DECOMPOSIÇÃO QR

Relatório da Aula Prática 5

Nome: Felipe Marques Esteves Lamarca
Matrícula: 211708306

Sumário

1 Introdução

Este relatório inicia com a implementação no Scilab de duas versões do Método de Gram-Schmidt para Decomposição QR. Além disso, apresentamos uma implementação do Método de Householder e outra que utiliza a implementação da Decomposição QR para o cálculo de autovalores.

2 Método de Gram-Schmidt Clássico

2.1 Desenvolvimento do algoritmo

A função que implementa o Método de Gram-Schmidt Clássico para encontrar a decomposição QR de uma determinada matriz com colunas linearmente independentes é a seguinte:

```
function[Q, R] = qr_GS(A)
    [m, n] = size(A); // le as dimensoes de A
    Q = zeros(m, n); // inicializa Q e R com zeros e as dimensoes
    de A
    R = zeros(n, n);

    for j = 1:n

        v = A(:, j); // v = a_j

        for i = 1:j-1
            R(i, j) = Q(:, i)' * A(:, j); // r_ij = q_i^T * a_j
            v = v - R(i, j) * Q(:, i); // v = v - v_ij * q_i
        end

        R(j, j) = norm(v, 2); // r_jj = norma 2 de v
        Q(:, j) = v/R(j, j); // q_j = v/r_jj
    end
endfunction
```

2.2 Aplicação em matrizes e teste de precisão

Nosso objetivo é utilizar essa implementação em uma série de testes com matrizes de ordens diferentes e verificar a precisão do método. Vejamos uma primeira matriz simples, 2×2:

$$\begin{bmatrix} 1 & 4 \\ 7 & 12 \end{bmatrix}$$

Não é uma matriz trivial e que normalmente saberíamos a decomposição de forma rápida. As matrizes Q e R resultantes do algoritmo implementado estão apresentadas abaixo e os valores das entradas condizem com os testes realizados na *internet*:

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

 Q  =
   0.1414214   0.9899495
   0.9899495  -0.1414214
```

```
 R  =
   7.0710678   12.445079
   0.          2.2627417
```

Podemos testar a precisão do método pelo cálculo de Q^TQ. Isso faz sentido na medida em que uma das propriedades fundamentais das matrizes ortogonais (ou, nesse caso, "ortonormais") é a de que sua transposta coincide com a sua inversa. Mais precisamente, devemos ter que $Q^TQ = I$, ou pelo menos uma matriz cujas entradas abaixo e acima da diagonal se aproximam muito de zero. Vejamos se isso ocorre nesse caso:

```
-->Q' * Q

 ans  =
   1.          -1.110D-16
  -1.110D-16    1.
```

Trata-se de um resultado bastante razoável. Vejamos se o mesmo ocorre para a seguinte matriz:

$$\begin{bmatrix} 7 & 8 \\ 5 & 9 \\ 3 & 10 \\ 11 & 12 \end{bmatrix}$$

Mais uma vez, o resultado foi testado previamente *online* e obtemos o mesmo *output* com o algoritmo implementado:

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

 Q  =
   0.490098   -0.14498
   0.35007     0.3614094
   0.210042    0.8677989
   0.770154   -0.3086894

 R  =
   14.282857   18.413683   0.   0.
   0.          7.0665603   0.   0.
```

Testemos mais uma vez se o resultado é razoável calculando Q^TQ:

```
-->Q' * Q

 ans  =
   1.          2.776D-17
   2.776D-17   1.
```

Façamos um último teste, dessa vez utilizando uma matriz um pouco maior, dessa vez 5×5:

$$\begin{bmatrix} 3 & 2 & 11 & 1 & 15 \\ 13 & 15 & 6 & 10 & 5 \\ 15 & 6 & 1 & 5 & 9 \\ 6 & 4 & 7 & 8 & 5 \\ 5 & 14 & 15 & 2 & 7 \end{bmatrix}$$

