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

Tarea 6
Sartenejas, 21 de Marzo de 2011

1. Sea $f(x) = \frac{1}{2}x^tAx$, donde $A = diag(1, 2, ...n)$. Entonces,

$$\nabla f(x) = Ax = (x_1, 2 * x_2, 3 * x_3, ..., n * x_n)^t$$

$$\nabla^2 f(x) = A = diag(1, 2, 3, ..., n)$$

Evidentemente el minimizador es $(0, 0, ..., 0)^t$ ya que la función es siempre positiva. Además, como es cuadrática estrictamente convexa, el mínimo es único.

La siguiente tabla muestra el número de iteraciones que tomaron los algoritmos estudiados:

n	Cauchy	Gradiente Espectral	Gradiente Conjugado
100	769	114	54
200	1536	164	78
500	3837	296	124

Los siguientes códigos fueron empleados para generar los resultado anteriores. En todos los casos se utilizó como criterio de parada $||x_k - x^*|| \leq 10^{-14}$ y como punto inicial $x_0 = (0{,}5; 0{,}5; ...; 0{,}5)$. Para el método del gradiente espectral se utilizó $\alpha_0 = 1{,}5$

Algoritmos:

```
classdef GradienteConjugado
    properties
        n
        xmin
        statfilename
    end
    methods(Abstract)
        [ret] = hessiano(x)
        [ret] = f(obj,arg)
        [ret] = grad(obj,arg)
    end
    methods
        function obj = GradienteConjugado(arg)
        end
        function [ret] = norma(obj,x)
```

```
            ret = 0;
            for i=1:obj.n
                ret = ret + x(i)*x(i);
            end
        end
        function [ret]=metodogradienteconjugado(obj,xk)
            parar = false;
            k=0;
            beta = 0;
            d = 0;

            fid_tex = fopen(strcat('stats',obj.statfilename,'-',int2str(obj.n),'-','.tex'),'w');
            fprintf(fid_tex,'\\begin{tabular}{|c|c|c|c|c|}\\hline\n\t$k$&$||x_k-x_*||$\\\\ \\hlin

            while ~parar
                %Calcular el grad para usarlo como condicion de parada
                normagrad = obj.norma(xk - obj.xmin);
                fprintf(fid_tex,'\n\t%d&%f\\\\ \\hline',k,normagrad);
                %Se encontro la solucion
                if normagrad < 10^-14
                    parar = true;
                else
                %Aun no se encontro, realizar otra iteracion
                    g = obj.grad(xk);
                    d = (-1 * g) + beta*d;
                    alpha = ((-1 * g)'*d)/(d'*obj.hessiano*d);
                    xk = xk+alpha*d;
                    g = obj.grad(xk);
                    beta = (g'*obj.hessiano*d) / (d'*obj.hessiano*d);
                    k = k+1;
                end
                if(k>250)
                    parar = true;
                end
            end
            ret = xk;
            fprintf(fid_tex,'\n\\end{tabular}\\\\\\\\');
            fclose(fid_tex);
        end
    end
end

classdef GradienteEspectral
    properties
        n
        xmin
        statfilename
    end
    methods(Abstract)
        [ret] = hessiano(x)
```

```
        [ret] = f(obj,arg)
        [ret] = grad(obj,arg)
    end
    methods
        function obj = GradienteEspectral(arg)
        end
        function [ret] = norma(obj,x)
            ret = 0;
            for i=1:obj.n
                ret = ret + x(i)*x(i);
            end
        end
        function [ret]=metodogradienteespectral(obj,xk)
            parar = false;
            k=0;
            alpha = 1.5;

            fid_tex = fopen(strcat('stats',obj.statfilename,'-',int2str(obj.n),'-','.tex'),'w');
            fprintf(fid_tex,'\\begin{tabular}{|c|c|c|c|c|}\\hline\n\t$k$&$||x_k-x_*||$\\\\ \\hlin

            while ~parar
                %Calcular el grad para usarlo como condicion de parada
                normagrad = obj.norma(xk - obj.xmin);
                fprintf(fid_tex,'\n\t%d&%f\\\\ \\hline',k,normagrad);
                %Se encontro la solucion
                if normagrad < 10^-14
                    parar = true;
                else
                %Aun no se encontro, realizar otra iteracion
                    g = obj.grad(xk);
                    d = (-1 * g) / alpha;
                    xk = xk+d;
                    g_s = obj.grad(xk);
                    y = g_s - g;
                    alpha = (d'*y)/(d'*d);
                    k = k+1;
                end
                if(k>500)
                    parar = true;
                end
            end
            ret = xk;
            fprintf(fid_tex,'\n\\end{tabular}\\\\\\\\');
            fclose(fid_tex);
        end
    end
end

classdef GradienteBLE
    properties
```

```
        n
        xmin
        statfilename
    end
    methods(Abstract)
        [ret] = hessiano(x)
        [ret] = f(obj,arg)
        [ret] = grad(obj,arg)
    end
    methods
        function obj = GradienteBLE(arg)
        end
        function [ret] = norma(obj,x)
            ret = 0;
            for i=1:obj.n
                ret = ret + x(i)*x(i);
            end
        end
        function [ret]=metodogradienteble(obj,xk)
            parar = false;
            k = 0;
            fid_tex = fopen(strcat('stats',obj.statfilename,'-',int2str(obj.n),'-','.tex'),'w');
            fprintf(fid_tex,'\\begin{tabular}{|c|c|c|c|c|}\\hline\n\t$k$&$||x_k-x_*||$\\\\ \\hlin

            while ~parar
                %Calcular el grad para usarlo como condicion de parada
                normagrad = obj.norma(xk - obj.xmin);
                fprintf(fid_tex,'\n\t%d&%f\\\\ \\hline',k,normagrad);
                %Se encontro la solucion
                if normagrad < 10^-14
                    parar = true;
                else
                    g = obj.grad(xk);
                    d = -1*g;
                    %busqueda lineal exacta, funcion cuadratica
                    alpha = (g'*g)/(g'*obj.hessiano*g);
                    xk = xk + alpha * d;
                    k = k+1;
                end
                if(k>5000)
                    parar = true;
                end
            end
            ret = xk;
            fprintf(fid_tex,'\n\\end{tabular}\\\\\\\\');
            fclose(fid_tex);
        end
    end
end
```

