Fundação Getulio Vargas

Escola de Matemática Aplicada (EMAp)

Nome: Lucas Machado Moschen

Data: 30 de maio de 2019

Professor Antônio Branco

Aula Prática 5

Inicialmente, inicializo as quatro matrizes que trabalharei com os três métodos, como forma de comparação de precisão.

Scilab 6.0.2 Console

```
--> A
 A  =

   1.  -1.   4.
   1.   4.   2.
   1.  -1.   0.
   1.   4.  -2.


--> B
 B  =

  -57.735027  -33.934582   69.949047
   51.208771   33.076221   37.146204
  -99.955773   25.678358   75.643296


--> C
 C  =

  -2.  -2.  -8.  -2.   1.
   8.  -4.   4.  -2.   2.
  -9.  -7.  -7.   8.   4.
   0.   6.   4.  -8.   8.
  -5.  -6.   7.  -6.   0.


--> D
 D  =

   0.   2.
   4.   3.
```

Exercício 1: O primeiro método a ser utilizado é o de Gram-Schmidt. Utilizarei a norma de Frobenius nas matrizes $I - Q' \cdot Q$ para comparar os resultados. Para a matriz A, obtemos um ótimo resultado com o método. Note que a matriz Q encontrada tem colunas de fato ortonormais (ela não é ortogonal, pois não é quadrada), pois o produto dela por sua transposta é a identidade, como esperado. Esse teste foi feito para ver se o código funcionava para matrizes simples. $erro = 0$, nesse exemplo.

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

   2.   3.   2.
   0.   5.  -2.
   0.   0.   4.

 Q  =

   0.5  -0.5   0.5
   0.5   0.5   0.5
   0.5  -0.5  -0.5
   0.5   0.5  -0.5


--> Test = Q'*Q
 Test  =

   1.   0.   0.
   0.   1.   0.
   0.   0.   1.
```

```
--> [Q,R] = gram_schimidt(B)
 R  =

   126.28075   8.6023127  -76.791457
   0.          53.206865   27.401429
   0.          0.          73.12217

 Q  =

  -0.4571958  -0.5638678   0.6877682
   0.4055153   0.5560909   0.7254794
  -0.7915361   0.6105866  -0.0255854


--> Test = Q'*Q
 Test  =

   1.          0.          1.969D-16
   0.          1.         -1.493D-16
   2.073D-16  -1.753D-16   1.
```

Para a matriz B, gerada uniformemente no intervalo [-100, 100), o resultado tem suas diferenças. A matriz que não é "exatamente" ortogonal, ela apresenta um pequeno erro. Computacionalmente, esses valores são tão próximos a zero que podem ser incluídos como. Nesse exemplo, $erro = 4.002 \cdot 10^{-16}$.

```
--> [Q,R] = gram_schimidt(C)
 R  =

   13.190906   4.9276373   5.7615451  -4.0937294  -1.6678157
   0.          10.803629   0.241511   -3.3162549   1.6863215
   0.          0.          12.678575  -7.6990418   1.0412787
   0.          0.          0.          9.2178407  -3.3867439
   0.          0.          0.          0.          8.1743755

 Q  =

  -0.1516196  -0.1159678  -0.5598759  -0.7936541  -0.1421798
   0.6064784  -0.6468665   0.0522119  -0.1367385   0.4385477
  -0.6822882  -0.3367323  -0.2356448   0.2469089   0.5519072
   0.          0.555369    0.3049138  -0.413406    0.6539789
  -0.379049   -0.3824811   0.7316501  -0.345756   -0.2348848


--> Test = Q'*Q
 Test  =

   1.          0.  -1.619D-16   0.          0.
   0.          1.   0.          0.          0.
  -1.619D-16   0.   1.          3.495D-16   0.
   0.          0.   3.495D-16   1.          0.
   0.          0.   0.          0.          1.
```

A matriz C também sofre desse mesmo erro. Ela é gerada uniformemente no intervalo [-10,10), mas eu busco os valores inteiros.

$erro = 4.795 \cdot 10^{-16}$

```
--> [Q,R] = gram_schimidt(D)
 R  =

   4.   3.
   0.   2.

 Q  =

   0.   1.
   1.   0.


