TRAVAUX PRATIQUES : METHODES ITERATIVES

Problème 1

Méthode de Heron d'Alexandrie pour l'extraction de racines carrées

La méthode de Newton (ou de Newton-Raphson) apparaît dans *Method of Fluxions* (Isaac Newton, publié en 1736, probablement rédigé en 1671) ainsi que dans *Analysis Aequationum* (Joseph Raphson, 1690). Elle avait cependant été déjà appliquée par Héron d'Alexandrie (Mathématicien grec, 1^{er} siècle ap. JC) pour le calcul approché de $\sqrt{2}$ (et vraisemblablement avant celà par les babyloniens). C'est encore de nos jours la méthode principalement utilisée par les calculateurs pour l'extraction de racines carrées.

Soient $\alpha \in \mathbb{R}_+^*$, et f_α la fonction définie sur $\mathbb{R}$ par :

$$f_\alpha(x) = x^2 - \alpha,$$

elle admet clairement deux zéros isolés : $\sqrt{\alpha}$ et $-\sqrt{\alpha}$.

Soit $x_0 \in \mathbb{R}^*$, et $(x_n)_{n\in\mathbb{N}}$ la suite définie par la relation de récurrence :

$$\forall\, n \in \mathbb{N}, \quad x_{n+1} = x_n - \frac{f_\alpha(x_n)}{f'_\alpha(x_n)},$$

c'est à dire par la méthode de Newton pour la recherche de zéros de f_α.

A - Existence et convergence de la suite $(x_n)_n$

1°) - Montrer que la suite $(x_n)_{n\in\mathbb{N}}$ est bien définie (*i.e.*, $\forall n \in \mathbb{N}$, $f'_\alpha(x_n) \neq 0$) et garde un signe constant.

2°) - Discuter de la convergence de $(x_n)_n$; on distinguera deux cas selon que $x_0 > 0$ ou $x_0 < 0$. (On étudiera la monotonie de la suite en montrant au préalable que $\forall n \in \mathbb{N}$, $x_{n+1} - \sqrt{\alpha} \geq 0$ lorsque $x_0 > 0$.)

B - Implémentation sous matlab

Saisir sous matlab le code suivant pour le calcul de $\sqrt{2} \approx 1,4142$.

```
N=3;
a=2;
x=1;
for i=1 : N
x=0.5*(x+a/x)
end
```

Le modifier pour calculer $\sqrt{3} \approx 1,7321$, etc...

Que dire (informellement) quant à la rapidité de convergence ?

Comment appliquer la méthode de Newton au calcul approché d'une racine n^{ieme} ?

Problème 2

Méthode du gradient conjugué appliquée à une fonction quadratique elliptique

On donne la fonction quadratique suivante :

$$f(x_1, x_2) = \frac{11}{8}\,{x_1}^2 + \frac{\sqrt{3}}{4}\,x_1 x_2 + \frac{9}{8}\,{x_2}^2 - 4\,x_1 + 7\,x_2$$

1°) - Justifier de l'existence d'un unique minimum de f sur $\mathbb{R}^2$.

2°) - Implémenter sous matlab les deux étapes de la méthode du gradient conjugué pour trouver le minimum de la fonction f en partant du point initial (10 , 10). On devra pour celà d'abord créer un fichier `Fquad.m` permettant le calcul de f et de son gradient en un point x. (Remarque : on pourra utiliser `norm()` qui retourne la norme d'un vecteur).

3°) - Retrouver ce résultat à l'aide de la fonction `quadprog`.

Problème 3

Méthode du gradient projeté pour le minimum d'une fonction quadratique sous contrainte

On considère la fonction f de l'exercice 2.

1°) - Justifier de l'existence d'un unique minimum u de f sur $\mathbb{R}_+ \times \mathbb{R}_+$.

2°) - Appliquer sous matlab la méthode du gradient projeté à pas fixe pour déterminer u. On partira du point $(10, 10)$ et on procèdera à 10 étapes.

3°) - Retrouver ce résultat à l'aide de `quadprog`.

