Álgebra Linear - Aula Prática 5

Iara Cristina Mescua Castro

5 de junho de 2022

MÉTODO DE GRAM-SCHMIDT

Escreva uma função Scilab function [Q,R] = qr_GS(A) que implementa o Método de Gram-Schmidt para determinar a decomposição QR de uma matriz A com colunas linearmente independentes. Testar a sua função com algumas matrizes de ordens diferentes. Para cada uma delas, testar a precisão do método (por exemplo, teste a ortogonalidade da matriz Q obtida calculando QTQ).

A decomposição QR pelo método de Gram-Schmidt funciona da seguinte forma:

$$A_{mxn} = Q_{mxn}R_{nxm}$$

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1m} \\ 0 & r_{22} & r_{23} & \cdots & r_{2m} \\ 0 & 0 & r_{33} & \cdots & r_{3m} \\ 0 & 0 & 0 & \cdots & r_{mm} \end{bmatrix}$$

Sabendo que r_{ij}:
0, se $i > j$
$||v_j||$, se $i = j$
$q_i^T a_j$, se $i < j$

Então:
$a_1 = r_{11}q_1$
$a_2 = r_{12}q_1 + r_{22}q_2$
$a_n = r_{1m}q_1 + r_{2m}q_2 + r_{3m}q_3 + \cdots + r_{mm}q_n$

É o mesmo que:
$a_1 = ||v_1||q_1$
$a_2 = (q_1^T a_2)q_1 + ||v_2||q_2$
$a_3 = (q_1^T a_3)q_1 + (q_2^T a_3)q_2 + ||v_3||q_3$

Código:

```
function [Q,R] = qr_GS(A)
[m,n] = size(A); // le dim de A
// inicializa as matrizes Q e R
Q = zeros(m,n);
R = zeros(n,m);
for j = 1:n
  v = A(:,j);
  for i = 1:(j-1)
    R(i,j) = Q(:,i)' * A(:,j); // r_ij caso i < j
    v = v - R(i,j)*Q(:,i);
  end
  R(j,j) = norm(v, 2); // r_ij, quando i = j
  Q(:,j) = v/(R(j,j));
end
endfunction

```

Para os testes, criei uma função que gera uma matriz aleatória quadrada de ordem n, e uma que gera uma matriz aleatória (m,n):

```
function [A] = matriz_aleat(n) //quadrada
//para gerar um numero aleatorio entre [a,b]
//   r = a + (b-a)*rand();

//matriz A(m,n)) com numeros aleatorios entre [1 e n^3]
A = floor(1 + ((n^3 - 1)*rand(n,n,'uniform')))
endfunction

function [A] = matriz_aleat2(n,m)
//para gerar um numero aleatorio entre [a,b]
//   r = a + (b-a)*rand();

//matriz A(m,n)) com numeros aleatorios entre [1 e n^3]
A = floor(1 + ((n^3 - 1)*rand(n,m,'uniform')))
endfunction
```

Testando com o **Método de Gram-Schmidt**:

ORDEM 5

Exemplo 1:

```
--> A = matriz_aleat(5)
 A  =

   7.     73.    36.    81.    76.
   84.    60.    107.   124.   106.
   26.    28.    106.   7.     8.
   49.    105.   66.    93.    103.
   103.   15.    124.   51.    115.

--> [Q,R] = qr_GS(A)
 Q  =

   0.0485468   0.6049133   0.1224189   0.397002    0.6775885
   0.5825613   0.0670061  -0.1131391   0.6540232  -0.4643121
   0.1803166   0.1038403   0.9481017  -0.1439537  -0.1925709
   0.3398274   0.6560436  -0.2693983  -0.5756098  -0.2241026
   0.7143311  -0.4340669  -0.0272173  -0.2501873   0.4878346
 R  =

   144.19085   89.943297   194.20096   145.46693   184.03387
   0.          113.46014   29.428184   96.908258   71.561578
   0.          0.          71.644755  -23.918726  -25.982104
   0.          0.          0.          45.957106   10.287639
   0.          0.          0.          0.          33.757493

--> Q*R
 ans  =

   7.     73.    36.    81.    76.
   84.    60.    107.   124.   106.
   26.    28.    106.   7.     8.
   49.    105.   66.    93.    103.
   103.   15.    124.   51.    115.
```

Sabendo que Q é uma matriz ortogonal, logo, $Q^T Q = I$. Então podemos testar sua ortogonalidade vendo o quão próximo esse produto com nosso Q se aproxima da matriz identidade.

```
--> Q'*Q
 ans  =

   1.          1.437D-16   7.014D-16   4.049D-16   1.151D-15
   1.437D-16   1.         -5.616D-16  -5.116D-16  -1.295D-15
   7.014D-16  -5.616D-16   1.         -7.851D-16  -2.315D-15
   4.049D-16  -5.116D-16  -7.851D-16   1.         -1.952D-15
   1.151D-15  -1.267D-15  -2.321D-15  -1.952D-15   1.
```

A diagonal é composta por 1's, e os demais índices estão muito próximos de 0 em cerca de 15 e 16 casas decimais.

Observação: Vale ressaltar que mesmo os números inteiros do produto QR sendo os mesmos, o produto não necessariamente é igual. Isso fica evidente ao realizar $Q * R - A$, que deveria ser 0, mas ao testar em um exemplo qualquer não é totalmente nulo.

```
--> [Q10,R10] = qr_GSM(A_10);

--> Q10*R10 - A_10
 ans  =

   0.          0.   1.137D-13   6.395D-14   0.          0.          5.684D-14  -5.684D-14   0.          5.684D-14
   0.          0.   0.          5.684D-14  -5.684D-14   5.329D-15   0.          1.137D-13   0.          0.
   2.842D-14   0.   0.          2.842D-14   0.          1.137D-13   0.          5.684D-14   0.         -5.684D-14
   0.          0.   0.          0.          5.684D-14   0.          1.137D-13   5.684D-14   0.          0.
  -1.137D-13   0.  -1.421D-14  -1.137D-13   0.          0.          2.398D-14   1.137D-13   0.         -1.137D-13
   0.          0.   0.         -1.137D-13   7.105D-15   0.          1.137D-13   0.         -1.137D-13   0.
   0.          0.   0.          2.842D-14   0.          0.          0.          7.105D-15   5.684D-14  -2.842D-14
   0.          0.   0.          0.          5.684D-14   0.          0.          1.137D-13   0.          0.
   0.          0.   0.          0.          0.          0.         -1.137D-13   5.684D-14   0.          0.
   0.          0.  -5.684D-14   0.          0.          0.         -1.137D-13  -1.137D-13   0.          4.263D-14
```

Exemplo 2:

```
--> A = matriz_aleat(5)
 A  =

   71.    16.    92.   78.   4.
   71.    91.    1.    15.   65.
   102.   34.    74.   76.   49.
   8.     68.    39.   85.   30.
   70.    123.   32.   42.   63.

--> [Q,R] = qr_GS(A)
 Q  =

   0.4450558  -0.3492842   0.530019   -0.4015835  -0.4875822
   0.4450558   0.2990329  -0.5487411   0.2339848  -0.5971926
   0.6393759  -0.4065803  -0.0295226   0.4465441   0.4749922
   0.0501471   0.5328676   0.6457186   0.5389879  -0.077955
   0.4387874   0.5825157  -0.0117883  -0.5422842   0.4170491
 R  =

   159.53056   126.74061   104.70094   112.67434   91.18629
   0.          115.68413  -22.499721   16.100621   50.802078
   0.          0.          70.83414    85.257624  -16.365806
   0.          0.          0.          29.161646   17.489072
   0.          0.          0.          0.          6.4422188

--> Q*R
 ans  =

   71.    16.    92.   78.   4.
   71.    91.    1.    15.   65.
   102.   34.    74.   76.   49.
   8.     68.    39.   85.   30.
   70.    123.   32.   42.   63.
```

```
--> Q'*Q
 ans  =

   1.          -5.386D-17  -1.248D-16  -1.834D-16   7.192D-16
  -5.386D-17   1.           8.960D-17  -8.074D-17   2.272D-15
  -1.248D-16   8.960D-17   1.           3.455D-16   9.284D-16
  -1.834D-16  -8.074D-17   3.455D-16   1.           2.667D-15
   7.469D-16   2.272D-15   9.111D-16   2.695D-15   1.
```

A diagonal é composta por 1's, e os demais índices estão muito próximos de 0 em cerca de 16 e 17 casas decimais.

