DEPARTAMENTO DE MATEMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Trabalho 2 – Matemática Numérica II

Liliana Carolina Vieira Pinho

Licenciatura em Matemática
Ano Letivo 2012/2013

1.Considere a função Rosenbrock

$$\boldsymbol{f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2}$$

a)Mostre que $x_* = [1 \quad 1]^T$ é o único minimizante local de f e que $\nabla^2 f(x_*)$ é definida positiva.

Tomemos como ponto de partida o seguinte teorema e definição:
Teorema: Seja $\nabla^2 f$ contínua em D aberto e $x_* \in$ D tal que $\nabla f(x_*) = 0$ e $\nabla^2 f(x_*)$ é definida positiva, então x_* é minimizante local estrito de f.
Definição: $\mathrm{x^T Ax} > 0, \forall \mathrm{x} \neq 0 \Rightarrow$ A definida positiva.

Comecemos então por provar que $\nabla^2 f(x_*)$ é definida positiva. Tomemos a matriz A como sendo $\nabla^2 f(x_*)$.

$$\frac{\delta f}{\delta x_1}(x_1, x_2) = 400{x_1}^3 + 2x_1 - 2 - 400x_1x_2$$

$$\frac{\delta f}{\delta x_2}(x_1, x_2) = -200{x_1}^2 + 200x_2$$

$$\frac{\delta^2 f}{\delta {x_1}^2}(x_1, x_2) = 1200{x_1}^2 + 2 - 400x_2$$

$$\frac{\delta^2 f}{\delta {x_2}^2}(x_1, x_2) = 200$$

$$\frac{\delta^2 f}{\delta x_1 \delta x_2}(x_1, x_2) = \frac{\delta^2 f}{\delta x_2 \delta x_1}(x_1, x_2) = -400x_1$$

Obtemos assim $\nabla^2 f(x_1, x_2) = \begin{bmatrix} 1200{x_1}^2 + 2 - 400x_2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$.

Logo $\nabla^2 f(1,1) = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$.

$\forall x \neq 0$,

$$x^T Ax > 0 \Leftrightarrow [x_1 \quad x_2] \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} > 0 \Leftrightarrow$$

$$\Leftrightarrow 802{x_1}^2 + 200{x_2}^2 - 800x_1x_2 > 0 \Leftrightarrow 802{x_1}^2 + 200{x_2}^2 > 800x_1x_2$$

Temos então que $\forall (x_1, x_2) \neq 0$, $802{x_1}^2 > 0$ e $200{x_2}^2 > 0$, logo $\mathrm{A} = \nabla^2 f(1,1)$ é definida positiva, pelo teorema podemos então concluir que $x_* = [1,1]^T$ é o único minimizante local de f.

b)Determine o minimizante de f usando o Método de Newton e o Método de BFGS. Considere $x_0 = [-1,2 \quad 1]^T$ e $B_0 = (\nabla f(x_0))^{-1}$. Faça uma tabela com os valores $\|x_k - x_*\|$ até 40 iterações para os dois métodos.

Considere a matriz gradiente e a matriz hessiana da função, respectivamente:

$$\nabla f(x_1, x_2) = \begin{bmatrix} 400{x_1}^3 - 400x_1x_2 + 2x_1 - 2 \\ -200{x_1}^2 + 200x_2 \end{bmatrix}$$

e

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 1200{x_1}^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

Construí duas funções auxiliares no MatLab para calcular o gradiente da função num determinado ponto, assim como a hessiana. Segue-se a função grad.m:

```
function [gradiente] = grad(x)
gradiente = [400*x(1)^3-400*x(1)*x(2)+2*x(1)-2; 200*x(2)-200*x(1)^2];
end
```

A matriz hessiana é dada pela função, hessi.m:

```
function [hessiana] = hessi(x)
hessiana = [1200*x(1)^2-400*x(2)+2 -400*x(1); -400*x(1) 200];
end
```

É necessário criar agora uma função auxiliar que implemente o método Newton para otimizar sem restrições, mN.m, e que obtenha o valor de cada iteração x_k.

```
function a = mN(x, it)
format long
a=x';
for k=1:it
p=(grad(x))'\(-fun(x));
x=x+p;
a=[a;(x)'];
end
```

Construindo uma função, BFGS.m, que implemente o método BFGS para otimizar sem restrições, e que obtenha o valor da cada iteração:

```
function [z] = BFGS(x, it)
format long
z = x;
B = inv(hessi(x));
for k=1:it
 s = -B*grad(x);
 gr = grad(x);
 x = x + s;
 y = grad(x)-gr;
 p = 1/(y'*s);
 B = (eye(2)-p*s*y')*B*(eye(2)-p*y*s')+p*s*(s');
 z = [z x];
end
end
```

