UNIVERSIDAD SIMÓN BOLÍVAR
CO5412: Optimización No Lineal I
Enero-Marzo 2011
Enrique Areyán

Tarea 3
Sartenejas, 14 de Febrero de 2011

1. <u>Definición:</u> Sea $f : \mathbb{R}^n \mapsto \mathbb{R}, f \in C^1$, decimos que ∇f es una función lipschitz continua si:

$$\exists \gamma > o / ||\nabla f(x) - \nabla f(y)|| \leq \gamma ||x - y||, \forall x, y \in \mathbb{R}^n$$

Para el caso en que $f : \mathbb{R} \mapsto \mathbb{R}$, tenemos la siguiente condición:

$$\exists \gamma > o / ||f'(x) - f'(y)|| \leq \gamma ||x - y||, \forall x, y \in \mathbb{R}$$

Hallaremos la constante γ para las siguientes funciones en el intervalo $[-a, a]$, con $a > 0 \in \mathbb{R}$

a) $f_1(x) = x \Rightarrow f_1'(x) = 1$. Sean $x, y \in \mathbb{R}$

$$|f_1'(x) - f_1'(y)| = |1 - 1| = |0| \leq \gamma |x - y|$$

Con $\gamma > 0$ cualquiera. Como se cumple para todo x, y, en particular se cumplira para $[-a, a]$, con $a > 0$

b) $f_2(x) = x^2 + x \Rightarrow f_2'(x) = 2x + 1$. Sean $x, y \in \mathbb{R}$

$$|f_2'(x) - f_2'(y)| = |2x + 1 - 2y - 1| = |2x - 2y| = |2(x - y)| = 2|x - y|$$

$$\Rightarrow 2|x - y| = |f_2'(x) - f_2'(y)| \leq 2|x - y|$$

Con $\gamma \geq 2$ cualquiera. Como se cumple para todo x, y, en particular se cumplira para $[-a, a]$, con $a > 0$

c) $f_3(x) = e^x - 1 \Rightarrow f_2'(x) = e^x$

2. Sea $f(x) = \frac{1}{2}||F(x)||^2$, donde $F : \mathbb{R}^n \mapsto \mathbb{R}^n, F \in C^1$. Consideremos:

$$x_{k+1} = x_k - \lambda_k (F'(x_k))^{-1} F(x_k)$$

Probaremos que si en la condición de Armijo usamos $\alpha = \frac{1}{2}$, entonces $\frac{f(x_{k+1})}{f(x_k)} \leq 1 - \lambda_k$ Por definición, la siguiente es la condición de Armijo:

$$f(x_{k+1}) \leq f(x_k) + \eta \alpha_k \nabla f(x_k)^t d_k$$

Por definición sabemos que $d_k = (F'(x_k))^{-1} F(x_k)$ y que la constante que antes llamamos η ahora llamamos α con valor $\frac{1}{2}$. Reemplanzando:

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2} \lambda_k \nabla f(x_k)^t (F'(x_k))^{-1} F(x_k)$$

Ahora debemos calcular $\nabla f(x)$. Antes debemos darnos cuenta del hecho que:

$$f(x) = \frac{1}{2}[F_1(x)^2 + F_2(x)^2 + ... + F_n(x)^2]$$

Donde cada $F_i : \mathbb{R}^n \mapsto \mathbb{R}$, con $i = 1, 2, ..., n$
EL gradiente de esta función es derivar parcialmente la función con respecto a las variables $x_1, x_2, ..., x_n$. Por ejemplo, la primera fila del gradiente queda:

$$\frac{\partial f(x)}{\partial x_1} = \frac{1}{2}\frac{\partial(F_1(x)^2 + F_2(x)^2 + ... + F_n(x)^2)}{\partial x_1} = \frac{1}{2}[\frac{\partial F_1(x)^2}{\partial x_1} + \frac{\partial F_2(x)^2}{\partial x_1} + ... + \frac{\partial F_n(x)^2}{\partial x_1}] =$$

