Programación: comparar la rapidez y precisión de varios métodos que realizan la descomposición QR

Objetivos. Medir el tiempo de ejecución y la magnitud de errores de varias funciones programadas anteriormente, cada una de las cuales realiza una descomposición QR.

Requisitos. Se supone están realizadas al menos dos de los siguientes cuatro métodos:

- `qrcgs`: con el método clásico de Gram–Schmidt;
- `qrmgs`: con el método modificado de Gram–Schmidt;
- `qrh`: con reflexiones de Householder;
- `qrg`: con rotaciones de Givens.

Además se necesita experiencia de generar datos pseudoaleatorios, medir el tiempo de ejecución de programas y la magnitud de errores.

1. Ajustar el nivel de los métodos. Si se utiliza un lenguaje de programación orientado a matrices, donde hay operaciones internas para realizar operaciones con matrices y vectores, entonces cada uno de los algoritmos QR se puede realizar a cada uno de los siguientes niveles:

> Nivel 0: la función tiene tres ciclos encajados y utiliza solamente operaciones con entradas de matrices.
>
> Nivel 1: la función tiene dos ciclos encajados y utiliza operaciones internas para calcular combinaciones lineales de vectores y productos puntos (axpy y dotproduct).
>
> Nivel 2: la función tiene un ciclo y utiliza operaciones internas para calcular productos de matrices por vectores o productos diádicos de vectores.
>
> Nivel 3: se usa una función interna que realiza la factorización qr (por ejemplo, la función `qr` en MATLAB).

El tiempo de ejecución dependerá mucho del nivel de realización del algoritmo, y se recomienda comparar realizaciones del mismo nivel, por ejemplo, `qrmgs2` y `qrh2`.

2. Prueba con una matriz y un método.

```
function [] = onetestqr(A, f, ftitle),
   t1 = cputime();
   [Q, R] = f(A);
   t2 = cputime();
   n = rows(A);
   e1 = norm(Q * R - A);
   e2 = norm(Q' * Q - eye(n));
   disp(ftitle);
   disp(n);
   display([t2 - t1, max(e1, e2)]);
end
```

Ejemplo de ejecución:

```
A = rand(100, 100);
onetest(A, @qrmgs2, 'QR with Modified Gram-Schmidt:');
```

3. Organizar muchas pruebas.

```
function [] = compareqr(),
   for n = [128, 256, 512],
      A = rand(n, n) + ones(n, n);
      onetestqr(A, @qrmgs2, 'QR with Modified Gram-Schmidt, level 2:');
      onetestqr(A, @qrh2, 'QR with Householder reflexions, level 2:');
   end
end
```

4. Llenar la tabla. En papel o en un documento electrónico construir y llenar una tabla similar a la siguiente. En cada celda se indica el tiempo de ejecución y el error.

	método `qrcgs2`	método `qrmgs2`	método `qrh2`	método `qrg2`
$n = 128$	$t =???, e =???$			
$n = 256$				
$n = 512$				

Por supuesto, las columnas deben corresponder a los métodos realizados. Se recomienda elegir los valores de n de tal manera que el tiempo de ejecución sea de 0.1 segundos a 200 segundos.

5. Conclusiones. Analizar los resultados obtenidos, intentar de explicarlos.