MASTER Imagerie Robotique Ingénierie pour le Vivant

An introduction to optimization

Practical work

Yassir IDSOUGOU / Abdelhak KADDOUR

TPS - PROMO 2019 -
12/03/2018

Supervisor: M.OMRAN Hassan

ECOLE D'INGENIEURS TELECOM PHYSIQUE STRASBOURG

3.3 First problem

The first problem that we're studying in this practical work is the next problem:

$$\min f(x) = \frac{1}{2}x^T Q x + q^T x \qquad s.t\ x \in R^2$$

1) We have to rank three different cases in term of speed of convergence when applying the steepest descend method

Here are the three cases:

$$Q = \begin{pmatrix} \gamma & 0 \\ 0 & \gamma \end{pmatrix} with\ \gamma > 0\ ; Q1 = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}\ ; Q2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

For each matrix, we calculate the ratio between the maximum eigenvalue and the minimum eigen value. Thanks to that, we can conclude that the method will be the fastest with Q1 and the slowest with Q.

2)

We had to write a Matlab routine to implement the steepest descend method and the conjugate gradient method, and compare their performance.

Conjugate method

```
q=[-3;-3];
x0=[-2;-7];

m=0;

deltaf=Q*x0+q;

g0=deltaf;
d0=-g0;

while norm(g0)>0.001

    alphak=-(g0'*d0)/(d0'*Q*d0);
    x0=x0+alphak*d0;
    g0=Q*x0+q;
    betak=(g0'*Q*d0)/(d0'*Q*d0);
    d0=-g0+betak*d0;

    m=m+1;
end
```

We found as a result: m=2 and x0 = $\begin{pmatrix}1.5000\\3.0000\end{pmatrix}$

Steepest Descend method

```
q=[-3;-3];
x0=[-2;-7];

m=0;

deltaf=Q*x0+q;

g0=deltaf;
d0=-g0;

while norm(g0)>0.001

    alphak=-(g0'*d0)/(d0'*Q*d0);
    x0=x0+alphak*d0;
    g0=Q*x0+q;
    betak=(g0'*Q*d0)/(d0'*Q*d0);
    d0=-g0+betak*d0;
    m=m+1;
end
```

We found as a result: m=7 and x0 = $\begin{pmatrix} 1.4997 \\ 2.9990 \end{pmatrix}$

Thanks to these results, we can conclude that Conjugate Gradient Method is more efficient than the Steepest descent method, especially because the number of steps with the first method is shorter than the second method and the final result is also more precise.

3.4 Second problem

We consider now the next function

$$f(x) = \frac{5}{2}{x_1}^2 + \frac{1}{2}{x_2}^2 + 2x_1x_2 - 3x_1 - x_2$$

1) We want to express the function in a standard quadratic form.

We have $f(x) = \frac{1}{2}x^TQx + q^Tx$ $with\ Q = \begin{pmatrix} 5 & 2 \\ 2 & 1 \end{pmatrix}$ $and\ q = \begin{pmatrix} -3 \\ -1 \end{pmatrix}$

2) We write the steps of the conjugate gradient algorithm to find the minimize of f starting from $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

Conjugate gradient algorithm

```
Q=[5 2;2 1];
x0=[0;0];
q=[-3;-1];

m=0;

deltaf=Q*x0+q;

g0=deltaf;
d0=-g0;

while norm(g0)>0.001

    alphak=-(g0'*d0)/(d0'*Q*d0);
    x0=x0+alphak*d0;
    g0=Q*x0+q;
    betak=(g0'*Q*d0)/(d0'*Q*d0);
    d0=-g0+betak*d0;

    m=m+1;
end
```

We found as a result : $x_0 = \begin{pmatrix} 1.0000 \\ -1.0000 \end{pmatrix}$ which is the minimizer of the function and m=2.

3.5 Third problem

We consider the next function : $f(x) = \frac{1}{2}x^T Qx + q^T x \ \ with\ Q = \begin{pmatrix} 2 & 2 \\ 2 & 10 \end{pmatrix}\ and\ q = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$

We want to find the minimizer of this function.