Por la regla de la cadena

$$\frac{F_1(x)\partial F_1(x)}{\partial x_1} + \frac{F_2(x)\partial F_2(x)}{\partial x_1} + ... + \frac{F_n(x)\partial F_n(x)}{\partial x_1}$$

Por lo que el gradiente (vector de dimensiones $n \times 1$) se expresa como:

$$\nabla f(x) = \begin{pmatrix} \frac{F_1(x)\partial F_1(x)}{\partial x_1} + \frac{F_2(x)\partial F_2(x)}{\partial x_1} + ... + \frac{F_n(x)\partial F_n(x)}{\partial x_1} \\ \frac{F_1(x)\partial F_1(x)}{\partial x_2} + \frac{F_2(x)\partial F_2(x)}{\partial x_2} + ... + \frac{F_n(x)\partial F_n(x)}{\partial x_2} \\ ... \\ \frac{F_1(x)\partial F_1(x)}{\partial x_n} + \frac{F_2(x)\partial F_2(x)}{\partial x_n} + ... + \frac{F_n(x)\partial F_n(x)}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial F_1(x)}{\partial x_1} & \frac{\partial F_2(x)}{\partial x_1} & ... & \frac{\partial F_n(x)}{\partial x_1} \\ \frac{\partial F_1(x)}{\partial x_2} & \frac{\partial F_2(x)}{\partial x_2} & ... & \frac{\partial F_n(x)}{\partial x_2} \\ ... \\ \frac{\partial F_1(x)}{\partial x_n} & \frac{\partial F_2(x)}{\partial x_n} & ... & \frac{\partial F_n(x)}{\partial x_n} \end{pmatrix} = \begin{pmatrix} F_1(x) \\ F_2(x) \\ ... \\ F_n(x) \end{pmatrix}$$

Si nos damos cuenta el gradiente lo podemos descomponer en una matriz M multiplicada por un vector V. Dicha matriz es la matriz jacobiana pero transpuesta. Es decir:

$$\nabla f(x)^t = (M \times V)^t = (V^t \times M^t) = F(x)^t \times F'(x)$$

Reemplazando esta definición del gradiente en la condición de armijo hallamos la expresión deseada:

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2}\lambda_k (F(x)^t F'(x))(F'(x_k))^{-1} F(x_k) = f(x_k) - \frac{1}{2}\lambda_k F(x)^t (F'(x)F'(x_k)^{-1})F(x_k) =$$

$$= f(x_k) - \frac{1}{2}\lambda_k F(x)^t I_n F(x_k) = f(x_k) - \frac{1}{2}\lambda_k F(x)^t F(x_k) = f(x_k) - \lambda_k \frac{F_1(x)^2 + F_2(x)^2 + ... + F_n(x)^2}{2} =$$

$$= f(x_k) - \lambda_k f(x_k) \Rightarrow \frac{f(x_{k+1})}{f(x_k)} \leq 1 - \lambda_k$$