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

 Q  =
   0.1392715   -0.0389762    0.7997826   -0.1242654    0.5692015
   0.6035098    0.3356759   -0.3984768    0.3278384    0.5067896
   0.6963575   -0.5135584   -0.0435445   -0.4380099   -0.23999
   0.278543    -0.0779524    0.388781     0.7459307   -0.4569179
   0.2321192    0.7848463    0.2202664   -0.3589078   -0.3909026

 R  =
   21.540659    17.873176    11.280992    12.34874     14.391389
   0.           12.551875    12.298788    1.6960648    1.5758739
   0.           0.           12.388666    0.1480729    13.098224
   0.           0.           0.           6.2136993   -2.9495784
   0.           0.           0.           0.           3.8911528
```

O resultado foi testado *online* e, de fato, os resultados correspondem à decomposição correta. O último passo antes de passarmos para a próxima seção é conferir se $Q^T Q$ retorna uma matriz próxima da identidade:

```
-->Q' * Q

 ans  =
   1.           3.886D-16   -2.914D-16    8.327D-17    1.721D-15
   3.886D-16    1.          -5.551D-17   -7.216D-16   -1.776D-15
  -2.914D-16   -5.551D-17    1.           5.829D-16    1.846D-15
   8.327D-17   -7.216D-16    5.829D-16    1.          -1.749D-15
   1.721D-15   -1.776D-15    1.846D-15   -1.749D-15    1.
```

De fato, um resultado razoável mais uma vez. Sigamos para uma adaptação do Método de Gram-Schmidt.

3 Método de Gram-Schmidt Modificado

3.1 Desenvolvimento do Algoritmo

Esta função é bastante semelhante à anterior, a menos do fato de que a matriz R é calculada de forma um pouco diferente. Nesse caso, calculamos $r_{ij} = q_i^T v$ ao invés de $r_{ij} = q_i^T a_j$:

```
function[Q, R]= qr_GSM(A)
    [m, n] = size(A) // le as dimensoes de A
    Q = zeros(m, n); // inicializa Q e R com zeros e as dimensoes
    de A
    R = zeros(n, n);

    for j = 1:n
        v = A(:, j); // v = a_j

        for i = 1:j-1
            R(i, j) = Q(:, i)' * v; // r_ij = q_i^T * v
            v = v - R(i, j) * Q(:, i); // v = v - v_ij * q_i
        end

        R(j, j) = norm(v, 2); // r_jj = norma 2 de v
```

```
        Q(:, j) = v/R(j, j); // q_j = v/r_jj
    end
endfunction
```

3.2 Aplicação em matrizes e teste de precisão

Vamos testar se a implementação se comporta de forma correta e, mais do que isso, vamos verificar qual é mais eficiente (ou mais precisa) do ponto de vista dos cálculos. Na prática, será melhor aquele que aproximar o máximo possível de 0 as entradas fora da diagonal. Vejamos para a primeira matriz que utilizamos na seção anterior, isto é:

$$\begin{bmatrix} 1 & 4 \\ 7 & 12 \end{bmatrix}$$

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

 Q  =
   0.1414214   0.9899495
   0.9899495  -0.1414214

 R  =
   7.0710678   12.445079
   0.          2.2627417
```

Ainda que tenhamos atingido o mesmo *output*, considere que desejamos verificar o quão próxima da identidade está a matriz identidade. Para isso, devemos simplesmente calcular a norma entre Q^TQ e I, verificando se há alguma diferença entre a norma calculada para o Q resultante do Método Clássico e para o Q resultante do Método Modificado. No caso dessa matriz, temos o seguinte:

```
-->norm(Q' * Q - eye())     // Metodo Classico
 ans  =
   2.220D-16

-->norm(Q' * Q - eye())     // Metodo Modificado
 ans  =
   2.220D-16
```

Nesse caso, não obtivemos qualquer indicação de que um método é mais preciso do que o outro. Vamos testar para a segunda matriz que utilizamos anteriormente, mais especificamente:

$$\begin{bmatrix} 7 & 8 \\ 5 & 9 \\ 3 & 10 \\ 11 & 12 \end{bmatrix}$$

Também nesse caso, os *outputs* são exatamente os mesmos:

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

 Q  =
   0.490098   -0.14498
```

```
   0.35007    0.3614094
   0.210042   0.8677989
   0.770154  -0.3086894

 R  =
   14.282857   18.413683   0.   0.
   0.          7.0665603   0.   0.
```

Para a precisão, também não há qualquer diferença no cálculo da norma.