ORDEM 10

Exemplo 1:

```
--> A = matriz_aleat(10)
 A  =

   901.   286.   410.   214.   439.   782.   140.   921.   912.   996.
   395.   251.   11.    689.   76.    55.    115.   944.   444.   158.
   565.   339.   197.   585.   256.   919.   536.   900.   598.   535.
   706.   392.   273.   420.   67.    460.   431.   809.   774.   213.
   679.   468.   344.   428.   765.   299.   614.   26.    792.   559.
   413.   336.   204.   319.   42.    3.     925.   2.     550.   431.
   141.   534.   301.   576.   344.   899.   100.   508.   409.   23.
   495.   204.   276.   426.   197.   838.   428.   408.   722.   576.
   420.   159.   295.   976.   213.   434.   943.   840.   477.   715.
   862.   19.    572.   252.   314.   777.   33.    502.   639.   932.

--> [Q,R] = qr_GS(A);

--> Q*R
 ans  =

   901.   286.   410.   214.   439.   782.   140.   921.   912.   996.
   395.   251.   11.    689.   76.    55.    115.   944.   444.   158.
   565.   339.   197.   585.   256.   919.   536.   900.   598.   535.
   706.   392.   273.   420.   67.    460.   431.   809.   774.   213.
   679.   468.   344.   428.   765.   299.   614.   26.    792.   559.
   413.   336.   204.   319.   42.    3.     925.   2.     550.   431.
   141.   534.   301.   576.   344.   899.   100.   508.   409.   23.
   495.   204.   276.   426.   197.   838.   428.   408.   722.   576.
   420.   159.   295.   976.   213.   434.   943.   840.   477.   715.
   862.   19.    572.   252.   314.   777.   33.    502.   639.   932.
```

Testando sua ortogonalidade:

```
--> Q'*Q
 ans  =

         column 1 to 9

   1.         -9.583D-17   3.464D-16  -1.295D-16  -3.435D-17  -1.245D-16   9.242D-17   1.409D-15   1.751D-15
  -9.583D-17   1.          1.997D-16   2.441D-16   1.745D-16   1.098D-16  -6.636D-17   7.400D-17   9.159D-16
   3.464D-16   1.997D-16   1.         -7.984D-16  -9.421D-16  -6.767D-16  -7.564D-16  -6.448D-16  -4.247D-15
  -1.295D-16   2.441D-16  -7.984D-16   1.          3.691D-16   6.522D-16   3.965D-16   4.338D-16   4.718D-16
  -3.435D-17   1.745D-16  -9.421D-16   3.691D-16   1.          4.764D-16  -5.575D-17  -2.171D-15  -7.182D-16
  -1.245D-16   1.098D-16  -6.767D-16   6.522D-16   4.764D-16   1.         -2.552D-16  -1.162D-15   7.772D-16
   9.242D-17  -6.636D-17  -7.564D-16   3.965D-16  -5.575D-17  -2.552D-16   1.         -1.466D-15  -1.776D-15
   1.409D-15   7.400D-17  -6.448D-16   4.338D-16  -2.171D-15  -1.162D-15  -1.466D-15   1.         -1.377D-14
   1.743D-15   9.379D-16  -4.272D-15   4.909D-16  -7.149D-16   8.038D-16  -1.754D-15  -1.373D-14   1.
  -2.141D-15  -7.855D-17   1.612D-15   1.232D-15   2.447D-15   1.652D-15   1.002D-15  -3.917D-16   2.512D-15

         column 10

  -2.130D-15
  -6.592D-17
   1.589D-15
   1.193D-15
   2.415D-15
   1.638D-15
   1.027D-15
  -3.886D-16
   2.512D-15
   1.
```

Exemplo 2:

```
--> [A] = matriz_aleat(10)
 A  =

   212.   561.   308.   502.   281.   410.   387.   538.   588.   649.
   756.   662.   933.   437.   128.   878.   922.   120.   483.   992.
   1.     726.   215.   270.   778.   114.   948.   226.   224.   50.
   330.   199.   313.   632.   212.   200.   344.   627.   840.   748.
   665.   544.   362.   405.   113.   562.   376.   761.   121.   410.
   628.   232.   292.   918.   686.   590.   734.   49.    286.   608.
   849.   231.   566.   44.    153.   685.   262.   672.   860.   854.
   686.   217.   483.   482.   697.   890.   499.   202.   849.   65.
   878.   883.   332.   264.   841.   504.   264.   391.   526.   828.
   69.    652.   593.   415.   406.   350.   525.   830.   993.   926.

--> [Q,R] = qr_GS(A);

--> Q*R
 ans  =

   212.   561.   308.   502.   281.   410.   387.   538.   588.   649.
   756.   662.   933.   437.   128.   878.   922.   120.   483.   992.
   1.     726.   215.   270.   778.   114.   948.   226.   224.   50.
   330.   199.   313.   632.   212.   200.   344.   627.   840.   748.
   665.   544.   362.   405.   113.   562.   376.   761.   121.   410.
   628.   232.   292.   918.   686.   590.   734.   49.    286.   608.
   849.   231.   566.   44.    153.   685.   262.   672.   860.   854.
   686.   217.   483.   482.   697.   890.   499.   202.   849.   65.
   878.   883.   332.   264.   841.   504.   264.   391.   526.   828.
   69.    652.   593.   415.   406.   350.   525.   830.   993.   926.
```

```
--> Q'*Q
 ans  =

   1.          2.660D-16   1.036D-16  -4.471D-17  -2.481D-17   1.266D-15  -8.507D-18   5.966D-17   1.622D-15   1.388D-15
   2.660D-16   1.         -6.759D-16  -1.448D-16  -5.628D-16  -2.395D-16   4.147D-16  -3.797D-16   2.082D-15  -1.013D-15
   1.036D-16  -6.759D-16   1.          2.603D-16   3.435D-16  -4.841D-17   5.735D-16   8.982D-16   7.078D-16   1.527D-15
  -4.471D-17  -1.448D-16   2.603D-16   1.          4.167D-16   1.223D-16  -6.702D-17   1.708D-16  -1.013D-15   1.138D-15
  -2.481D-17  -5.628D-16   3.435D-16   4.167D-16   1.         -1.032D-16   3.864D-16   3.936D-16  -3.886D-16   9.437D-16
   1.266D-15  -2.395D-16  -4.841D-17   1.223D-16  -1.032D-16   1.         -2.565D-15  -3.115D-15  -7.064D-15  -6.072D-15
  -8.507D-18   4.147D-16   5.735D-16  -6.702D-17   3.864D-16  -2.565D-15   1.         -6.941D-16  -3.539D-15  -3.164D-15
   5.966D-17  -3.797D-16   8.982D-16   1.708D-16   3.936D-16  -3.115D-15  -6.941D-16   1.         -5.839D-15  -3.941D-15
   1.633D-15   2.114D-15   6.917D-16  -1.046D-15  -4.006D-16  -7.091D-15  -3.509D-15  -5.835D-15   1.         -2.276D-14
   1.369D-15  -9.994D-16   1.538D-15   1.149D-15   9.371D-16  -6.095D-15  -3.151D-15  -3.940D-15  -2.276D-14   1.
```

Matrizes Não-Quadradas

Exemplo 1: A(m,n), onde $n > m$ Mais colunas do que linhas.

```
--> [A] = matriz_aleat2(4,5)
 A  =

   19.   46.   61.   36.   62.
   41.   57.   32.   30.   52.
   6.    16.   33.   50.   27.
   29.   28.   36.   50.   16.

--> [Q,R] = qr_GS(A)
 Q  =

   0.351671   0.7882146   0.2206797  -0.4542529   0.2934409
   0.758869  -0.0351237  -0.6077742   0.2312891  -0.509446
   0.111054   0.3094431   0.4389893   0.836182   -0.6412227
   0.536761  -0.5307815   0.6238584  -0.2023841   0.4931437
 R  =

   54.027771   76.23857    68.723917   67.816976   72.851424
   0.          24.345028   38.06062    16.255095   46.905332
   0.          0.          30.958238   42.853629   3.9123292
   0.          0.          0.          22.275462   3.2021185
   0.          0.          0.          0.          8.172D-14

--> Q*R
 ans  =

   19.   46.   61.   36.   62.
   41.   57.   32.   30.   52.
   6.    16.   33.   50.   27.
   29.   28.   36.   50.   16.
```

Como esperado, a matriz Q tem tamanho (m,n) e a matriz R tem tamanho (n,n).

Exemplo 2: A(m,n), onde $m > n$ Mais linhas do que colunas.

```
--> [A] = matriz_aleat2(5,4)
 A  =

   40.    72.    15.    71.
   66.    35.    115.   88.
   71.    119.   91.    85.
   6.     113.   112.   52.
   103.   42.    49.    18.

--> [Q,R] = qr_GS(A)
 Q  =

   0.2719013   0.2720543  -0.5260868   0.5835269
   0.4486372  -0.158366    0.7370279   0.4756791
   0.4826248   0.4191274  -0.1694781   0.0484924
   0.0407852   0.7813992   0.3491348  -0.3682373
   0.7001459  -0.3386064  -0.171478   -0.5433926
 R  =

   147.1122   126.7264   138.46574   114.53163
   0.         137.9979   94.93431    75.543318
   0.         0.         92.145079   28.169063
   0.         0.         0.          58.482623

--> Q*R
 ans  =

   40.    72.    15.    71.
   66.    35.    115.   88.
   71.    119.   91.    85.
   6.     113.   112.   52.
   103.   42.    49.    18.
```

Como esperado, a matriz Q tem tamanho (m,n) e a matriz R tem tamanho (n,n).

