Aula Prática 1 (ALN)

Raphael F. Levy

March 21, 2021

1 Análise da função *Gaussian_Elimination_1*

Fazendo alguns testes com a função *Gaussian_Elimination_1*, temos:

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

   1.   2.   3.
   6.   4.   1.
   2.   3.   5.

--> b = [1; 2; 3]
 b  =

   1.
   2.
   3.

--> [x,C]=Gaussian_Elimination_1(A,b)
 x  =

   1.6666667
  -2.3333333
   1.3333333
 C  =

   1.   2.       3.
   6.  -8.      -17.
   2.   0.125    1.125

--> A\b
 ans  =

   1.6666667
  -2.3333333
   1.3333333

"OK!"
---------------------------------
--> A = [2 3 -1; -4 0 2; 1 1 3]
 A  =

```

```
   2.    3.   -1.
  -4.    0.    2.
   1.    1.    3.

--> b = [6; -1; 0]
 b  =

   6.
  -1.
   0.

--> [x,C]=Gaussian_Elimination_1(A,b)
 x  =

  -0.047619
   1.8333333
  -0.5952381
 C  =

   2.     3.          -1.
  -2.     6.           0.
   0.5   -0.0833333    3.5

--> A\b
 ans  =

  -0.047619
   1.8333333
  -0.5952381

"OK!"
---------------------------------
--> A = [2 0; 1 -1]
 A  =

   2.    0.
   1.   -1.

--> b = [2; -1]
 b  =

   2.
  -1.

--> [x,C]=Gaussian_Elimination_1(A,b)
 x  =

   1.
   2.
 C  =

   2.     0.
   0.5   -1.

--> A\b
 ans  =

```

```
   1.
   2.

"OK!"
 ---------------------------------
--> A = [1 2; 2 1]
 A  =

   1.   2.
   2.   1.

--> b = [1; 0]
 b  =

   1.
   0.

--> [x,C]=Gaussian_Elimination_1(A,b)
 x  =

  -0.3333333
   0.6666667
 C  =

   1.   2.
   2.  -3.

--> A\b
 ans  =

  -0.3333333
   0.6666667

"OK!"
 ---------------------------------
-->  A = [3 -2; 6 -4]
 A  =

   3.  -2.
   6.  -4.

--> b = [9; 18]
 b  =

   9.
   18.

--> [x,C]=Gaussian_Elimination_1(A,b)
 x  =

   Nan
   Nan
 C  =

   3.  -2.
   2.   0.
```

```
--> A\b
 ans  =

   3.
   0.

"Erro: Deu problema porque o pivoteamento da matriz eliminou a
    linha 2"
---------------------------------
--> A = [2 0; 4 0]
 A  =

   2.   0.
   4.   0.

--> b = [2; -1]
 b  =

   2.
  -1.

--> [x,C]=Gaussian_Elimination_1(A,b)
 x  =

   Nan
  -Inf
 C  =

   2.   0.
   2.   0.

--> A\b
 ans  =

  -3.831D-17
   0.
```

Erro: A multiplicação $Ax = b$ leva a um resultado impossível, já que $2x = 2$ e simultaneamente $4x = -1$, e não é possível que isso seja válido para um único x. Como a segunda coluna é composta por zeros, y pode tomar qualquer valor

2 Testando a função *Gaussian_Elimination_1* com *A*1 e *b*1

Usando a função inicial dada, temos o seguinte resultado para as entradas A1 e b1:

```
--> A1 = [1 -2 5 0; 2 -4 1 3; -1 1 0 2; 0 3 3 1]
 A1  =

   1.  -2.   5.   0.
   2.  -4.   1.   3.
  -1.   1.   0.   2.
   0.   3.   3.   1.

```

```
--> b1 = [1; 0; 0; 0]
 b1  =

   1.
   0.
   0.
   0.

--> [x,C]=Gaussian_Elimination_1(A1,b1)
 x  =

   Nan
   Nan
   Nan
   Nan
 C  =

   1.   -2.     5.     0.
   2.    0.    -9.     3.
  -1.   -Inf   -Inf    Inf
   0.    Inf    Nan    Nan

--> A1\b1
 ans  =

  -0.3247863
  -0.1709402
   0.1965812
  -0.0769231
```

Erro: Fazendo o pivoteamento da matriz $A1$, é possível ver que o pivô que deveria "aparecer" na segunda linha será 0, e a função *Gaussian_Elimination_1* não consegue trabalhar com um pivô 0.

3 Testando a *Gaussian_Elimination_2* com $A1$ e $b1$ e $A2$ e $b2$

Utilizando a função *Gaussian_Elimination_2* desenvolvida para trocar as linhas da matriz caso o elemento na posição (j,j) seja nulo, temos:

```

--> A1=[1 -2 5 0; 2 -4 1 3; -1 1 0 2; 0 3 3 1]
 A1  =

   1.   -2.    5.    0.
   2.   -4.    1.    3.
  -1.    1.    0.    2.
   0.    3.    3.    1.

--> b1=[1; 0; 0; 0]
 b1  =

   1.
   0.
   0.
```

```
   0.

--> [x,C]=Gaussian_Elimination_2(A1,b1)
 x  =

  -0.3247863
  -0.1709402
   0.1965812
  -0.0769231
 C  =

   1.  -2.   5.   0.
  -1.  -1.   5.   2.
   2.   0.  -9.   3.
   0.  -3.  -2.   13.

--> A1\b1
 ans  =

  -0.3247863
  -0.1709402
   0.1965812
  -0.0769231

"OK!"
-----------------------------------
--> A2=[0 10^-20 1; 10^-20 1 1; 1 2 1]
 A2  =

   0.           1.000D-20   1.
   1.000D-20    1.          1.
   1.           2.          1.

