Table of Contents

Mini Projet - Optimisation Continue

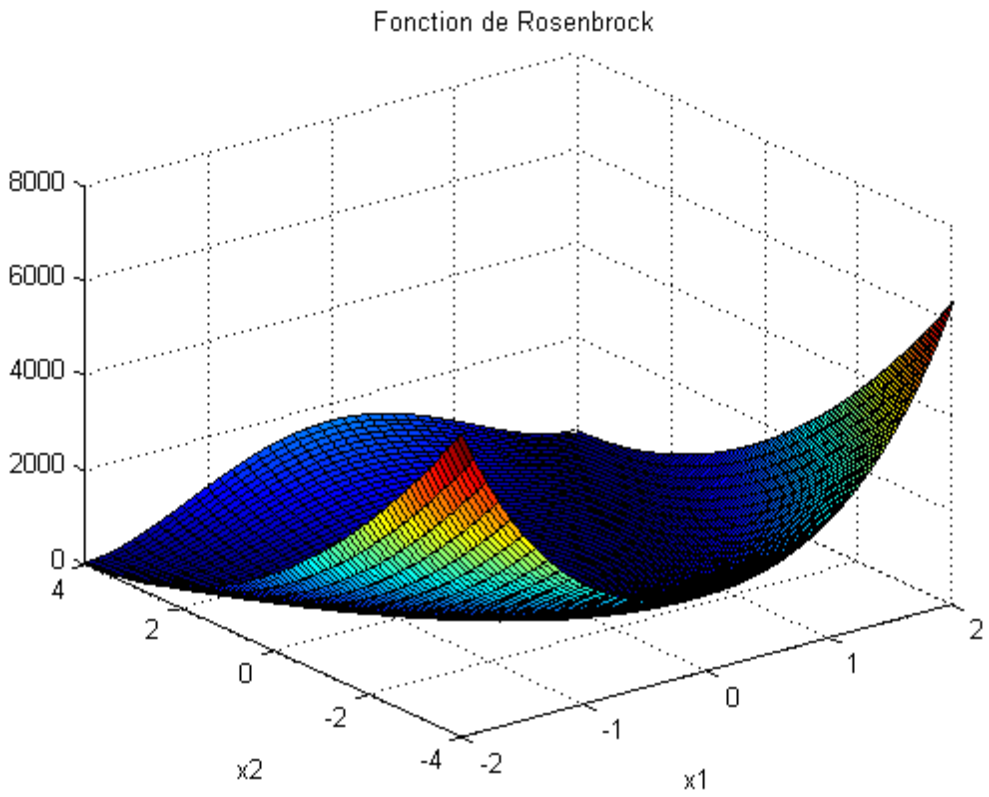
DAVY - MARCHET

```
close all;
clear;
```

Question 1

Nous allons représenter la fonction de Rosenbrock pour -2<x1<2 et -4<x2<4.

```
[X,Y] = meshgrid (-2:0.1:2, -4:0.1:4);
R=100.*(Y - X.^2).^2 + (1 - X).^2;
figure;
surf(X,Y,R);
title('Fonction de Rosenbrock');
xlabel('x1');
ylabel('x2');
```