Correndo as duas funções na linha de comando do MatLab obtemos:

- Para o método de Newton, com 1000 iterações

[>> mN([-1.2 1]', 1000)]:

x_1	x_2
-1.200000000000000	1.000000000000000
-1.087755102040816	1.000000000000000
-0.995786371514961	1.000000000000000

-0.995786371514961	-1.372456222065513
⋮	⋮
0.999999827014714	0.999999652747018
0.999999827014714	0.999999763558102
0.999999855335100	0.999999763558102

- Para o método de BFGS, com 80 iterações [>> BFGS([-1.2 1]', 80)]:

x_1	x_2
-1.200000000000000	1.000000000000000
-1.175280898876405	1.380674157303372
-1.150704065777814	1.323508193387195
0.764345961436917	-3.130258170757544
-1.146233341195833	1.319286017101812
⋮	⋮
1.000000000000014	1.000000000000029
0.999999999999999	0.999999999999997
1.000000000000000	1.000000000000000

Observando os dois quadros anteriores podemos concluir que os métodos convergem para o único mínimo local da função de Rosenbrock, ou seja, para $x_* = [1 \quad 1]^T$.

Para construir a tabela $\|x_k - x_*\|$ para as 40 primeiras iterações executei algumas alterações nas funções anteriores, obtendo as seguintes funções:

mNR.m

```
function [n] = mNR(x, it)
format long
n = [];
for k=1:it
p=(grad(x))'\(-fun(x));
x=x+p;
m = norm(x-[1 1]');
n = [n; m];
end
end
```

BFGSR.m

```
function [n] = BFGS(x, it)
format long
B = inv(hessi(x));
n=norm(x-[1 1]');
for k=1:it
 s = -B*grad(x);
 gr = grad(x);
 x = x + s;
 y = grad(x)-gr;
```

```
    p = 1/(y'*s);
    B = (eye(2)-p*s*y')*B*(eye(2)-p*y*s')+p*s*(s');
    m=norm(x-[1 1]');
    n = [n ; m];
  end
  end
```

Correndo ambas as "novas" funções no MatLab obtemos as seguintes tabelas:

- Para o método de Newton obtemos a seguinte tabela

[>> mNR([-1.2 1]', 40)]:

Iterações	x_1	Iterações	x_2
1	2.087755102040816	⋮	⋮
2	1.995786371514961	36	1.002587978726703
3	3.100276079052029	37	0.957276176301643
4	2.755020186912646	38	0.798986745327414
5	2.126103772064385	39	0.878955759363632
⋮	⋮	40	0.927829152681187

- Para o método BFGS obtemos a seguinte tabela

[>> BFGSR([-1.2 1]', 40)]:

Iterações	x_1	Iterações	x_2
1	2.200000000000000	⋮	⋮
2	2.208338697540568	36	1.331788671390998
3	2.174898970008002	37	1.272561528382526
4	4.136975390668954	38	1.141865841624014
5	2.169852786614191	39	0.844321868841273
⋮	⋮	40	1.127774737002655

Podemos verificar que aparentemente, analisando só as 40 primeiras iterações, seria de esperar que o método de Newton convergisse mais rapidamente. Porém essa discrepância não seria significativa pois a diferença entre $\|x_{40} - x_*\|_{BFGS} - \|x_{40} - x_*\|_N = 0{,}199945584321468$.

c) Se usarmos o método de descida mais rápida para resolver o problema da alínea anterior, observa-se que são necessárias 5264 iterações para reduzir a norma do gradiente para 10^{-5}. Quantas iterações dos métodos de Newton e BFGS são necessárias para fazer o mesmo?

De modo a saber quantas iterações são necessárias para alcançar a convergência, executei mais algumas alterações nos métodos.

- No método Newton [>> mNanalise([-1.2 1]')]:

```
function a = mNanalise(x)
format long
a=x';
while norm(x)>10^(-5)
p=(grad(x))'\(-fun(x));
x=x+p;
a=[a;(x)'];
end
```

Resulta num ciclo infinito, na verdade este método converge para o minimizante mas nunca chega a esse valor.

- No método BFGS [>> BFGSanalise([-1.2 1]')]:

```
function [z] = BFGSanalise(x)
format long
z = x;
B = inv(hessi(x));
while norm(x)>10^(-5)
 s = -B*grad(x);
 gr = grad(x);
 x = x + s;
 y = grad(x)-gr;
 p = 1/(y'*s);
 B = (eye(2)-p*s*y')*B*(eye(2)-p*y*s')+p*s*(s');
 z = [z x];
end
end
```

A convergência dá-se na iteração 82.