--> Test = Q'*Q
 Test  =

   1.   0.
   0.   1.
```

Por fim, uma matriz simples. O objetivo com a matriz D será explanada no exercício 3. Nesse caso, $erro = 0$, como no exemplo 1.Note que a matriz Q é a matriz de permutação das linhas.

Exercício 2: Agora, utilizando o Método de Gram-Schmidt modificado, realizei os mesmos procedimentos do item anterior. Por fim, comparei os erros, segundo a norma de Frobenius novamente. Na matriz A, como era de se esperar, o resultado foi idêntico ao anterior. Digo que era esperado, pois esse método visa reduzir o erro numérico introduzido a cada iteração. Se o erro numérico foi ínfimo sem a modificação, assim será com o modificado. Com a matriz B, já foi observada a melhora que o método modificado traz. Ele não é tão nítido, mas é um ganho. Nesse exemplo, $erro = 3.442 \cdot 10^{-16} < 4 \cdot 10^{-16}$. Mas também noto que menos valores da matriz são diferentes de 0. O erro introduzido, pode ser devido à normalização. Na matriz C, obtive algo não esperado. O erro em relação a ortogonalidade da matriz Q é maior do que sem a modificação, $erro = 6.099 \cdot 10^{-16}$. Não sei exatamente o porquê deste acontecimento. Mas acredito que seja devido à normalização. Para a matriz D, obtemos os mesmo resultados do item anterior, como esperado. O erro é 0 novamente.

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

   2.   3.   2.
   0.   5.  -2.
   0.   0.   4.

 Q  =

   0.5  -0.5   0.5
   0.5   0.5   0.5
   0.5  -0.5  -0.5
   0.5   0.5  -0.5


--> Test = Q'*Q
 Test  =

   1.   0.   0.
   0.   1.   0.
   0.   0.   1.
```

```
--> [Q,R] = gram_schimidt_modificado(B)
 R  =

   126.28075   8.6023127  -76.791457
   0.          53.206865   27.401429
   0.          0.          73.12217

 Q  =

  -0.4571958  -0.5638678   0.6877682
   0.4055153   0.5560909   0.7254794
  -0.7915361   0.6105866  -0.0255854


--> Test = Q'*Q
 Test  =

   1.          0.   2.481D-16
   0.          1.   0.
   2.377D-16   0.   1.
```

```
--> [Q,R] = gram_schimidt_modificado(C)
 R  =

   13.190906   4.9276373   5.7615451  -4.0937294  -1.6678157
   0.          10.803629   0.241511   -3.3162549   1.6863215
   0.          0.          12.678575  -7.6990418   1.0412787
   0.          0.          0.          9.2178407  -3.3867439
   0.          0.          0.          0.          8.1743755

 Q  =

  -0.1516196  -0.1159678  -0.5598759  -0.7936541  -0.1421798
   0.6064784  -0.6468665   0.0522119  -0.1367385   0.4385477
  -0.6822882  -0.3367323  -0.2356448   0.2469089   0.5519072
   0.          0.555369    0.3049138  -0.413406    0.6539789
  -0.379049   -0.3824811   0.7316501  -0.345756   -0.2348848


--> Test = Q'*Q
 Test  =

   1.          0.  -1.619D-16   0.          0.
   0.          1.   0.          0.          0.
  -1.619D-16   0.   1.          3.495D-16   1.523D-16
   0.          0.   3.495D-16   1.          0.
   0.          0.   1.523D-16   0.          1.
```

```
--> [Q,R] = gram_schimidt_modificado(D)
 R  =

   4.   3.
   0.   2.

 Q  =

   0.   1.
   1.   0.
```