Question 2

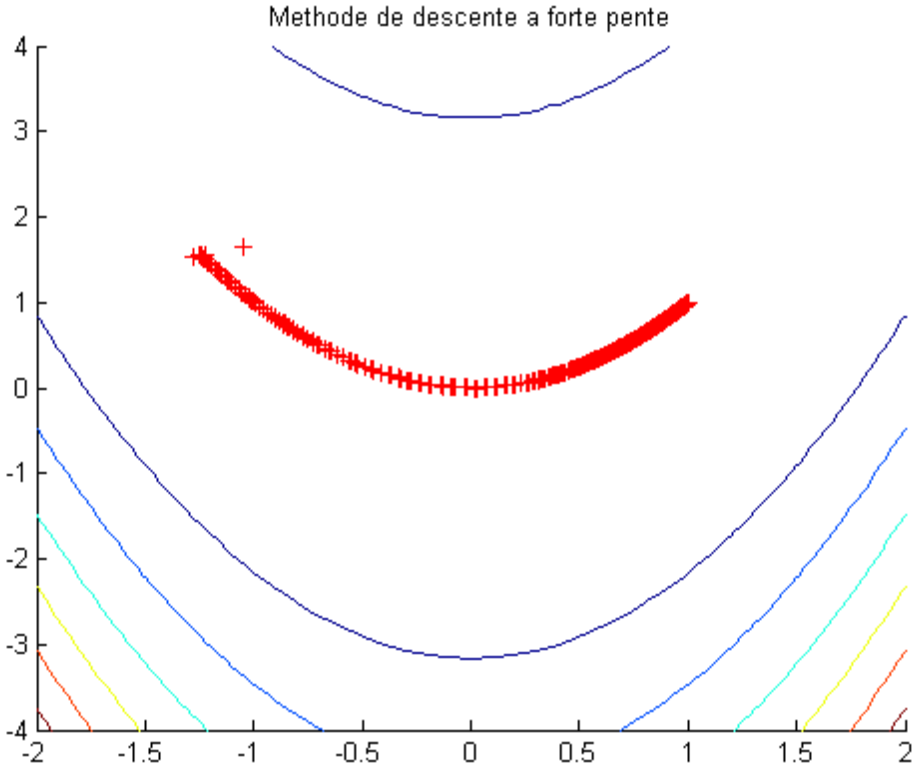
Nous allons utiliser la méthode des plus fortes pente avec préconditionnement. Pour cela, le pas va être déterminé de manière linéaire à chaque itération de façon à respecter les conditions de Wolf. La matrice Dk choisie est la matrice identité. Le pas ne doit pas être trop grand, ni trop petit. Pour cela, nous allons utilisé l'algortihme de Fletcher et Lemaéchal.

Pour calculer dk, on utilisera la fonction gradient_Rosenbrock

```
hold off;
x=[-1.5;1.5];
epsilon = 0.001;
beta1 = 0.001;
beta2 = 0.99;
lambda = 20;
figure;
title('Methode de descente a forte pente');
hold on;
contour(X,Y,R);
S = steepest_descent(x,epsilon, beta1, beta2, lambda);
```

```
iterations =

        1589
```

```
Vecteur trouvé


x =

    0.9989
    0.9978

Fonction coût en ce point

ans =

   1.2421e-06

Qualité de la solution

ans =

  -4.4535e-04

Temps de calcul: Plus forte pente
Elapsed time is 3.573111 seconds.
```



Résultats obtenus

Le vecteur trouvé est le vecteur: xf = (0.9989;0.9978). On attendait le vecteur (1;1). De plus, le nombre d'itérations est de 1589. Enfin, la durée que mettent l'algorithme pour converger vers une solution satisfaisant les conditions imposées est de 3,6 sec.

Premières conclusions

Ainsi, cette méthode n'est pas adapté à notre problème: la solution toruvée n'est pas la solution attendue

Question 3

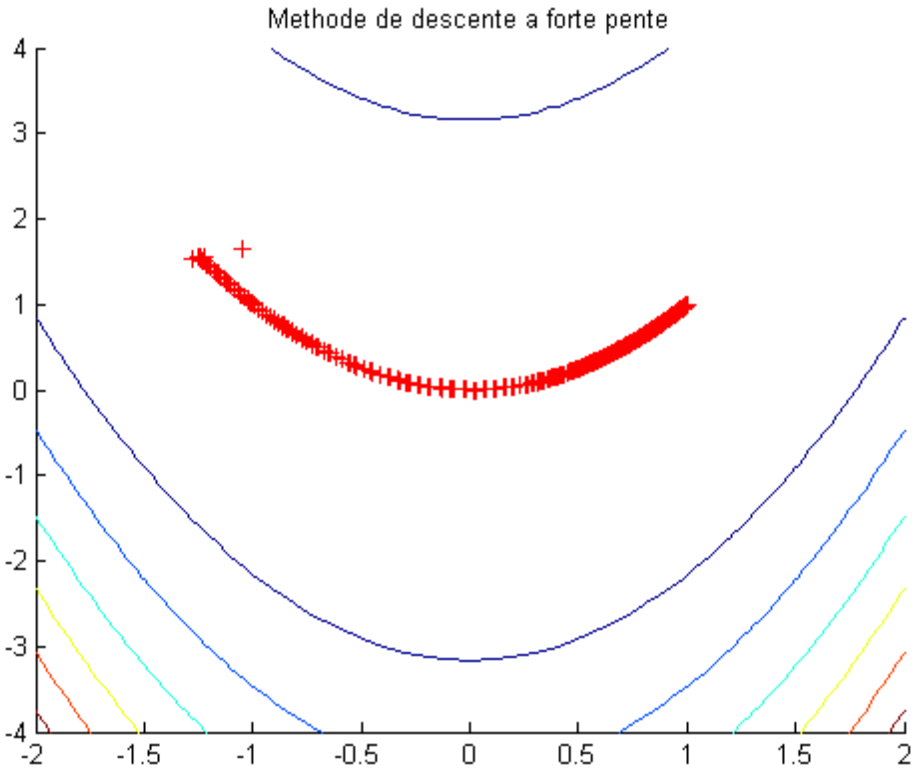
Nous allons utiliser la méthode de Newton avec recherche linéaire. Pour cela, nous utilisons le fichier newton_lineaire.m.

