$\mathscr{O}$*ptimisation* $\mathscr{L}$*inéaire*

Yasmine 5132609061
Justine 5132609063
Gabriel 5132609041

May 2016

1 Introduction

Le sujet de ce document sert à résoudre le problème:

$$min\{f^T.x : A_{eq}.x = b_{eq}, A_{ineq}.x \leq b_{ineq}\}$$

où $(f, x) \in (R^n)^2, A_{eq} \in \mathscr{M}_{k,n}, B_{eq} \in R^k, A_{ineq} \in \mathscr{M}_{j,n}, B_{ineq} \in R^j$
Ce document consiste deux parties :

1 La partie théorique: on reformulera ce problème sous la forme standard, présentera en détail de l'algorithme du simplexe et prendra un exemple.

2 La partie informatique: On expliquera le code de l'algorithme du simplexe et testera le code par quelques samples.

2 Partie Théorique

Dans cette partie, on d'abord mettra en forme standard et ensuite présentera en détail de l'algorithme du simplexe. A la fin, on prendra un exemple

2.1 Mettre en forme standard

On rappelera que la forme standard:

$$min\{c^T.x : A.x = b, x \geq 0\}$$

où $c \in R^n, A \in \mathscr{M}_{m,n}, b \in R^m$, alors que l'on aura deux problèmes à résoudre : les constraintes d'inégalité et les variables libres.

2.1.1 Pour les constraintes d'inégalité

D'après la remarque 24.2 dans le polycopie[1], on ajoutera variables auxiliaires $z_{aux} \in R^j, z_{aux} > 0$,tel que $A_{aux}.x + z_{aux} = b_{ineq}$. Ici, pour facilement trouver les bases, on ajoutera les variables artificielles $z_{arti} \in R^k, z_{arti} > 0$ telles que $A_{arti}.x + z_{arti} = b_{eq}$,donc

$$A = \begin{bmatrix} A_{eq} & I_k & 0_{j,j} \\ A_{ineq} & 0_k & I_{j,j} \end{bmatrix}, x_{std} = \begin{bmatrix} x \\ z_{arti} \\ z_{aux} \end{bmatrix}, b = \begin{bmatrix} b_{eq} \\ b_{ineq} \end{bmatrix}, c = \begin{bmatrix} f \\ 0_{m,1} \end{bmatrix}$$

De cette façons, les inégalités devient les égalités, donc on a $A.x_{std} = b$. Ensuite, il nous reste de transformer les variables libres aux variables positives.

2.1.2 Pour les variables libres

D'après la remarque 24.4[1], on ajoutera les variables $(x_+, x_-) \in (R^n)^2$, $x_+ > 0, x_- > 0$, telles que $x = x_+ - x_-$, ainsi, les matrices deviennent

$$A = \begin{bmatrix} A_{eq} & -A_{eq} & I_k & 0_{j,j} \\ A_{ineq} & -A_{ineq} & 0_k & I_{j,j} \end{bmatrix}, x_{std} = \begin{bmatrix} x_+ \\ x_- \\ z_{arti} \\ z_{aux} \end{bmatrix}, b = \begin{bmatrix} b_{eq} \\ b_{ineq} \end{bmatrix}, c = \begin{bmatrix} f \\ -f \\ 0_{m,1} \end{bmatrix}$$

Ouf! On arrive à mettre le problème en forme standard, et donc le problème devient

$$min\{c^T.x_{std} : A.x_{std} = b, x_{std} \geq 0\}$$

où $(c, x_{std}) \in (R^{2n+m})^2, A \in \mathscr{M}_{m,2n+m}, B \in R^m$.

2.2 Présenter en détail l'algoritme du simplexe

On utilisera la version tableau de l'algoruthme du simplexe. On d'abords discutera dans le cas le plus simple(non variables artificielles) et ensuite le cas des deux phrases.

