Relatório da Aula Prática 1 de ALN

Aluno: Sillas Rocha da Costa

Primeira Questão

Ao implementar o algoritmo base fornecido no arquivo da aula prática, no arquivo `Gaussian_Elimination_1.sci` foi possível observar os resultados esperados com testes simples, tanto as matrizes A e vetores b de entrada podem ser observados no arquivo da questão 1 (quest_1.sci), a seguir estão as saídas observadas:

- ```
  --> A_1 = [3 -2; 9 -4];

  --> b_1 = [1; 7];

  --> [x_1, C_1] = Gaussian_Elimination_1(A_1, b_1)
   x_1  =

     1.6666667
     2.
   C_1  =

     3.  -2.
     3.   2.
  ```
- ```
  --> A_2 = [1 -1 1; 0 2 3; 0 0 4];

  --> b_2 = [6; 8; 8];

  --> [x_2, C_2] = Gaussian_Elimination_1(A_2, b_2)
   x_2  =

     5.
     1.
     2.
   C_2  =

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

- ```
  --> A_3 = [1 -1; 2 1];

  --> b_3 = [4; 8];

  --> [x_3, C_3 = Gaussian_Elimination_1(A_3, b_3)
    >

  --> [x_3, C_3] = Gaussian_Elimination_1(A_3, b_3)
   x_3  =

     4.
     0.
   C_3  =

     1.  -1.
     2.   3.
  ```
- ```
  --> A_4 = [2 3; 4 -6];

  --> b_4 = [-2 12];

  --> b_4 = [-2; 12];

  --> [x_4, C_4] = Gaussian_Elimination_1(A_4, b_4)
   x_4  =

     1.
    -1.3333333
   C_4  =

     2.   3.
     2.  -12.
  ```

Segunda Questão

Na segunda questão foi possível verificar que a implementação ainda não estava perfeita, pois apresentava problemas na decomposição com pivôs zero, possível verificar no arquivo da [questão 2 (quest_2.sci)](quest_2.sci):

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

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

  --> [x1, C1] = Gaussian_Elimination_1(A1, b1)
   x1  =

     Nan
     Nan
     Nan
     Nan
   C1  =

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

Terceira Questão

A terceira, no arquivo questão 3 (quest_3.sci), ao aprimorar o algoritmo no arquivo `Gaussian_Elimination_2.sci` com a seguinte adição no algoritmo:

```
for j=1:(n-1)

    //O pivô está na posição (j,j)
    // Verifica se o pivô é 0

    if C(j,j) == 0 then
        // Se for então verifica outras linhas para substituir com um pivô
diferente de 0

        vec = C(j+1:n, j);
        k = j + find(vec,1);

        C([j k],:) = C([k j],:);
        P([j k],:) = P([k j],:);
    end
```

Obtemos estes resultados:
```

-

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

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

--> [x1, P1, C1] = Gaussian_Elimination_2(A1, b1)
 x1  =

  -0.3247863
  -0.1709402
   0.1965812
  -0.0769231
 P1  =

   1.   0.   0.   0.
   0.   0.   1.   0.
   0.   1.   0.   0.
   0.   0.   0.   1.
 C1  =

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

-

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

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

--> [x2, P1, C2] = Gaussian_Elimination_2(A2, b2)
 x2  =

   0.0076336
  -0.0839695
   0.1603053
 P1  =

   0.   1.   0.
   1.   0.   0.
   0.   0.   1.
 C2  =

  -10.    1.      1.
   0.   -10.      1.
  -0.1  -0.21    1.31
```

Quarta Questão

Já na [questão 4 (quest_4.sci)](quest_4.sci), ao aprimorar o algoritmo para que ele substitua pelo pivô de maior valor, chemos na `Gaussian_Elimination_3` com as seguintes alterações:

```
for j=1:(n-1)
    //O pivô está na posição (j,j)
    // Verifica se o pivô é 0
    if C(j,j) == 0 then
        // Se for então verifica outras linhas para substituir com um pivô
diferente de 0
       [num, index] = max(abs(C(j+1:n,j)));
       k = j + index

        C([j k],:) = C([k j],:);
        P([j k],:) = P([k j],:);
    end
```

E chegamos nos resultados:

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

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

  --> [x2, P2, C2] = Gaussian_Elimination_3(A2, b2)
   x2  =

     0.0076336
    -0.0839695
     0.1603053
   P2  =

     0.   1.   0.
     1.   0.   0.
     0.   0.   1.
   C2  =

    -10.    1.      1.
     0.    -10.     1.
    -0.1   -0.21    1.31
  ```

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

  --> b3=b2;

  --> [x3, P3, C3] = Gaussian_Elimination_3(A3, b3)
   x3  =

     0.
    -1.
     1.
   P3  =

     1.   0.   0.
     0.   1.   0.
     0.   0.   1.
   C3  =

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

Quinta Questão

Na questão 5 (quest_5.sci) chegamos na implementação final da Eliminação Gaussiana, a `Gaussian_Elimination_4`, onde a substituição pelo maior pivô é o primeiro passo mesmo que o pivô seja não nulo:

```
for j=1:(n-1)
    //O pivô está na posição (j,j)
    // Verifica se o pivô é 0

     [num, index] = max(abs(C(j:n,j)));
     k = j + index - 1;

     C([j k],:) = C([k j],:);
     P([j k],:) = P([k j],:);
```

Para obter os resultados seguintes:

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

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

  --> [x2, P2, C2] = Gaussian_Elimination_4(A2, b2)
   x2  =

     0.0076336
    -0.0839695
     0.1603053
   P2  =

     0.   1.   0.
     1.   0.   0.
     0.   0.   1.
   C2  =

    -10.   1.     1.
     0.   -10.    1.
    -0.1  -0.21   1.31
  ```
  ```

-

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

--> b3=[1; 0; 0];

--> [x3, P3, C3] = Gaussian_Elimination_4(A3, b3)
 x3  =

   1.
  -1.
   1.
 P3  =

   0.   0.   1.
   0.   1.   0.
   1.   0.   0.
 C3  =

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

Sexta Questão

Por fim, na [questão 6 (quest_6.sci)](), foi necessário implementar a função [Resolve_com_LU]() sem uma função de base, entretando, a implementação acaba sendo bem simples e se resumindo a apenas:

```
Y(1,:)=B(1,:);
for i=2:n
    Y(i,:)=B(i,:)-C(i,1:i-1)*Y(1:i-1,:);
end

X(n,:)=Y(n,:)/C(n,n);
for i=n-1:-1:1
    X(i,:)=(Y(i,:)-C(i,i+1:n)*X(i+1:n,:))/C(i,i);
end
```

Assim, com a implementação preparada, podemos verificar ela em ação:

- ```
  --> X1=Resolve_com_LU(C1, P1, B1)
   X1  =

    -2.034188   -1.9316239   1.4529915   0.8119658
    -0.6495726  -0.7008547   0.6068376   0.4273504
     0.5470085   0.9059829  -0.2478632   1.008547
     0.3076923   0.3846154  -0.0769231   0.6923077


  --> X2=Resolve_com_LU(C2, P2, B2)
   X2  =

     0.   3.   3.
     0.  -2.  -2.
     1.   1.   2.


  --> X3=Resolve_com_LU(C3, P3, B3)
   X3  =

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