Resumidamente verifica-se que o método BFGS converge mais rapidamente para o minimizante do que o método da descida mais rápida. O método de Newton nunca consegue reduzir a norma do gradiente em menos de 10^{-5}.

2. A raiz quadrada de uma matriz A é a matriz $A^{\frac{1}{2}}$ tal que $A^{\frac{1}{2}}A^{\frac{1}{2}} = A$. Mostre que qualquer matriz simétrica definida positiva A tem uma raiz quadrada, e essa matriz também é simétrica definida positiva.

Sendo A uma matriz simétrica é diagonalizável, ou seja,

$$\exists\, M\ invertível : MDM^{-1} = A$$

Como A é simétrica podemos afirmar que M é ortogonal, ou seja, $M^{-1} = M^T$. M é constituída pelas bases ortonormadas dos subespaços próprios, e D é a matriz diagonal cujos os seus elementos diferentes de 0 são os valores próprios de A.

Por hipótese, A é definida positiva, logo os seus valores próprios também o são.

$\forall p \neq 0, p^T Ap > 0 \Rightarrow p^T \lambda p > 0 \Leftrightarrow \lambda > 0$ pois $\forall p \neq 0, p^T p > 0$.

Pretendemos encontrar a matriz $\mathrm{A}^{\frac{1}{2}}$ tal que $\mathrm{A}^{\frac{1}{2}}\mathrm{A}^{\frac{1}{2}} = \mathrm{A}$. Sabemos que a raiz quadrada de uma matriz diagonal é imediata, se $D = [d_{i,i}]$, então $D^{\frac{1}{2}} = [\sqrt{d_{i,i}}], i = 1,2, \dots, n$.

Note-se que,

$A = MDM^{-1} = MD^{\frac{1}{2}}D^{\frac{1}{2}}M^{-1} = MD^{\frac{1}{2}}M^{-1}MD^{\frac{1}{2}}M^{-1} = (MD^{\frac{1}{2}}M^{-1})(MD^{\frac{1}{2}}M^{-1})$

De onde podemos concluir que $\mathrm{A}^{\frac{1}{2}} = MD^{\frac{1}{2}}M^{-1}$. Sabemos que $\mathrm{A}^{\frac{1}{2}}$ é simétrica pois resulta do produto de uma matriz com a sua transposta (uma vez que a matriz diagonal apenas altera os valores da diagonal principal).

Falta provar que $\mathrm{A}^{\frac{1}{2}}$ é definida positiva.

$$\forall p \neq 0, p^T(MDM^{-1})p > 0 \Leftrightarrow p^T\left(MD^{\frac{1}{2}}D^{\frac{1}{2}}M^{-1}\right)p > 0$$

Como $\mathrm{D}^{\frac{1}{2}}$ é constituída por elementos positivos ao retirarmos da inequação anterior esta permanece verdadeira, ou seja,

$$p^T MD^{\frac{1}{2}}M^{-1}p > 0 \Leftrightarrow p^T A^{\frac{1}{2}}p > 0$$

Logo $\mathrm{A}^{\frac{1}{2}}$ é simétrica e definida positiva.

3. Considere uma matriz B definida positiva com valores próprios $\lambda_1, \lambda_2, \dots, \lambda_n$ onde $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Considere a função $\Psi(B) = tr(B) - \ln(\det(B))$. Mostre que $\Psi(B) > 0$.

Segundo a forma canônica de Jordan, qualquer matriz A é semelhante a outra, T, constituída por blocos triangulares superiores, em que a diagonal contêm os valores próprios de A.

Podemos então afirmar que B é semelhante a T, sendo esta formada por blocos triangulares superiores.

Matrizes semelhantes têm o mesmo traço e o mesmo determinante, logo:

$$\det(B) = \det(T) = \prod_{i=1}^{n} \lambda_i$$

e

$$tr(B) = tr(T) = \sum_{i=1}^{n} \lambda_i$$

sendo $\lambda_1, \lambda_2, \dots, \lambda_n$ são os valores próprios da matriz B.

Assim,

$$\Psi(B) = tr(B) - \ln(\det(B)) = \sum_{i=1}^{n} \lambda_i - \ln\left(\prod_{i=1}^{n} \lambda_i\right) =$$

$$= \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \ln(\lambda_i) = \sum_{i=1}^{n} (\lambda_i - \ln(\lambda_i))$$

Ora sabendo que B é uma matriz definida positiva sabemos que os seus valores próprios são todos positivos. Como $\forall x > 0, x > \ln(x)$, deste modo, $(\lambda_i - \ln(\lambda_i)) > 0, i = 1,2, \dots, n$ logo $\Psi(B) > 0$.