2.2.1 Cas non variables artificielles

Étape 1 : établir le tableau du simplexe

Chercher c_B , c_N , B , N Tout d'abord, toutes les bases se trouvent à la droite de matrice A (correspondant aux variables auxiliaires), donc

$$c_B = [0_{m,1}],\ B = [I_m],\ c_N = \begin{bmatrix} f \\ -f \end{bmatrix}, N = \begin{bmatrix} A_{ineq} & -A_{ineq} \end{bmatrix}$$

Établir le tableau D'après la Chapitre 3.3.3, on obtenira le tableau ci-dessous

$$T = \begin{array}{c|c|c|c} N & B & b & y_B \\ \hline {c_B}^T.B^{-1}.N - {c_N}^T & 0_{1,m} & {c_B}^T.B^{-1}.b & \end{array}$$

où y_B note l'indice de base

Étape 2 : Chercher la colonne du pivot On cherche $u \in [1, 2n+m]$ tel que $T_{m,u}$ $estplusgrand$

si $T_{m,u} \leq 0$ alors, cette solution est déjà optimal, et on passera à Étape 6.

si $T_{m,u} > 0$ alors, la colonne du pivot est u, on passera à Étape 3.

Étape 3 : Chercher la ligne du pivot On cherche $v \in [1, m]$ tel que $T_{v,u} > 0$ *et* $\frac{T_{v,2n+m+1}}{T_{v,u}}$ *est plus petit*

si tel v *n'existe pas* alors, il n'y a pas de solution optimale, on termine.

si $\frac{T_{v,2n+m+1}}{T_{v,u}} \neq 0$ alors, la ligne du pivot est v, et ensuite on arrive à trouver le pivot, et on passera à Étape 4.

si $\frac{T_{v,2n+m+1}}{T_{v,u}} = 0$ alors, on se trouve dans le cas dégénéré, et on passera à Étape 5.

Étape 4 : Méthode de Gausse On fait pivot de gausse en ligne. En même temps, on modifiera y_B car on change la base dans cette étape. On passera à Étape 2.

Étape 5 : Anti-cyclage Pour ne pas se tromper dans un cyclage. On change le colonne du pivot, et choisit le premier $u \in [1, 2n+m]$ tel que $T_{m,u} > 0$. Ensuite on passera à Étape 3. (Cette fois dans Étape 3, même si c'est le cas dégénéré, on continuera à passer à Étape 4, sinon on va tromper dans un cyclage)

Étape 6: Obtenir le résultat

$$min\{f^T.x : A_{eq}.x = b_{eq}, A_{ineq}.x \leq b_{ineq}\} = T_{m,2n+m}$$

, l'information des valeurs de x se trouve dans le 2n+m+1-ième colonne de T: $soit\ i \in [1, 2n],\ si\ i \in y_B, alors\ x_i = T_{y_B{}^{-1}(i),2n+m+1},\ sinon\ x_i = 0$, et ensuite on obtient le résultat finale: $soit\ i \in [1, n], x_i = x_{i+n} - x_i$.

2.2.2 Cas des deux phrases

On a justement besoin de modifier un peu algorithme précédant.

Étape 0 : Changer c D'abord, on a besoin d'éliminate les variables artificielles, on pose $c = \begin{bmatrix} 0_{2n,1} \\ 1_{k,1} \\ 0_{m-k,1} \end{bmatrix}$, ensuite on passe à Étape 1.

Étape 3 : Version modifié pour éliminer les variables artificielles On met une priorité sur la ligne qui corrspondant à éliminer une variable artificielle, c'est-à-dire s'il y a pleusieurs lignes v telles que $\frac{T_{v,2n+m+1}}{T_{v,u}}$ est un minimun, on choissira l'un qui corrspond à éliminer une variable artificielle. Ensuite, on continue à Étape 3. Finalement on détermine si c'est le cas dégénéré ou pas,etc.

Étape 4 : Version modifiée pour éliminer les variables artificielles Après le processus de pivot de gausse, si une colonne correspondant à une variable mais pas une base, alors on peut éliminer cette colonne. Ensuite on continue l'algorithme, et on passera à Étape 2. Pour connecter la colonne et le vrai indice d'un variable, on définit une fonction $transf()$ de l'indice de la colonne à l'indice d'une variable, cette fonction change à chaque fois on éliminate une variable.