3. El siguiente código fue empleado en matlab para el método del gradiente con búsqueda lineal inexacta. Se utilizó una estrategia orientada a objetos, con una clase base en la que se implementó el método y 3 clases hijas, cada uno con la implementación de la función y el gradiente para cada caso:

```
classdef GradienteBLI
    %GrandienteBLI
    %   Calcula el minimo de una funcion con el metodo del gradiente
    %   y para el calculo del paso utiliza un backtraking con condicion
    %   de Armijo.
    %   Esta clase debe ser extendida por otra clase que implemente la
```

```
%   funcion que se quiere minimizar

properties
    eta = 10^-4;
    ro  = 1/2;
    xmin = [0;0];
    statfilename = '';
end
methods(Static)
    function [ret] = norma(x)
        ret = sqrt(x(1)*x(1)+x(2)*x(2));
    end
end
methods(Abstract)
    [ret] = f(obj,arg)
    [fd_x,fd_y] = grad_f(obj,arg)
end
methods
    function obj = GradienteBLI(arg)
        if ~isempty(arg)
            obj.eta = arg(1);
            obj.ro = arg(2);
        end
    end
    function [d_x,d_y] = d(obj,arg)
        [fd_x,fd_y] = obj.grad_f(arg);
        d_x= -1 * fd_x;
        d_y= -1 * fd_y;
    end
    function [lambda] = backtracking(obj,xk)
        eta_local = obj.eta;
        ro_local = obj.ro;
        lambda = 1;
        parar = false;
        k=0;
        while ~parar
            [d1,d2] = obj.d(xk);
            d = [d1;d2];
            [g1,g2] = obj.grad_f(xk);
            g = [g1;g2];
            %fprintf('\n\t====K=%d\n',k);
            %fprintf('\n\t\tComparando %f con %f\n',obj.f(xk + lambda * d),obj.f(xk)+eta_loca
            if(obj.f(xk + lambda * d) > obj.f(xk)+eta_local*lambda*g'*d) %condicion de armijo
                lambda = lambda*ro_local;
                xk = xk+lambda*d;
            else
                parar = true;
            end
            k=k+1;
        end
```

```
        end
        function [ret]=metodogradiente(obj,xk)

            fid = fopen(strcat('stats',obj.statfilename,'.txt'),'w');
            fprintf(fid,'k\tkx\t||xk-xmin||\t||grad_f(xk)||\n');

            fid_tex = fopen(strcat('stats',obj.statfilename,'.tex'),'w');
            fprintf(fid_tex,'\\underline{Funcion:} $%s$ \\\\ \nParametros: $\\eta = %f; \\rho = %
            fprintf(fid_tex,'\\begin{tabular}{|c|c|c|c|}\\hline\n\t$k$&$x_k$&$||x_k-x_*||$&$||gra

            parar = false;
            k = 0;
            while ~parar
                [g1,g2] = obj.grad_f(xk);
                normagrad = obj.norma([g1;g2]);
                fprintf(fid,'%d\t(%f,%f)\t%f\t%f\n',k,xk(1),xk(2),obj.norma(xk-obj.xmin),normagra
                fprintf(fid_tex,'\n\t%d&(%f,%f)&%f&%f\\\\ \\hline',k,xk(1),xk(2),obj.norma(xk-obj
                if normagrad < 10^-5 %condicion de terminacion del algoritmos
                    parar = true;
                else
                    [d1,d2] = obj.d(xk);
                    d = [d1;d2];
                    lambda = obj.backtracking(xk);
                    xk = xk+lambda*d;
                end
                k = k+1;
                if(k>1000) % Seguro contra estancamiento del algoritmo
                    parar=true;
                end
            end
            fprintf(fid_tex,'\n\\end{tabular}\\\\\\\\');

            fclose(fid_tex);
            fclose(fid);

            ret = xk;
        end
    end
end
```

Para la función 1:

```
classdef f1 < GradienteBLI
    % Funcion particular
    %   Implementa la funcion 1 de la tarea: 3x1^2+ 2x1x2 + x2^2

    properties
    end
```

```
    methods
        function obj = f1(arg)
              obj = obj@GradienteBLI(arg);
              obj.xmin = [0;0];
              obj.statfilename = 'f_1';
         end

        function [ret]=f(obj,arg)
            ret = 3*arg(1)*arg(1)+ 2*arg(1)*arg(2) + arg(2)^2;
        end
        function [fd_x,fd_y]=grad_f(obj,arg)
            fd_x = 6*arg(1)+2*arg(2);
            fd_y = 2*arg(1)+2*arg(2);
        end
    end
end
```

Para la función 2:

```
classdef f2 < GradienteBLI
    % Funcion particular
    %   Implementa la funcion 2 de la tarea.

    properties
    end
    methods
        function obj = f2(arg)
              obj = obj@GradienteBLI(arg);
              obj.xmin = [1;1];
              obj.statfilename = 'f_2';
         end
        function [ret]=f(obj,arg)
            ret = 1/2*(arg(1)^4-2*arg(1)^2*arg(2)+arg(2)^2+1-2*arg(1)+arg(1)^2);
        end
        function [fd_x,fd_y]=grad_f(obj,arg)
            fd_x = 2*arg(1)^3-2*arg(1)*arg(2)+arg(1)-1;
            fd_y = -1*arg(1)^2+arg(2);
        end
    end
end
```

