

Lista de Exercícios – Métodos Numéricos para Engenharia TC Prof^a Polliana Cândida Oliveira Martins 2020/1

Fazer os exercícios que seguem manualmente. Organizar cada procedimento numérico e destacar o resultado.

1ª QUESTÃO: Considere o sistema de duas equações lineares a seguir:

$$0,0003x_1 + 1,566x_2 = 1,569$$

 $0,3454x_1 - 2,436x_2 = 1,018$

- (a) Resolva o sistema usando o método de eliminação de Gauss arredondando em quatro algarismos significativos.
- (b) Troque a ordem das equações e resolva o sistema com o método de eliminação de Gauss arredondando em quatro algarismos significativos.

Verifique as respostas substituindo a solução de volta nas equações. Teça comentários em relação aos resultados obtidos.

2ª QUESTÃO: Determine a inversa da matriz A abaixo usando o método de Gauss Jordan.

$$A = \begin{bmatrix} -4/5 & -3/5 & -2/5 \\ A = \begin{bmatrix} -3/5 & -6/5 & -4/5 \end{bmatrix} \\ -2/5 & -4/5 & -6/5 \end{bmatrix}$$

3ª QUESTÃO: Considere o sistema no qual $[A]{x}={b}$ e onde:

$$A = \begin{bmatrix} 2 & 3 & 1 & 5 \\ 1 & 3.5 & 1 & 7.5 \\ 1.4 & 2.7 & 5.5 & 12 \\ -2 & 1 & 3 & 28 \end{bmatrix}$$
 $b = \{11\ 13\ 21.6\ 30\}^T$

- a) Resolva o sistema linear utilizando o Método de Gauss;
- Resolva o sistema linear utilizando decomposição LU utilizando as Matrizes L e U oriundas do Método de Gauss.

Fazer os exercícios que seguem utilizando Linguagens de programação Matlab® ou Octave®. Anexar todas as rotinas programadas para obter os resultados.

5ª QUESTÃO Resolva o sistema linear da Questão 3 utilizando fatoração LU pelo Método de Crout.

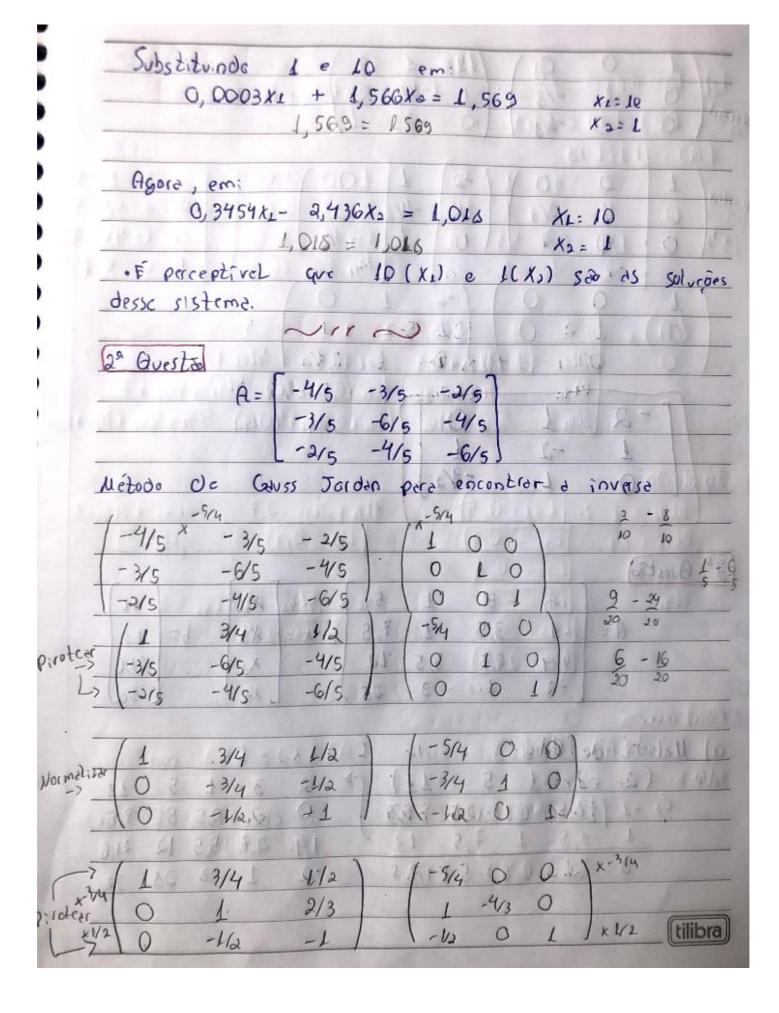
6ª QUESTÃO Considere o sistema abaixo.