Correction

Problème 1

Méthode de Heron d'Alexandrie pour l'extraction de racines carrées

A - Existence et convergence de $(x_n)_n$

1°) - Puisque $f'_\alpha(x_n) = 2x_n$ il suffit de montrer que $x_n \neq 0$ pour justifier de l'existence de x_{n+1}. La relation de récurrence s'écrit :

$$x_{n+1} = x_n - \frac{x_n^2 - \alpha}{2x_n} = \frac{1}{2}(x_n + \frac{\alpha}{x_n}) = \frac{1}{2x_n}(x_n^2 + \alpha) \tag{1}$$

Ainsi lorsque $x_0 > 0$ (resp. $x_0 < 0$), une récurrence (immédiate) montre que tous les x_n ($n \in \mathbb{N}$) sont bien définis et > 0 (resp. < 0). La suite $(x_n)_n$ est donc bien définie dès-lors que $x_0 \neq 0$.

2°) - *Cas où* $x_0 > 0$.

$$x_{n+1} - \sqrt{\alpha} = \frac{1}{2x_n}(x_n^2 + \alpha) - \sqrt{\alpha} = \frac{1}{2x_n}(x_n^2 + \alpha - 2x_n\sqrt{\alpha}) = \frac{1}{2x_n}(x_n - \sqrt{\alpha})^2 \geq 0$$

Alors, pour tout $n \in \mathbb{N}$,

$$x_{n+2} - x_{n+1} = \frac{1}{2x_{n+1}}(x_{n+1}^2 + \alpha) - x_{n+1} = \frac{1}{2x_{n+1}}(-x_{n+1}^2 + \alpha) \leq 0$$

La suite $(x_n)_n$ est donc décroissante à partir du rang 1. Puisqu'elle est minorée par 0, elle converge vers $l \geq 0$. Par passage à la limite dans (1), l vérifie :

$$l = \frac{1}{2l}(l^2 + \alpha) \implies l = \lim_{n\to\infty} x_n = \sqrt{\alpha}$$

Cas où $x_0 < 0$. Pour tout $n \in \mathbb{N}$, $x_n < 0$, et

$$-x_{n+1} = \frac{1}{2}(-x_n + \frac{\alpha}{-x_n})$$

La cas où $x_0 > 0$ montre alors que la suite $(-x_n)_n$ converge vers $\sqrt{\alpha}$ et donc lorsque $x_0 < 0$:

$$\lim_{n\to\infty} x_n = -\sqrt{\alpha}.$$

Remarque. Un théorème du cours nous assurait déjà 'grossièrement' de l'existence et de la convergence de $(x_n)_n$ vers $\pm\sqrt{\alpha}$ pour un choix de x_0 suffisament proche de $\pm\sqrt{\alpha}$.

B - Implémentation sous matlab

Il suffit de 3 itérations pour obtenir $\sqrt{2} = 1,4142$ à 10^{-4} près... On constate que la convergence est très rapide ! Pour obtenir une racine n^{ieme} il suffit d'appliquer la méthode de Newton à $f(x) = x^n - \alpha$ (la méthode converge pour un choix adéquat du point initial puisque les zéros de f sont isolés et la dérivée est continue et y prend une valeur non nulle). S'en souvenir si l'on a besoin dans l'avenir d'approcher la valeur d'une racine n^{ieme} en assembleur, ou en C sans utiliser la bibliothèque `math.h` (pour un gain d'espace mémoire dans un système informatique embarqué, par exemple).

Méthode du gradient conjugué appliquée à une fonction quadratique elliptique

1°) - La matrice Hessienne est :

$$H = \begin{pmatrix} \frac{11}{4} & \frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & \frac{9}{4} \end{pmatrix}$$

avec $\det(H) = 6 > 0$ et $\mathrm{tr}(H) = 5 > 0$; donc H est définie positive et f est elliptique. Ainsi f admet un unique minimum sur $\mathbb{R}^2$.