```
-->norm(Q' * Q - eye())     // Metodo Classico
 ans  =
   1.176D-16

-->norm(Q' * Q - eye())     // Metodo Modificado
 ans  =
   1.176D-16
```

Vejamos se isso também ocorre na matriz 5×5 que elegemos anteriormente:

$$\begin{bmatrix} 3 & 2 & 11 & 1 & 15 \\ 13 & 15 & 6 & 10 & 5 \\ 15 & 6 & 1 & 5 & 9 \\ 6 & 4 & 7 & 8 & 5 \\ 5 & 14 & 15 & 2 & 7 \end{bmatrix}$$

```
-->[Q, R] = qr_GSM(A)
 Q  =
   0.1392715  -0.0389762   0.7997826  -0.1242654   0.5692015
   0.6035098   0.3356759  -0.3984768   0.3278384   0.5067896
   0.6963575  -0.5135584  -0.0435445  -0.4380099  -0.23999
   0.278543   -0.0779524   0.388781    0.7459307  -0.4569179
   0.2321192   0.7848463   0.2202664  -0.3589078  -0.3909026

 R  =
   21.540659   17.873176   11.280992   12.34874    14.391389
   0.          12.551875   12.298788   1.6960648   1.5758739
   0.          0.          12.388666   0.1480729   13.098224
   0.          0.          0.          6.2136993  -2.9495784
   0.          0.          0.          0.          3.8911528
```

Quanto à precisão, testemos:

```
-->norm(Q' * Q - eye())     // Metodo Classico
 ans  =
   4.184D-15

-->norm(Q' * Q - eye())     // Metodo Modificado
 ans  =
   2.196D-15
```

Nesse caso houve uma diferença considerável: a distância entre a matriz Q^TQ calculada utilizando o Q do Método Modificado é quase duas vezes mais próxima da matriz identidade que a matriz Q^TQ calculada utilizando o Q do Método Clássico.

4 Método de Householder

4.1 Desenvolvimento do Algoritmo

Considere as seguintes funções, documentadas, que executam o algoritmo de Householder para encontrar uma matriz U, primeiro, e construir a matriz Q em seguida:

```
function[U, R] = qr_House(A)
    [m, n] = size(A) // le as dimensoes de A
    k = min(m-1, n) // obtem o numero k de vetores necessarios
    U = zeros(m, k) // inicializa U com zeros

    for j = 1:k
        x = A(j:m, j);

        if x(1) > 0
            x(1) = x(1) + norm(x, 2)        // calcula x - Hx
        else
            x(1) = x(1) - norm(x, 2)
        end

        u = x / norm(x, 2)  // u = (x-Hx) / norma 2 de (x = Hx)
        U(j:m, j) = u       // guarda o vetor u em U
        A(j:m, j:n) = A(j:m, j:n) - 2 * u * (u' * A(j:m, j:n))
    end

    R = triu(A);
endfunction


function[Q] = constroi_Q_House(U)
    [m, k] = size(U)      // le as dimensoes de U
    Q = eye(m, m)       // inicializa Q como uma identidade mxm

    for j = 1:k
        u = U(j:m, j)
        Q = Q - 2 * (Q * u) * u'
    end
endfunction
```

4.2 Aplicação em matrizes e teste de precisão

4.2.1 Aplicação com as matrizes de teste anteriores

Façamos o teste para a primeira matriz definida neste relatório:

```
-->A = [1 4; 7 12]
 A  =
   1.    4.
   7.    12.