Para la función 3:

```
classdef f3 < GradienteBLI
    % Funcion particular
    %   Implementa la funcion 3 de la tarea.

    properties
```

```
    end
    methods
        function obj = f3(arg)
              obj = obj@GradienteBLI(arg);
              obj.xmin = [1;1];
              obj.statfilename = 'f_3';
         end
        function [ret]=f(obj,arg)
            ret = 100*(arg(2)-arg(1)^2)^2+(1-arg(1))^2;
        end
        function [fd_x,fd_y]=grad_f(obj,arg)
            fd_x = 100*(-4*arg(1)*arg(2)+4*arg(1)^3)-2+2*arg(1);
            fd_y = 100*(2*arg(2)-2*arg(1)^2);
        end
    end
end
```

Y el código utilizado para ejecutar las pruebas simultáneamente:

```
clear
clc

f1 = f1([]); %sin parametros toma los valores por defecto, eta = 10^-4 y ro = 1/2
[xf1] = f1.metodogradiente([1;1]) % punto de partida (1,1)^t

%f1 = f1([0.00001;0.75]); %el primer parametro es eta y el segundo ro
%[xf1] = f1.metodogradiente([1;1])

f2 = f2([]); %sin parametros toma los valores por defecto, eta = 10^-4 y ro = 1/2
[xf2] = f2.metodogradiente([2;2])

%f3 = f3([0.01;0.001]); %el primer parametro es eta y el segundo ro
%[xf3] = f3.metodogradiente([1.2;1.2])

f3 = f3([0.01;0.001]); %el primer parametro es eta y el segundo ro
[xf3] = f3.metodogradiente([-1.2;1])
```

4. Antes de presentar los resultados, analizaremos las funciones con las condiciones de optimalidad:

$\underline{f_1}$: $f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$

$$\vec{0} = \nabla f(\bar{x}) = \begin{pmatrix} \frac{\partial f(\bar{x})}{\partial x_1} \\ \frac{\partial f(\bar{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 6x_1 + 2x_2 \\ 2x_1 + 2x_2 \end{pmatrix} \Rightarrow x_1 = x_2 = 0$$

$$\nabla^2 f(\bar{x}) = \begin{pmatrix} 6 & 2 \\ 2 & 2 \end{pmatrix}$$

Luego, f_1 es una función estrictamente convexa, de donde se desprende que el mínimo global es $(0,0)^t$

$\underline{f_2}$: $f(x) = 1/2(x_1^2 - x_2)^2 + 1/2(1-x_1)^2$

$$\nabla f(\bar{x}) = \begin{pmatrix} \frac{\partial f(\bar{x})}{\partial x_1} \\ \frac{\partial f(\bar{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1^3 - 2x_1x_2 + x_1 - 1 \\ x_2 - x_1^2 \end{pmatrix}$$

Se cumple que $\nabla f(1,1) = (0,0)^t$, luego $(1,1)^t$ cumple con la condición de primer orden. Verifiquemos el hessiano.

$$\nabla^2 f(\bar{x}) = \begin{pmatrix} 6x_1^2 - 2x_2 + 1 & -2x_1 \\ -2x_1 & 1 \end{pmatrix}$$

$$\nabla^2 f(1,1) = \begin{pmatrix} 6(1)^2 - 2(1) + 1 & -2(1) \\ -2(1) & 1 \end{pmatrix} = \begin{pmatrix} 5 & -2 \\ -2 & 1 \end{pmatrix}$$

Como $\nabla f(1,1) = \vec{0}$ y $\nabla^2 f(1,1)$ es una matriz positivo definida, entonces $(1,1)^t$ es un mínimo local.