1) Find the formula for α_k in terms of Q, $g_k = \nabla f(x_k)$ and d_k.

First, we know that : $\alpha_k = \arg\ min_{\alpha \geq 0} f(x_k + \ \alpha d_k)$.

$h(\alpha) = f(x_k + \ \alpha d_k)$. After calculating $h(\alpha)$ and $h'(\alpha)$ we have: $\alpha_k = \frac{-1}{2}\frac{(x_k{}^T Q d_k + d_k{}^T Q x_k) - q^T d_k}{d_k{}^T Q d_k}$.

After that , knowing that $x_k{}^T Q d_k = d_k{}^T Q x_k$, We found : $\alpha_k = \frac{-d_k{}^T g_k}{d_k{}^T Q d_k}$.

2) We can now implement the DFP algorithm starting from $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

DFP algorithm

```
Q=[2 2;2 10];
q=[2;0];
x0=[0;0];
m=0;

deltaf=Q*x0+q;
g0=deltaf;
d0=-g0;
gk1=g0;
Hk=inv(Q);

while norm(gk1)>0.001
    gk=gk1;
    dk=-Hk*gk;
    alphak=(-dk'*gk)/(dk'*Q*dk);
    x0=x0+alphak*dk;
    deltaxk=alphak*dk;
    gk1=Q*x0+q;
    deltagk=gk1-gk;
    Hk=Hk+(deltaxk*deltaxk')/(deltaxk'*deltagk)-
(Hk*deltagk)*(Hk*deltagk)'/(deltagk'*Hk*deltagk);
    m=m+1;

end
```

We found as a result $x_0 = \begin{pmatrix} -1.2500 \\ 0.2500 \end{pmatrix}$

4.1 Third problem

We want to solve this problem: $\min ||x - x0||^2 \ with \ Ax = b$

We have $||x - x0||^2 = ||x* - x0||^2 + ||x - x*||^2 + 2(x* - x0)^T(x - x*)$

We show that for x*=$A^T(AA^T)^{-1}b + (I - A^T(AA^T)^{-1}A)x0$, $2(x* - x0)^T(x - x*)$=0.

Thanks to this result, $||x - x0||^2 = ||x* - x0||^2 + ||x - x*||^2 > ||x* - x0||^2$.

We showed that the problem has a unique solution.

4.2 Fourth problem

We had to plot the curves corresponding to the next equations:

