (Serth and Heenan, 1986) que les méthodes de moindres carrés ne sont pas suffisantes.

En effet, il faut d'abord détecter les valeurs aberrantes avant d'appliquer la méthode de moindres carrés.

Une fois que les variables aberrantes sont détectées, il faut les éliminer avec, par exemple, la méthode expliquée au chapitre 4.

Quand les variables aberrantes sont éliminées ou s'il n'y en a pas, les méthodes de moindres carrés deviennent efficaces.

Il faudrait donc compléter cette étude par une étude des résultats de détection et d'élimination de variables aberrrantes.

REFERENCES

Richard J. Hanson (1986) : Linear least squares with bounds and linear contraints, SIAM J. Sci. Stat. Comput., Vol. 7, n° 3, pp.826-834.

C.L. Lawson and R.J. Hanson (1974) : Solving least squares problems, Prentice Hall, Englewood Cliffs, New-Jersey.

K. Haskell and R.J. Hanson (1981) : An algorithm for linear least squares problems with equality and nonnegativity constraints, Math.Prog. 21, pp. 98-118.

J. Stoer and K. Schittkowski (1979) : A factorization method for the solution of constrained least squares problems, Numer. Math. 31, pp. 431-463.

J.W. Daniel, W.B. Gragg, L. Kaufman, G.W. Stewart (1976) : Stable algorithms for updating the Gram-Schmidt QR factorization, Math.Comput., 31, pp. 772-795.

P.E. Gill, W. Murray and M.H. Wright (1981) : Practical optimization, Academic Press.

R. Crane, B. Garbow, K.Hillstrom and M. Minkoff (1977) : LCLSQ: An implementation of an algorithm for linearly contrained linear least squares problems, Argonne Natl. Labs. Rept. ANL 80116.

R.W. Serth and W.A. Heenan (May 1986) : Gross error detection and data reconciliation in steam-metering systems, AIChE Journal, Vol. 32, n° 5, p. 733.

W.A. Heenan and R.W. Serth (November 1986) : Detecting errors in process data, Chemical Engineering, 10 November, pp. 99-103.