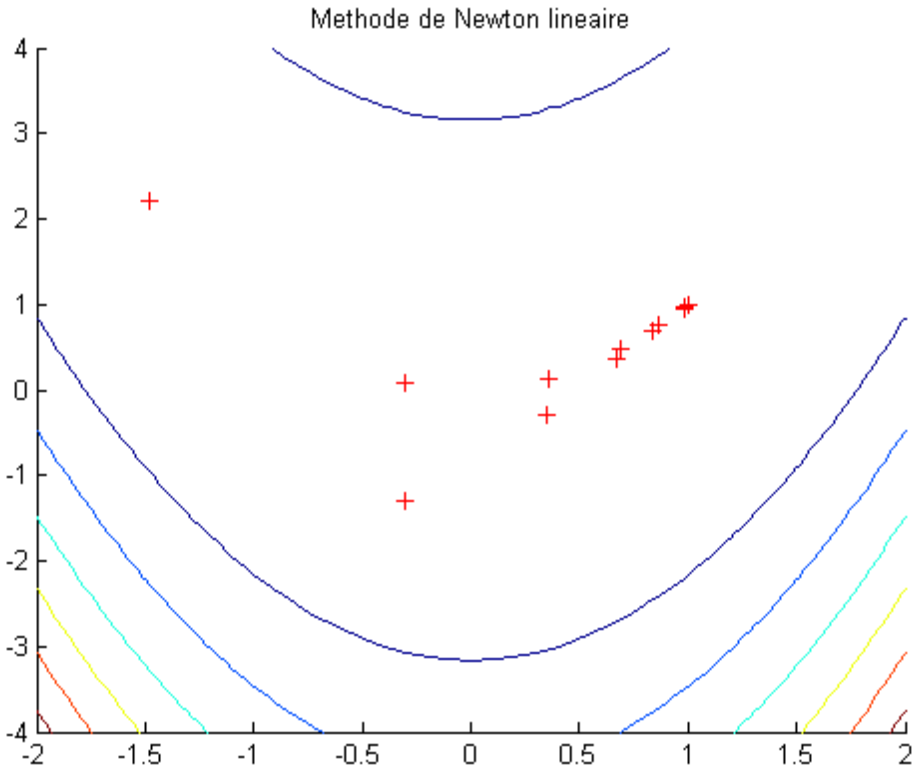
```
hold off;
x=[-1.5;1.5];
epsilon = 0.001;
beta1 = 0.001;
beta2 = 0.99;
lambda = 25;
figure;
title('Methode de Newton lineaire');
hold on;
contour (X,Y,R);
N = newton_lineaire(x, epsilon, beta1, beta2, lambda);
```

```
iterations =

    13

Vecteur trouvé

x =

    1.0000
    1.0000

Fonction coût en ce point

ans =

   1.3191e-11

Qualité de la solution

ans =

  -1.5657e-06
```

Temps de calcul: Newton Lineaire

Elapsed time is 0.065856 seconds.