-->[U, R] = qr_House(A)
 U  =
   0.755454
   0.6552017

```

```
 R  =
  -7.0710678   -12.445079
   0.          -2.2627417

-->[Q] = constroi_Q_House(U)
 Q  =
  -0.1414214   -0.9899495
  -0.9899495    0.1414214
```

Calculemos a norma:

```
-->norm(Q' * Q - eye())
 ans  =
   2.220D-16
```

Para este caso, não houve qualquer diferença no valor da norma. Passemos para o teste da segunda função:

```
-->A = [7 8; 5 9; 3 10; 11 12]
 A  =

   7.    8.
   5.    9.
   3.    10.
   11.   12.

-->[U, R] = qr_House(A)
 U  =

   0.8631622   0.
   0.2027834   0.8353052
   0.1216701   0.5316829
   0.4461236  -0.139923
 R  =

  -14.282857   -18.413683
   0.          -7.0665603
   0.           0.
   0.           0.

-->[Q] = constroi_Q_House(U)
 Q  =

  -0.490098    0.14498     0.1050639  -0.8530805
  -0.35007    -0.3614094  -0.8635523   0.0333419
  -0.210042   -0.8677989   0.4494363   0.0285404
  -0.770154    0.3086894   0.2030914   0.5199302
```

Calculada a norma, notamos que o resultado foi desvantajoso em relação às versões do método de Gram-Schmidt:

```
-->norm(Q' * Q - eye())
 ans  =
   3.028D-16
```

Vejamos a terceira matriz:

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

```
   0.7547422    0.           0.           0.
   0.3998119    0.8235056    0.           0.
   0.4613215   -0.2973477   -0.9563612    0.
   0.1845286   -0.0415437    0.0793451    0.9428486
   0.1537738    0.4813492    0.2812074   -0.3332215

 R  =
  -21.540659   -17.873176   -11.280992   -12.34874    -14.391389
   0.          -12.551875   -12.298788   -1.6960648   -1.5758739
   0.           0.           12.388666    0.1480729    13.098224
   0.           0.           0.          -6.2136993    2.9495784
   0.           0.           0.           0.          -3.8911528

-->[Q] = constroi_Q_House(U)
 Q  =
  -0.1392715    0.0389762    0.7997826    0.1242654   -0.5692015
  -0.6035098   -0.3356759   -0.3984768   -0.3278384   -0.5067896
  -0.6963575    0.5135584   -0.0435445    0.4380099    0.23999
  -0.278543     0.0779524    0.388781    -0.7459307    0.4569179
  -0.2321192   -0.7848463    0.2202664    0.3589078    0.3909026
```

Para a norma:

```
-->norm(Q' * Q - eye())
 ans  =
   1.289D-15
```

Obtivemos um resultado excelente. Se a norma obtida pelo calculo da norma entre Q^TQ e a identidade, para Q do Método de Gram-Schmidt Modificado, já era baixa, agora ela é ainda menor — isto é, estamos ainda mais próximos da matriz Identidade.

4.2.2 Teste das funções com a matriz $\begin{bmatrix} 0,70000 & 0,70711 \\ 0,70001 & 0,70711 \end{bmatrix}$

Testemos os métodos para a matriz enunciada acima:

```
-->[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.
```

```
-->[Q, R] = qr_GSM(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.
```

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

   0.9238782
   0.3826867
 R  =

  -0.9899566  -1.0000046
   0.         -0.0000071

-->[Q] = constroi_Q_House(U)
 Q  =

  -0.7071017  -0.7071118
  -0.7071118   0.7071017

-->Q'*Q
 ans  =

   1.   0.
   0.   1.
```

Os resultados são extremamente interessantes, em especial para o caso do Método de Householder. Se as duas versões do Método de Gram-Schmidt são efetivas em encontrar uma matriz cujos valores são próximos daqueles esperados para uma matriz identidade, o Método de Householder é extremamente preciso e retorna **exatamente** a identidade.