Implementación de la función: (la misma para todas las pruebas)

```
classdef f1GradienteConjugado < GradienteConjugado
    % Funcion particular
    %   Implementa la funcion 1 de la tarea 6: 0:5x^tAx,
    %   donde A = diag(1,2,...,n).

    properties
    end
    methods
        function obj = f1GradienteConjugado(arg)
              obj = obj@GradienteConjugado(arg);
              obj.statfilename = 'f_1GradienteConjugado';
              obj.n = arg;
              obj.xmin = zeros(arg,1);
        end
        function [ret]=f(obj,arg)
            ret = 0;
            for i=1:obj.n
                ret = ret +arg(i)*arg(i)*i;
            end
            ret = 0.5*ret;
        end
        function [g] = grad(obj,arg)
            g = zeros(obj.n,1);
            for i=1:obj.n
               g(i) = i*arg(i);
            end
        end
        function H = hessiano(obj)
            H = zeros(obj.n,obj.n);
            for i=1:obj.n
                for j=1:obj.n
                    if i == j
                        H(i,j) = i;
                    else
                        H(i,j) = 0;
                    end
                end
            end
        end
    end
end
```

Para iniciar las corridas:

```
f1 = f1GradienteConjugado(100);
xk = zeros(f1.n,1);
for i=1:f1.n
    xk(i) = 0.5;
```

```
end
[xf1] = f1.metodogradienteconjugado(xk);
```

2. Profesora, el esquema de la siguiente prueba lo vi en el libro de Nocedal. Aquí lo presento con algunos detalles que faltaban en la demostración original:

   Consideremos el método del gradiente conjugado aplicado a una función cuadrática, $f(x) = \frac{1}{2}x^tQx - bx$, estrictamente convexa. Sea Q el hessiano de la función cuadrática a minimizar y $g_i = \nabla f(x_i) = Qx_i - b$ y d_i la dirección del método.

   El método del gradiente conjugado es el que sigue: $x_{k+1} = x_k + \alpha_k d_k$ con $\alpha_x = \frac{-g_k^t d_k}{d_k^t Q d_k}$ y $d_{k+1} = -g_{k+1}^t + \beta_k d_k$. De esto podemos deducir que:

$$g_{k+1} = Qx_{k+1} - b \Rightarrow g_{k+1} + b = Qx_{k+1} \Rightarrow Q^{-1}(g_{k+1} + b) = x_{k+1}$$

$$g_k = Qx_k - b \Rightarrow g_k + b = Qx_k \Rightarrow Q^{-1}(g_k + b) = x_k$$

$$\Rightarrow Q^{-1}(g_{k+1} + b) = Q^{-1}(q_k + b) + \alpha_k d_k \Rightarrow g_{k+1} + b = g_k + b + \alpha_k Q d_k$$

$$\Rightarrow g_{k+1} = g_k + \alpha_k Q d_k (1)$$

   Se demostrará que:

$$gen\{g_0, g_1, ..., g_k\} = gen\{g_0, Qg_o, ..., Q^k g_0\} = gen\{d_0, d_1, ..., d_k\} (2)$$