Étape 7 : Reformuler le tableau Après le calcul pour c (c'est-à-dire on a éliminé toutes les variables artificielles), on obtient un tableau $T \in \mathscr{M}_{m+1,2n+j+1}$(on ne compte pas que y_B dans le matrice T car il est le record d'indice de base, et il ne participe pas dans pivot de gausse)

si $T_{m+1,2n+j+1} \neq 0$ c'est-à-dire le minimun de la somme des variables artificielles n'est pas 0, on n'a pas de solution optimale. Terminé.

si $T_{m+1,2n+j+1} = 0$ alors, on reviendra au cas non variables artificielles. Ensuite, on reformulera le tableau. On pose ici y_N la séquence des indices des variables non basiques $y_N = [1, 2n] \cup [2n + k + 1, 2n + m]/y_B$:

$$c = \begin{bmatrix} f \\ -f \\ 0_{j,1} \end{bmatrix}, pour\ tout\ i \in [1, m], c_{B\,i,1} = c_{transf^{-1}(y_B(i)),1},$$

$$pour\ tout\ i \in [1, 2n + j], c_{N\,i,1} = c_{transf^{-1}(y_N(i)),1},$$

B est la matrice constituée par les colonnes correspondantes aux variables basiques. N est la matrice constituée par les colonnes correspondantes aux variables basique. On peut alors calculer $c_B{}^T.B^{-1}.N - c_N{}^T$. On met chaque sa valeur à la derrière ligne et aux colonnes correspondantes aux variables non basiques de façon itérative; on met 0 à la derrière ligne et aux colonnes correspondantes aux variables non basiques. On met $c_B{}^T.B^{-1}.b$ à $T_{m+1,2n+j+1}$. Finalement, on arrive à reformuler le tableau. Ensuite, on passera à Étape 2. On finira le calcule un jour! Bon courage!

2.3 Exemple

On prend l'exemple de Exercice 3.3 dans la polycopie utilise la méthode ci-dessus. Voici, le problème :

$$\begin{cases} min & 3x_1 - 2x_2 + x_3 \\ s.c. & 2x_1 + x_2 - x_3 \leq 5 \\ & 4x_1 + 3x_2 + x_3 \geq 3 \\ & -x_1 + x_2 + x_3 = 3 \\ & x \geq 0 \end{cases}$$

Initialiser les paramètre

$$f = \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, A_{eq} = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix}, b_{eq} = \begin{bmatrix} 2 \end{bmatrix}, A_{ineq} = \begin{bmatrix} 2 & 1 & -1 \\ -4 & -3 & -1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, b_{ineq} = \begin{bmatrix} 5 \\ -3 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$m = 3, k = 1, j = 5, n = k + j = 6$$

Mettre au forme standard

$$A = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & -1 & -2 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -4 & -3 & -1 & 4 & 3 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 5 \\ -3 \\ 0 \\ 0 \\ 0 \end{bmatrix}, c = \begin{bmatrix} 3 \\ -2 \\ 1 \\ -3 \\ 2 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix},$$

x_7 *est une variable artificielle,* $x_8 \sim x_{12}$ *sont variables auxiliaires.*

Car c'est le cas des deux phrases, on passe à Étape 0

Étape 0 on doit premièrement calculer $minx_7$, ici,

$$c = \begin{bmatrix} 0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0 \end{bmatrix}, c_B = \begin{bmatrix} 1\\0\\0\\0\\0\\0 \end{bmatrix}, c_N = \begin{bmatrix} 0\\0\\0\\0\\0\\0 \end{bmatrix}, B = \begin{bmatrix} 1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1 \end{bmatrix}, N = \begin{bmatrix} -1&1&1&1&-1&-1\\2&1&-1&-2&-1&1\\-4&-3&-1&4&3&1\\-1&0&0&1&0&0\\0&-1&0&0&1&0\\0&0&-1&0&0&1 \end{bmatrix},$$

On calcule ${c_B}^T.B^{-1}.N - {c_N}^T$ et ${c_B}^T.B^{-1}.b$, et on forme le tableau T ci-dessus:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}		
-1	1	1	1	-1	-1	1	0	0	0	0	0	2	x_7
2	1	-1	-2	-1	1	0	1	0	0	0	0	5	x_8
-4	-3	-1	4	3	1	0	0	1	0	0	0	-3	x_9
-1	0	0	1	0	0	0	0	0	1	0	0	0	x_{10}
0	-1	0	0	1	0	0	0	0	0	1	0	0	x_{11}
0	0	-1	0	0	1	0	0	0	0	0	1	0	x_{12}
-1	1	1	1	-1	-1	0	0	0	0	0	0	2	