---

2) MÉTODO DE GRAM-SCHMIDT MODIFICADO

> Escreva uma função Scilab function [Q,R] = qr_GSM(A) que implementa o Método de Gram-Schmidt Modificado. Testar a sua função com as mesmas matrizes usadas nos testes do item anterior. Comparar a precisão dos dois Métodos.

Para o método modificado, vamos usar v em vez de a_j para $r_i j$ quando i ¡ j:

Código:

```
function [Q,R] = qr_GSM(A)
[m,n] = size(A); // le dim de A
//inicializa as matrizes Q e R
Q = zeros(m,n);
R = zeros(n,n);
//loop
for j = 1:n
  v = A(:,j);
  for i = 1:(j-1)
    R(i,j) = Q(:,i)' * v; // r_ij caso i < j
    v = v - R(i,j)*Q(:,i);
  end
  R(j,j) = norm(v, 2); // r_ij , quando i = j
  Q(:,j) = v/(R(j,j));
end
endfunction
```

Agora a função é mais estável em relação a ortogonalidade.

Testando com o **Método de Gram-Schmidt Modificado**:

ORDEM 5

Exemplo 1:

```
--> A
 A  =

   7.     73.    36.    81.    76.
   84.    60.    107.   124.   106.
   26.    28.    106.   7.     8.
   49.    105.   66.    93.    103.
   103.   15.    124.   51.    115.

--> [Q,R] = qr_GSM(A)
 Q  =

   0.0485468   0.6049133   0.1224189   0.397002    0.6775885
   0.5825613   0.0670061  -0.1131391   0.6540232  -0.4643121
   0.1803166   0.1038403   0.9481017  -0.1439537  -0.1925709
   0.3398274   0.6560436  -0.2693983  -0.5756098  -0.2241026
   0.7143311  -0.4340669  -0.0272173  -0.2501873   0.4878346
 R  =

   144.19085   89.943297   194.20096   145.46693   184.03387
   0.          113.46014   29.428184   96.908258   71.561578
   0.          0.          71.644755  -23.918726  -25.982104
   0.          0.          0.          45.957106   10.287639
   0.          0.          0.          0.          33.757493

--> Q*R
 ans  =

   7.     73.    36.    81.    76.
   84.    60.    107.   124.   106.
   26.    28.    106.   7.     8.
   49.    105.   66.    93.    103.
   103.   15.    124.   51.    115.
```

Assim como antes, podemos testar sua ortogonalidade vendo o quão próximo $Q^T Q = I$ com nosso Q se aproxima da matriz identidade.

```
--> Q'*Q
 ans  =

   1.          1.437D-16   7.499D-16   3.811D-16   1.254D-15
   1.437D-16   1.         -2.802D-17  -2.414D-16  -1.084D-16
   7.499D-16  -2.802D-17   1.         -9.686D-17  -2.529D-17
   3.811D-16  -2.414D-16  -9.686D-17   1.         -2.195D-17
   1.254D-15  -8.066D-17  -2.182D-17  -2.195D-17   1.
```

Observação: Vale ressaltar que aqui também, mesmo os números inteiros sendo os mesmos, o produto não necessariamente é igual. Isso fica evidente ao realizar $Q * R - A$, que deveria ser 0, mas assim como na GS, não é totalmente.

```
--> [Q10,R10] = qr_GSM(A_10);

--> Q10*R10 - A_10
 ans  =

   0.          0.   1.137D-13   6.395D-14   0.          0.          5.684D-14  -5.684D-14   0.          5.684D-14
   0.          0.   0.          5.684D-14  -5.684D-14   5.329D-15   0.          1.137D-13   0.          0.
   2.842D-14   0.   0.          2.842D-14   0.          1.137D-13   0.          5.684D-14   0.         -5.684D-14
   0.          0.   0.          0.          5.684D-14   0.          1.137D-13   5.684D-14   0.          0.
  -1.137D-13   0.  -1.421D-14  -1.137D-13   0.          0.          2.398D-14   1.137D-13   0.         -1.137D-13
   0.          0.   0.         -1.137D-13   7.105D-15   0.          1.137D-13   0.         -1.137D-13   0.
   0.          0.   0.          2.842D-14   0.          0.          0.          7.105D-15   5.684D-14  -2.842D-14
   0.          0.   0.          0.          5.684D-14   0.          0.          1.137D-13   0.          0.
   0.          0.   0.          0.          0.          0.         -1.137D-13   5.684D-14   0.          0.
   0.          0.  -5.684D-14   0.          0.          0.         -1.137D-13  -1.137D-13   0.          4.263D-14
```

Exemplo 2:

```
--> A
 A  =

   71.    16.    92.   78.   4.
   71.    91.    1.    15.   65.
   102.   34.    74.   76.   49.
   8.     68.    39.   85.   30.
   70.    123.   32.   42.   63.

--> [Q,R] = qr_GSM(A)
 Q  =

   0.4450558  -0.3492842   0.530019   -0.4015835  -0.4875822
   0.4450558   0.2990329  -0.5487411   0.2339848  -0.5971926
   0.6393759  -0.4065803  -0.0295226   0.4465441   0.4749922
   0.0501471   0.5328676   0.6457186   0.5389879  -0.077955
   0.4387874   0.5825157  -0.0117883  -0.5422842   0.4170491
 R  =

   159.53056   126.74061   104.70094   112.67434   91.18629
   0.          115.68413  -22.499721   16.100621   50.802078
   0.          0.          70.83414    85.257624  -16.365806
   0.          0.          0.          29.161646   17.489072
   0.          0.          0.          0.          6.4422188

--> Q*R
 ans  =

   71.    16.    92.   78.   4.
   71.    91.    1.    15.   65.
   102.   34.    74.   76.   49.
   8.     68.    39.   85.   30.
   70.    123.   32.   42.   63.
```

```
--> Q'*Q
 ans  =

   1.          -5.386D-17  -9.379D-17  -3.222D-16   9.735D-16
  -5.386D-17   1.          -3.557D-17  -8.074D-17   7.584D-16
  -9.379D-17  -3.557D-17   1.          -7.368D-16   2.407D-15
  -3.222D-16  -8.074D-17  -7.368D-16   1.          -1.989D-16
   9.735D-16   7.306D-16   2.433D-15  -1.989D-16   1.
```

ORDEM 10

Exemplo 1:

```
--> A
 A  =

   901.   286.   410.   214.   439.   782.   140.   921.   912.   996.
   395.   251.   11.    689.   76.    55.    115.   944.   444.   158.
   565.   339.   197.   585.   256.   919.   536.   900.   598.   535.
   706.   392.   273.   420.   67.    460.   431.   809.   774.   213.
   679.   468.   344.   428.   765.   299.   614.   26.    792.   559.
   413.   336.   204.   319.   42.    3.     925.   2.     550.   431.
   141.   534.   301.   576.   344.   899.   100.   508.   409.   23.
   495.   204.   276.   426.   197.   838.   428.   408.   722.   576.
   420.   159.   295.   976.   213.   434.   943.   840.   477.   715.
   862.   19.    572.   252.   314.   777.   33.    502.   639.   932.

--> [Q,R] = qr_GSM(A);

--> Q*R
 ans  =

   901.   286.   410.   214.   439.   782.   140.   921.   912.   996.
   395.   251.   11.    689.   76.    55.    115.   944.   444.   158.
   565.   339.   197.   585.   256.   919.   536.   900.   598.   535.
   706.   392.   273.   420.   67.    460.   431.   809.   774.   213.
   679.   468.   344.   428.   765.   299.   614.   26.    792.   559.
   413.   336.   204.   319.   42.    3.     925.   2.     550.   431.
   141.   534.   301.   576.   344.   899.   100.   508.   409.   23.
   495.   204.   276.   426.   197.   838.   428.   408.   722.   576.
   420.   159.   295.   976.   213.   434.   943.   840.   477.   715.
   862.   19.    572.   252.   314.   777.   33.    502.   639.   932.
```