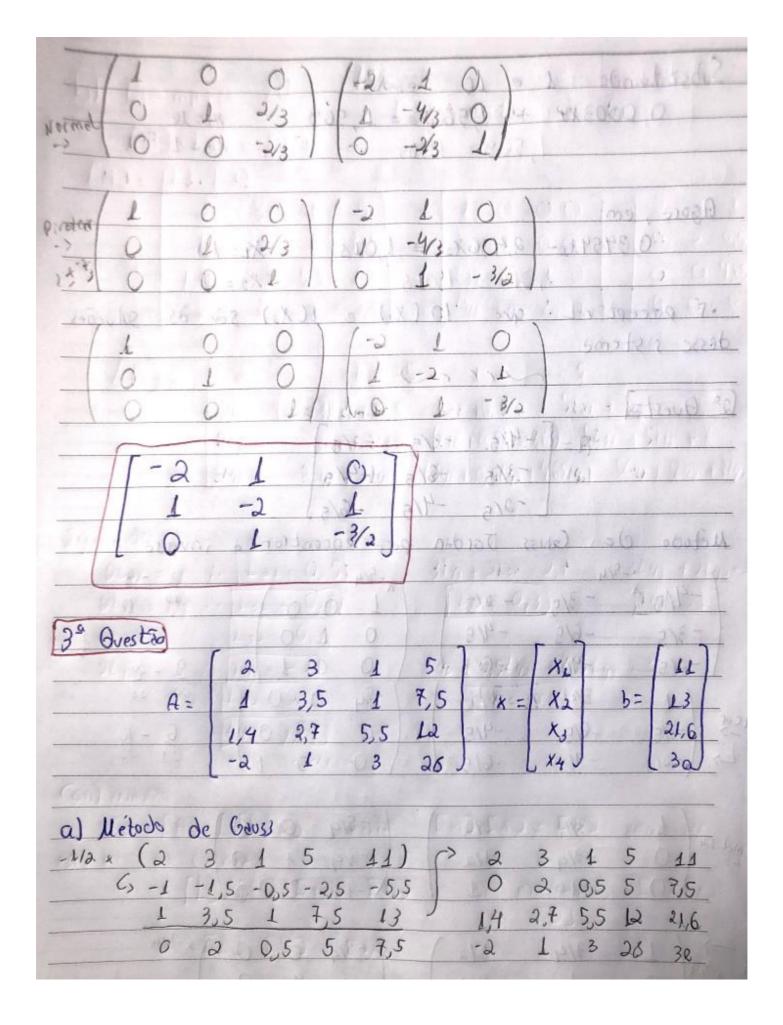
$$2x_1 - 6x_2 - x_3 = -38$$
$$-3x_1 - x_2 + 7x_3 = -34$$
$$-8x_1 + x_2 - 2x_3 = -20$$

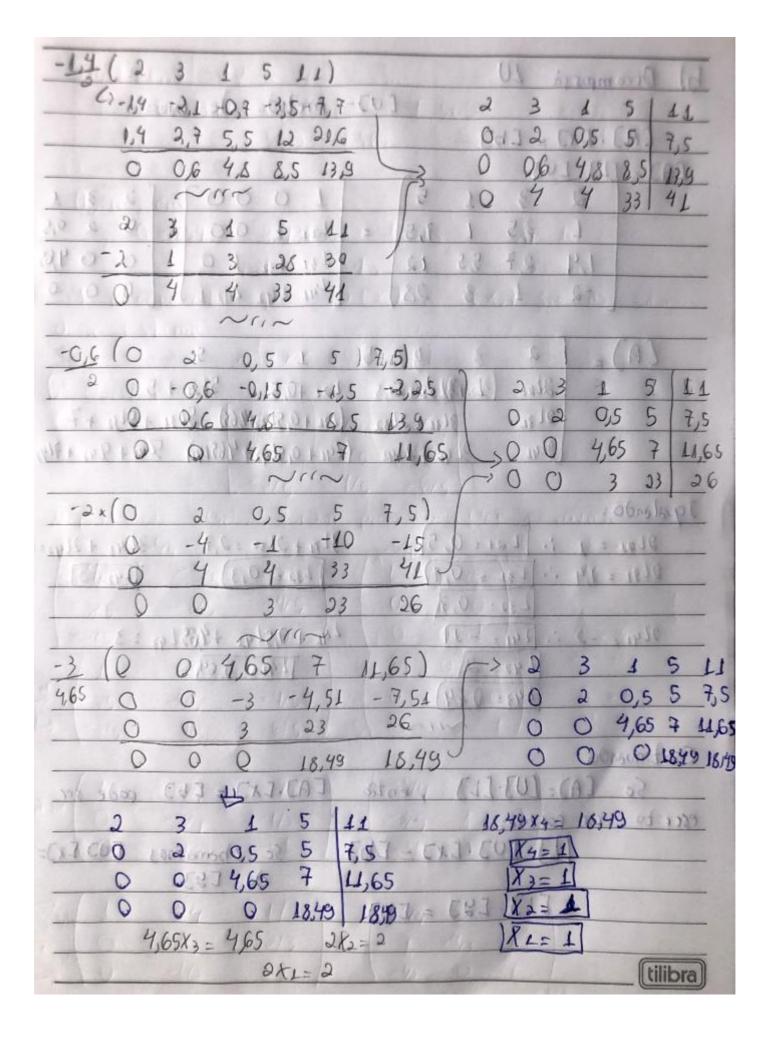
- a) Utilize o método de Gauss Siedel para encontrar a solução do sistema;
- b) Utilize o método de Jacobi para encontrar a solução do sistema;
- c) Compare os resultados obtidos.