Étape 2 $u = 2$

Étape 3 $v = 1$ alors, le pivot se trouve dans la position (1,2)

Étape 4 On fait pivot de gausse, ensuite, on trouve que x_7 n'est plus une base, alors on l'éliminate, le tableau devient:

x_1	x_2	x_3	x_4	x_5	x_6	x_8	x_9	x_{10}	x_{11}	x_{12}		
-1	1	1	1	-1	-1	0	0	0	0	0	2	x_2
3	0	-2	-3	0	2	1	0	0	0	0	3	x_8
-7	0	2	7	0	-2	0	1	0	0	0	3	x_9
-1	0	0	1	0	0	0	0	1	0	0	0	x_{10}
-1	0	1	1	0	-1	0	0	0	1	0	2	x_{11}
0	0	-1	0	0	1	0	0	0	0	1	0	x_{12}
0	0	0	0	0	0	0	0	0	0	0	0	

On trouve que la solution est optimale, et $minx_7 = 0$, alors on passe à Étape 7.

Étape 7

$$c = \begin{bmatrix} 3 \\ -2 \\ 1 \\ -3 \\ 2 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, c_B = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, c_N = \begin{bmatrix} 3 \\ 1 \\ -3 \\ 2 \\ -1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, N = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 \\ 3 & -2 & -3 & 0 & 2 \\ -7 & 2 & 7 & 0 & -2 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix},$$

On calcule ${c_B}^T.B^{-1}.N - {c_N}^T$ et ${c_B}^T.B^{-1}.b$, et on forme le tableau T ci-dessus:

x_1	x_2	x_3	x_4	x_5	x_6	x_8	x_9	x_{10}	x_{11}	x_{12}		
-1	1	1	1	-1	-1	0	0	0	0	0	2	x_2
3	0	-2	-3	0	2	1	0	0	0	0	3	x_8
-7	0	2	7	0	-2	0	1	0	0	0	3	x_9
-1	0	0	1	0	0	0	0	1	0	0	0	x_{10}
-1	0	1	1	0	-1	0	0	0	1	0	2	x_{11}
0	0	-1	0	0	1	0	0	0	0	1	0	x_{12}
-1	0	-3	1	0	3	0	0	0	0	0	-4	

Étape 2 $u = 6$

Étape 3 $v = 6$ On trouve que $\frac{T_{6,6}}{T_{6,12}} = 0$, c'est le cas dégénéré, on passe à Étape 5.

Étape 5 $u = 4$

Étape 3 $v = 4$ On trouve le pivot se trouve dans (4,4).

Étape 3 Voici le tableau :

x_1	x_2	x_3	x_4	x_5	x_6	x_8	x_9	x_{10}	x_{11}	x_{12}		
0	1	1	0	-1	-1	0	0	-1	0	0	2	x_2
0	0	-2	0	0	2	1	0	3	0	0	3	x_8
0	0	2	0	0	-2	0	1	-7	0	0	3	x_9
-1	0	0	1	0	0	0	0	1	0	0	0	x_4
0	0	1	0	0	-1	0	0	-1	1	0	2	x_{11}
0	0	-1	0	0	1	0	0	0	0	1	0	x_{12}
0	0	-3	0	0	3	0	0	-1	0	0	-4	

On revient à Étape 2.

Étape 2 $u = 6$

Étape 3 $v = 6$, c'est le cas dégénéré, on passe à Étape 5.

Étape 5 $u = 6$

Étape 3 $v = 6$

Étape 4 Voici le tableau :

x_1	x_2	x_3	x_4	x_5	x_6	x_8	x_9	x_{10}	x_{11}	x_{12}		
0	1	0	0	-1	0	0	0	-1	0	1	2	x_2
0	0	0	0	0	0	1	0	3	0	-2	3	x_8
0	0	0	0	0	0	0	1	-7	0	2	3	x_9
-1	0	0	1	0	0	0	0	1	0	0	0	x_4
0	0	0	0	0	0	0	0	-1	1	1	2	x_{11}
0	0	-1	0	0	1	0	0	0	0	1	0	x_6
0	0	0	0	0	0	0	0	-1	0	-3	-4	

Voilà, on obtient que $min\{3x_1 - 2x_2 + x_3\}$=-4,avec $x_1 = 0, x_2 = 2, x_3 = 0$

3 Partie Informatique

Dans cette partie, on expliquera notre code de l'algorithme du simplexe et testera notre code par quelques samples.