Testando sua ortogonalidade:

```
--> Q'*Q
 ans  =

   1.          -9.583D-17   4.246D-16  -6.100D-17  -6.670D-17  -1.505D-16   7.717D-17   1.278D-15   1.745D-15  -1.769D-15
  -9.583D-17   1.           8.551D-17   8.545D-18  -8.417D-17  -1.310D-16  -1.314D-16   7.874D-17   2.498D-16   1.197D-16
   4.246D-16   8.551D-17   1.          -3.716D-17  -7.469D-17  -1.562D-16  -7.674D-17   4.397D-16   1.665D-16  -2.984D-16
  -6.100D-17   8.545D-18  -3.716D-17   1.           3.520D-17   6.677D-17  -1.939D-16  -1.379D-16  -2.776D-17   2.776D-16
  -6.670D-17  -8.417D-17  -7.469D-17   3.520D-17   1.           6.869D-18   8.447D-18   3.877D-17  -2.082D-17   1.665D-16
  -1.505D-16  -1.310D-16  -1.562D-16   6.677D-17   6.869D-18   1.           1.339D-16  -7.694D-17  -1.110D-16  -1.665D-16
   7.717D-17  -1.314D-16  -7.674D-17  -1.939D-16   8.447D-18   1.339D-16   1.          -1.933D-16   0.           1.388D-16
   1.278D-15   7.874D-17   4.397D-16  -1.379D-16   3.877D-17  -7.694D-17  -1.933D-16   1.          -5.551D-17  -5.551D-17
   1.746D-15   2.145D-16   1.627D-16  -4.243D-17  -1.937D-17  -1.402D-16   4.660D-18  -8.384D-18   1.          -1.492D-16
  -1.801D-15   9.835D-17  -2.689D-16   2.405D-16   1.865D-16  -1.543D-16   1.264D-16  -4.504D-17  -1.492D-16   1.
```

Exemplo 2:

```
--> A
 A  =

   212.   561.   308.   502.   281.   410.   387.   538.   588.   649.
   756.   662.   933.   437.   128.   878.   922.   120.   483.   992.
   1.     726.   215.   270.   778.   114.   948.   226.   224.   50.
   330.   199.   313.   632.   212.   200.   344.   627.   840.   748.
   665.   544.   362.   405.   113.   562.   376.   761.   121.   410.
   628.   232.   292.   918.   686.   590.   734.   49.    286.   608.
   849.   231.   566.   44.    153.   685.   262.   672.   860.   854.
   686.   217.   483.   482.   697.   890.   499.   202.   849.   65.
   878.   883.   332.   264.   841.   504.   264.   391.   526.   828.
   69.    652.   593.   415.   406.   350.   525.   830.   993.   926.

--> [Q,R] = qr_GSM(A);

--> Q*R
 ans  =

   212.   561.   308.   502.   281.   410.   387.   538.   588.   649.
   756.   662.   933.   437.   128.   878.   922.   120.   483.   992.
   1.     726.   215.   270.   778.   114.   948.   226.   224.   50.
   330.   199.   313.   632.   212.   200.   344.   627.   840.   748.
   665.   544.   362.   405.   113.   562.   376.   761.   121.   410.
   628.   232.   292.   918.   686.   590.   734.   49.    286.   608.
   849.   231.   566.   44.    153.   685.   262.   672.   860.   854.
   686.   217.   483.   482.   697.   890.   499.   202.   849.   65.
   878.   883.   332.   264.   841.   504.   264.   391.   526.   828.
   69.    652.   593.   415.   406.   350.   525.   830.   993.   926.
```

```
--> Q'*Q
 ans  =

   1.          2.660D-16   1.165D-16  -2.773D-17  -9.951D-17   1.303D-15   5.036D-19  -1.896D-16   1.934D-15   1.318D-15
   2.660D-16   1.         -3.199D-17  -1.639D-16  -6.146D-17   4.553D-17   4.288D-17  -1.443D-16   2.776D-16   1.388D-17
   1.165D-16  -3.199D-17   1.          4.009D-19   9.895D-17  -2.808D-16  -1.622D-16  -2.575D-16  -9.992D-16  -5.551D-16
  -2.773D-17  -1.639D-16   4.009D-19   1.          1.303D-16  -2.671D-17   2.590D-16   3.787D-16  -2.429D-16   5.551D-16
  -9.951D-17  -6.146D-17   9.895D-17   1.303D-16   1.          9.645D-17   3.597D-17   2.393D-17   1.388D-16   2.776D-17
   1.303D-15   4.553D-17  -2.808D-16  -2.671D-17   9.645D-17   1.          6.992D-18   8.606D-19   4.163D-17  -3.849D-17
   5.036D-19   4.288D-17  -1.622D-16   2.590D-16   3.597D-17   6.992D-18   1.          3.809D-17  -2.776D-17   5.551D-17
  -1.896D-16  -1.443D-16  -2.575D-16   3.787D-16   2.393D-17   8.606D-19   3.809D-17   1.         -3.123D-17   1.388D-17
   1.924D-15   2.691D-16  -1.003D-15  -2.429D-16   1.010D-16   6.457D-18  -7.573D-18  -2.941D-17   1.          0.
   1.320D-15   8.959D-18  -5.295D-16   5.579D-16   3.709D-17  -5.587D-17   4.103D-17   8.267D-18   0.          1.
```

Matrizes Não-Quadradas

Exemplo 1: A(m,n), onde $n > m$ Mais colunas do que linhas.

```
--> A
 A  =

   19.   46.   61.   36.   62.
   41.   57.   32.   30.   52.
   6.    16.   33.   50.   27.
   29.   28.   36.   50.   16.

--> [Q,R] = qr_GSM(A)
 Q  =

   0.351671   0.7882146   0.2206797  -0.4542529   0.3607406
   0.758869  -0.0351237  -0.6077742   0.2312891  -0.5657068
   0.111054   0.3094431   0.4389893   0.836182    0.2295622
   0.536761  -0.5307815   0.6238584  -0.2023841  -0.7050839
 R  =

   54.027771   76.23857    68.723917   67.816976   72.851424
   0.          24.345028   38.06062    16.255095   46.905332
   0.          0.          30.958238   42.853629   3.9123292
   0.          0.          0.          22.275462   3.2021185
   0.          0.          0.          0.          1.354D-14

--> Q*R
 ans  =

   19.   46.   61.   36.   62.
   41.   57.   32.   30.   52.
   6.    16.   33.   50.   27.
   29.   28.   36.   50.   16.
```

Como esperado, a matriz Q tem tamanho (m,n) e a matriz R tem tamanho (n,n).

Exemplo 2: A(m,n), onde $m > n$ Mais linhas do que colunas.

```
--> A
 A  =

   40.    72.    15.    71.
   66.    35.    115.   88.
   71.    119.   91.    85.
   6.     113.   112.   52.
   103.   42.    49.    18.

--> [Q,R] = qr_GSM(A)
 Q  =

   0.2719013   0.2720543  -0.5260868   0.5835269
   0.4486372  -0.158366    0.7370279   0.4756791
   0.4826248   0.4191274  -0.1694781   0.0484924
   0.0407852   0.7813992   0.3491348  -0.3682373
   0.7001459  -0.3386064  -0.171478   -0.5433926
 R  =

   147.1122   126.7264   138.46574   114.53163
   0.         137.9979   94.93431    75.543318
   0.         0.         92.145079   28.169063
   0.         0.         0.          58.482623

--> Q*R
 ans  =

   40.    72.    15.    71.
   66.    35.    115.   88.
   71.    119.   91.    85.
   6.     113.   112.   52.
   103.   42.    49.    18.
```

Como esperado, a matriz Q tem tamanho (m,n) e a matriz R tem tamanho (n,n).

4) MÉTODO DE HOUSEHOLDER

Escreva uma função Scilab function [U,R] = qr_House(A) que implementa o Método de Householder para determinar a decomposição QR de uma matriz A. A matriz U, triangular inferior, deve conter em suas colunas os vetores unitários que geraram as matrizes dos refletores de Householder usadas para gerar a decomposição QR. Escreva também uma função Scilab function [Q] = constroi_Q_House(U) que constrói a matriz ortogonal Q da decomposição A = QR a partir da matriz U retornada pela função function [U,R] = qr_House(A).

No método de Householder, vamos decompor A, mas nossa função inicialmente retornará as matrizes U, R em vez de Q e R.

