Programación: ortogonalización con respecto al producto interno no canónico usando el método de Gram–Schmidt modificado

Objetivos. Aplicar el método de Gram–Schmidt modificado a un ejemplo, usando el producto interno inducido por una matriz real simétrica positiva definida.

Requisitos. Operaciones con vectores (operaciones lineales y el producto punto), ciclos, la idea del método de Gram–Schmidt modificado.

Producto interno no canónico

Los siguientes comandos están escritos en el lenguaje de MATLAB y se pueden ejecutar en MATLAB o en sus análogos libres (GNU Octave, Scilab, FreeMat).

1. Producto interno asociado a una matriz positiva definida. Sea A una matriz real simétrica positiva definida. Denotemos por $\langle \cdot, \cdot \rangle_A$ al producto interno asociado a esta matriz:

$$\langle x, y \rangle_A \coloneqq x^\top A y.$$

Ejecute los siguientes comandos en el intérprete:

```
A = [1 -1 -1; -1 4 1; -1 1 3];
A' - A          # the matrix is autoadjoint
eig(A)          # the eigenvalues are strictly positive
x = [3; -1; 4]
y = [1; 5; -3]
x' * A * y
y' * A * x
```

2. Generación de matrices aleatorias positivas definidas. Recordemos que una matriz $A \in \mathcal{M}_n(\mathbb{R})$ es simétrica y positiva semidefinida ($A^\top = A$ y $x^\top A x \geq 0$ para cada $x \in \mathbb{R}^n$) si, y sólo si, existe una matriz $B \in \mathcal{M}_n(\mathbb{R})$ tal que $A = B^\top B$. Además, si una matriz A es simétrica positiva semidefinida y $\varepsilon > 0$, entonces la matriz $A + \varepsilon I_n$ es positiva definida.

```
B = rand(3, 3);
A = B' * B + 0.1 * eye(3);
A' - A
eig(A)
```

Proyección ortogonal

3. Vectores *A*-conjugados. Sea $A \in \mathcal{M}_n(\mathbb{R})$ una matriz real simétrica positiva definida. Dos vectores $x, y \in \mathbb{R}^n$ se llaman A-*ortogonales* o A-conjugados si

$$\langle x, y \rangle_A = 0,$$

esto es, si

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

4. Proyección ortogonal de un vector al otro con respecto al producto interno no canónico. Sea A una matriz real simétrica positiva definida. Dados dos vectores a y v en $\mathbb{R}^3$, calculemos $\lambda \in \mathbb{R}$ tal que $a \perp_A (v - \lambda a)$, esto es,

$$a^\top A(v - \lambda a) = 0.$$

Denotemos λa por u y $v - \lambda a$ por w. Verifiquemos que se cumple la identidad de Pitágoras para los vectores u, w, v (con respecto a la norma $\| \cdot \|_A$).

```
A = [1 -1 -1; -1 4 1; -1 1 3];
a = [-3; 2; 1]
v = [-8; 1; 4]
lambda = (a' * A * v) / (a' * A * a)
u = lambda * a
w = v - u
a' * A * w
[(u' * A * u) + (w' * A * w), (v' * A * v)]
```

Proceso de ortogonalización de Gram–Schmidt modificado para un producto interno no canónico

5. Ortogonalización de Gram–Schmidt. Dada una matriz $A \in \mathcal{M}_3(\mathbb{R})$ real simétrica positiva definida y tres vectores $a_1, a_2, a_3 \in \mathbb{R}^3$, construimos tres vectores A-ortonormales $b_1, b_2, b_3 \in \mathbb{R}^4$ aplicando el algoritmo de Gram–Schmidt modificado a los vectores dados a_1, a_2, a_3. Al final comprobamos que los vectores b_1, b_2, b_3 son A-ortogonales a pares calculando su matriz de Gram con respecto al producto $\langle \cdot, \cdot \rangle_A$. El siguiente código se puede guardar en un archivo `MGSAexample.m`.

```
function [] = MGSAexample(),
   a1 = [-2; 5; 1]
   a2 = [-1; 6; 3]
   a3 = [-11; 20; 9]

   b1 = a1;
   b2 = a2;
   b3 = a3;

   # Step 1
   nu1 = sqrt(b1' * A * b1)
   b1 = b1 / nu1
   lambda21 = b1' * A * b2
   b2 = b2 - lambda21 * b1
   lambda31 = b1' * A * b3
   b3 = b3 - lambda31 * b1

   # Step 2
   nu2 = sqrt(b2' * A * b2)
   b2 = b2 / nu2
   lambda32 = b2' * A * b3
   b3 = b3 - lambda32 * b2

   # Step 3
   nu3 = sqrt(b3' * A * b3)
   b3 = b3 / nu3

   # Verify that b1, b2, b3 form an A-orthonormal list
   b1b2b3 = [b1 b2 b3]
   G = b1b2b3' * A * b1b2b3
end
```

Descomposición QR generalizada

6. Descomposición QR generalizada de una matriz de tres columnas usando el método de Gram–Schmidt modificado. El siguiente algoritmo construya una descomposición de la matriz dada V de tres columnas en un producto QR tal que R es triangular superior y las columans de Q son A-conjugadas.

```
function [Q, R] = QRMGSA3(V, A),
   [n, m] = size(V);
   Q = V; R = eye(m);
   # primer paso
   R(1, 1) = sqrt(Q(:, 1)' * A * Q(:, 1));
   Q(:, 1) = Q(:, 1) / R(1, 1);
   R(1, 2) = Q(:, 1)' * A * Q(:, 2);
   Q(:, 2) = Q(:, 2) - R(1, 2) * Q(:, 1);
   R(1, 3) = Q(:, 1)' * A * Q(:, 3);
   Q(:, 3) = Q(:, 3) - R(1, 3) * Q(:, 1);
   # segundo paso
   R(2, 2) = sqrt(Q(:, 2)' * A * Q(:, 2));
   Q(:, 2) = Q(:, 2) / R(2, 2);
   R(2, 3) = Q(:, 2)' * A * Q(:, 3);
   Q(:, 3) = Q(:, 3) - R(2, 3) * Q(:, 2);
   # tercer paso
   R(3, 3) = sqrt(Q(:, 3)' * A * Q(:, 3));
   Q(:, 3) = Q(:, 3) / R(3, 3);
end
```

Prueba:

```
function [] = testQRA3(),
   n = 5; m = 3;
   B = rand(n, n);
   A = B' * B;     # A es real simetrica positiva definida
   V = rand(n, m);
   [Q, R] = QRMGSA3(V, A);
   display(Q);
   display(R);
   display(norm(Q' * A * Q - eye(m)));
   display(norm(Q * R - V));
end
```

7. Problema principal. Generalice el algoritmo anterior al caso de una matriz V rectangular de tamaño general $n \times m$, $n \geq m$, con columnas linealmente independientes. Escriba la función correspondiente QRMGSA. Haga pruebas con datos pequeños y con datos grandes.