3.1 Explication de notre code

Il y a 4 functions dans notre code:Aform(),simplexe(),findPivot(),projet() que on discutera après. On expliquera d'abord les variables dans notre code.

3.1.1 La notation des variables

A le tableau T

$AStd$ la partie à la gauche et à haut du tableau A

$BStd$ la partie à la droite et à haut du tableau A

$cbindex$ l'ensemble de l'indices correspondants à la variable basique

$cnindex$ l'ensemble de l'indice correspondant à la variable non basique

$eqNum$ k

ineqNum j

Num m

arti l'ensemble de l'indice des variables artificielles qui restent dans le tableau

max_index u

min_index v

transf le liste qui a l'indice qui corresponde au colonne de tableau et qui a les valeurs qui correspondent au l'indice des variables, c'est pour simplifier la vie quand on éliminer les colonnes.

3.1.2 Aform()

Cette fonction est pour établir le tableau A. Cette fonction est utilisé dans Étape 1 et Étape 7.

```
function [A] = Aform(AStd,BStd,fStd,cbindex,cnindex,transf)
     line = size(AStd,1);
     n=size(AStd,2);
     %cb is the coefficient of base in fStd
     cb=zeros(line,1);
     cbindexreal =zeros(1,line);
     for i= 1:line
          cbindexreal(i)=find(transf==cbindex(i),1);
          cb(i) =fStd(cbindex(i));
     end
     %B is consisted of the colone of base
     B = AStd(:,cbindexreal);
     %cn is the coefficient of not base in fStd
     cn=zeros(n-line,1);
     cnindexreal =zeros(1,n-line);
     for i= 1:n-line
          cnindexreal(i)=find(transf==cnindex(i),1);
          cn(i)=fStd(cnindex(i));
     end
     N = AStd(:,cnindexreal);
     %we get the lastline in the matrix here
     temps = (cb')*(B^-1)*N-cn';
     lastline = zeros(1,n);
     %in the lastline,only the colone of not base has value
     for i = 1:size(cnindex,2)
          lastline (find(transf==cnindex(i),1)) =temps(i);
     end
     %the initial value of the question(solution possible)
     val =cb'*BStd;
     %Finally we get the big matrice A
     A = [AStd,BStd;lastline,val];
```

3.1.3 findPivot()

Cette fonction est pour trouver la ligne de pivot s'on la colonne de pivot. Le output *min_col* est pour déterminer si c'est le cas dégénéré. Elle est utilisé dans Étape 3.

```
function [min_index,min_col] =findPivot(A,arti,max_index)
[line,col]=size(A);
Num = ((col-1)-(line-1))/2;

 firstPos=find(A(1:line-1,max_index)>0,1);
 if isempty(firstPos)
     error('pas de solution optimal');
end
%we set the priority to eliminate the variable artificiel here
 if ~isempty(arti)
     min_index=firstPos;
     min_col= A(firstPos,col)/A(firstPos,max_index);
     for i = arti -2*Num
         if A(i,max_index)>0
             if A(i,col)/A(i,max_index) < min_col
                 min_index = i;
                 min_col = A(i,col)/A(i,max_index);
             end
         end
     end
     for i = setdiff(1:line-1,arti- 2*Num)
         if A(i,max_index)>0
             if A(i,col)/A(i,max_index) < min_col
                 min_index = i;
                 min_col = A(i,col)/A(i,max_index);
             end
         end
     end

 else
     min_index=firstPos;
     min_col= A(firstPos,col)/A(firstPos,max_index);
     for i = 1:line-1
         if A(i,max_index)>0
             if A(i,col)/A(i,max_index) < min_col
                 min_index = i;
                 min_col = A(i,col)/A(i,max_index);
             end
         end
     end
end
```

3.1.4 simplexe()