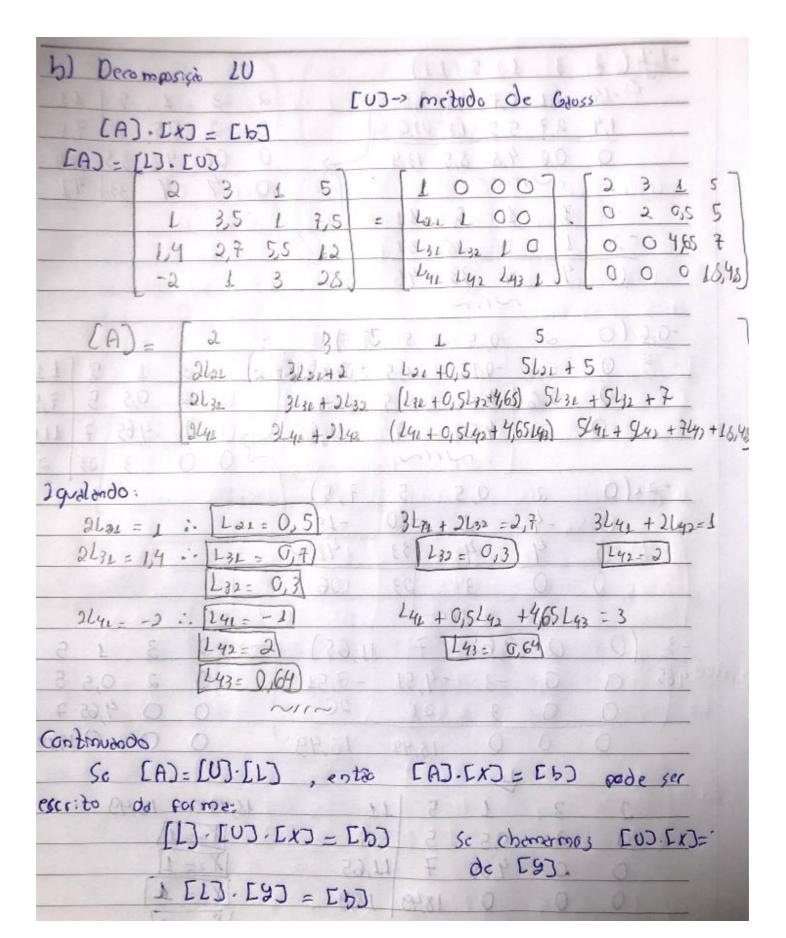
Aluno: Roberto Gabriel Mangabeira Santana **Matrícula:** 190019620