$\underline{f_3}$: $f(x) = 100(x_2 - x_1^2) + (1-x_1)^2$

$$\nabla f(\bar{x}) = \begin{pmatrix} \frac{\partial f(\bar{x})}{\partial x_1} \\ \frac{\partial f(\bar{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 100(4x_1^3 - 4x_1x_2) + 2x_1 - 2 \\ 100(2x_2 - 2x_1^2) \end{pmatrix}$$

Se cumple que $\nabla f(1,1) = (0,0)^t$, luego $(1,1)^t$ cumple con la condición de primer orden. Verifiquemos el hessiano.

$$\nabla^2 f(\bar{x}) = \begin{pmatrix} 100(12x_1^2 - 4x_2) & 100(-4x_1) \\ 100(-4x_1) & 200 \end{pmatrix}$$

$$\nabla^2 f(1,1) = \begin{pmatrix} 800 & -400 \\ -400 & 200 \end{pmatrix}$$

Como esta matriz es positiva semidefinida, no podemos concluir nada acerca del punto $(1,1)$

<u>Resultados del método del gradiente con búsqueda lineal inexacta:</u>

<u>Funcion:</u> f_1
Parametros: $\eta = 0{,}000100; \rho = 0{,}500000; x_0 = (1{,}000000; 1{,}000000)$; $x_* = (0{,}000000, 0{,}000000)$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$
0	(1.000000,1.000000)	1.414214	8.944272
1	(-1.000000,0.000000)	1.000000	6.324555
2	(0.500000,0.500000)	0.707107	4.472136
3	(-0.500000,0.000000)	0.500000	3.162278
4	(0.250000,0.250000)	0.353553	2.236068
5	(-0.250000,0.000000)	0.250000	1.581139
.	...	...	...
30	(0.000031,0.000031)	0.000043	0.000273
31	(-0.000031,0.000000)	0.000031	0.000193
32	(0.000015,0.000015)	0.000022	0.000136
33	(-0.000015,0.000000)	0.000015	0.000097
34	(0.000008,0.000008)	0.000011	0.000068
35	(-0.000008,0.000000)	0.000008	0.000048
36	(0.000004,0.000004)	0.000005	0.000034
37	(-0.000004,0.000000)	0.000004	0.000024
38	(0.000002,0.000002)	0.000003	0.000017
39	(-0.000002,0.000000)	0.000002	0.000012
40	(0.000001,0.000001)	0.000001	0.000009

Funcion: f_2
Parametros: $\eta = 0{,}000100; \rho = 0{,}500000; x_0 = (2{,}000000; 2{,}000000); x_* = (1{,}000000, 1{,}000000)$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$
0	(2.000000,2.000000)	1.414214	9.219544
1	(0.875000,2.250000)	1.256234	3.101004
2	(1.555664,1.878906)	1.039826	2.303929
3	(0.435931,2.149498)	1.280438	3.000587
4	(1.004043,1.659633)	0.659645	1.457961
5	(1.330115,1.496750)	0.596437	1.089522
.	...	...	...
118	(1.000037,1.000093)	0.000100	0.000018
119	(1.000037,1.000075)	0.000083	0.000034
120	(1.000028,1.000075)	0.000080	0.000020
121	(1.000030,1.000070)	0.000076	0.000015
122	(1.000019,1.000060)	0.000063	0.000033
123	(1.000025,1.000055)	0.000060	0.000018
124	(1.000021,1.000054)	0.000058	0.000012
125	(1.000023,1.000042)	0.000048	0.000033
126	(1.000015,1.000043)	0.000046	0.000017
127	(1.000018,1.000040)	0.000044	0.000010
128	(1.000016,1.000039)	0.000042	0.000008

