Aula prática 5

Decomposição QR

Lucas Emanuel Resck Domingues

30 de Maio de 2019

1. A função Scilab **gram_schmidt** foi escrita no arquivo **Functions.sce**. Vejamos como ela se comporta para as seguintes matrizes:

```
--> A = rand(4, 4) * 100;

--> [Qa1, Ra1] = gram_schmidt(A)
 Ra1  =

   85.16904   98.909975   53.564207   44.659712
   0.         102.54677   100.27503   76.836812
   0.         0.          48.660708   20.700703
   0.         0.          0.          29.04781

 Qa1  =

   0.248124    0.4095319   0.6877266   0.545665
   0.887698   -0.2434305  -0.3350021   0.2012648
   0.0002596   0.8283912  -0.5547803   0.0773737
   0.3878488   0.2946069   0.3271461  -0.8097873


--> Qa1'*Qa1
 ans  =

   1.          5.873D-16  -7.987D-16  -1.870D-16
   5.873D-16   1.         -5.448D-16  -1.001D-15
  -7.987D-16  -5.448D-16   1.          2.386D-15
  -1.870D-16  -1.001D-15   2.386D-15   1.
```

Figura 1: Matriz $A_{4\times 4}$ e método de Gram-Schmidt.

Para essa matriz 4×4, a função encontra uma matriz R triangular superior. Além disso, dado que buscamos Q tal que $Q^TQ = I$, temos que $Q^TQ \approx I$, ou seja, isso indica uma boa precisão do método.

```
--> B = rand(6, 5) * 100;

--> [Qbl, Rbl] = gram_schmidt(B)
 Rbl  =

   150.71533   83.068667   106.58475   73.897665   105.67205
   0.          47.299125   29.249008   56.363217  -8.3941876
   0.          0.          39.505961   74.201963   3.6240813
   0.          0.          0.          26.314283  -30.02526
   0.          0.          0.          0.          11.575245

 Qbl  =

   0.1534175   0.1842717   0.2904741  -0.1047928   0.7948613
   0.1436239   0.4087507   0.8122135  -0.0787684  -0.3674989
   0.5861307  -0.2648143  -0.1157704  -0.586152   -0.3031515
   0.4329443  -0.1425279   0.0432697   0.7985833  -0.1671982
   0.2040994   0.83909    -0.4901839   0.0148925  -0.0939038
   0.6190224  -0.0667379  -0.0194624   0.0358147   0.3232122


--> Qbl'*Qbl
 ans  =

   1.          6.438D-16   2.680D-16  -2.062D-15  -3.358D-15
   6.438D-16   1.         -1.376D-15   2.433D-15   6.106D-16
   2.680D-16  -1.376D-15   1.          3.018D-15   4.219D-15
  -2.062D-15   2.433D-15   3.018D-15   1.          1.885D-14
  -3.341D-15   5.880D-16   4.164D-15   1.884D-14   1.
```

Figura 2: Matriz $A_{6\times5}$ e método de Gram-Schmidt.

Para essa matriz 6×5, obtemos uma R triangular superior. Além disso, $Q^TQ \approx I$ indica uma boa precisão do método.

Concluímos que a função **gram_schmidt** cumpre o esperado.

2. A função Scilab **gram_schmidt_modificado** foi escrita no arquivo **Functions.sce**. Vejamos como ela se comporta para as mesmas matrizes do item anterior.

```
--> [Qa2, Ra2] = gram_schmidt_modificado(A)
 Ra2  =

   85.16904    98.909975   53.564207   44.659712
   0.          102.54677   100.27503   76.836812
   0.          0.          48.660708   20.700703
   0.          0.          0.          29.04781

 Qa2  =

   0.248124     0.4095319   0.6877266   0.545665
   0.887698    -0.2434305  -0.3350021   0.2012648
   0.0002596    0.8283912  -0.5547803   0.0773737
   0.3878488    0.2946069   0.3271461  -0.8097873


--> Qa2'*Qa2
 ans  =

   1.           5.873D-16  -8.638D-16  -1.554D-16
   5.873D-16    1.          4.415D-16  -5.361D-16
  -8.638D-16    4.415D-16   1.         -1.366D-16
  -1.554D-16  -5.361D-16  -1.366D-16    1.
```

Figura 3: Matriz $A_{4\times 4}$ e método de Gram-Schmidt modificado.

A função encontra uma matriz R triangular superior e temos que $Q^TQ \approx I$, ou seja, o método tem boa aproximação. Comparando a precisão desse método com o do item anterior:

```
--> norm(Qa1'*Qa1 - eye(4, 4), 'fro')
 ans  =

   4.031D-15


--> norm(Qa2'*Qa2 - eye(4, 4), 'fro')
 ans  =

   1.882D-15
```

Figura 4: Comparação da precisão entre os métodos de Gram-Schmidt e Gram-Schmidit modificado utilizando a norma de Frobenius.