Basicamente, vamos criar um loop para transformar A na matriz R da seguinte forma:

$$
\begin{gathered}
A = R_0 \\
Q_1 A = Q_1 R = R_1 \\
Q_2 Q_1 A = Q_2 R_1 = R_2 \\
\vdots \\
Q_j Q_{j-1} \cdots Q_1 A = Q_j R_{j-1} = R_j \\
\vdots \\
Q_k \cdots Q_1 A = R
\end{gathered}
$$

Para a reconstrução de Q, basta lembrar que:

$$
\begin{gathered}
Q_k \cdots Q_2 Q_1 A = R \\
= Q^T A = R \\
A = QR \\
Q = (Q_k \cdots Q_2 Q_1)^T \\
Q = Q_1^T Q_2^T \cdots Q_k^T \\
Q = Q_1 Q_2 \cdots Q_k
\end{gathered}
$$

Então vamos criar um loop para multiplicar as matrizes Q pelo refletor de Householder: $I - 2uu^T$
Juntando as duas funções:

Código:

```
function [Q, U, R] = Householder(A)
[m,n] = size(A);  //Ler dim de A
k = min(m-1, n); //Tamanho de U depende do tamanho de A
for j = 1:k
  x = A(j:m, j); //Coluna j da linha j ate a final
   if x(1) > 0 then
     x(1) = x(1) + norm(x, 2);
   else
     x(1) = x(1) - norm(x,2);
  end

  u = x/norm(x,2);
  U(j:m, j) = u; //Guarda o vetor u em U

   I = eye(m - j + 1);
  H = I - 2*u*u';
  A(j:m, j:m) = A(j:m, j:m) - 2*u*u'*A(j:m, j:m);
end

R = triu(A) // R e a matriz A alterana apos varias iteracoes
//limpa os zeros
Q = constroi_Q(U); // Constroi Q a partir de U
endfunction

function Q = constroi_Q(U)
```

```
    [m,k] = size(U);
    Q = eye(m,m);
    for j = 1:k
      u = U(1:m, j)
      //Q = Q(I - 2uu^T)
      Q = Q - 2*(Q*u)*u';
    end
    endfunction

```

Testando com o **Método de Householder**:

ORDEM 5

Exemplo 1:

```
--> A
 A  =

   7.     73.    36.    81.    76.
   84.    60.    107.   124.   106.
   26.    28.    106.   7.     8.
   49.    105.   66.    93.    103.
   103.   15.    124.   51.    115.

--> [Q, U, R] = Householder(A)
 Q  =

  -0.0485468   0.6049133  -0.1224189   0.397002    0.6775885
  -0.5825613   0.0670061   0.1131391   0.6540232  -0.4643121
  -0.1803166   0.1038403  -0.9481017  -0.1439537  -0.1925709
  -0.3398274   0.6560436   0.2693983  -0.5756098  -0.2241026
  -0.7143311  -0.4340669   0.0272173  -0.2501873   0.4878346
 U  =

   0.7240673   0.          0.          0.
   0.4022839  -0.7965793   0.          0.
   0.1245165  -0.0001164   0.9815997   0.
   0.2346656   0.2887315  -0.1908799  -0.8911918
   0.4932767  -0.5311267   0.0051802  -0.4536267
 R  =

  -144.19085  -89.943297  -194.20096  -145.46693  -184.03387
   0.          113.46014   29.428184   96.908258   71.561578
   0.          0.         -71.644755   23.918726   25.982104
   0.          0.          0.          45.957106   10.287639
   0.          0.          0.          0.          33.757493
```

Curiosamente, ao usar a função qr(A) do próprio scilab, é possível observar que a decomposição retorna as matrizes exatamente iguais às da nossa função de Householder.

```
--> Q*R
 ans  =

   7.     73.    36.    81.    76.
   84.    60.    107.   124.   106.
   26.    28.    106.   7.     8.
   49.    105.   66.    93.    103.
   103.   15.    124.   51.    115.

--> [Q, R] = qr(A)
 Q  =

  -0.0485468   0.6049133  -0.1224189   0.397002    0.6775885
  -0.5825613   0.0670061   0.1131391   0.6540232  -0.4643121
  -0.1803166   0.1038403  -0.9481017  -0.1439537  -0.1925709
  -0.3398274   0.6560436   0.2693983  -0.5756098  -0.2241026
  -0.7143311  -0.4340669   0.0272173  -0.2501873   0.4878346
 R  =

  -144.19085  -89.943297  -194.20096  -145.46693  -184.03387
   0.          113.46014   29.428184   96.908258   71.561578
   0.          0.         -71.644755   23.918726   25.982104
   0.          0.          0.          45.957106   10.287639
   0.          0.          0.          0.          33.757493
```

Observação: Vale ressaltar que mesmo os números inteiros sendo os mesmos, o produto não necessariamente é igual. Isso fica evidente ao realizar $Q * R - A$, que deveria ser 0, mas assim como nos outros casos, não é totalmente.

```
--> [Q, U, R] = Householder(A_10);

--> Q10*R10 - A_10
 ans  =

   0.          0.   1.137D-13   6.395D-14   0.          0.          5.684D-14  -5.684D-14   0.          5.684D-14
   0.          0.   0.          5.684D-14  -5.684D-14   5.329D-15   0.          1.137D-13   0.          0.
   2.842D-14   0.   0.          2.842D-14   0.          1.137D-13   0.          5.684D-14   0.         -5.684D-14
   0.          0.   0.          0.          5.684D-14   0.          1.137D-13   5.684D-14   0.          0.
  -1.137D-13   0.  -1.421D-14  -1.137D-13   0.          0.          2.398D-14   1.137D-13   0.         -1.137D-13
   0.          0.   0.         -1.137D-13   7.105D-15   0.          1.137D-13   0.         -1.137D-13   0.
   0.          0.   0.          2.842D-14   0.          0.          0.          7.105D-15   5.684D-14  -2.842D-14
   0.          0.   0.          0.          5.684D-14   0.          0.          1.137D-13   0.          0.
   0.          0.   0.          0.          0.          0.         -1.137D-13   5.684D-14   0.          0.
   0.          0.  -5.684D-14   0.          0.          0.         -1.137D-13  -1.137D-13   0.          4.263D-14
```

Assim como antes, podemos testar sua ortogonalidade vendo o quão próximo $Q^T Q = I$ com nosso Q se aproxima da matriz identidade.

```
--> Q'*Q
 ans  =

   1.         -1.583D-16   3.908D-17  -1.431D-16  -2.842D-16
  -1.583D-16   1.         -1.813D-17  -1.565D-17   2.195D-16
   3.908D-17  -1.813D-17   1.         -3.173D-18  -9.626D-18
  -1.431D-16  -1.565D-17  -3.173D-18   1.          2.967D-16
  -2.842D-16   2.195D-16  -7.891D-18   2.550D-16   1.
```

Exemplo 2:

```
--> A
 A  =

   71.    16.    92.   78.   4.
   71.    91.    1.    15.   65.
   102.   34.    74.   76.   49.
   8.     68.    39.   85.   30.
   70.    123.   32.   42.   63.

--> [Q, U, R] = Householder(A)
 Q  =

  -0.4450558   0.3492842   0.530019    0.4015835  -0.4875822
  -0.4450558  -0.2990329  -0.5487411  -0.2339848  -0.5971926
  -0.6393759   0.4065803  -0.0295226  -0.4465441   0.4749922
  -0.0501471  -0.5328676   0.6457186  -0.5389879  -0.077955
  -0.4387874  -0.5825157  -0.0117883   0.5422842   0.4170491
 U  =

   0.8500164   0.          0.          0.
   0.2617925   0.838632    0.          0.
   0.3760962  -0.1502667  -0.8341482   0.
   0.0294977   0.3249272   0.5413793   0.9646442
   0.2581053   0.4105346   0.1053813  -0.2635557
 R  =

  -159.53056  -126.74061  -104.70094  -112.67434  -91.18629
   0.         -115.68413   22.499721  -16.100621  -50.802078
   0.          0.          70.83414    85.257624  -16.365806
   0.          0.          0.         -29.161646  -17.489072
   0.          0.          0.          0.          6.4422188
```

```
--> Q*R
 ans  =

   71.    16.    92.   78.   4.
   71.    91.    1.    15.   65.
   102.   34.    74.   76.   49.
   8.     68.    39.   85.   30.
   70.    123.   32.   42.   63.

--> Q'*Q
 ans  =

   1.           8.033D-17   1.081D-16  -1.396D-17  -1.239D-16
   8.033D-17   1.          -1.583D-16  -9.961D-17   1.756D-16
   1.081D-16  -1.583D-16   1.          -5.015D-16   2.684D-16
  -1.396D-17  -9.961D-17  -5.015D-16   1.           4.066D-16
  -1.517D-16   1.756D-16   2.719D-16   4.343D-16   1.
```

ORDEM 10

Exemplo 1:

```
--> A
 A  =

   901.   286.   410.   214.   439.   782.   140.   921.   912.   996.
   395.   251.   11.    689.   76.    55.    115.   944.   444.   158.
   565.   339.   197.   585.   256.   919.   536.   900.   598.   535.
   706.   392.   273.   420.   67.    460.   431.   809.   774.   213.
   679.   468.   344.   428.   765.   299.   614.   26.    792.   559.
   413.   336.   204.   319.   42.    3.     925.   2.     550.   431.
   141.   534.   301.   576.   344.   899.   100.   508.   409.   23.
   495.   204.   276.   426.   197.   838.   428.   408.   722.   576.
   420.   159.   295.   976.   213.   434.   943.   840.   477.   715.
   862.   19.    572.   252.   314.   777.   33.    502.   639.   932.

--> [Q, U, R] = Householder(A);

--> Q*R
 ans  =

   901.   286.   410.   214.   439.   782.   140.   921.   912.   996.
   395.   251.   11.    689.   76.    55.    115.   944.   444.   158.
   565.   339.   197.   585.   256.   919.   536.   900.   598.   535.
   706.   392.   273.   420.   67.    460.   431.   809.   774.   213.
   679.   468.   344.   428.   765.   299.   614.   26.    792.   559.
   413.   336.   204.   319.   42.    3.     925.   2.     550.   431.
   141.   534.   301.   576.   344.   899.   100.   508.   409.   23.
   495.   204.   276.   426.   197.   838.   428.   408.   722.   576.
   420.   159.   295.   976.   213.   434.   943.   840.   477.   715.
   862.   19.    572.   252.   314.   777.   33.    502.   639.   932.
```