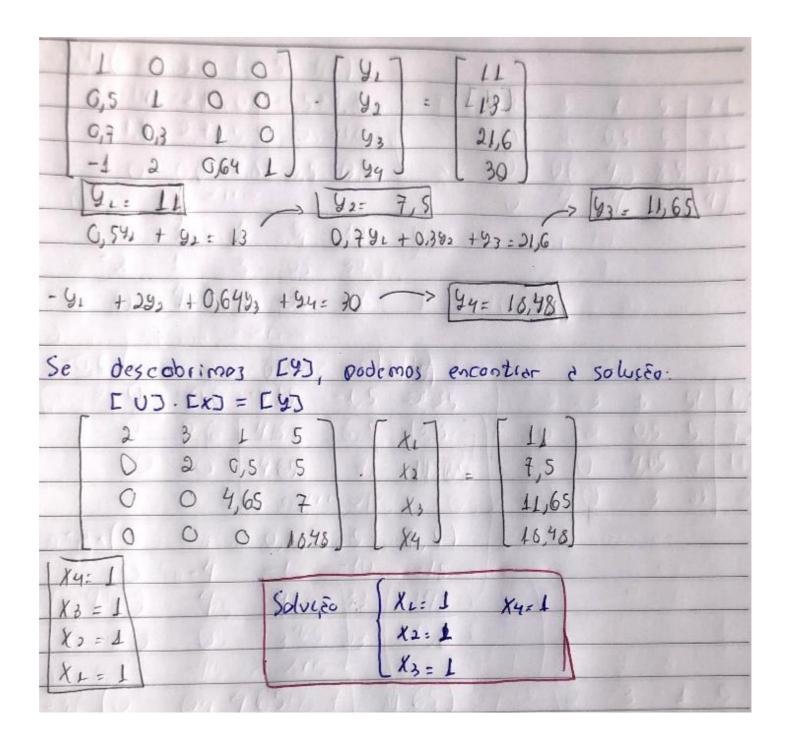
Listo 03 - Roberto G. M. Santana
L' questão
$0,0003 \times 1 + 1,566 \times 2 = 1,569$
0,3454x1 + (-) 2,436x0 = LO18
2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
a) Mitodo de Garss
$\frac{-0.3454.}{0.0003} \left(\frac{0.0003 \times 1.1.}{0.0003 \times 1.1.} + 1.566 \times 1.1. \right) - 1.569. \left(\frac{-0.3459}{0.0003} \right)$
- 0,3459 XL - 1802,388X = -1806,442
0,34548, - 2,436x= 1,018
OX, - 1605, 424 = -1805, 424
TENTALEY WAS A LOUR IN BEAUTIFUL TO
0,0003×1 + 1,566×1= 1,569
OX1 + X21 = 1
[Xa=1]
XL=40
The state of the s
b) 0,3454XL - 2,436X2 = 1,018
0,000311 + 1,5601 = 1,569
$\frac{-0.0003}{0.3454}\left(\begin{array}{c} 0.3454 \chi_{L} & -2.436 \chi_{2} \\ \hline 0.3454 \end{array}\right) = 1.018\left(\begin{array}{c} -0.0003 \\ \hline 0.3454 \end{array}\right)$
-0,0003 kz + 0,0021 xz = +0,000 8
0,0003x1 + 1,566x2 = 1,569
Oxe + 1,5681 x2 = 1,5682
The second secon
0,3454X2 - 2,436 Ks = 1,018 X2= 1,0000
OX2 + 1,5681X2 = 1,5682 XL=10 /











5ª Questão

```
[A] \times [x] = [b]
```

Usando decomposição por LU, temos:

 $[A] = [L] \times [U]$, onde L é uma matriz triangular inferior e U é uma matriz triangular superior. Usando o método de resolução de Crout, é possível descobrir [L] e [U], e assim descobrir [x].

- Lógica programada, no Octave, de Crout para a resolução desse sistema linear:

```
<unnamed> ☑ Lista3Exer5_Crout.m ☑ Questao5Lista2.m ☑
  1 %Metodo de Crout para resolucao de sistema linear
  2 clear all; close all; clc;
  4 A = [2, 3, 1, 5; 1, 3.5, 1, 7.5; 1.4, 2.7, 5.5, 12; -2, 1, 3, 28];
     b = [11, 13, 21.6, 30]';
  7 function [somatorio] = somatorio(final, L, U, i, j)
  8 -somatorio = 0;
  9 for inicio = 1 : final
 10
11
         somatorio += L(i, inicio) *U(inicio, j);
       endfor
 12 endfunction
 13 %Primeira coluna de L e diagonal de U
 14 - for i = 1 : 4
 15 L(i, 1) = A(i, 1);
16 U(i, i) = 1;
 17 endfor
 18
 19 %Primeira Linha de U
 20 \bigcirc for j = 1 : 4
 21 \text{ T} \text{ U}(1, j) = A(1, j)/L(1, 1);
 22 endfor
 23
 24 %Demais linhas de L e U;
 25 □ for i = 2 : 4

26 □ for j = 2 : 4

27 □ if i >= j

28 □ L(i, j) =
            L(i, j) = A(i, j) - somatorio(j - 1, L, U, i, j);
 29
          else
            U(i, j) = (A(i, j) - somatorio(i - 1, L, U, i, j))/L(i, i);
 30
 31
          endif
 32 endfor
 33 endfor
 34 fprintf("Solucao do Sistema: ")
 35 y = L b;
     x = U \setminus y
 37 fprintf("Prova real (A * x = B)")
 38 A*x
                                                                       eol: CRLF line: 37 col: 32
```