Question 4

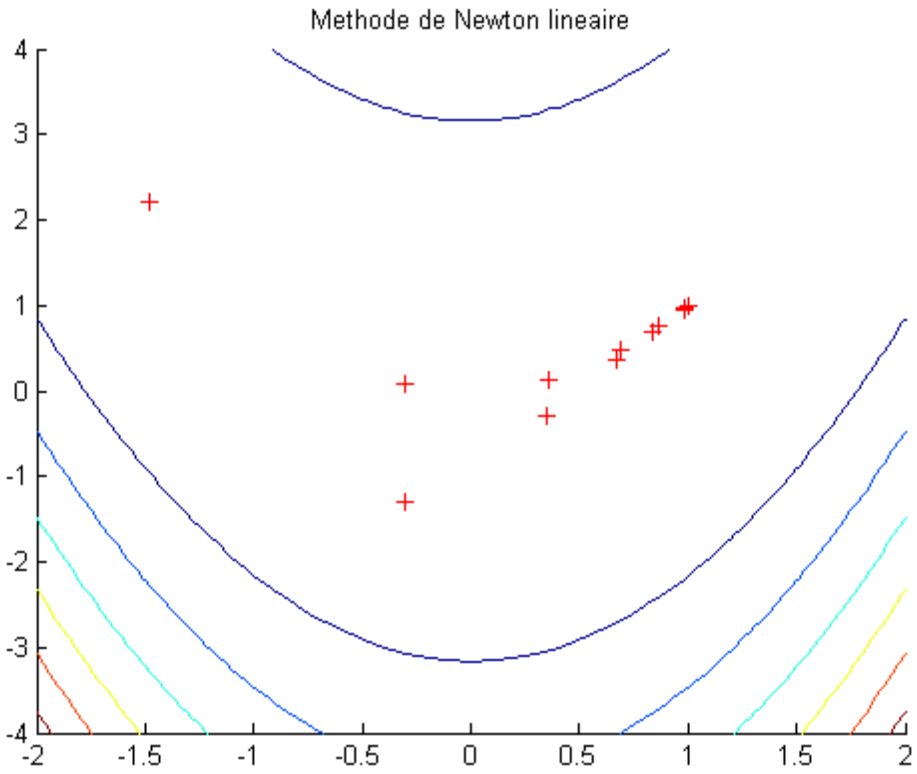
Nous allons utiliser la methode quasi-Newton BFGS Pour cela, nous utilisons le fichier BFGS.m

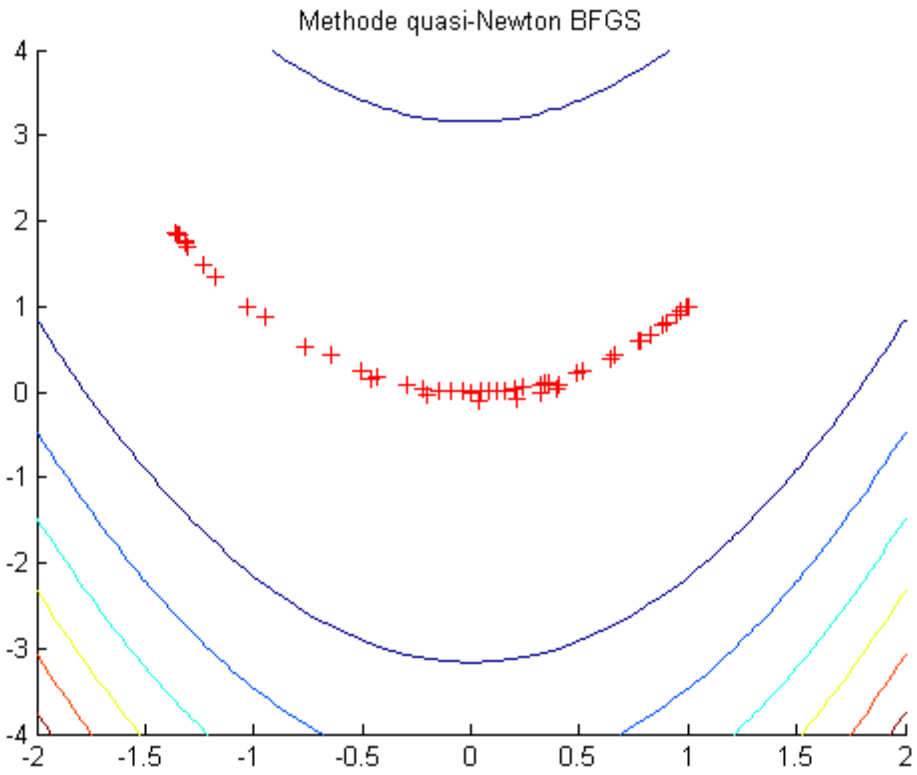
```
hold off;
x=[-1.5;1.5];
epsilon = 0.001;
beta1 = 0.001;
beta2 = 0.99;
lambda = 20;
figure;
title('Methode quasi-Newton BFGS');
hold on;
contour (X,Y,R);
B = BFGS(x, epsilon, beta1, beta2, lambda);
```

```
iterations =

    66

Vecteur trouvé

x =

    1.0000
```

```
    1.0000
```

```
Fonction coût en ce point

ans =

   3.5065e-11

Qualité de la solution

ans =

   3.3338e-05

Temps de calcul: BFGS
Elapsed time is 0.123966 seconds.
```





Published with MATLAB® R2013a