TP1 Optimisation Continue : Compte rendu

Constanza Corentin

Juin 2022

Table des matières

1 Objectif **3**

2 Méthodes d'uzawa **3**

3 Cas quadratiques **4**

4 Cas Non Quadratique **6**

5 Conclusion **7**

1 Objectif

Le but de ce TP est d'implémenter la méthode numériques d'Uzawa pour résoudre approximativement certains problèmes de minimisation avec contraintes.
Nous considérons l'ensemble suivant des contraintes en $\mathbb{R}^n$ avec $n \in \mathbb{N}^*$:

$$U = \{x \in \mathbb{R}^n,\ \theta_i(x) \leq 0,\ \forall i \in 1, m\}$$

avec $m \in \mathbb{N}^*$ et $\theta_1, \theta_2, ..., \theta_m : \mathbb{R}^n \to \mathbb{R}$ des fonctions données.
Nous allons considérer dans ce TP des fonctions affines de la forme :

$$\theta_i(x) = \langle C_i, x\rangle + d_i, \quad i \in 1, m$$

avec $C_1, ..., C_m$ des vecteurs dans $\mathbb{R}^n$ et $d_1, ..., d_m \in \mathbb{R}$.
Pour une fonction $f : \mathbb{R}^n \to \mathbb{R}$ donnée de classe C^1 on considère le problème de minimisation avec contraintes :

$$\min_{x \in U} \ f(x)$$

c'est à dire, trouver $x^* \in U$ tel que

$$f(x^*) \leq f(x), \quad \forall x \in U.$$

Dans ce TP nous allons considérer successivement le cas quadratique et le cas non-quadratique.

2 Méthodes d'uzawa

En appliquant l'algo du cours on obtients le programme suivante :

```
function X = uzawa(x0, T, L, gradL, nbC, rho)
    %nbC est le nombre de contrainte
    %T est le tableau stokant les contraites

    Nmax = 1000;
    eps = 10^(-6);
    % On prend p=0 pour la premiere it ration
    p=zeros(nbC,1);

    X=[];
    for i = 1:Nmax
        x = wolfe(x0,eps,Nmax, @(x) L(x,p), @(x) gradL(x,p));
        X=[X x];
        if (i>1 & norm(X(:,end-1)-x)<eps)
            break;
        end
        p=max(p+rho*T(x), zeros(nbC,1));
    end
```

Pour minimiser L nous utilisons la méthodes de recherche linéaire de Wolf vu u TP précédent. Pour rappel elle s'écrit comme ceci :

```
function x = wolfe(x0,eps,Nmax, f, Grad)
    fprintf("---------------------------Wolfe------------------------------\n");
    fprintf("Avec la condition initiale :\n");
    disp(x0);

    x = x0;
    X = [];
    g = Grad(x);
    GRAD = [norm(g)];
    i=0;

    while norm(g)>eps
        i=i+1;
        if(i>Nmax)
            fprintf("Le nombre d'it ration maximal a   t   atteint, on obtient :");
```

```
            break;
        end


        % Calcul de rho
        w1 = 0.1; w2 = 0.9;
        rhoInf = 0; rhoSup = Inf; rho = 100;
        j=0;
        while 1
            j=j+1;
            if j==10000 % On s'assure que la recherche du pas aboutis
                fprintf("La recherche du rho n'a pas aboutis, on a la solution suivante
                disp(x);
                return;
            end
            if f(x-rho*g)>f(x)+w1*rho*-norm(g)^2 %premi re condition de Wolfe
                rhoSup = rho;
                rho = 1/2*(rhoInf+rhoSup);
            elseif dot(Grad(x-rho*g), -g) < -w2*norm(g)^2 % deuxi me condition de wo
                rhoInf = rho;
                if rhoSup == Inf
                    rho = 2*rhoInf;
                else
                    rho = 1/2*(rhoInf+rhoSup);
                end
            else
                break;% On sort de la boucle si les deux conditions sont v rifi e
            end
        end

        % Calcul de l'approximation
        x = x-rho*g;
        g=Grad(x);
        GRAD = [GRAD norm(g)];


    end
end
```

3 Cas quadratiques

Nous utiliserons la même fonction que durant le TP 1 pour rappel : $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$

$$A = \begin{pmatrix} 2 & -1.5 & -0.5 \\ -1.5 & 2 & 0 \\ -0.5 & 0 & 2 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Nous utiliserons le script suivant :

```

A = [2 -1.5 -0.5 ;-1.5 2 0; -0.5 0 2];
b = [1; 2; 1];
x0 = [1/2;1/2;1/2];

f = @(x) 1/2*dot(A*x,x)-dot(b,x);
GradF = @(x) A*x-b;

% On defini les contraintes, on prend di = 10
c1=[-3;-5;-3]; t1 = @(x) dot(c1,x)+10;
```

```
c2=[-8;-2;-2]; t2 = @(x) dot(c2,x)+10;

T = @(x) [t1(x);t2(x)];


% On definition la fonction lagrangienne et de son gradient
L = @(x,p) f(x)+dot(p,T(x));
gradL = @(x,p) GradF(x)+[c1 c2]*p;

str=evalc("x=uzawa(x0,T,L,gradL,2,0.01);");

x(:,end)
%
```

On remarque que nos contraintes sont assez "permisive" (ie le p.min est le même que dans le cas sans contrainte). Verifions cela, on obtient la sortie :

```
>> TP2_simple_quadratique

ans =

    3.6660
    3.7491
    1.4164
```

Ce qui est bien le resultat du TP1.