- Resultado na tela de comando:

```
Command Window
Solucao do Sistema: x =

1.00000
1.00000
1.00000
1.00000

Prova real (A * x = B) ans =

11.000
13.000
21.600
30.000
```

6ª Questão

Antes de isolar cada X do sistema, é necessário observar se cada termo a^{ii} é maior que a soma do que o módulo dos outros elementos da sua linha. Por meio dessa observação, e organizando as linhas para que isso ocorra, tem-se o novo sistema:

$$-8x_1 + x_2 - 2x_3 = -20$$
$$2x_1 - 6x_2 - x_3 = -38$$
$$-3x_1 - x_2 + 7x_3 = -34$$

Agora, é possível calcular pelo Método de Gauss Siedel e Jacobi:

a) Gauss Siedel:

- Código no Octave

```
ed> 🛛 Lista3Exer6_Siedel.m 🔀
                          Questao5Lista2.m
%Metodo de Gauss Siedel para sistema linear
clear all; close all; clc;
k = 1; x1(1) = 0; x2(1) = 0; x3(1) = 0;
%Erro estabelecido
err = 1e-5;
%Iteracoes
\frac{1}{2} for k = 2: 100
  x1(k) = (20 + x2(k-1) - 2*x3(k-1))/8; %estabelece o 1°x1

x2(k) = (38 + 2*x1(k) - x3(k-1))/6; %Usa esse x1 no calculo de x2

x3(k) = (-34 + 3*x1(k) + x2(k))/7; %Usa x1 e x2 no calculo de x3
  %Calculo do Erro
  el = abs((x1(k) - x1(k-1))/x1(k));
  e2 = abs((x2(k) - x2(k-1))/x2(k));
  e3 = abs((x3(k) - x3(k-1))/x3(k));
 if (e1 + e2 + e3) <= err
   break
  endif
endfor
%Resultado
```

- Resposta na tela de comando

```
Command Window

Solucao encontrada em 8 interacoes por Gauss Siedel:

x1 = 3.999999

x2 = 8.000000

x3 = -2.000000
```

b) Jacobi

- Código no Octave

```
ed > 🗵 Lista3Exer6_Siedel.m 🗵 Questao5Lista2.m 🗵
%Metodo de Jacobi para sistema linear
clear all; close all; clc;
k = 1; x1(1) = 0; x2(1) = 0; x3(1) = 0;
%Erro estabelecido
err = 1e-5;
%Iteracoes
lfor k = 2: 100
 x1(k) = (20 + x2(k-1) - 2*x3(k-1))/8; %estabelece o 1°x1, usando valor da iteracao anterior dos x
  x2(k) = (38 + 2*x1(k-1) - x3(k-1))/6; %estabelece o 1°x2, usando valor da iteracao anterior dos x
             +3*x1(k-1) +x2(k-1))/7; %estabelece o 1°x3, usando valor da iteracao anterior dos x
 x3(k) = (-34)
  %Calculo do Erro
  el = abs((x1(k) - x1(k-1))/x1(k));
  e2 = abs((x2(k) - x2(k-1))/x2(k));
 e3 = abs((x3(k) - x3(k-1))/x3(k));
 if (e1 + e2 + e3) <= err
   break
 endif
endfor
```

- Resposta na tela de comando

```
Command Window
Solucao encontrada em 15 interacoes por Jacobi:
x1 = 3.999999
x2 = 8.000000
x3 = -1.999998
```

c) Comparando os dois resultados

Os dois métodos convergiram para as raízes 4, 8 e -2. Como o método de Gauss Siedel atualiza constantemente os valores para usá-los no cálculo dos outros "x's" e o método de Jacobi só atualiza após uma iteração completa, era de se esperar que este último tivesse mais iterações — o que ocorreu na realidade, 15 contra 8 iterações. Como conclusão, o método de Gauss Siedel se torna uma boa escolha pois precisa de um menor custo computacional