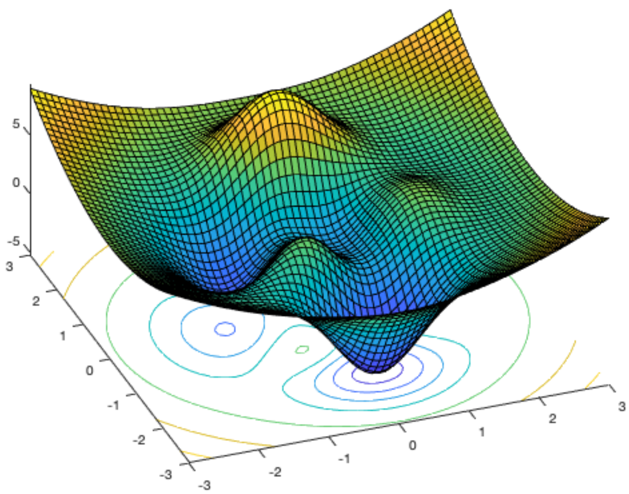
Optimisation sous contrainte
1A Ensem NRJ

Objectif : Programmation et illustration graphique de la progression des algorithmes du gradient, gradient projeté et Uzawa.
Livrable : Compte-rendu par binômes en fin de séance (4h).



1 Implémentation de l'algorithme du gradient

L'énergie dépensée par un procédé industriel lors d'un cycle de fonctionnement est paramétré par la fonction :

$$f(x,y) = 3(1-x)^2 e^{(-(x^2)-(y+1)^2)} - 10(\frac{x}{5} - x^3 - y^5)e^{(-x^2-y^2)} - \frac{1}{3}e^{(-(x+1)^2-y^2)} + 0.5(x^2+y^2)$$

où x et y sont deux paramètres intervenant sur ce cycle. On souhaite déterminer les valeurs de x et y permettant de minimiser la dépense d'énergie. Une première approche consiste à utiliser un algorithme du gradient à pas fixe.

— Analyser et tester le programme naïf (gradient à pas fixe) ci-dessous pour différents pas et positions initiales prises sur un cercle de rayon 3 (Insérer une boucle for).

Programme matlab de l'algorithme du gradient à pas fixe : (faire un copier coller)

```
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Affichage de la fonction
%%%%%%%%%%%%%%%%%%%%%%%%%%%
[x,y]=meshgrid(-3:.1:3,-3:.1:3);

z =  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
    - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
    - 1/3*exp(-(x+1).^2 - y.^2)+0.5*(x.^2+y.^2);

figure(1)
clf
hold on
surfc(x,y,z)
view(3)
%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialisation
%%%%%%%%%%%%%%%%%%%%%%%%%
% Point Initial
X=[3;-3];
% Choix du pas
rho=0.1;

test=1;
compteur=1;
precision=1.e-3
maxiter=100;
while test
    compteur=compteur+1;
    %%%%%%%%%%%%%%%%%%%
    % Affichage progression de X
    %%%%%%%%%%%%%%%%%%%
    x=X(1);
    y=X(2);
    z=3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
        - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
```

```
        - 1/3*exp(-(x+1).^2 - y.^2)+0.5*(x.^2+y.^2);

    plot3(X(1),X(2),z+0.1,'.r','MarkerSize',15)
    view(2)
    pause(.1)

    %Calcul du gradient
    g=[.8*x+x/5+(exp(-(x+1)^2-y^2)*(2*x+2))/3 ...
         +3*exp(-(y+1)^2-x^2)*(2*x-2)+exp(-x^2-y^2)*(30*x^2-2) ...
         -6*x*exp(-(y+1)^2-x^2)*(x-1)^2-2*x*exp(-x^2-y^2)*(10*x^3-2*x+10*y^5);
         .8*y+y/5+(2*y*exp(-(x+1)^2-y^2))/3+50*y^4*exp(-x^2-y^2) ...
         -3*exp(-(y+1)^2-x^2)*(2*y+2)*(x-1)^2 ...
         -2*y*exp(-x^2-y^2)*(10*x^3-2*x+10*y^5)];
    %%%%%%%%%%%%%%%%%%%%%
    % Mise à jour de X
    %%%%%%%%%%%%%%%%%%%%%
    Xold=X;
    X=X-rho*g;

    test=norm(Xold-X)>precision && compteur<maxiter;
end

if compteur==maxiter
    display('warning: Nombre d"iteration maximum atteint')
end

%%%%%%%%%%%%%%%%%%%%%%%%%
% Affichage dernier point
%%%%%%%%%%%%%%%%%%%%%%%%%
x=X(1);
y=X(2);
z=3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
    - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
    - 1/3*exp(-(x+1).^2 - y.^2)+0.5*(x.^2+y.^2);
plot3(X(1),X(2),z+.1,'*g','MarkerSize',15)
```