$$4{x_2}^2 = 20 - x_1$$

$$x_2 = {x_1}^4 - 10$$

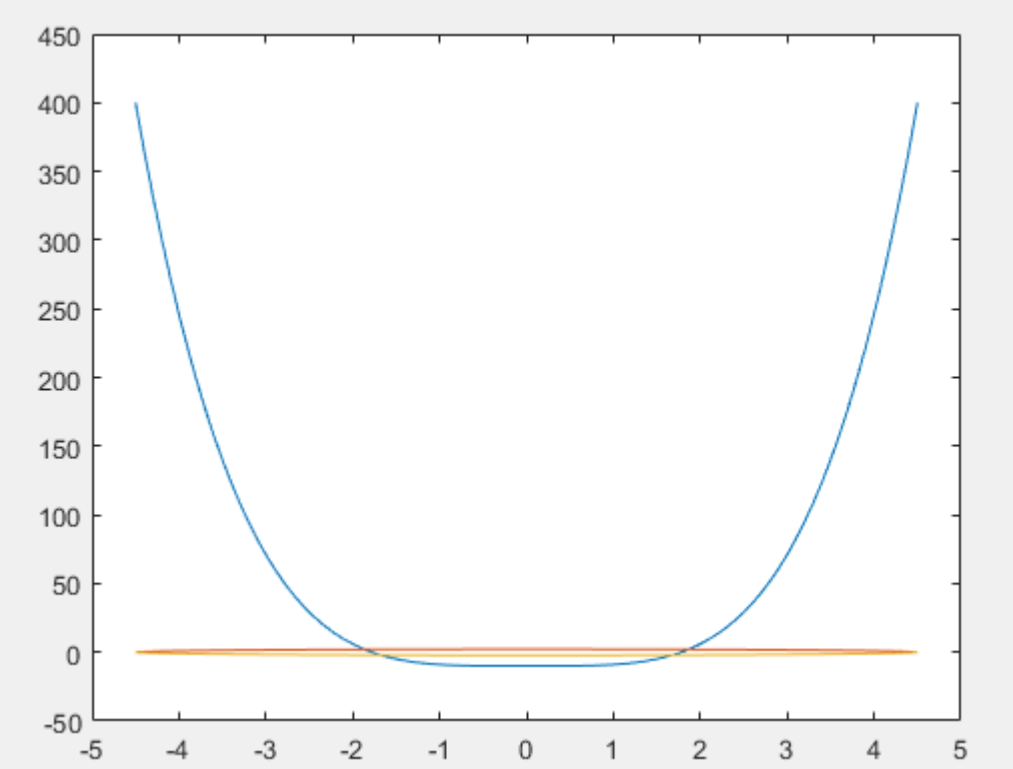
So we have to place all the next points: $(x_1, {x_1}^4 - 10), (x_1, +/-\sqrt{\frac{-{x_1}^2+20}{4}})$

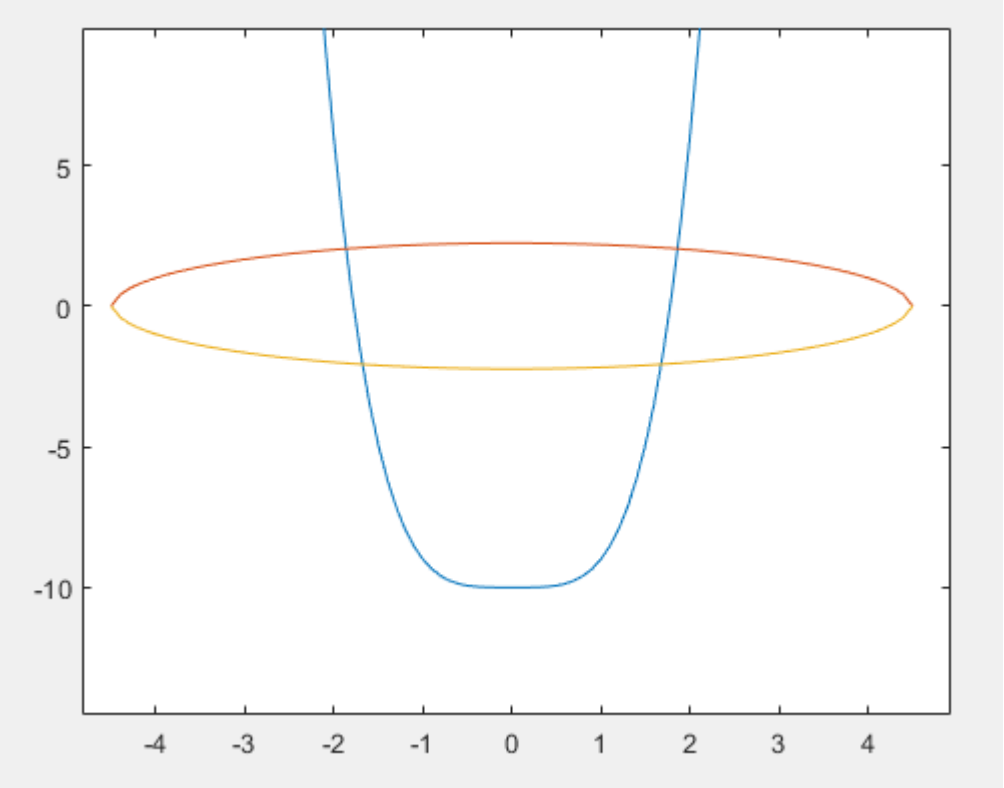
```
x1=-4.5:0.1:4.5;
x2=-1:0.1:1;

plot(x1,x1.^4-10);
hold on;

plot(x1,sqrt((-x1.^2+20)/4));
hold on;

plot(x1,-sqrt((-x1.^2+20)/4));
hold on;
```