Foi utilizada a norma de Frobenius na matriz $Q^TQ - I$ para o cálculo da precisão do método. Uma matriz ideal teria norma 0. Observe que o método de Gram-Schmidt modificado obteve uma melhor precisão para essa matriz.

```
--> [Qb2, Rb2] = gram_schmidt_modificado(B)
 Rb2  =

   150.71533   83.068667   106.58475   73.897665   105.67205
   0.          47.299125   29.249008   56.363217  -8.3941876
   0.          0.          39.505961   74.201963   3.6240813
   0.          0.          0.          26.314283  -30.02526
   0.          0.          0.          0.          11.575245

 Qb2  =

   0.1534175   0.1842717   0.2904741  -0.1047928   0.7948613
   0.1436239   0.4087507   0.8122135  -0.0787684  -0.3674989
   0.5861307  -0.2648143  -0.1157704  -0.586152   -0.3031515
   0.4329443  -0.1425279   0.0432697   0.7985833  -0.1671982
   0.2040994   0.83909    -0.4901839   0.0148925  -0.0939038
   0.6190224  -0.0667379  -0.0194624   0.0358147   0.3232122


--> Qb2'*Qb2
 ans  =

   1.          6.438D-16   2.657D-16  -2.049D-15  -3.275D-15
   6.438D-16   1.          1.373D-16   0.         -2.220D-16
   2.657D-16   1.373D-16   1.          5.070D-16   1.443D-15
  -2.049D-15   0.          5.070D-16   1.          1.804D-16
  -3.298D-15  -2.572D-16   1.426D-15   1.853D-16   1.
```

Figura 5: Matriz $A_{6\times 5}$ e método de Gram-Schmidt modificado.

A função encontra uma matriz R triangular superior e temos que $Q^T Q \approx I$. Comparando a precisão desse método com o do item anterior:

```
--> norm(Qb1'*Qb1 - eye(5, 5), 'fro')
 ans  =

   2.849D-14


--> norm(Qb2'*Qb2 - eye(5, 5), 'fro')
 ans  =

   6.001D-15
```

Figura 6: Comparação da precisão entre os métodos de Gram-Schmidt e Gram-Schmidit modificado utilizando a norma de Frobenius.

O cálculo da precisão do método é análogo ao anterior. Observe que o método de Gram-Schmidt modificado obteve uma melhor precisão para essa matriz.

Concluímos que a função **gram_schmidt_modificado** cumpre o esperado e tem melhor precisão do que a função **gram_schmidt**.

3. A função Scilab **householder** foi escrita no arquivo **Functions.sce**. Vejamos como ela se comporta para as mesmas matrizes do item anterior.

```
--> [Ua3, Ra3] = householder(A)
 Ra3  =

   85.16904   98.909975   53.564207   44.659712
   0.         102.54677   100.27503   76.836812
   0.         0.         -48.660708  -20.700703
   0.         0.          0.         -29.04781

 Ua3  =

  -0.6131378   0.          0.          0.
   0.7238976  -0.6164086   0.          0.
   0.0002117   0.6720645   0.6947951   0.
   0.3162819   0.4103288   0.7192078   0.
```

Figura 7: Matriz $A_{4\times4}$ e método de Householder.

A função encontra uma matriz R triangular superior e nos retorna uma matriz U com os vetores unitários que geram as matrizes de Householder. Comparando a precisão desse método com a dos itens anteriores:

```
--> norm(Qa1'*Qa1 - eye(4, 4), 'fro')
 ans  =

   4.031D-15


--> norm(Qa2'*Qa2 - eye(4, 4), 'fro')
 ans  =

   1.882D-15


--> norm(Qa3'*Qa3 - eye(4, 4), 'fro')
 ans  =

   1.238D-15
```

Figura 8: Comparação da precisão entre os métodos utilizando a norma de Frobenius.