Changons maintenant les contraites, on utilise le script suivant :

Listing 1 –

```

A = [2 -1.5 -0.5 ;-1.5 2 0; -0.5 0 2];
b = [1; 2; 1];
x0 = [1/2;1/2;1/2];

f = @(x) 1/2*dot(A*x,x)-dot(b,x);
GradF = @(x) A*x-b;

% On defini les contraintes, on prend di = 10
c1=[3;0;2]; t1 = @(x) dot(c1,x)+10;
c2=[1;0;-2]; t2 = @(x) dot(c2,x)+10;

T = @(x) [t1(x);t2(x)];


% On definition la fonction lagrangienne et de son gradient
L = @(x,p) f(x)+dot(p,T(x));
gradL = @(x,p) GradF(x)+[c1 c2]*p;

str=evalc("x=uzawa(x0,T,L,gradL,2,0.01);");

x(:,end)
%
```

Et on à la sortie suivante :

```
>> TP2_simple_quadratique

ans =

   -5.0000
   -2.7500
    2.5000
```

On voit bien que l'algo c'est arreté avant le minimum sans contraite. Il s'est donc arrété sur la frontière de notre espace de contraite.

4 Cas Non Quadratique

Nous utiliserons la même fonction que durant le TP 1 pour rappel : $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle + \exp(x_1) + \exp(x_2) + \exp(x_2)$

$$A = \begin{pmatrix} 2 & -1.5 & -0.5 \\ -1.5 & 2 & 0 \\ -0.5 & 0 & 2 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Nous utiliserons le script suivant :

```
A = [2 -1.5 -0.5 ;-1.5 2 0; -0.5 0 2];
b = [1; 2; 1];
x0 = [1/2;1/2;1/2];

g = @(x) exp(x(1)) +  exp(x(2)) + exp(x(3));
gradg = @(x)[exp(x(1));exp(x(2)); exp(x(3))];


f = @(x) 1/2*dot(A*x,x)-dot(b,x) + g(x);
GradF = @(x) A*x-b +gradg(x);

% On defini les contraintes, on prend di = 10
c1=[-17;-18;-23]; t1 = @(x) dot(c1,x)+10;
c2=[-80;-20;-20]; t2 = @(x) dot(c2,x)+10;

T = @(x) [t1(x);t2(x)];


% On definition la fonction lagrangienne et de son gradient
L = @(x,p) f(x)+dot(p,T(x));
gradL = @(x,p) GradF(x)+[c1 c2]*p;

str=evalc("x=uzawa(x0,T,L,gradL,2,0.01);");

x(:,end)
%
```

On remarque que nos contraintes sont assez "permisive" (ie le p.min est le même que dans le cas sans contrainte). Verifions cela, on obtient la sortie :

```
>> TP2_non_quad

ans =

   0.199471
   0.402113
   0.033061
```

Ce qui est bien le resultat du TP1.

Changons maintenant les contraites, on utilise le script suivant :

Listing 2 –

```
A = [2 -1.5 -0.5 ;-1.5 2 0; -0.5 0 2];
b = [1; 2; 1];
```

```
x0 = [1/2;1/2;1/2];

g = @(x) exp(x(1)) +  exp(x(2)) + exp(x(3));
gradg = @(x)[exp(x(1));exp(x(2)); exp(x(3))];


f = @(x) 1/2*dot(A*x,x)-dot(b,x) + g(x);
GradF = @(x) A*x-b +gradg(x);

% On defini les contraintes, on prend di = 10
c1=[1;0;0]; t1 = @(x) dot(c1,x)+10;
c2=[0;0;1]; t2 = @(x) dot(c2,x)+10;

T = @(x) [t1(x);t2(x)];


% On definition la fonction lagrangienne et de son gradient
L = @(x,p) f(x)+dot(p,T(x));
gradL = @(x,p) GradF(x)+[c1 c2]*p;

str=evalc("x=uzawa(x0,T,L,gradL,2,0.01);");

x(:,end)
%
```

Et on à la sortie suivante :

```
>> TP2_non_quad

ans =

  -10.0214
   -6.5168
   -9.9437
```

On voit bien que l'algo c'est arreté avant le minimum sans contraite. Il s'est donc arrété sur la frontière de notre espace de contraite.

5 Conclusion

Nous avons donc bien un programe permettant l'implémentation d'Uzawa avec differentes contrainte, et ce dans le cas quadratiques comme non quadratique. Lorsque nos contraites inclus le points de minimum, celui est trouver. Sinon notre programme renvoie le points qui minimise le plus la fonction sur la frontière de notre espaces de contraite.

L'objectif premier du TP est donc remplis, cependant il y a de nombreuse amélioration possible qui n'ont pu être réalisé par manque de temps. On peut citer par exemple le tracé de graphique montrant la convergence de notre méthodes ou encore des plot 3d afin de mieux visualiser.