Remarque : L'expression symbolique du gradient g a été obtenue via la commande :

```
syms x y
z =  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2)...
- 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2)...
```

```
- 1/3*exp(-(x+1).^2 - y.^2)+0.5*(x.^2+y.^2);
g=jacobian(z)
```

- Observer (graphiquement) la progression obtenue au cours des itérations. Quelle analyse faites vous des résultats obtenus ? Comment y remédier ?
- On se propose d'adapter le pas à chaque itération afin de garantir la décroissance de la fonction f et la convergence de l'algorithme.
  Notons X_k la position, f_k la valeur de f au point X_k et ρ_k la valeur du pas au début de l'itération k. On adapte alors le pas ρ_k à chaque itération suivant le schéma :

$$f_{test} = \min_{\rho \in \{\frac{1}{2}\rho_k,\ \rho_k,\ 2\rho_k\}} f(X_k - \rho g(X_k))$$

  Si $f_{test} \leq f_k$ alors

$$\rho_{k+1} = \arg \min_{\rho \in \{\frac{1}{2}\rho_k,\ \rho_k,\ 2\rho_k\}} f(X_k - \rho g(X_k)) \tag{1}$$

$$X_{k+1} = X_k - \rho_{k+1} g(X_k) \tag{2}$$

  sinon

$$X_{k+1} = X_k \tag{3}$$

$$\rho_{k+1} = \rho_k / 4 \tag{4}$$

  fin si
  Modifier le programme matlab pour parvenir à ce résultat.
- Tester le programme réalisé pour différentes positions initiales prises sur un cercle de rayon 3 (faire une boucle for).
- Conclure sur la méthode

2 Implémentation de l'algorithme du gradient projeté

On souhaite à présent que la solution trouvée respecte la contrainte

$$AX \geq b$$

avec $X = (x, y)$, $A = [-1\ 1]$ et $b = 1$, (les paramètres x et y sont liés entre eux par cette contrainte).
- Le domaine de recherche $K = \{(x, y) : AX \geq b\}$ est il convexe ?
- Déterminer à l'aide des conditions de Kuhn et Tucker, l'opérateur de projection de X sur K,

$$\Pi_K(X) = \arg \min_{Z \in K} \frac{1}{2} \|X - Z\|^2$$

- Modifier l'algorithme à pas variable pour prendre en compte cette contrainte.

3 Implémentation de l'algorithme d'Uzawa

La contrainte est maintenant définie par :

$$c(X) = y - \cosh(x) + 1 \geq 0$$

- Le domaine de recherche $K = \{(x, y) : y - \cosh(x) + 1 \geq 0\}$ est il convexe ?
- Afin de ne pas calculer l'opérateur de projection (difficile), formuler le problème dual et mettre en oeuvre l'algorithme d'Uzawa :

  $\lambda_0 \in C^+ = \{\lambda : \lambda \geq 0\}$ donné
  Pour $k = 0, 1, 2, \ldots$
  1. Pour λ_k fixé, calculer : $X_k(\lambda_k) = \arg\min_X L(X, \lambda_k)$ (problème sans contrainte)
  2. $\lambda_{k+1} = \Pi_{C^+}(\lambda_k - \rho_k c(X_k)) = \max(0, \lambda_k - \rho_k c(X_k))$
  3. Retourner en 1 tant que $\|\lambda_{k+1} - \lambda_k\| > \epsilon$.