--> b2=[1; 0; 0]
 b2  =

   1.
   0.
   0.

--> [x,C]=Gaussian_Elimination_2(A2,b2)
 x  =

  -1.000D+20
   0.
   1.
 C  =

   1.000D-20    1.          1.
   0.           1.000D-20   1.
   1.000D+20   -1.000D+40   1.000D+40

--> A2\b2
 ans  =

   1.
  -1.
```

```
   1.
```

Erro: A função não devolveu o resultado esperado porque haverá uma troca entre a primeira coluna e a segunda, dado que $A2_{1,1}$ é 0, porém o pivô substituto não é o maior encontrado na matriz, e é com ele que a eliminação gausseana é corretamente feita, portanto o resultado foi diferente do esperado.

4 Testando a *Gaussian_Elimination_3* com $A2$ e $b2$ e $A3$ e $b3$

Utilizando a função *Gaussian_Elimination_3* desenvolvida para trocar as linhas da matriz caso o elemento na posição (j,j) seja nulo, temos:

```

--> A2=[0 10^-20 1; 10^-20 1 1; 1 2 1]
 A2  =

   0.           1.000D-20   1.
   1.000D-20    1.          1.
   1.           2.          1.

--> b2=[1; 0; 0]
 b2  =

   1.
   0.
   0.

--> [x,C]=Gaussian_Elimination_3(A2,b2)
 x  =

   1.
  -1.
   1.
 C  =

   1.           2.          1.
   1.000D-20    1.          1.
   0.           1.000D-20   1.


--> A2\b2
 ans  =

   1.
  -1.
   1.

"Ok!"
---------------------------------
--> A3=[10^-20 10^-20 1; 10^-20 1 1; 1 2 1]
 A3  =

   1.000D-20   1.000D-20   1.
```

```
   1.000D-20   1.           1.
   1.          2.           1.

--> b3=b2
 b3  =

   1.
   0.
   0.

--> [x,C]=Gaussian_Elimination_3(A3,b3)
 x  =

   0.
  -1.
   1.
 C  =

   1.000D-20   1.000D-20   1.
   1.          1.          0.
   1.000D+20   1.         -1.000D+20

--> A3\b3
 ans  =

   1.
  -1.
   1.
```

Erro: Apesar de ter pivôs não nulos, nenhum dos dois primeiros é o maior em módulo, já que $10^{-20} < 1$. Com isso, a *Gaussian_Elimination_3* não fez trocas de linhas já que não há um pivô nulo,e como o maior pivô está na última linha, a eliminação não foi feita corretamente.

5 Testando a *Gaussian_Elimination_4* com $A3$ e $b3$

Utilizando a função *Gaussian_Elimination_4* desenvolvida para sempre escolher a linha com o maior pivô, temos:

```

--> A3=[10^-20 10^-20 1; 10^-20 1 1; 1 2 1]
 A3  =

   1.000D-20   1.000D-20   1.
   1.000D-20   1.          1.
   1.          2.          1.

--> b3=b2
 b3  =

   1.
   0.
   0.
```

```
 
--> [x,C,P]=Gaussian_Elimination_4(A3,b3)
 x  =

   1.
  -1.
   1.
 C  =

   1.           2.           1.
   1.000D-20    1.           1.
   1.000D-20   -1.000D-20    1.
 P  =

   0.    0.    1.
   0.    1.    0.
   1.    0.    0.

"Ok!"
```

6 Testando a função *Resolve_com_LU*

Infelizmente a função *Resolve_com_LU* não funcionou corretamente para que os testes pedidos pudessem ser feitos, visto que ela está apenas retornando a matriz B passada.

```

--> A3=[10^-20 10^-20 1; 10^-20 1 1; 1 2 1]
 A3  =

   1.000D-20    1.000D-20    1.
   1.000D-20    1.           1.
   1.           2.           1.

--> b3=b2
 b3  =

   1.
   0.
   0.

--> [x,C1,P1]=Gaussian_Elimination_4(A3,b3)
 x  =

   1.
  -1.
   1.
 C1  =

   1.           2.           1.
   1.000D-20    1.           1.
   1.000D-20   -1.000D-20    1.
 P1  =

   0.    0.    1.
```

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

--> B1 = [2 4 -1 5; 0 1 0 3; 2 2 -1 1; 0 1 1 5]
 B1  =

   2.    4.  -1.    5.
   0.    1.   0.    3.
   2.    2.  -1.    1.
   0.    1.   1.    5.

--> [X]=Resolve_com_LU(C1,B1,P1)
 X  =

   2.    4.  -1.    5.
   0.    1.   0.    3.
   2.    2.  -1.    1.
   0.    1.   1.    5.
---------------------------------
--> A2=[0 10^-20 1; 10^-20 1 1; 1 2 1]
 A2  =

   0   1.000D-20   1.
   1.000D-20   1.          1.
   1.          2.          1.

--> b2=[1; 0; 0]
 b2  =

   1.
   0.
   0.

--> [x,C2,P2]=Gaussian_Elimination_4(A2,b2)
 x  =

   1.
  -1.
   1.
 C2  =

   1.          2.          1.
   1.000D-20   1.          1.
   0.          1.000D-20   1.
 P2  =

   0.    0.    1.
   0.    1.    0.
   1.    0.    0.

--> B2=[1 1 2; 1 -1 0; 1 0 1]
 B2  =

   1.    1.    2.
   1.   -1.    0.
   1.    0.    1.
```

```
--> [X]=Resolve_com_LU(Cb,B2,P2)
 X  =

   1.    1.    2.
   1.   -1.    0.
   1.    0.    1.
```