MÉTODOS COMPUTACIONAIS PARA ENGENHARIA CIVIL

LISTA DE EXERCÍCIOS – SISTEMAS LINEARES E NÃO-LINEARES

**QUESTÃO 01:**

>> A = [0 3 8 -5 -1 6; 3 12 -4 8 5 -2; 8 0 0 10 -3 7;

3 1 0 0 0 4; 0 0 4 -6 0 2; 3 0 5 0 0 -6];

>> b = [34; 20; 45; 36; 60; 28];

>> x = GaussPivo(A,b)

x =

-67.1576

-284.9310

202.6158

168.6108

672.6847

130.6010

>> r = b - A\*x

**r =**

**1.0e-11 \***

**0**

**-0.0227**

**0.1023**

**-0.1080**

**0**

**-0.0114**

**QUESTÃO 02:**

>> A = [1 2 -3 0 3; 2 5 -1 1 4; -3 -1 50 1 -19;

0 1 1 6 0; 3 4 -19 0 39];

>> b = [17; 41; -45; 30; 51];

>> U = Cholesky(A)

U =

1 2 -3 0 3

0 1 5 1 -2

0 0 4 -1 0

0 0 0 2 1

0 0 0 0 5

>> = triang\_inf(U',b)

d =

17.0000

7.0000

-7.2500

7.8750

1.2250

>> x = triang\_sup(U,d)

**x =**

**-2.2488**

**7.9688**

**-0.8588**

**3.8150**

**0.2450**

**QUESTÃO 03:** (Ver Anexos)

>> A = [67 0 0 0 0; -67 161 0 0 -36; 0 -161 182 0 0;

0 0 -182 212 0; 0 0 0 0 36]

A =

67 0 0 0 0

-67 161 0 0 -36

0 -161 182 0 0

0 0 -182 212 0

0 0 0 0 36

>> b = [180; 740; 3850; 4720; 710]

b =

180

740

3850

4720

710

(a) Cálculo da matriz inversa usando o método de Crout, a partir da decomposição LU:

>> C = Crout(A)

A matriz U é igual a

1.0000 0 0 0 0

0 1.0000 0 0 -0.2236

0 0 1.0000 0 -0.1978

0 0 0 1.0000 -0.1698

0 0 0 0 1.0000

A matriz L é igual a

67 0 0 0 0

-67 161 0 0 0

0 -161 182 0 0

0 0 -182 212 0

0 0 0 0 36

A matriz inversa C é igual a

0.0149 0 0 0 0

0.0062 0.0062 0 0 0.0062

0.0055 0.0055 0.0055 0 0.0055

0.0047 0.0047 0.0047 0.0047 0.0047

0 0 0 0 0.0278

(b) Comparativo entre as concentrações inicial e final, após um aumento de 40% na carga de cloreto no lago Michigan:

>> conc\_inicial = C\*b

**conc\_inicial =**

**2.6866**

**10.1242**

**30.1099**

**48.1132**

**19.7222**

>> b40=[180; 740; 3850; 4720; 994]

b40 =

180

740

3850

4720

994

>> conc\_final = C\*b40

**conc\_final =**

**2.6866**

**11.8882**

**31.6703**

**49.4528**

**27.6111**

Pode-se observar um aumento na concentração de cloreto nos lagos que recebem carga direta ou indiretamente do lago Michigan.

(c) Análise do condicionamento do sistema:

Pode-se analisar o condicionamento do sistema através do número de condicionamento de uma matriz, definido com o conceito de norma, neste caso, norma uniforme (módulo máximo).

>> condicionamento = cond(A,inf)

condicionamento =

10.9444

Pode-se também inverter a matriz invertida C e verificar se o resultado está suficientemente próximo da matriz de coeficientes original.

>> Cinv = Crout(C)

Cinv =

67.0000 0 0 0 0

-67.0000 161.0000 0 0 -36.0000

0 -161.0000 182.0000 0 0

0 -0.0000 -182.0000 212.0000 0

0 0 0 0 36.0000

Ou ainda, multiplicando a matriz inversa C pela matriz dos coeficientes original e verificar se o resultado está próximo da matriz identidade.

>> A\*C

ans =

1.0000 0 0 0 0

-0.0000 1.0000 0 0 -0.0000

0 0 1.0000 0 0

0.0000 0.0000 0 1.0000 0.0000

0 0 0 0 1.0000

Como se pode observar nas três opções de análise, e ainda através da comparação entre as concentrações inicial e final de cloreto nos lagos, **o sistema apresenta um bom condicionamento**.

