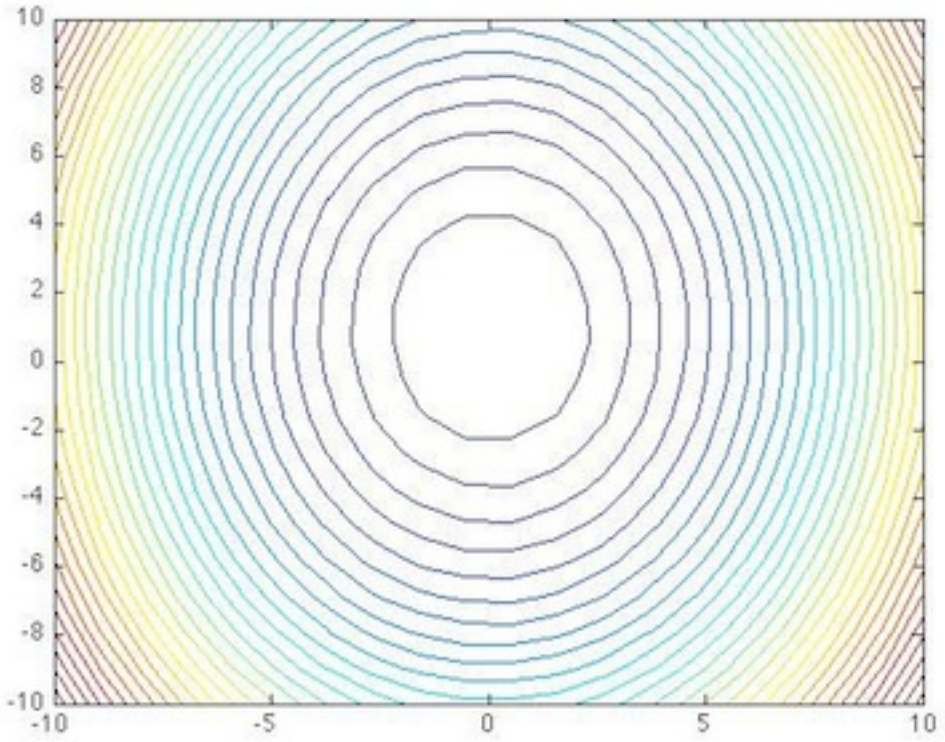
UNIVERSIDAD SIMÓN BOLÍVAR
CO5412: Optimización No Lineal I
Enero-Marzo 2011
Enrique Areyán

Tarea 5
Sartenejas, 4 de Marzo de 2011

1. Sea $f(x) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$

   a) Modelo cuadrático para $x_c = (0, -1)^t$, con $H_c = \nabla^2 f(x_c)$:



Código utilizado en Matlab para generar las curvas de nivel

```
x=linspace(-10,10,20);
y=linspace(-10,10,20);
[x,y] = meshgrid(x,y);
%Graficar el objeto
f1 = f1RegionConfianza([]);
for i=1:length(x)
    for j=1:length(y)
        z(i,j) = f1.aprox_cuadratica([0;-1],[x(i,j);y(i,j)]);
        %z(i,j) = f1.aprox_cuadratica([0;0.5],[x(i,j);y(i,j)]);
    end
```

```
end

contour(x,y,z,30)
```