2°) - Ci–dessous, le code pour appliquer la méthode du gradient conjugué à la fonction f à partir du point initial $x_0 = (10\ ,\ 10)$. On devra d'abord définir dans un fichier `Fquad.m` :

```
function [f,gradf]=Fquad(x)
f=11/8*x(1)^2+sqrt(3)/4*x(1)*x(2)+9/8*x(2)^2-4*x(1)+7*x(2);
if nargout >1
   gradf(1)=11/4*x(1)+sqrt(3)/4*x(2)-4;
   gradf(2)=9/4*x(2)+sqrt(3)/4*x(1)+7;
end
```

On peut alors implémenter directement la méthode du gradient conjugué (elle converge en 2 étapes, cf. fig 1) :
(attention : il faut travailler avec des matrices lignes et non des vecteurs car `gradf` retourné par `Fquad` est une matrice ligne !)

```
x0=[10  10];
%%% valeur de f et du gradient en x0
[f,d0]=Fquad(x0)
%%% matrice hessienne
H=[11/4 sqrt(3)/4; sqrt(3)/4 9/4]
%%% pas dans la direction de descente
r0=d0*d0'/(d0*H*d0')
%%% point x1
x1=x0-r0*d0
%%% valeur de f et du gradient en x1
[fx1,gradfx1]=Fquad(x1)
%%% direction de descente
d1=gradfx1+norm(gradfx1)^2/norm(d0)^2*d0
%%% pas dans la direction de descente
r1=gradfx1*d1'/(d1*H*d1')
%%% point x2 minimum de f1
x2=x1-r1*d1
%%% valeur de f et du gradient en x2
[fx2,gradfx2]=Fquad(x2)
```

Les résultats obtenus :

```
f=323.3013
d0=[27.8301 33.8301]
H=[2.7500 0.4330;
    0.4330 2.2500]
r0=0.3476
x1=[0.3256 -1.7601]
fx1=-10.2404
gradfx1=[-3.8666 3.1808]
d1=[-3.5030 3.6228]
r1=0.4794
x2=[2.0052 -3.4970]
fx2=-16.2499
gradfx2=1.0e-014 *
    [-0.0888 -0.1776]
```

3°) - Avec la fonction `quadprog`, il suffit de saisir le code :

```
H=[11/4 sqrt(3)/4; sqrt(3)/4 9/4];
b=[-4 ; 7];
[xmin,fval]=quadprog(H,b)
```

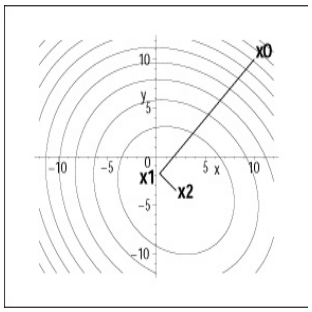

Figure 1: Points successifs

Problème 3

Méthode du gradient projeté pour le minimum d'une fonction quadratique sous contrainte

1°) - f est elliptique et admet donc un unique minimum sur le fermé convexe $\mathbb{R}_+ \times \mathbb{R}_+$.
2°) - Puisque $\det(H) = 6$ et $\mathrm{tr}(H) = 5$, H a pour valeurs propres $\lambda_1 = 2$ et $\lambda_2 = 3$. On prendra donc comme pas de descente $r = 2/(\lambda_1 + \lambda_2) = 2/5$.
Le code est le suivant (attention ici aussi l'appel de `Fquad.m` impose de travailler avec des matrices lignes et non avec des vecteurs).

```
N=10;
x=[10 10];
r=2/5;
for i=1:N
    [f,gradf]=Fquad(x);
    x=x-r*gradf;
    if x(1)<0
         x(1)=0;
    end
    if x(2)<0
         x(2)=0;
    end
end
x
f=Fquad(x)
```

3°) - Il suffit de saisir le code :

```
H=[11/4 sqrt(3)/4; sqrt(3)/4 9/4];
b=[-4 ; 7];
[x,f]=quadprog(H,b,[],[],[],[],[0 0])
```

on retrouve le même résultat :

```
x =
    1.4545
    0.0000
f =
   -2.9091
```