Para melhor comparar o ganho do método modificado. Tomei uma matriz aleatória um pouco maior, apliquei ambos os métodos e obtive um ganho com a modificação, que ficou bem claro. Seguem os prints.

```
--> norm(e1)
 ans  =

   2.561D-15


--> norm(e2)
 ans  =

   1.036D-15
```

```
Scilab 6.0.2 Console
 A  =

   27.   25.   10.   39.   33.   30.   17.
   13.   14.   17.   33.   24.   38.   14.
   1.    28.   39.   17.   7.    8.    28.
   21.   21.   20.   10.   1.    23.   31.
   16.   11.   21.   37.   34.   11.   14.
   10.   26.   22.   4.    3.    17.   31.
   20.   4.    22.   19.   34.   36.   22.
   17.   18.   19.   16.   0.    32.   4.
   12.   29.   31.   1.    7.    32.   38.
   4.    36.   32.   21.   20.   10.   9.


--> [Q1,R1] = gram_schimidt(A)
 R1  =

   50.447993   52.33112   54.888209   61.231375   52.509522   75.899947   56.573113
   0.          50.80801   47.465715   11.389809   2.7774731   11.574371   32.307222
   0.          0.         28.271724   7.1624224   14.219565   11.304654   17.095447
   0.          0.         0.          38.893324   30.91246   -4.0432833  -17.804274
   0.          0.         0.          0.          23.712012   0.0262791   5.6479621
   0.          0.         0.          0.          0.          27.424222  -1.7220645
   0.          0.         0.          0.          0.          0.          22.882632

 Q1  =

   0.5352046  -0.0592005  -0.5859717   0.2853946   0.1927759  -0.0788982   0.1096222
   0.2576911   0.010131    0.0840034   0.4243443  -0.1632672   0.6962605   0.3205272
   0.0198224   0.5306776   0.4500254   0.1676036  -0.2992159  -0.1476296   0.2823222
   0.4162703  -0.0154285  -0.0748453  -0.3799367  -0.3376456  -0.3317336   0.1660432
   0.3171583  -0.1101647   0.3120006   0.4268094   0.0009239  -0.4958603   0.044689
   0.1982239   0.3075641  -0.1230525  -0.2766353   0.0859622  -0.0486707   0.2822374
   0.3964479  -0.3296048   0.5618551  -0.1425735   0.4434943   0.1015488  -0.1858713
   0.3369807   0.0071922   0.0057424  -0.1223048  -0.5910753   0.2113475  -0.6061278
   0.2378687   0.3257766   0.087741   -0.4603361   0.2778032   0.2667228   0.1403884
   0.0792896   0.6268832  -0.0745452   0.2452564   0.3194126  -0.0527963  -0.5240859
```

```
--> [Q2,R2] = gram_schimidt_modificado(A)
 R2  =

   50.447993   52.33112    54.888209   61.231375   52.509522   75.899947   56.573113
   0.          50.80801    47.465715   11.389809   2.7774731   11.574371   32.307222
   0.          0.          28.271724   7.1624224   14.219565   11.304654   17.095447
   0.          0.          0.          38.893324   30.91246   -4.0432833  -17.804274
   0.          0.          0.          0.          23.712012   0.0262791   5.6479621
   0.          0.          0.          0.          0.          27.424222  -1.7220645
   0.          0.          0.          0.          0.          0.          22.882632

 Q2  =

   0.5352046  -0.0592005  -0.5859717   0.2853946   0.1927759  -0.0788982   0.1096222
   0.2576911   0.010131    0.0840034   0.4243443  -0.1632672   0.6962605   0.3205272
   0.0198224   0.5306776   0.4500254   0.1676036  -0.2992159  -0.1476296   0.2823222
   0.4162703  -0.0154285  -0.0748453  -0.3799367  -0.3376456  -0.3317336   0.1660432
   0.3171583  -0.1101647   0.3120006   0.4268094   0.0009239  -0.4958603   0.044689
   0.1982239   0.3075641  -0.1230525  -0.2766353   0.0859622  -0.0486707   0.2822374
   0.3964479  -0.3296048   0.5618551  -0.1425735   0.4434943   0.1015488  -0.1858713
   0.3369807   0.0071922   0.0057424  -0.1223048  -0.5910753   0.2113475  -0.6061278
   0.2378687   0.3257766   0.087741   -0.4603361   0.2778032   0.2667228   0.1403884
   0.0792896   0.6268832  -0.0745452   0.2452564   0.3194126  -0.0527963  -0.5240859


--> e1 = eye(size(Q1,2),size(Q1,2)) - Q1'*Q1
 e1  =

   2.220D-16  -2.006D-16   0.          0.          0.         -5.690D-16   0.
  -2.006D-16   0.          7.529D-16   2.183D-16  -3.886D-16   2.776D-16   4.441D-16
   0.          7.529D-16  -2.220D-16  -2.848D-16   0.         -4.718D-16  -1.325D-15
   0.          2.183D-16  -2.848D-16  -2.220D-16   3.608D-16   0.         -3.123D-16
   0.         -3.546D-16   0.          3.634D-16   0.          1.943D-16   8.188D-16
  -5.406D-16   2.810D-16  -4.711D-16   1.234D-16   1.943D-16   0.          1.429D-15
   0.          4.882D-16  -1.329D-15  -3.298D-16   8.384D-16   1.421D-15  -4.441D-16
```