   A través de un argumento inductivo. Para el caso base $k = 0$ tenemos que:

$$gen\{g_0\} = gen\{d_0\}$$

   Lo cual es cierto ya que por definición $d_0 = -g_0$. Tomaremos entonces la hipótesis inductiva (2) como cierta en k y demostraremos que se cumple para $k+1$. La estrategia para demostrar la igualdad será demostrar que el conjunto del lado derecho está contenido en el del izquierdo y viceversa.

   Como sabemos que (2) se cumple para k, entonces:

$$g_k \in gen\{g_0, Qg_o, ..., Q^k g_0\}$$

$$d_k \in gen\{g_0, Qg_o, ..., Q^k g_0\}$$

   Multiplicando la segunda expresión por el hessiano Q, obtenemos que:

$$Qd_k \in gen\{Qg_0, Q^2 g_o, ..., Q^{k+1} g_0\}$$

   Pero por lo demostrado al principio $(1) = g_{k+1} = g_k + \alpha_k Q d_k$ y las proposiciones anteriores se deduce que:

$$g_{k+1} \in gen\{g_0, Qg_o, ..., Q^k g_0\} \cup gen\{Qg_0, Q^2 g_o, ..., Q^{k+1} g_0\} \Rightarrow g_{k+1} \in gen\{g_0, Qg_o, ..., Q^k g_0, Q^{k+1} g_0\}$$

   Por lo tanto:

$$gen\{g_0, g_1, ..., g_k, g_{k+1}\} \subset gen\{g_0, Qg_o, ..., Q^k g_0, Q^{k+1} g_0\}$$

Ahora se demostrará la segunda inclusión. Para ello multiplicamos una de las hipotesis inductivas por Q:

$$Q^{k+1}q_0 = Q(Q^k q_0) \in gen\{Qd_0, Qd_1, ..., Qd_k\}$$

Por otra parte, sabemos que por la regla de actualización se cumple que:

$$Qd_i = (r_{i+1} - r_i)/\alpha_i, i = 0, 1, ..., k \Rightarrow Q^{k+1}q_o \in gen\{q_0, q_1, ..., q_{k+1}\}$$

Finalmente,

$$gen\{q_0, Qq_0, ..., Q^{k+1}q_0\} \subset gen\{q_0, q_1, ..., q_k, q_{k+1}\}$$

De lo que se concluye que se cumple la tesis inductiva para $k+1$

3. Consideremos las extensiones del método de gradiente conjugado para el caso no cuadrático:

$$x_{k+1} = x_k + \alpha_k d_k$$

$$d_k = -\nabla f(x_k) + \beta^i_{k-1} d_{k-1}$$

Donde $\alpha_k = argmin f(x_k + \alpha x_k)$, encontrado por búsqueda lineal exacta. Que d_k es una dirección de descenso para i =Fletcher-Reeves, Polak-Riviere se demostrará en ambos casos como sigue.

$$\nabla f(x_k)^t d_k = \nabla f(x_k)^t(-\nabla f(x_k) + \beta_k d_{k-1}) = -\nabla f(x_k)^t \nabla f(x_k) + \beta_k \nabla f(x_k)^t d_{k-1} =$$

$$-||f(x_k)||^2 + \beta_k \nabla f(x_k)^t d_{k-1} = -||f(x_k)||^2 + \beta_k \nabla f(x_{k-1} + \alpha_{k-1} d_{k-1})^t d_{k-1}$$

Analizamo la expresión anterior término a término:

a) El primer termino de la expresión anterior es estrictamente negativo ya que la norma es estrictamente positiva.

b) El segundo termino es cero, ya que α_{k-1} es el resultado de búsqueda lineal exacta y por lo tanto minimiza a $f(x_{k-1} + \alpha_{k-1} d_{k-1})$ en la dirección d_{k-1}, de lo que por condiciones de optimalidad de primer orden se deduce que $\nabla f(x_k) = 0 \Rightarrow \nabla f(x_k)^t d_{k-1} = 0 \Rightarrow \beta_k f(x_k)^t d_{k-1} = 0$

Por lo tanto se cumple que $\nabla f(x_k)^t d_k < 0$ y la dirección es de descenso, cuando la búsqueda es exacta, sin importar la extensión que se use.