Testando sua ortogonalidade:

```
--> Q'*Q
 ans  =

   1.          1.519D-17   9.769D-17   7.003D-17   1.226D-16  -1.555D-16   2.928D-17  -1.250D-16   1.735D-18   1.527D-16
   1.519D-17   1.          1.334D-16   5.910D-17   1.252D-16   1.266D-17   1.216D-17  -2.076D-17   5.551D-17  -7.806D-17
   9.769D-17   1.334D-16   1.         -6.934D-17   1.116D-16  -4.603D-16   2.620D-16  -2.898D-17  -2.498D-16   4.163D-17
   7.003D-17   5.910D-17  -6.934D-17   1.          5.963D-17   1.324D-16  -1.427D-17   9.375D-17   5.551D-17   2.776D-17
   1.226D-16   1.252D-16   1.116D-16   5.963D-17   1.         -8.392D-17  -1.342D-16  -7.774D-17   9.021D-17   5.551D-17
  -1.555D-16   1.266D-17  -4.603D-16   1.324D-16  -8.392D-17   1.         -2.986D-16   3.388D-17   1.110D-16   0.
   2.928D-17   1.216D-17   2.620D-16  -1.427D-17  -1.342D-16  -2.986D-16   1.          5.000D-18   2.776D-17  -1.388D-17
  -1.250D-16  -2.076D-17  -2.898D-17   9.375D-17  -7.774D-17   3.388D-17   5.000D-18   1.         -5.551D-17  -8.327D-17
   3.330D-17   7.006D-17  -2.517D-16   2.832D-17   8.781D-17   1.647D-16   7.316D-17  -7.422D-17   1.          1.006D-16
   1.570D-16  -7.401D-17   4.647D-17   4.000D-17   3.225D-17  -3.274D-17  -3.215D-17  -8.955D-17   1.006D-16   1.
```

Exemplo 2:

```
--> A
 A  =

   212.   561.   308.   502.   281.   410.   387.   538.   588.   649.
   756.   662.   933.   437.   128.   878.   922.   120.   483.   992.
   1.     726.   215.   270.   778.   114.   948.   226.   224.   50.
   330.   199.   313.   632.   212.   200.   344.   627.   840.   748.
   665.   544.   362.   405.   113.   562.   376.   761.   121.   410.
   628.   232.   292.   918.   686.   590.   734.   49.    286.   608.
   849.   231.   566.   44.    153.   685.   262.   672.   860.   854.
   686.   217.   483.   482.   697.   890.   499.   202.   849.   65.
   878.   883.   332.   264.   841.   504.   264.   391.   526.   828.
   69.    652.   593.   415.   406.   350.   525.   830.   993.   926.

--> [Q, U, R] = Householder(A);

--> Q*R
 ans  =

   212.   561.   308.   502.   281.   410.   387.   538.   588.   649.
   756.   662.   933.   437.   128.   878.   922.   120.   483.   992.
   1.     726.   215.   270.   778.   114.   948.   226.   224.   50.
   330.   199.   313.   632.   212.   200.   344.   627.   840.   748.
   665.   544.   362.   405.   113.   562.   376.   761.   121.   410.
   628.   232.   292.   918.   686.   590.   734.   49.    286.   608.
   849.   231.   566.   44.    153.   685.   262.   672.   860.   854.
   686.   217.   483.   482.   697.   890.   499.   202.   849.   65.
   878.   883.   332.   264.   841.   504.   264.   391.   526.   828.
   69.    652.   593.   415.   406.   350.   525.   830.   993.   926.
```

Testando sua ortogonalidade:

```
--> Q'*Q
 ans  =

   1.           1.440D-16  -2.134D-17   1.032D-16   1.145D-17  -1.719D-17  -4.384D-17  -7.075D-17  -1.145D-16  -4.163D-17
   1.440D-16    1.         -1.080D-16  -9.672D-17   9.106D-17   7.364D-18   2.917D-16  -3.137D-16   8.327D-17   1.388D-16
  -2.134D-17  -1.080D-16   1.          -8.364D-17   5.174D-17  -6.844D-19  -7.882D-17   6.717D-17   8.327D-17  -5.551D-17
   1.032D-16  -9.672D-17  -8.364D-17   1.           9.467D-17  -4.979D-16  -9.241D-17   3.485D-16  -2.706D-16   4.025D-16
   1.145D-17   9.106D-17   5.174D-17   9.467D-17   1.          -1.093D-17   3.440D-17  -6.002D-17  -4.441D-16  -2.776D-16
  -1.719D-17   7.364D-18  -6.844D-19  -4.979D-16  -1.093D-17   1.          -5.115D-17   4.335D-17  -2.359D-16  -1.583D-16
  -4.384D-17   2.917D-16  -7.882D-17  -9.241D-17   3.440D-17  -5.115D-17   1.          -5.716D-17  -2.637D-16   1.943D-16
  -7.075D-17  -3.137D-16   6.717D-17   3.485D-16  -6.002D-17   4.335D-17  -5.716D-17   1.          -3.123D-17  -9.714D-17
  -1.074D-16   6.907D-17   1.280D-16  -3.183D-16  -4.231D-16  -2.346D-16  -2.454D-16  -4.458D-17   1.           3.331D-16
  -6.307D-17   1.222D-16  -8.624D-17   3.936D-16  -3.003D-16  -1.802D-16   1.535D-16  -9.315D-17   3.331D-16   1.
```

Matrizes Não-Quadradas

Exemplo 1: A(m,n), onde $n > m$ Mais colunas do que linhas, então k = m - 1;

```
 A  =

   19.   46.   61.   36.   62.
   41.   57.   32.   30.   52.
   6.    16.   33.   50.   27.
   29.   28.   36.   50.   16.

--> [Q, U, R] = Householder(A)
 Q  =

  -0.351671   0.7882146  -0.2206797   0.4542529
  -0.758869  -0.0351237   0.6077742  -0.2312891
  -0.111054   0.3094431  -0.4389893  -0.836182
  -0.536761  -0.5307815  -0.6238584   0.2023841
 U  =

   0.8220921   0.          0.
   0.4615474  -0.8595497   0.
   0.0675435   0.1423321   0.8061332
   0.3264604  -0.4908319   0.5917341
 R  =

  -54.027771  -76.23857   -68.723917  -67.816976  -72.851424
   0.          24.345028   38.06062    16.255095   46.905332
   0.          0.         -30.958238  -42.853629  -3.9123292
   0.          0.          0.         -22.275462  -3.2021185

--> Q*R
 ans  =

   19.   46.   61.   36.   62.
   41.   57.   32.   30.   52.
   6.    16.   33.   50.   27.
   29.   28.   36.   50.   16.
```

Como esperado, a matriz Q é (m,m), (4,4). U é (m, m-1), (4,3). R tem tamanho (m,n), ou seja, (4,5).

Exemplo 2: A(m,n), onde $m > n$ Mais linhas do que colunas, então k = n.

```
--> A
 A  =

   40.    72.    15.    71.
   66.    35.    115.   88.
   71.    119.   91.    85.
   6.     113.   112.   52.
   103.   42.    49.    18.

--> [Q, U, R] = Householder(A)
 Q  =

  -0.2719013   0.2720543   0.5260868   0.5835269  -0.4845463
  -0.4486372  -0.158366   -0.7370279   0.4756791  -0.0645307
  -0.4826248   0.4191274   0.1694781   0.0484924   0.7485527
  -0.0407852   0.7813992  -0.3491348  -0.3682373  -0.3609129
  -0.7001459  -0.3386064   0.171478   -0.5433926  -0.2654455
 U  =

   0.7974651   0.          0.          0.
   0.2812895  -0.7919367   0.          0.
   0.3025993   0.1994452   0.7945115   0.
   0.0255718   0.4878391   0.58799    -0.755988
   0.4389821  -0.3083356  -0.1517211  -0.6545855
 R  =

  -147.1122  -126.7264  -138.46574  -114.53163
   0.         137.9979   94.93431    75.543318
   0.         0.        -92.145079  -28.169063
   0.         0.         0.          58.482623
   0.         0.         0.          0.
```

```
--> Q*R
 ans  =

   40.    72.    15.    71.
   66.    35.    115.   88.
   71.    119.   91.    85.
   6.     113.   112.   52.
   103.   42.    49.    18.
```

Como esperado, a matriz Q é (m,m), (5,5). U é (m, n), (5,4). R tem tamanho (m,n), ou seja, (5,4).