É possível calcular Q a partir da matriz U e isso foi feito. O cálculo da precisão do método é feito da mesma forma que anteriormente. Observe que o método de Householder tem melhor precisão para essa matriz.

```
--> [Ub3, Rb3] = householder(B)
 Rb3  =

   150.71533   83.068667   106.58475   73.897665   105.67205
   0.          47.299125   29.249008   56.363217  -8.3941876
   0.          0.         -39.505961  -74.201963  -3.6240813
   0.          0.          0.          26.314283  -30.02526
   0.          0.          0.          0.          11.575245
   0.          0.          0.          0.          0.

 Ub3  =

  -0.6506084   0.          0.          0.          0.
   0.1103766  -0.5291443   0.          0.          0.
   0.4504482  -0.1296753   0.6611416   0.          0.
   0.3327227  -0.0456311   0.0888822  -0.2839968   0.
   0.1568527   0.834853    0.7101797   0.914535   -0.6680474
   0.4757258   0.0642564   0.2250257   0.2880479   0.7441187
```

Figura 9: Matriz $A_{6\times 5}$ e método de Householder.

A função retorna as matrizes R e U que esperamos. Comparando a precisão desse método com as dos itens anteriores:

```
--> norm(Qb1'*Qb1 - eye(5, 5), 'fro')
 ans  =

   2.849D-14


--> norm(Qb2'*Qb2 - eye(5, 5), 'fro')
 ans  =

   6.001D-15


--> norm(Qb3'*Qb3 - eye(6, 6), 'fro')
 ans  =

   1.832D-15
```

Figura 10: Comparação da precisão entre os métodos utilizando a norma de Frobenius.

O cálculo da precisão do método é análogo ao anterior. Observe que o método de Householder teve melhor precisão para esta matriz.

Concluímos que a função **householder** cumpre o esperado e tem melhor precisão do que as funções **gram_schmidt** e **gram_schmidt_modificado**.

4. 2) a) Queremos aproximar $s(t) = s_0 + v_0 t + \frac{1}{2}gt^2$. Formemos A e b:

$$A = \begin{bmatrix} 1 & 0,5 & 0,25 \\ 1 & 1 & 1 \\ 1 & 1,5 & 2,25 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}, \; b = \begin{bmatrix} 11 \\ 17 \\ 21 \\ 23 \\ 18 \end{bmatrix}$$

Precisamos encontrar $\mathbf{x}$ tal que $A^T A\mathbf{x} = A^T b$. Porém, agora, podemos encontrá-lo fatorando A em Q e R e resolvendo $R\mathbf{x} = Q^T b$. Utilizando a função **gram_schmidt_modificado** para a fatoração QR e a função **Gaussian_Elimination_4**, da aula prática 1, para a resolução do sistema:

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

--> Gaussian_Elimination_4(R, Q'*b)
 ans  =

   1.9175258
   20.306333
  -4.9720177
```

Figura 11: Fatoração QR e resolução do sistema.

Então obtemos $\mathbf{x} = (s_0, v_0, \frac{g}{2}) \approx (1,92; 20,31; -4,97)$.
Os resultados foram muito próximos daqueles obtidos na aula prática 4, então, neste caso, não seria necessária a fatoração QR através do método de Gram-Schmidt modificado.

b) $s_0 \approx 1,92$m, $v_0 \approx 20,31$m/s e $g \approx -9,94$m/s^2.

c) Resolvendo $0 = s_0 + v_0 t + \frac{1}{2} g t^2$, obtemos $t \approx 4,18$s.

3) a) Vamos aproximar $y = p(t) = ce^{kt}$. Podemos aproximar $\ln y = \ln c + kt$, um modelo linear, então $\mathbf{x} = (\ln c, k)$. Calculamos b aplicando ln nas populações. Realizamos a fatoração QR da matriz A utilizando a função **householder**, encontramos a matriz Q a partir da matriz U e, com a função **Gaussian_Elimination_4**, da aula prática 1, resolvemos $R^T R\mathbf{x} = R^T Q^T b$. Considerando $t = 0$ para 1950, temos:

```
--> Gaussian_Elimination_4(R'*R, R'*Q'*b)
 ans  =

   5.0455774
   0.0121502
```

Figura 12: Resolução do sistema $R^T R\mathbf{x} = R^T Q^T b$.

Calculamos a solução de $R^T R\mathbf{x} = R^T Q^T b$ pois R^T não é invertível.
Novamente, os resultados foram muito próximos daqueles obtidos na aula prática 4. Sendo assim, a fatoração QR pelo método de Householder não é muito necessária neste caso.
Concluímos que $c \approx e^{5,04} \approx 155,33$ e $k \approx 0,01$. Logo, $p(t) \approx 155,33e^{0,01t}$. Portanto, a taxa de crescimento é $p'(t) \approx 1,89e^{0,01t}$.

b) A população estimada dos EUA em 2010 é $p(60) = ce^{k60} \approx 322,01$ milhões de pessoas. Em 2010, a população dos EUA foi de $309,3$ milhões de pessoas. Nosso modelo superestimou a população. Sabemos que um modelo exponencial não se adequa muito bem a um crescimento populacional de uma população já grande.