450
400
350
300
250
200
150
100
50
0
-50
-5
-4
-3
-2
-1
0
1
2
3
4
5


5
0
-5
-10
-4
-3
-2
-1
0
1
2
3
4

We have to formalize a last square optimization problem which permits to find the intersections of the previous two curves and Implement the Gauss-Newton method to solve the problem and run it for different initial conditions.

Thanks to the curves, we can notice that there are 4 different solutions to the problem. Knowing that the algorithm is efficient to solve local problems, we can determine the 4 initials which are near to the different intersections.

Gauss-Newton Method

```
function [f1] = f1(x)
    x1=x(1,1);
    x2=x(2,1);
    f1= 4*x2^2-20+x1^2;

end

function [f2] = f2(x)
    x1=x(1,1);
    x2=x(2,1);
    f2= x2+10-x1^4;

end
```

```
xk=[2;2];
xk1=xk-inv(J(xk)'*J(xk))*J(xk)'*[f1(xk);f2(xk)];

while norm(xk1-xk)/norm(xk)>0.001
    xk=xk1;
    xk1=xk-inv(J(xk)'*J(xk))*J(xk)'*[f1(xk);f2(xk)];

end
```

Of course , J expression has been calculated in the algorithm

RESULTS

```
xk=[2;2];
xk1 =

    1.8625
    2.0329


xk=[1.5;-2];
xk1 =

    1.6780
   -2.0727


xk=[-1.5;-2];
xk1 =

   -1.6780
   -2.0727


xk=[-2;2];
xk1 =

   -1.8625
    2.0329
```

Depending on the initial condition xk , we find a different xk1 solution of the problem. And as we could think , there are 4 different solutions.

4.3 Fifth problem

We want to find the good a* and b* value which permit to solve the problem.

We find analytically the final result we can implement using Matlab

```
A= -sum(t.*t);
B= -sum(t);
C= -sum(y);
D= -sum(t.*y);

P=[A B;B -9];
p=[D;C];

Z=inv(P)*p;
```

We found as a result a=4.9885 and b=2.0345.

4.4 Fifth problem

We want know to solve a nonlinear least-squares problem

```
xk=[0.5;1.25;0.1];
a=xk(1,1);
w=xk(2,1);
phi=xk(3,1);


F=zeros(26,1);
J=zeros(26,3);

for i=1:26
    J(i,1)=-sin(w*t(1,i)+phi);
    J(i,2)=-a*t(1,i)*cos(w*t(1,i)+phi);
    J(i,3)=-a*cos(w*t(1,i)+phi);
    F(i)=y(1,i)-a*sin(w*t(1,i)+phi);

end

xk1=xk-inv(J'*J)*J'*F;
 while norm(xk1-xk)/norm(xk)>0.001
    xk=xk1;

        a=xk(1,1);
        w=xk(2,1);
        phi=xk(3,1);


    F=zeros(26,1);
    J=zeros(26,3);

    for i=1:26
        J(i,1)=-sin(w*t(i)+phi);
        J(i,2)=-a*t(i)*cos(w*t(i)+phi);
        J(i,3)=-a*cos(w*t(i)+phi);
        F(i)=y(i)-a*sin(w*t(i)+phi);

    end

    xk1=xk-inv(J'*J)*J'*F;

end
```

During this algorithm, during each step, we calculate the J and F Matrix which are used for each iteration : we have : $x_{k+1} = x_k - \left(J(x_k)^T J(x_k)\right)^{-1} J(x_k)^T f(x_k)$.We found as a result : a=0.9573

w= 1.0086 phi =-0.0409.