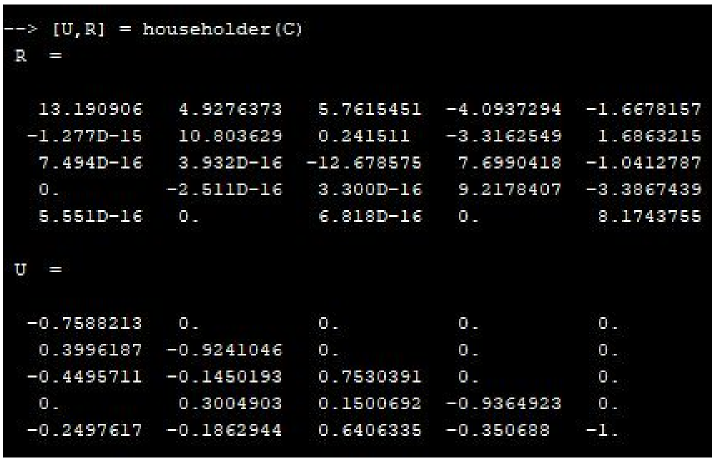
Exercício 3: Na questão 3 como um todo, observei que o erro introduzido é maior. Devo isso a divisão pela norma para inserir na matriz U. Para calcular a matriz R, não fiz isso. A fim de retirar a raiz quadrada da soma dos quadrados do vetor u, eu não o normalizo e para o cálculo de H, faço: $H = I - 2 \cdot \frac{u \cdot u^T}{u^T \cdot u}$ (observei um pequeno ganho de estabilidade).

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

  -2.  -3.          -2.
   0.  -5.           2.
   0.   1.678D-16    4.
   0.  -3.355D-16    0.

 U  =

   0.8660254   0.           0.
   0.2886751   0.9128709    0.
   0.2886751  -0.1825742   -0.8944272
   0.2886751   0.3651484   -0.4472136
```

Para calcular o erro nesse exercício, a partir de U, calculo a matriz Q e continuo como nos exercícios anteriores. Note que com a matriz A, a matriz R tem uma diferença aos exercícios anteriores. Perguntei - me se isso era possível. De fato é, basta trocarmos os sinais das primeira e segunda colunas de Q, e o resultado permanecerá igual. O erro aumentou consideravelmente, se utilizarmos o U. Sugiro uma implementação onde o U de saída não esteja com colunas normalizadas, pois retirar-se-ia a raiz quadrada de jogo. $erro = 18.36 \cdot 10^{-16}$. Para a matriz B, obtive um

```
--> [U,R] = householder(B)
 R  =

   126.28075   8.6023127  -76.791457
  -6.331D-15  -53.206865  -27.401429
   1.157D-14   0.          73.12217

 U  =

  -0.8535795   0.           0.
   0.2375381   0.8364135    0.
  -0.463657    0.5480989   -1.
```

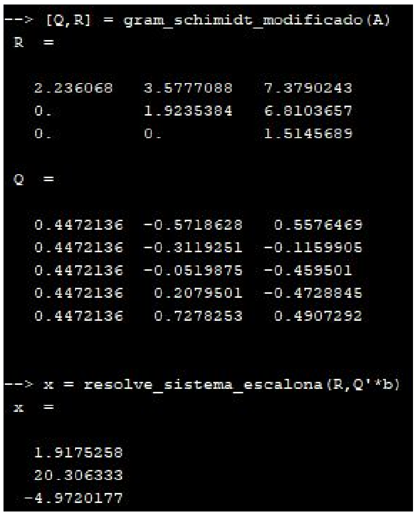
resultado interessante. Apenas a segunda linha de R foi multiplicada por -1. Os valores absolutos são todos muito próximos. Porém o erro da ortogonalidade de Q é $3.188 \cdot 10^{-16}$, um valor um pouco menor aos métodos anteriores. Na matriz C, já vemos os valores onde R deveria ser 0 ficarem, às vezes, um pouco mais distantes. O erro da ortogonalidade também aumenta consideravelmente. $erro = 11.86 \cdot 10^{-16}$. A matriz D foi selecionada para esse exercício. O erro, como era de se esperar, é 0. Mas o interessante é ver um 0 na diagonal. Logo, quando usávamos a função sign(A(j,j)), ela retornava 0 e Hx acabava zerando. Esse problema foi percebido e quis mostrá-lo aqui.