Cette fonction utilise fonction findPivot(), chaque fois on l'utilise, il optimise la solution pour une fois. Elle correspondant à Étape 2, Étape 3, Étape 4, Étape 5.

```
function [A,cbindex,cnindex,arti,transf,n] =simplexe(A,cbindex,cnindex,arti,transf)
[line,col]=size(A);
Num = ((col-1)-(line-1))/2;
cindextotal = [cbindex,cnindex];
[max_lastline,max_index]=max(A(line,1:col-1));
[min_index,min_col]=findPivot(A,arti,max_index);

 if min_col ==0%Etape5: if it's the cas degenere
     max_index = find(A(line,1:col-1)>0,1);
     [min_index,min_col]=findPivot(A,arti,max_index);
end
A(min_index,:)=A(min_index,:)/A(min_index,max_index);
for i = [1:min_index-1,min_index+1:line]
     A(i,:)=A(i,:)-A(min_index,:)*A(i,max_index);
end
 if ismember(transf(min_index+2*Num),arti)&&A(line,max_index)<=0
     A(:,min_index+2*Num)=[];
     col=col-1;
     arti=setdiff(arti,transf(min_index+2*Num));
     if min_index +2*Num ~=col
          for i = min_index +2*Num : col-1
               transf(i)=transf(i+1);%col number to x number
          end
     end
     transf(col)=[];

     %update cbindex,cnindex
     cbindex(min_index) = transf(max_index);
     cnindex=setdiff(cnindex,transf(max_index));

 else
     cbindex(min_index) = transf(max_index);
     cnindex=setdiff(cindextotal,cbindex);
end
n=col-1;
```

3.1.5 projet()

C'est la function principale dans notre projet! Elle correspondant à tout ce que on fait avant. Elle nous donne le solution du problème *val* et la valeur de chaque x_i x quand il est optimal.

```
function [x,val]=projet(f,Aeq,Beq,Aineq,Bineq)
% min(f'.x)
% Aeq .x = Beq ; A .x = B
%eqNum is the number of equation
[eqNum,Num2]=size(Aeq);
%ineqNum is the number of inequation
[ineqNum,Num]=size(Aineq);
%line is the number of row in the matrice not include the lastrow
```

```
line=eqNum+ineqNum;

%text if the dimension is right
if (~isempty(Aeq))&&(~isempty(Aineq))
    if ((Num~=Num2)||~isequal(size(f),[Num,1])||~isequal(size(Beq),[eqNum,1])||~isequal( size (Bineq),[ineqNum,1]))
        error('Dimension error');
    end
else if isempty(Aineq)
        Num = Num2;
    end
end
%we make this peroblem in standard form,which is xi = ui - vi and ui¿=0,vi¿0
fStd = [f;−f;zeros( line ,1) ];

AStd = [Aeq, −Aeq,eye(eqNum),zeros(eqNum,ineqNum);Aineq, −Aineq,zeros(ineqNum,eqNum),eye(ineqNum)];
BStd = [Beq; Bineq];

%n is the number of colone not include the last colone
n = size(AStd,2);
%transf is a list whose index is the index of colone of matrice and whose value is the index of xi, it's used when we delete a colone
transf= 1: n;
% if it's the cas 2 phrase
if ~isempty(Aeq)
    %Etape 0:change c
    arti = 2∗Num+1:2∗Num+eqNum;
    %ff is the new fStd here
    ff=zeros(n,1);
    for i =arti;
        ff(i)=1;
    end
    %index of base
    cbindex = (2∗Num+1):n;
    cnindex = 1:2∗Num;
    %Etape 1
    A = Aform(AStd,BStd,ff,cbindex,cnindex,transf);
    %Etape 2,Etape 3,Etape 4
    while ~isempty(arti)
        [A,cbindex,cnindex,arti, transf ,n]=simplexe(A,cbindex,cnindex,arti,transf);
    end
    if abs(A(line+1,n+1))>0.001
        error('pas de solution optimal');
    end
    %Etape 7
    A = Aform(A(1:line,1:n),A(1:line,n+1),fStd,cbindex,cnindex,transf);

    max_lastline=max(A(line+1,1:n));
    %Etape2,Etape3,Etape4
    while max_lastline>0
        [A,cbindex,cnindex,arti, transf ,n]=simplexe(A,cbindex,cnindex,arti,transf);
        max_lastline = max(A(line+1,1:n));
    end
% if it's the cas non variable artificiel
```

```
else
    %index of base
    cbindex = (2*Num+1):n;
    cnindex = 1:2*Num;
    %Etape 1
    A = Aform(AStd,BStd,fStd,cbindex,cnindex,transf);
    max_lastline =max(A(line+1,1:n));
    %Etape2,Etape3,Etape4
    while max_lastline>0
        disp(A);
        [A,cbindex,cnindex,arti,transf,n]=simplexe(A,cbindex,cnindex,[],
         transf);
        max_lastline = max(A(line+1,1:n));
    end
end

%Etape 6: we get the result here
val = A(line+1,n+1);
x=zeros(1,Num);
for i = 1:line
    if cbindex(i)<=2*Num
        if cbindex(i) <=Num
            x(cbindex(i)) = x(cbindex(i)) + A(i,n+1);
        else
            x(cbindex(i)-Num) = x(cbindex(i)-Num) + A(i,n+1);
        end
    end
end
```