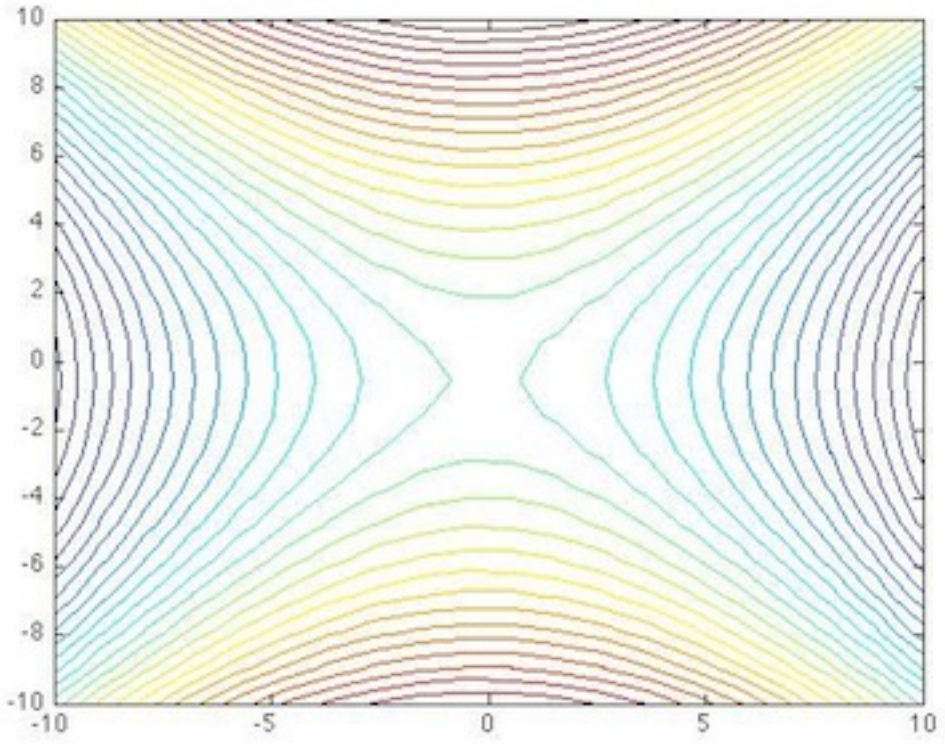
b) El PRC para δ_c cualquiera es:

$$min(S) : f(x_k) + \nabla f(x_k)^t S + \frac{1}{2} S^t \nabla^2 f(x_k) S$$

$$\text{sujeto a } ||S|| \leq \delta_c, \text{ donde } \delta_c > 0 \text{ dado}$$

c) Profesora: Intenté varias estrategias para graficar la familia de soluciones pero no lo logré. Sin embargo por teoría yo se que al aumentar el radio δ_c aumentará la región de confianza y la intersección entre la solución y las curvas de nivel ocurrirá en un punto más cercano a la solución en el modelo cuadrático (que no necesariamente es la solución de la función original).

d) Modelo cuadrático para $x_c = (0; 0{,}5)^t$, con $H_c = \nabla^2 f(x_c)$:



2. El siguiente es el código del dogleg implementado en MatLab:

```
function S = metododogleg(obj,xc,delta)
    %Definiciones
    [grad_x,grad_y]= obj.grad_f(xc);
    g = [grad_x;grad_y];
    h = obj.hessiano(xc);
    Sn = -1*(h\g);
    %Comienza el metodo dogleg
    if(obj.norma(Sn) < delta)
```

```
            S = Sn;
        else
            lambda = (obj.norma(g) *  obj.norma(g)) / ( g'*h*g);
            Scp = -lambda * g;
            if(obj.norma(Scp) > delta)
                S = (delta* Scp) / obj.norma(Scp);
            else
                %este polinomio esta calculado para el caso R^2
                p = [Sn(1)^2-2*Sn(1)*Scp(1)+Scp(1)^2+Sn(2)^2-2*Sn(2)*Scp(2)
                      +Scp(2)^2 2*Sn(1)*Scp(1)-2*Scp(1)^2+2*Sn(2)*Scp(2)
                      -2*Scp(2)^2 Scp(1)^2+Scp(2)^2];
                raices_p = roots(p);
                cantraices = length(raices_p);
                lambda_techo = 0.5;
                for i=1:cantraices
                    if raices_p(i)>0 && raices_p(i)<1
                        lambda_techo = raices_p(i);
                    end
                end
                S = lambda_techo * Sn + (1-lambda_techo) * Scp;
            end
        end
    end
```