4.1) Testar as suas funções com as mesmas matrizes usadas nos testes dos itens anteriores. Comparar a precisão dos Métodos.

Para comparar as funções é mais conveniente o uso de matrizes com ordem grande, e podemos conferir sem precisar abrir as matrizes. Criei uma função que faz a a norma dos erros para

$$Q * R - A$$

```
function erro = media_erro_qr(A) //QR - A -> 0
[m,n] = size(A);
erro = abs(norm(A,2))/ (m*n)
endfunction
```

Analogamente, podemos calcular a média dos erros de

$$Q' * Q - I$$

com:

```
function erro = media_erro_qq(A) //Q'Q - I -> 0
[m,n] = size(A);
A = A - eye(m,n);
erro = abs(norm(A),2)/ (m*n)
endfunction
```

Primeiro subtraímos A da matriz identidade para obter os resíduos, e então calculamos a média dividindo sua soma pelo total de índices $m * n$.

ORDEM 50

Erro QR
Exemplo 1:

```
--> [A] = matriz_aleat(50); [Q,R] = qr_GS(A);

--> erro = media_erro_qr(Q, R, A)
 erro  =

   9.751D-14

--> [Q,R] = qr_GSM(A);

--> erro = media_erro_qr(Q, R, A)
 erro  =

   1.053D-13

--> [Q, U, R] = Householder(A);

--> erro = media_erro_qr(Q, R, A)
 erro  =

   1.144D-12
```

Erro QR
Exemplo 2:

```
--> [A] = matriz_aleat(50); [Q,R] = qr_GS(A);

--> erro = media_erro_qr(Q, R, A)
 erro  =

   1.032D-13

--> [Q,R] = qr_GSM(A); erro = media_erro_qr(Q, R, A)
 erro  =

   9.988D-14

--> [Q, U, R] = Householder(A); erro = media_erro_qr(Q, R, A)
 erro  =

   8.796D-13
```

Erro QR
Exemplo 3:

```
--> [A] = matriz_aleat(50); [Q,R] = qr_GS(A);

--> erro = media_erro_qr(Q, R, A)
 erro  =

   1.033D-13

--> [Q,R] = qr_GSM(A); erro = media_erro_qr(Q, R, A)
 erro  =

   9.786D-14

--> [Q, U, R] = Householder(A); erro = media_erro_qr(Q, R, A)
 erro  =

   8.993D-13
```

ORDEM 100

Erro QR
Exemplo 1:

```
--> [A] = matriz_aleat(100); [Q,R] = qr_GS(A);

--> erro = media_erro_qr(Q, R, A)
 erro  =

   4.075D-13

--> [Q,R] = qr_GSM(A); erro = media_erro_qr(Q, R, A)
 erro  =

   4.064D-13

--> [Q, U, R] = Householder(A); erro = media_erro_qr(Q, R, A)
 erro  =

   3.058D-12
```

ORDEM 50

Erro QQ

Exemplo 1:

```
--> [A] = matriz_aleat(50); [Q,R] = qr_GS(A);

--> erro = media_erro_qq(Q)
 erro  =

   1.283D-16

--> [Q,R] = qr_GS(A); erro = media_erro_qq(Q)
 erro  =

   1.283D-16

--> [Q, U, R] = Householder(A); erro = media_erro_qq(Q)
 erro  =

   1.155D-18
```

Exemplo 2:

```
--> [A] = matriz_aleat(50); [Q,R] = qr_GS(A);

--> erro = media_erro_qq(Q)
 erro  =

   1.732D-16

--> [Q,R] = qr_GS(A); erro = media_erro_qq(Q)
 erro  =

   1.732D-16

--> [Q, U, R] = Householder(A); erro = media_erro_qq(Q)
 erro  =

   1.310D-18
```

ORDEM 100

```
--> [A] = matriz_aleat(100); [Q,R] = qr_GS(A);

--> erro = media_erro_qq(Q)
 erro  =

   7.647D-17

--> [Q,R] = qr_GSM(A); erro = media_erro_qq(Q)
 erro  =

   2.485D-18

--> [Q, U, R] = Householder(A); erro = media_erro_qq(Q)
 erro  =

   4.326D-19
```

Todos os exemplos tiveram boa precisão. O método de Householder obtém um Q com maior ortogonalidade, visto que tem menor erro para $Q'Q - I$ mas o método de Gram-Smitch se mostra superior para o produto $Q * R$ que se aproxima mais de A.

4.2) Testar as suas funções com a matriz A = [0,70000 0,70711; 0,70001 0,70711]. Comparar a ortogonalidade das matrizes Q produzidas pelos Métodos.

```
--> A = [0.70000 0.70711; 0.70001 0.70711]
 A  =

   0.7       0.70711
   0.70001   0.70711

--> [Q,R] = qr_GS(A)
 Q  =

   0.7071017   0.7071118
   0.7071118  -0.7071017
 R  =

   0.9899566   1.0000046
   0.          0.0000071

--> Q'*Q
 ans  =

   1.          2.301D-11
   2.301D-11   1.
```


```
--> A = [0.70000 0.70711; 0.70001 0.70711]
 A  =

   0.7       0.70711
   0.70001   0.70711

--> [Q,R] = qr_GS(A)
 Q  =

   0.7071017   0.7071118
   0.7071118  -0.7071017
 R  =

   0.9899566   1.0000046
   0.          0.0000071

--> Q'*Q
 ans  =

   1.          2.301D-11
   2.301D-11   1.
```


Tanto a função GS quando a GSM tiveram o mesmo erro de 11 casas decimais.

```
--> [Q, U, R] = Householder(A)
 Q  =

  -0.7071017  -0.7071118
  -0.7071118   0.7071017
 U  =

   0.9238782
   0.3826867
 R  =

  -0.9899566  -1.0000046
   0.          -0.0000071

--> Q'*Q
 ans  =

   1.   0.
   0.   1.
```

A função de Householder teve o melhor desempenho, alcançando uma ortogonalidade perfeita neste exemplo.

4.3) Seja a matriz

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 7 \\ 4 & 2 & 3 \\ 4 & 2 & 2 \end{bmatrix}$$

Calcule a decomposição QR (reduzida) usando os métodos de Gram-Schmidt, Householder e a função "qr" do Scilab. Compare os três resultados. Comente.
Testando a função de Gram-Schmidt:

```
--> A = [1,2,3;4,5,6;7,8,7;4,2,3;4,2,2]
 A  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.

--> [Q,R] = qr_GS(A)
 Q  =

   0.1010153   0.3161731   0.5419969
   0.404061    0.3533699   0.5161875
   0.7071068   0.3905667  -0.5247906
   0.404061   -0.5579525   0.3871406
   0.404061   -0.5579525  -0.1204438
 R  =

   9.8994949   9.4954339   9.6974644
   0.          3.2919196   3.0129434
   0.          0.          1.9701157

--> Q*R
 ans  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.
```

```
--> Q'*Q
 ans  =

   1.          -1.036D-15   2.961D-17
  -1.036D-15   1.           5.205D-15
   1.573D-17   5.233D-15   1.
```

Testando a função de Gram-Schmidt Modificada:

```
--> A
 A  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.

--> [Q,R] = qr_GSM(A)
 Q  =

   0.1010153   0.3161731   0.5419969
   0.404061    0.3533699   0.5161875
   0.7071068   0.3905667  -0.5247906
   0.404061   -0.5579525   0.3871406
   0.404061   -0.5579525  -0.1204438
 R  =

   9.8994949   9.4954339   9.6974644
   0.          3.2919196   3.0129434
   0.          0.          1.9701157

--> Q*R
 ans  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.
```

```
--> Q'*Q
 ans  =

   1.          -1.036D-15   6.331D-17
  -1.036D-15   1.           1.422D-16
   7.799D-18   1.422D-16   1.
```