4.2.3 Teste das funções com a matriz $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 7 \\ 4 & 2 & 3 \\ 4 & 2 & 2 \end{bmatrix}$

```
-->[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] = 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
```

```
-->[U, R] = qr_House(A)
 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] = constroi_Q_House(U)
 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
```

```
-->[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.
```

As versões do Método de Gram-Schmidt obtiveram exatamente o mesmo resultado, e o mesmo vale para os métodos de Householder e a função `qr()` do Scilab. Vamos

calcular as normas com relação à identidade para verificar como cada um dos modelos se comporta:

```
-->norm(Q' * Q - eye())      // GS Classico
 ans  =
   8.057D-15

-->norm(Q' * Q - eye())      // GS Modificado
 ans  =
   1.859D-15

-->norm(Q' * Q - eye())      // Householder
 ans  =
   1.064D-15

-->norm(Q' * Q - eye())      // qr do Scilab
 ans  =
   3.639D-16
```

Veja que a implementação de decomposição QR do próprio Scilab não é a mais precisa — na verdade, fica em terceiro lugar no ranking entre todos os métodos utilizados. O método "vencedor"é o de Householder, com a menor distância entre a matriz $Q^T Q$ da identidade, seguida do Método de Gram-Schmidt modificado. Notamos, desse modo, que os desempenhos das funções implementadas são muito positivos.

5 Algoritmo QR para Autovalores

5.1 Desenvolvimento do Algoritmo

Utilizaremos a seguinte implementação para calcular os autovalores a partir da decomposição QR:

```
function[S] = espectro(A, tol)
    [Q, R] = qr_GS(A) // decompoe A

    current = 0

    while current < tol // itera ate alcancar a tolerancia definida
        d1 = diag(A)
        A = Q' * A * Q
        d2 = diag(A)
        [Q, R] = qr_GS(A)
        current = norm(d1 - d2, %inf)
    end

    S = diag(A)
endfunction
```

5.2 Aplicação em matrizes

Os testes serão realizados com algumas matrizes utilizadas em outro relatório, mais especificamente na Aula Prática 3. Na ocasião, geramos algumas matrizes aleatoria-

mente garantindo a simetria. Como já conhecemos os autovalores dessas matrizes, será simples determinar se nosso método está funcionando corretamente.

Vejamos a primeira matriz:

$$\begin{bmatrix} 149 & 91 & 31 \\ 91 & 89 & 10 \\ 32 & 10 & 17 \end{bmatrix}$$

Obtemos o seguinte resultado:

```
-->[S] = espectro(A, 10^(-5))   // funcao implementada
 S  =
   218.64813
   30.032559
   6.3193109

-->spec(A)      // funcao do Scilab
 ans  =
   219.68933 + 0.i
   29.278105 + 0.i
   6.0325675 + 0.i
```

Os resultados são razoavelmente parecidos — ainda há, no entanto, uma diferença perceptível entre os valores calculados pela função *built-in* do Scilab e a minha implementação.

Vejamos esta outra matriz:

$$\begin{bmatrix} 144 & 66 & 48 \\ 66 & 59 & 65 \\ 48 & 65 & 89 \end{bmatrix}$$

```
-->[S] = espectro(A, 10^(-5))   // funcao implementada
 S  =
   210.90539
   78.509979
   2.584633

-->spec(A)      // funcao do Scilab
 ans  =
   2.4003425
   67.442678
   222.15698
```

Dessa vez os erros são ainda mais claros. Para um dos autovalores, nosso modelo apresenta uma diferença de praticamente 12 unidades em relação a um dos autovalores e, em outro caso, de mais de 11 unidades. Vamos testar se os resultados são mais razoáveis com uma matriz e valores menores:

$$\begin{bmatrix} 2 & 7 \\ 7 & 9 \end{bmatrix}$$

```
-->[S] = espectro(A, 10^(-5))
 S  =
   12.169811
  -1.1698113
```

```
-->spec(A)
 ans  =
  -2.3262379
   13.326238
```

Essa mudança também não surtiu efeitos muito positivos. A diferença, no entanto, é menor entre os autovalores calculados. Na prática, o que verificamos é que o método implementado é funcional para calcular autovalores, mas não apresentou uma precisão tão alta, pelo menos nos exemplos aleatórios que foram apresentados neste relatório.