Programación: el método de direcciones conjugadas

Objetivos. Programar una función que resuelva sistemas de ecuaciones lineales de la forma $Ax = b$, donde $A^\top = A$ y $A > 0$, usando el método iterativo con direcciones A-conjugadas y con búsqueda exacta sobre cada recta.

Requisitos. Solución de sistemas simétricas positivas definidas por medio de la minimización de un funcional, búsqueda exacta sobre cada recta, ortogonalización de una lista de vectores con respecto al producto $\langle \cdot, \cdot \rangle_A$.

1. Vectores A-conjugados (repaso). Sea $A \in \mathcal{M}_m(\mathbb{R})$ y sean $x, y \in \mathbb{R}^m$. Se dice que x, y son A-conjugados si

$$x^\top A y = 0.$$

2. Ortogonalización con respecto al producto interno asociado a una matriz simétrica positiva definida (repaso). En las clases pasadas hemos programado una función `QRMGSA` con la siguiente entrada y salida:

> Entrada: un par de matrices V y A, tales que $V \in \mathcal{M}_{m\times n}(\mathbb{R})$, $m \geq n$, $\mathrm{r}(V) = n$, $A \in \mathcal{M}_m(\mathbb{R})$, $A^\top = A$, $A > 0$.
>
> Salida: un par de matrices Q y R, tales que $V = QR$, $Q \in \mathcal{M}_{m\times n}(\mathbb{R})$, $Q^\top A Q = I_n$, $R \in \mathcal{M}_n(\mathbb{R})$, $R_{j,k} = 0$ para cada $j > k$.

En otras palabras, la matriz dada V debe tener columnas linealmente independientes, y la matriz dada A debe ser simétrica y positiva definida. Se construyen matrices Q y R tales que $V = QR$, R es triangular superior, y las columnas de Q son ortonormales con respecto al producto interno asociado a la matriz A.

3. El mínimo de una forma cuadrática sobre una recta (repaso). Sea A una matriz real simétrica estrictamente positiva definida de orden n, y sea $b \in \mathbb{R}^n$. Dado un vector $x \in \mathbb{R}^n$ y una *dirección* $p \in \mathbb{R}^n \setminus \{0\}$, consideremos la función

$$g(\alpha) = \frac{1}{2}(x + \alpha p)^\top A(x + \alpha p) - b^\top (x + \alpha p).$$

Escriba g como un polinomio de grado 2 en la variable α, calcule g' y encuentre el punto $\alpha_{\min}$ donde g alcanza su valor mínimo. Se recomienda usar la notación $r = b - Ax$.

4. El cambio del residuo al cambiar el vector (repaso). Sean A, b, x, p, r los mismos que en el ejercicio anterior. Definimos el vector nuevo y como

$$y = x + \alpha p.$$

Exprese el residuo nuevo $b - Ay$ a través del residuo previo:

$$b - Ay = \underbrace{b - Ax}_{r} + ???.$$

5. Idea del método de direcciones conjugadas. Aplicar el método iterativo con la búsqueda exacta sobre cada recta, tomando como direcciones algunos vectores A-conjugados.

6. Algoritmo. Complete el programa usando las fórmulas de los Ejercicios 3 y 4.

```
function [x, s] = conjdir(A, b, tol, smax),
   # construct a matrix of conjugate directions
   n = length(b);
   V = eye(n);
   [Q, R] = QRMGSA(V, A);
   # now apply the iterative method
   x = zeros(n, 1);
   r = b;
   s = 0;
   while (s < smax) && (norm(r) >= tol),
      s = s + 1;
      p = Q(:, s); # use columns of Q as directions
      q = A * p;
      alpha = ???;
      x = ???;
      r = ???;
   end
end
```

7. Pruebas pequeñas. Haga pruebas de la función anterior con datos de tamaño $n = 3$ o $n = 4$. Para generar una matriz real simétrica positiva definida A, se puede utilizar el siguiente código:

```
B = rand(n, n);
A = B' * B + 0.01 * eye(n);
```

Analice cómo se cambia el número de pasos al cambiar n.

8. Pruebas con matrices grandes. Elija varios valores de n de tal manera que el tiempo de ejecución sea de 0.1 segundos a 100 segundos. Analice cómo se cambia el tiempo de ejecución al cambiar n.

9. Prueba en el plano con un dibujo. El siguiente código consiste de dos funciones y se debe guardar en un archivo `plotconjdir.m`

```
function [] = plotconjdir(),
   A = [3 -2; -2 2];
   sol = [1; -6]; # exact solution
   b = A * sol;
   n = length(b);
   xsequence = zeros(n, 6);
   for s = 1 : 6,
      xsequence(:, s) = conjdir(A, b, 1.0E-8, s);
   end
   disp(xsequence);
   r = max(abs(xsequence(:, 1) - sol));
   plotdomain = [sol(1) - r, sol(1) + r, sol(2) - r, sol(2) + r];
   plot(xsequence(1, :), xsequence(2, :), '-w', 'linewidth', 3);
   hold on;
   ezcontourf(@(var1, var2) f(var1, var2, A, b, sol), plotdomain, 100);
end

function result = f(var1, var2, A, b, sol),
   [size1, size2] = size(var1);
   result = zeros(size1, size2);
   for j = 1 : size1,
      for k = 1 : size2,
         x = [var1(j, k); var2(j, k)];
         result(j, k) = sqrt(0.5 * (x - sol)' * A * (x - sol));
      end
   end
end
```