Testando a função de Householder:

```
--> A
 A  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.

--> [Q, U, R] = Householder(A)
 Q  =

  -0.1010153  -0.3161731   0.5419969  -0.6842085  -0.3576711
  -0.404061   -0.3533699   0.5161875   0.3280084   0.5812274
  -0.7071068  -0.3905667  -0.5247906   0.0093972  -0.2682612
  -0.404061    0.5579525   0.3871406   0.3655973  -0.4918175
  -0.404061    0.5579525  -0.1204438  -0.5389987   0.4694651
 U  =

   0.741962    0.          0.
   0.2722923   0.7865551   0.
   0.4765114   0.1191972  -0.9800408
   0.2722923  -0.4284409   0.1842055
   0.2722923  -0.4284409  -0.0747553
 R  =

  -9.8994949  -9.4954339  -9.6974644
   0.         -3.2919196  -3.0129434
   0.          0.          1.9701157
   0.          0.          0.
   0.          0.          0.
```

```
--> Q*R
 ans  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.

--> Q'*Q
 ans  =

   1.          6.649D-18   5.672D-17   3.840D-17   6.364D-17
   6.649D-18   1.          5.678D-17  -4.382D-17  -1.492D-16
   5.672D-17   5.678D-17   1.         -1.067D-16  -1.329D-16
   3.840D-17  -4.382D-17  -1.067D-16   1.          1.562D-16
   6.364D-17  -9.371D-17  -1.607D-16   1.007D-16   1.
```

Testando a função qr do Scilab:

```
--> A
 A  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.

--> [Q, R] = qr(A)
 Q  =

  -0.1010153  -0.3161731   0.5419969  -0.6842085  -0.3576711
  -0.404061   -0.3533699   0.5161875   0.3280084   0.5812274
  -0.7071068  -0.3905667  -0.5247906   0.0093972  -0.2682612
  -0.404061    0.5579525   0.3871406   0.3655973  -0.4918175
  -0.404061    0.5579525  -0.1204438  -0.5389987   0.4694651
 R  =

  -9.8994949  -9.4954339  -9.6974644
   0.         -3.2919196  -3.0129434
   0.          0.          1.9701157
   0.          0.          0.
   0.          0.          0.

--> Q*R
 ans  =

   1.   2.   3.
   4.   5.   6.
   7.   8.   7.
   4.   2.   3.
   4.   2.   2.
```

```
--> Q'*Q
 ans  =

   1.          4.407D-17   5.254D-18   1.913D-17   7.260D-17
   4.407D-17   1.          1.353D-17   2.488D-16   1.971D-16
   5.254D-18   1.353D-17   1.         -1.242D-16  -5.542D-17
   1.913D-17   2.488D-16  -1.242D-16   1.         -7.352D-17
   7.260D-17   1.971D-16  -5.542D-17  -7.352D-17   1.
```

Comentários:

Assim como nos testes, a função de Grand-Smitch modificada teve uma leve melhora de desempenho na ortogonalidade de Q em comparação com o método original. Visto que o erro se encontra em cerca de 17 e até 18 casas decimais. Além disso, também mostramos novamente que a função QR do Scilab utiliza o método de Householder, visto que as matrizes obtidas na decomposição são exatamente iguais, e consequentemente sua precisão em ortogonalidade também.

5) ALGORITMO QR para AUTOVALORES

Escreva uma função Scilab function [S] = espectro(A, tol) que calcula os autovalores de uma matriz simétrica A usando o Algoritmo QR. Os autovalores calculados devem ser devolvidos no vetor S. Use como critério de parada a norma infinito da diferença entre dois espectros consecutivos menor do que uma tolerância tol dada (10^{-3}, 10^{-4}, 10^{-5}, ...). Teste a sua função com matrizes simétricas das quais você saiba quais são os autovalores.

Para essa função, vamos usar a função de HouseHolder para encontrar a decomposição QR da nossa matriz A. Depois, vamos ter um A_1, tal que $A_1 = R * Q$. Essa troca ocorre pois:

$$A = QR$$

$$R = Q^T A$$

$$A_1 = Q^T A Q$$

Feito isso, podemos concluir que A_1 e A são semelhantes, ou seja, tem os mesmos autovalores. A partir daí iremos continuar gerando matrizes A com RQ de decomposições QR até obter um R_{k-1} que converge para uma matriz triangular superior, cujo os autovalores serão a diagonal de A_k.

Código:

```
function [S] = espectro(A, tol)

erro = 1;
while erro > tol
[Q, U, R] = Householder(A);
A1 = R*Q;
[Q1, U1, R1] = Householder(A1);
A = R1*Q1;

//definir erro baseado nas diagonais convergindo para os autovalores
erro = abs(norm(diag(A), 'inf') - norm(diag(A1),'inf'));
end
S = diag(A1);
endfunction
```

Para os testes, vou usar uma função de gerar matrizes simétricas aleatórias usada na AP3:

```
function A = Matriz_Simetrica_Aleat(n)
// gera numeros aleatorios entre -n^2 e n^2
A = floor(-((n^2)*rand(n,n,'uniform')) + ((n^2)*rand(n,n,'uniform')))
// define a simetria da matriz
A = tril(A) + triu(A', 1)
endfunction
```

Ordem 5:

```
--> A = Matriz_Simetrica_Aleat(5)
 A  =

  -4.   -8.   -19.   14.   -6.
  -8.   -6.    0.    2.   -23.
  -19.   0.    7.   -2.    4.
   14.   2.   -2.   -3.    5.
  -6.   -23.   4.    5.    7.

--> [S] = espectro(A, 10^-10)
 S  =

  -32.817128
   26.815011
   24.187566
  -15.673790
  -1.5116588

--> spec(A)
 ans  =

  -32.817128
  -15.673790
  -1.5116588
   24.187534
   26.815042
```

```
--> A = Matriz_Simetrica_Aleat(5)
 A  =

   9.    14.  -17.   7.   -3.
   14.   18.  -5.   -2.   -2.
  -17.  -5.   -16.  -2.   -1.
   7.   -2.   -2.   -13.  -11.
  -3.   -2.   -1.   -11.  -15.

--> [S] = espectro(A, 10^-10)
 S  =

   33.215398
  -25.919437
  -24.822361
   6.0781630
  -5.5517638

--> spec(A)
 ans  =

  -25.921430
  -24.820367
  -5.5517638
   6.0781630
   33.215398
```

Com essa tolerância os autovalores já alcançam seus respectivos valores. Note que a função retorna eles de acordo com sua posição na matriz, enquanto o spec() do scilab retorna eles em ordem crescente.

Ordem 10:

```
--> A = Matriz_Simetrica_Aleat(10)
 A  =

  -27.  -47.  -48.   63.  -36.   23.  -48.   56.   53.   9.
  -47.  -43.   40.  -6.    6.    94.  -67.  -5.   -7.   -40.
  -48.   40.   45.  -22.   20.  -78.  -11.  -56.  -13.  -29.
   63.  -6.   -22.  -9.    1.   -18.   27.   19.   53.   50.
  -36.   6.    20.   1.    71.  -39.   39.  -10.  -2.    1.
   23.   94.  -78.  -18.  -39.   66.   13.   9.   -34.  -18.
  -48.  -67.  -11.   27.   39.   13.   60.   2.    7.   -9.
   56.  -5.   -56.   19.  -10.   9.    2.   -71.  -20.   2.
   53.  -7.   -13.   53.  -2.   -34.   7.   -20.  -25.  -23.
   9.   -40.  -29.   50.   1.   -18.  -9.    2.   -23.   41.

--> [S] = espectro(A, 10^-10)
 S  =

   196.44277
  -187.27972
   168.91168
  -121.01915
   108.75053
  -88.796797
  -55.674532
   50.047719
   35.963152
   0.6543457
```

Mas sabendo que a função spec() do scilab ordena os autovalores, podemos compará-los lado a lado ordenando nossos autovalores com a função gsort()

```
--> gsort(S,'g','i')
 ans  =

  -187.27972
  -121.01915
  -88.796797
  -55.674532
   0.6543457
   35.963152
   50.047719
   108.75053
   168.91168
   196.44277

--> spec(A)
 ans  =

  -187.27972
  -121.01915
  -88.796797
  -55.674532
   0.6543457
   35.963152
   50.047719
   108.75053
   168.91168
   196.44277
```

Confirmamos que a precisão da função está boa, sendo igual em todas as casas decimais da visualização.

Ordem 15:

```
--> A = Matriz_Simetrica_Aleat(15); [S] = espectro(A, 10^-10)
 S  =

  -553.04128
   534.84984
  -477.06323
  -406.10277
   399.69374
   368.99207
  -329.65343
  -266.09758
   254.22859
   233.88893
  -218.27204
   152.32933
   128.88835
  -72.684161
   4.0436415

--> spec(A)
 ans  =

  -553.04128
  -477.06323
  -406.10277
  -329.65343
  -266.09758
  -218.27204
  -72.684161
   4.0436415
   128.88835
   152.32933
   233.88893
   254.22859
   368.99207
   399.69374
```

Ordem 50: Para confirmar a precisão em uma ordem maior, sem ter que abrir as matrizes, basta utilizar a mesma lógica dos testes anteriores e pegar o módulo da subtração de ambos para verificar a diferença.

```
--> A = Matriz_Simetrica_Aleat(50); [S] = espectro(A, 10^-10);

--> erro = abs(norm(S,2) - norm(spec(A), 2))
 erro  =

   1.381D-08
```

O erro foi relativamente baixo.