  Remarques : Au point 1, on pourra ré-utiliser l'algorithme du gradient à pas variable des questions précédentes pour minimiser le Lagrangien. Au point 2, on mettra en oeuvre une stratégie de pas variable sur ρ_k comme à la section 2.
- Voici une ébauche de programme :

```
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Affichage de la fonction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[x,y]=meshgrid(-3:.1:3,-3:.1:3);

z =  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
    - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
    - 1/3*exp(-(x+1).^2 - y.^2)+0.5*(x.^2+y.^2);
figure(1)
clf
hold on
surfc(x,y,z)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tracer de la contrainte y-cosh(x)+1=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=-3:.1:3;
b=-1;
```

```
y=min(max(-3,cosh(x)+b),3);
z=3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
    - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
    - 1/3*exp(-(x+1).^2 - y.^2)+0.5*(x.^2+y.^2);
figure(1)
plot3(x,y,z+.1,'b','MarkerSize',15)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Nombre itérations maximum
maxiter=100;

for theta=2*pi/10:2*pi/10:2*pi
    %%%%%%%%%%%%%%%%%%%
    % Point de départ
    %%%%%%%%%%%%%%%%%%
    X=3*[cos(theta);sin(theta)];
    lambda=0;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % calcule minimum de L par rapport X
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [Fval,X]=minimum_L(X,lambda); %(A Faire)
    L=Fval;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialisation boucle while
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    rho=1;
    compteur=0;
    test=1;
    while test  && compteur<maxiter

        compteur=compteur+1;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Affichage point courant
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        figure(1)
        z=objective(X); % Evaluation de la fonction f (A faire)
        plot3(X(1),X(2),z+.1,'.r','MarkerSize',15)
        pause(0.1)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Iteration point 2 avec pas variable rho/2, rho, et 2 rho
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    contx=contrainte(X); % Evaluation de c(X) : A faire

    lambda1=max(0,lambda-rho/2*contx);
    % Calcul point 1 pour lambda1
    [L1,X1]=minimum_L(X,lambda1);

    lambda2=max(0,lambda-rho*contx);
     % Calcul point 1 pour lambda2
    [L2,X2]=minimum_L(X,lambda2);

    lambda3=max(0,lambda-2*rho*contx);
     % Calcul point 1 pour lambda3
    [L3,X3]=minimum_L(X,lambda3);
    % Sélection du meilleur résultat
    [L,ind]=max([L1,L2,L3]);
    %%%%%%%%%%%%%%%%%%%%%%%%%%
    % mise à jour et adaptation du pas
    %%%%%%%%%%%%%%%%%%%%%%%%%%
    if ind==1
        rho=rho/2;
        lold=lambda;
        lambda=lambda1;
        X=X1;
    elseif ind==2
        lold=lambda;
        lambda=lambda2;
        X=X2;
    else
        lold=lambda;
        lambda=lambda3;
        rho=2*rho;
        X=X3;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % test arret
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    test=(norm(lold-lambda)>1.e-2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Affichage
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(1)
    z=objective(X);
    plot3(X(1),X(2),z+.2,'*g','MarkerSize',15)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Alerte arret prematuré
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if compteur==maxiter
        display('warning: Nombre d"iteration maximum atteint')
    end
end
```

- Ecrire les scripts des fonctions manquantes :

```
function [zval,gval]=objective(x)
% Calcule zval la valeur de f au point x et gval la valeur de gradient

zval=...

if nargout>1
gval= ...
end

function [cval,aval]=contrainte(x)
% Calcule cval la valeur de c(x) au point x et aval la valeur de gradient

cval=...
if nargout>1
aval= ...
end


function [Lval,Xval]=minimum_L(X,lambda)
% Calcule par l'algorithme du gradient à pas variable le minimum
% de la fonction Lagrangienne par rapport à X (lambda fixé).

...
```

- Tester le programme réalisé pour différentes positions initiales prises sur un cercle de rayon 3. Conclure sur le résultat obtenu.