Y el código de las funciones implementadas para esta tarea:

```
classdef f1RegionConfianza < RegionConfianza
    % Funcion particular
    %   Implementa la funcion 1 de la tarea: 3x1^2+ 2x1x2 + x2^2

    properties
    end
    methods
        function [d_x,d_y] = d(obj,arg)
        end
        function obj = f1RegionConfianza(arg)
              obj = obj@RegionConfianza(arg);
              obj.xmin = [0;0];
              obj.statfilename = 'f_1DogLeg';
        end
        function [ret]=f(obj,arg)
            ret = 10*(arg(2)-arg(1)*arg(1))^2+(1-arg(1))^2;
        end
        function [fd_x,fd_y]=grad_f(obj,arg)
            fd_x = -40*arg(1)*arg(2)+40*arg(1)^3-2+2*arg(1);
            fd_y = 20*arg(2)-20*arg(1)*arg(1);
        end
        function H = hessiano(obj,arg)
            H = [-40*arg(2)+120*arg(1)^2+2,-40*arg(1);-40*arg(1),20];
        end
```

```
    end
end

classdef f2RegionConfianza < RegionConfianza
    % Funcion particular
    %   Implementa la funcion 1 de la tarea: 3x1^2+ 2x1x2 + x2^2

    properties
    end
    methods
        function [d_x,d_y] = d(obj,arg)
        end
        function obj = f2RegionConfianza(arg)
              obj = obj@RegionConfianza(arg);
              obj.xmin = [0;0];
              obj.statfilename = 'f_1RegionConfianza';
        end
        function [ret]=f(obj,arg)
            ret = 100*(arg(2)-arg(1)*arg(1))^2+(1-arg(1))^2;
        end
        function [fd_x,fd_y]=grad_f(obj,arg)
            fd_x = -400*arg(1)*arg(2)+400*arg(1)^3-2+2*arg(1);
            fd_y = 200*arg(2)-200*arg(1)*arg(1);
        end
        function H = hessiano(obj,arg)
            H = [-400*arg(2)+1200*arg(1)^2+2,-400*arg(1);-400*arg(1),200];
        end
    end
end
```

Sea $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. Los siguientes fueron los valores de S_c obtenidos:

x_c	S_c
$(1{,}2; 1{,}2)^t$	$(-0{,}0041; 0{,}2302)^t$
$(-1{,}2; 1)^t$	$(0{,}0247; 0{,}3807)^t$

3. El siguiente es el código de Región de Confianza implementado en MatLab:

```
classdef RegionConfianza < Base
    properties
    end
    methods
        function [ret] = aprox_cuadratica(obj,x,s)
            f = obj.f(x);
            [g1,g2] = obj.grad_f(x);
            g = [g1;g2];
            h = obj.hessiano(x);
            ret = f + g'*s+0.5*(s'*h*s);
```

```
        end
        function obj = RegionConfianza(arg)
            obj = obj@Base(arg);
        end
        function [ret] = metodoregionconfianza(obj,x)
            delta_barra = 100;
            eta = 1/8;
            %Para establecer el delta_0 resuelvo el PRC y busco la norma
            s = obj.metododogleg(x,1)
            delta = obj.norma(s);
            parar = false;
            k=0;
            while ~parar
                [g1,g2] = obj.grad_f(x);
                normagrad = obj.norma([g1;g2]);
                if normagrad < 10^-5 %condicion de terminacion del algoritmo
                    parar = true;
                else
                    %(1) Obtener Sx como solucion de PRC (sol. aprox). Usa dogleg.
                    s = obj.metododogleg(x,delta);
                    %(2) Evaluo que tan buena es la aprox. cuadratica
                    ro = (obj.f(x) - obj.f(x+s))/
                     (obj.aprox_cuadratica(x,[0;0]) - obj.aprox_cuadratica(x,s));
                    %(3) Decision segun el valor de ro
                    if ro < 0.25
                        delta = 0.25 * obj.norma(s);
                    elseif (ro > 0.75) && (obj.norma(s) == delta)
                        delta = min(2*delta,delta_barra);
                    end
                    %(4) Confio en la region y actualizo el proximo iterado
                    if ro > eta
                        x = x + s;
                    end
                    k = k+1;
                end
                if(k>1000)
                    parar = true;
                end
            end
            k
            ret = x;
        end
    end
end
```

Resultados: Las siguientes tablas muestran los resultados del algoritmo:

Punto de partida = $(1{,}2; 1{,}2)^t$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$
0	(1.200000,1.200000)	0.282843	125.169325
1	(1.157697,1.331820)	0.367386	4.549666
2	(1.099039,1.204446)	0.227172	1.843894
3	(1.040372,1.078931)	0.088657	1.662306
4	(1.016460,1.032619)	0.036537	0.288978
5	(1.001689,1.003163)	0.003586	0.100740
6	(1.000071,1.000139)	0.000156	0.001299
7	(1.000000,1.000000)	0.000000	0.000002