3.2 Exemple

On prend le même exemple que dans la partie théorique. Voici le résultat qu'on obtient par fonction projet

```
>> [x,val]=projet([3;-2;1],[-1,1,1],[2],[2,1,-1;-4,-3,-1;-1,0,0;0,-1,0;0,0,-1],[5;-3;0;0;0])
    -1     1     1     1    -1    -1     1     0     0     0     0     0     2
     2     1    -1    -2    -1     1     0     1     0     0     0     0     5
    -4    -3    -1     4     3     1     0     0     1     0     0     0    -3
    -1     0     0     1     0     0     0     0     0     1     0     0     0
     0    -1     0     0     1     0     0     0     0     0     1     0     0
     0     0    -1     0     0     1     0     0     0     0     0     1     0
    -1     1     1     1    -1    -1     0     0     0     0     0     0     2

    -1     1     1     1    -1    -1     1     0     0     0     0     0     2
     3     0    -2    -3     0     2    -1     1     0     0     0     0     3
    -7     0     2     7     0    -2     3     0     1     0     0     0     3
    -1     0     0     1     0     0     0     0     0     1     0     0     0
    -1     0     1     1     0    -1     1     0     0     0     1     0     2
     0     0    -1     0     0     1     0     0     0     0     0     1     0
     0     0     0     0     0     0    -1     0     0     0     0     0     0

    -1     1     1     1    -1    -1     0     0     0     0     0     2
     3     0    -2    -3     0     2     1     0     0     0     0     3
    -7     0     2     7     0    -2     0     1     0     0     0     3
    -1     0     0     1     0     0     0     0     1     0     0     0
    -1     0     1     1     0    -1     0     0     0     1     0     2
     0     0    -1     0     0     1     0     0     0     0     1     0
    -1     0    -3     1     0     3     0     0     0     0     0    -4

     0     1     1     0    -1    -1     0     0    -1     0     0     2
     0     0    -2     0     0     2     1     0     3     0     0     3
     0     0     2     0     0    -2     0     1    -7     0     0     3
    -1     0     0     1     0     0     0     0     1     0     0     0
     0     0     1     0     0    -1     0     0    -1     1     0     2
     0     0    -1     0     0     1     0     0     0     0     1     0
     0     0    -3     0     0     3     0     0    -1     0     0    -4
```

```
     0     1     0     0    -1     0     0     0    -1     0     1     2
     0     0     0     0     0     0     1     0     3     0    -2     3
     0     0     0     0     0     0     0     1    -7     0     2     3
    -1     0     0     1     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     0    -1     1     1     2
     0     0    -1     0     0     1     0     0     0     0     1     0
     0     0     0     0     0     0     0     0    -1     0    -3    -4


x =

     0     2     0


val =

    -4
```

Voici le résultat qu'on obtient par lingprog()

```
>> [x,val]=linprog([3;-2;1],[2,1,-1;-4,-3,-1],[5;-3],[-1,1,1],[2],[0;0;0])
Optimization terminated.

x =

    0.0000
    2.0000
    0.0000


val =

   -4.0000
```

On vous donnera quelques exemples dans le document "sample.txt".

4 Conclusion

Merci pour votre attention!

References

[1] Yi-Shuai.N. *Optimisation – Theorie et Algorithmes.* SJTU-ParisTech, 2015-2016.