```
--> [U,R] = householder(C)
 R  =

   13.190906   4.9276373   5.7615451  -4.0937294  -1.6678157
  -1.277D-15   10.803629   0.241511   -3.3162549   1.6863215
   7.494D-16   3.932D-16  -12.678575   7.6990418  -1.0412787
   0.         -2.511D-16   3.300D-16   9.2178407  -3.3867439
   5.551D-16   0.          6.818D-16   0.          8.1743755

 U  =

  -0.7588213   0.          0.          0.          0.
   0.3996187  -0.9241046   0.          0.          0.
  -0.4495711  -0.1450193   0.7530391   0.          0.
   0.          0.3004903   0.1500692  -0.9364923   0.
  -0.2497617  -0.1862944   0.6406335  -0.350688   -1.
```

Exercício 4:

2) Inicialmente, tenho as matrizes $A = [1, 0.5, 0.25; 1, 1, 1; 1, 1.5, 2.25; 1, 2, 4; 1, 3, 9]$ e $b = [11; 17; 21; 23; 18]$, onde $Ax = b$. Utilizo o método de Gram-Schmidt modificado para resolver o sistema $Rx = Q^T b$. Para resolver esse sistema, construo uma função (anexada com as outras) que resolve sistemas lineares já escalonados. Chego ao resultado exatamente ao igual ao visto na Aula Prática 4.



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

   2.236068    3.5777088   7.3790243
   0.          1.9235384   6.8103657
   0.          0.          1.5145689

 Q  =

   0.4472136  -0.5718628   0.5576469
   0.4472136  -0.3119251  -0.1159905
   0.4472136  -0.0519875  -0.459501
   0.4472136   0.2079501  -0.4728845
   0.4472136   0.7278253   0.4907292

--> x = resolve_sistema_escalona(R,Q'*b)
 x  =

   1.9175258
   20.306333
  -4.9720177
```

a) A aproximação quadrática é $s(t) = 1.92 + 20.31t - 4.97t^2$

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

c) c) O objeto atinge o chão tem o mesmo significado de $s(t) = 0$. Isto é, queremos encontrar a raiz positiva do polinômio acima mostrado. Aplicando a fórmula de Bháskara no próprio Scilab encontro um valor aproximadamente de $t = 4,17s$ para o objeto tocar no chão.

O resultado exatamente igual fortalece apenas a ideia de que os métodos de resolução equivalentes. A estabilidade não foi percebida, pois a matriz $A^T A$ era uma matriz praticamente escalonada, o que reduz a instabilidade.

3) Lembro que nessa questão, resolve-se o sistema linear $ln(p(t)) = ln(c) + kt$. Assim, terei as matrizes para resolução: $A = [1, 0; 1, 10; 1, 20; 1, 30; 1, 40; 1, 50]$ e $b =$ ln([150;179;203;227;250;281]). Agora, como no exercício anterior, deve-se realizar a decomposição QR e resolver um sistema já escalonado, $Rx = Q^T b$. Note que a função de Householder retorna os vetores u. Para encontrar $b' = Q^T b$, basta aplicar as matrizes Q_1 *e* Q_2 a matriz b. Os resultados foram extremamente similares, novamente.

Scilab 6.0.2 Console

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

  -2.4494897  -61.237244
   0.          41.833001
   0.          2.495D-16
   0.          2.083D-15
   0.          8.152D-16
   0.          1.303D-15

 U  =

   0.8391211   0.
   0.2432595  -0.7698446
   0.2432595   0.034893
   0.2432595   0.1901488
   0.2432595   0.3454046
   0.2432595   0.5006604


--> b_linha = (eye(6,6) - 2*U(:,2)*(U(:,2))')*(eye(6,6) - 2*U(:,1)*(U(:,1))')*b
 b_linha  =

  -13.103134
   0.5082789
   0.036134
   0.0325142
   0.0136613
   0.0151913


--> x = resolve_sistema_escalona(R,b_linha)
 x  =

   5.0455774
   0.0121502
```

Assim encontro que $c = e^{x(1)} \approx 155.3334$ e $k \approx 0.0122$. A taxa de crescimento da população é p'(t) e $p'(t) = kce^{kt} = 1.89e^{0.0122t}$

b) Para estimar a população em 2010, basta calcularmos $p(60) = e^{x(1)}e^{x(2)\cdot 60} \approx 322.012$ milhões.