Punto de partida = $(-1{,}2; 1)^t$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$
0	(-1.200000,1.000000)	2.200000	232.867688
1	(-1.115989,1.219582)	2.127352	16.596754
2	(-0.937910,0.851937)	1.943558	15.321802
3	(-0.782010,0.591286)	1.828280	10.696315
4	(-0.598277,0.328505)	1.733607	11.810667
5	(-0.366172,0.080209)	1.646949	15.130758
6	(-0.250144,0.049110)	1.570686	4.695863
7	(-0.250144,0.049110)	1.570686	4.695863
8	(-0.071162,-0.022061)	1.480539	6.158268
9	(0.020172,-0.005518)	1.403969	2.249279
10	(0.262895,0.017957)	1.227898	10.951387
11	(0.288038,0.080888)	1.162608	1.255314
12	(0.375970,0.111735)	1.085555	6.735680
13	(0.415853,0.170004)	1.014949	0.898350
14	(0.486502,0.230768)	0.924877	1.189648
15	(0.603849,0.350618)	0.760679	3.818454
16	(0.708015,0.490434)	0.587292	3.302197
17	(0.800120,0.631709)	0.419035	2.870447
18	(0.874241,0.758803)	0.272013	1.998784
19	(0.934161,0.869066)	0.146555	1.406976
20	(0.972482,0.944253)	0.062169	0.593909
21	(0.993753,0.987092)	0.014340	0.190247
22	(0.999482,0.998931)	0.001188	0.013752
23	(0.999997,0.999993)	0.000008	0.000113
24	(1.000000,1.000000)	0.000000	0.000000

4. Ahora resolvemos el problema utilizando el método de Cauchy con BLI:
   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
6	(1.111177,1.235115)	0.260076	0.091519
9	(1.072981,1.152146)	0.168744	0.280383
10	(1.073203,1.151975)	0.168686	0.070330
...	...	...	...
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

Para el punto de partida el $(-1{,}2; 1)^t$ el método de Cauchy no converge
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
...	...	...	
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
...	...	...	

Ahora resolvemos el problema utilizando el método de Newton con BLI:

Parametros: $\eta = 0{,}000100; \rho = 0{,}500000; x_0 = (1{,}200000; 1{,}200000); x_* = (1{,}000000, 1{,}000000)$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$	$recortes$
0	(1.200000,1.200000)	0.282843	125.169325	0
1	(1.195918,1.430204)	0.472715	0.399820	0
2	(1.098284,1.196688)	0.219877	4.784866	1
3	(1.064488,1.131993)	0.146904	0.656352	0
4	(1.011992,1.021372)	0.024507	1.265832	0
5	(1.004261,1.008481)	0.009491	0.034658	0
6	(1.000050,1.000083)	0.000097	0.008020	0
7	(1.000000,1.000000)	0.000000	0.000001	0

Parametros: $\eta = 0{,}000100; \rho = 0{,}500000; x_0 = (-1{,}200000; 1{,}000000); x_* = (1{,}000000, 1{,}000000)$

k	x_k	$\|\|x_k - x_*\|\|$	$\|\|grad_f(x_k)\|\|$	$recortes$
0	(-1.200000,1.000000)	2.200000	232.867688	0
1	(-1.175281,1.380674)	2.208339	4.639426	0
2	(-0.206083,-0.897180)	2.248094	204.198411	1
3	(-0.199699,0.039839)	1.536616	2.402668	0
4	(0.395301,-0.197783)	1.341769	89.520633	1
5	(0.403722,0.162920)	1.027740	1.181190	0
6	(0.697692,0.400320)	0.671570	29.193895	1
7	(0.714220,0.509836)	0.567389	0.496534	0
8	(0.849707,0.703509)	0.332407	7.035722	1
9	(0.881693,0.776360)	0.253004	0.239369	0
10	(0.979904,0.950567)	0.053362	4.208566	0
11	(0.986765,0.973658)	0.029480	0.012284	0
12	(0.999877,0.999581)	0.000437	0.076654	0
13	(0.999996,0.999992)	0.000009	0.000004	0

Nota: El código utilizado es el mismo entregado en tareas anteriores.

Comparación: Los siguientes puntos pueden resaltarse al comparar el método de región de confianza, Cauchy y Newton:

a) Todos los algoritmos presentan distintos comportamientos en función del punto de partida

b) Región de Confianza y Newton siempre convergen

c) Dependiendo del punto de partida, Región de Confianza y Newton tardan sólo 7 iteraciones o considerablemente más iteraciones

d) Cauchy es relativamente mucho más lento que los demás métodos

e) El método de Cauchy, desde el segundo punto de partida, es el único algoritmo que no converge

f) Los resultados del algoritmo de Región de Confianza son los más precisos y para este caso convergen a la solución exacta