Funcion: f_3
Parametros: $\eta = 0{,}010000; \rho = 0{,}001000; x_0 = (1{,}200000; 1{,}200000); x_* = (1{,}000000, 1{,}000000)$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$
0	(1.200000,1.200000)	0.282843	125.169325
1	(1.084400,1.248000)	0.261968	34.274055
2	(1.115495,1.233585)	0.260578	5.465491
3	(1.110470,1.235734)	0.260334	1.064244
4	(1.111400,1.235216)	0.260262	0.219824
5	(1.111180,1.235214)	0.260167	0.098826
.	...	...	...
47	(1.000068,1.000138)	0.000154	0.000888
48	(1.000069,1.000138)	0.000154	0.000062
49	(1.000040,1.000083)	0.000093	0.001515
50	(1.000041,1.000083)	0.000092	0.000037
51	(1.000041,1.000083)	0.000092	0.000037
52	(1.000025,1.000050)	0.000056	0.000023
53	(1.000025,1.000050)	0.000055	0.000022
54	(1.000015,1.000030)	0.000033	0.000015
55	(1.000015,1.000030)	0.000033	0.000013
56	(1.000009,1.000018)	0.000020	0.000014
57	(1.000009,1.000018)	0.000020	0.000008

Funcion: f_3
Parametros: $\eta = 0{,}010000; \rho = 0{,}001000; x_0 = (-1{,}200000; 1{,}000000); x_* = (1{,}000000, 1{,}000000)$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$
0	(-1.200000,1.000000)	2.200000	232.867688
1	(-0.984400,1.088000)	1.986350	49.030587
2	(-1.027272,1.064209)	2.028288	1.826163
3	(-1.026883,1.062424)	2.027844	1.774738
4	(-1.026089,1.060837)	2.027002	1.775047
5	(-1.025311,1.059241)	2.026178	1.775453
.	...	...	...
990	(0.317685,0.097831)	1.131134	1.151805
991	(0.318656,0.098449)	1.130054	1.149153
992	(0.319625,0.099068)	1.128977	1.146512
993	(0.320590,0.099686)	1.127902	1.143882
994	(0.321553,0.100305)	1.126829	1.141263
995	(0.322512,0.100923)	1.125757	1.138654
996	(0.323468,0.101541)	1.124688	1.136056
997	(0.324421,0.102159)	1.123621	1.133469
998	(0.325371,0.102777)	1.122556	1.130893
999	(0.326319,0.103395)	1.121493	1.128327
1000	(0.327263,0.104013)	1.120432	1.125772

Según los resultados obtenidos se puede concluir que el método funciona bien pero depende en gran medida del valor de los parámetros empleados en la condición de Armijo. Por ejemplo, para la función 3 se tuvo que modificar tanto el valor de η como de ρ para que el algoritmo lograra la convergencia. En general estos fueron los resultados:

a) Para la función 1 el algoritmo siempre convergió al mínimo global $(0,0)^t$, sigsageando, relativamente rápido y con un error muy pequeño.

b) Para la función 2 el algoritmo también convergió, aunque mucho más lento al mínimo local $(1,1)^t$ y con un error más grande.

c) Para la función 3 el algoritmo se corrió con dos conjuntos de parámetros distintos. El comportamiento del mismo fue muy distinto, fuertemente relacionado al punto de partida. En el primer caso el punto convergió a un punto estacionario, pero que no se puede asegurar que es mínimo. En el segundo caso convergió a un punto que evidentemente no es estacionario por lo tanto, no puede ser mínimo.

5. La siguiente tabla contiene un resumen de los datos obtenidos por el algoritmo:

	η	ρ	x_0	$\bar{x}$	x_*	$\nabla f(\bar{x})$	¿Mínimo?
f_1	10^{-4}	0,5	$(1,1)^t$	$(0{,}000001; 0{,}000001)^t$	$(0,0)^t$	≈ 0	Sí, global
f_2	10^{-4}	0,5	$(2,2)^t$	$(1{,}000016, 1{,}000039)^t$	$(1,1)^t$	≈ 0	Sí, local
f_3	0,010000	0,001000	$(1{,}2; 1{,}2)^t$	$(1{,}000009, 1{,}000018)^t$	$(1,1)^t$	≈ 0	?
f_3	0,010000	0,001000	$(-1{,}2; 1)^t$	$(0{,}327263, 0{,}104013)^t$	$(1,1)^t$	$(-0{,}9; -0{,}6)$	No