Programación: el método iterativo con direcciones aleatorias y con la búsqueda exacta sobre cada recta

Objetivos. Programar el método iterativo con direcciones aleatorias y con la búsqueda exacta sobre cada recta.

Requisitos. Solución de sistemas simétricas positivas definidas por medio de la minimización de un funcional.

1. El mínimo de una forma cuadrática sobre una recta. 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 $r = b - Ax$.

2. El cambio del residuo al cambiar el vector. 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:

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

3. Algoritmo.

```
function [x, s] = randomiter(A, b, tol, smax),
   n = length(b);
   x = zeros(n, 1);
   r = b;
   s = 0;
   while (s < smax) && (norm(r) >= tol),
      p = 2 * rand(n, 1) - ones(n, 1);
      q = A * p;
      alpha = (p' * r) / (p' * q);
      x = x + alpha * p;
      r = r - alpha * q;
      s = s + 1;
   end
end
```

4. Una modificación del algoritmo que guarda los pasos.

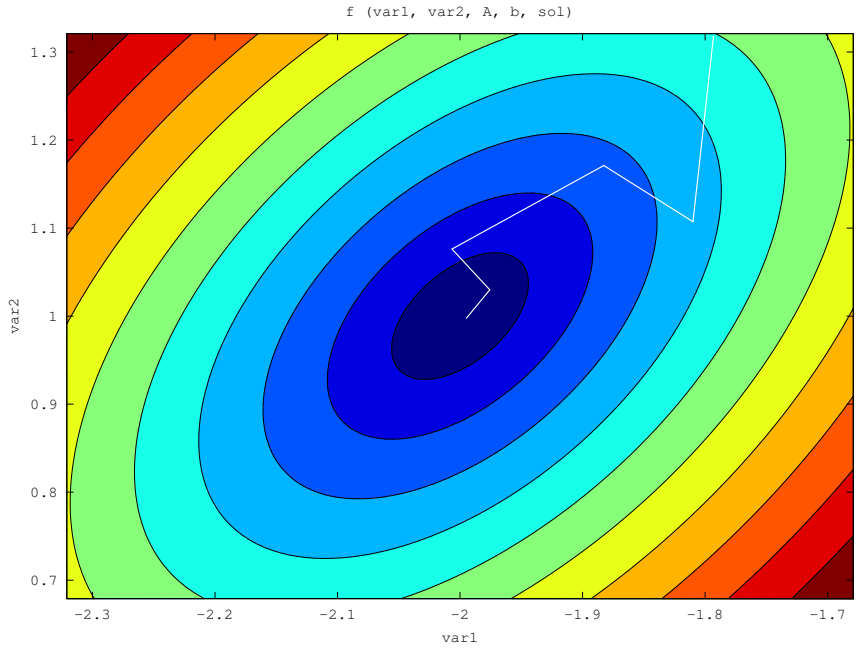
```
function [xsequence] = randomitersequence(A, b, tol, smax),
   n = length(b);
   xsequence = zeros(n, smax);
   x = zeros(n, 1); r = b; s = 0;
   while (s < smax) && (norm(r) >= tol),
      p = 2 * rand(n, 1) - ones(n, 1);
      q = A * p;
      alpha = (p' * r) / (p' * q);
      x = x + alpha * p;
      r = r - alpha * q;
      s = s + 1;
      xsequence(:, s) = x;
   end
end
```

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

```
function [] = plotrandomiter(),
   A = [3 -2; -2 2];
   sol = [1; -6]; # exact solution
   b = A * sol;
   n = length(b);
   xsequence = randomitersequence(A, b, 1.0E-8, 6);
   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
```

El dibujo correspondiente:



6. Pruebas con matrices y vectores de tamaños grandes. Escriba una función que haga pruebas de la función `randomiter` con matrices y vectores aleatorios de tamaños grandes. Observe cómo se cambia el tiempo de ejecución y el número de pasos al aumentar n. Para generar matrices aleatorias simétricas y positivas definidas se pueden usar los siguientes comandos:

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