**QUESTÃO 04:** (Ver Anexos)

Devido à grande dimensão da matriz, os seus elementos foram atribuídos conforme o script abaixo:

A = zeros(12);

%Nó 1 (H,V)

A(1,1) = -cosd(60);

A(1,6) = -1;

A(1,10) = 1;

A(2,1) = -sind(60);

A(2,11) = 1;

%Nó 2 (H,V)

A(3,1) = cosd(60);

A(3,2) = -1;

A(3,7) = -cosd(60);

A(4,1) = sind(60);

A(4,7) = sind(60);

%Nó 3 (H,V)

A(5,2) = 1;

A(5,3) = -1;

A(6,8) = 1;

%Nó 4 (H,V)

A(7,3) = 1;

A(7,4) = -cosd(45);

A(7,9) = cosd(45);

A(8,4) = sind(45);

A(8,9) = sind(45);

%Nó 5 (H,V)

A(9,4) = cosd(45);

A(9,5) = 1;

A(10,4) = -sind(45);

A(10,12) = 1;

%Nó 6 (H,V)

A(11,5) = -1;

A(11,6) = 1;

A(11,7) = cosd(60);

A(11,9) = -cosd(45);

A(12,7) = -sind(60);

A(12,8) = -1;

A(12,9) = -sind(45);

b = zeros(12,1);

b(12) = 3500;

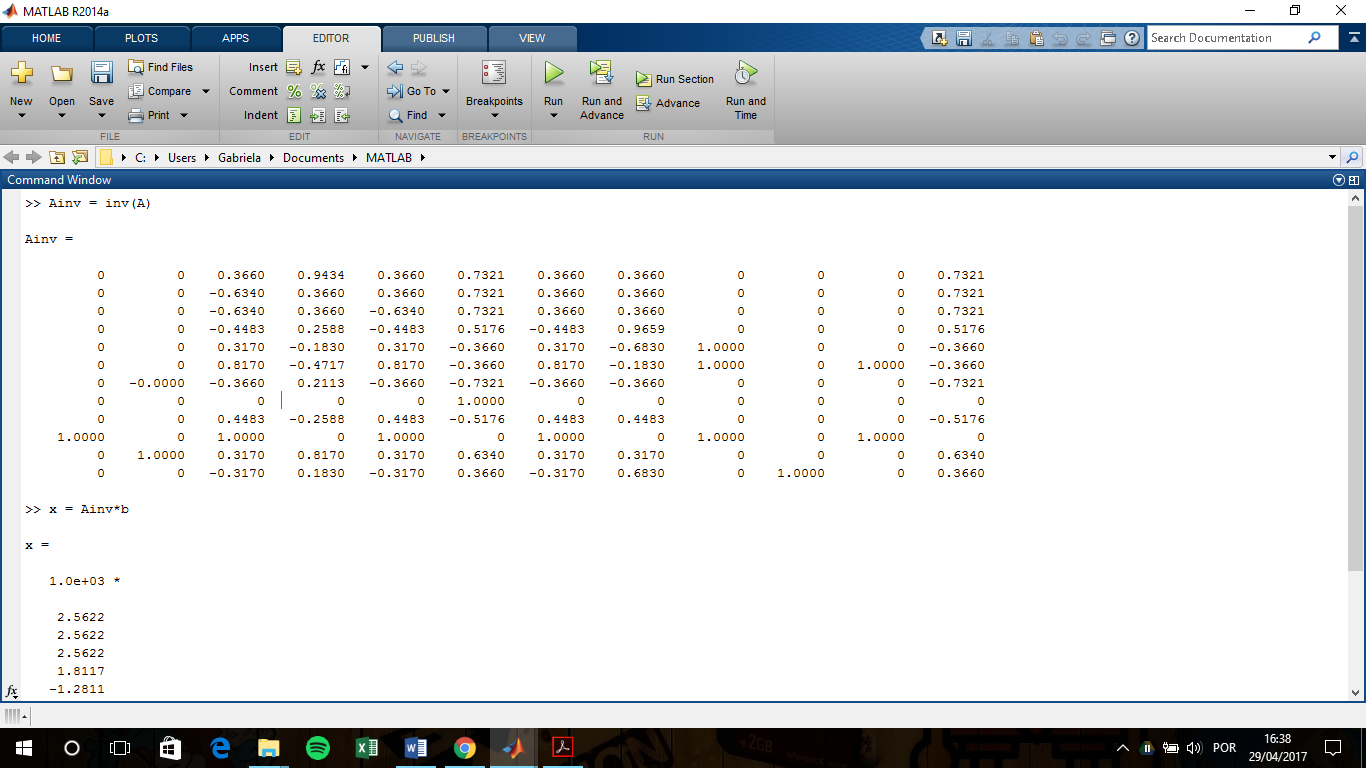
%Cálculo da Matriz Inversa por Gauss-Jordan

Aginv = GaussJordanInv(A);

xgauss = Aginv\*b;

>> Aginv = GaussJordanInv(A)

Aginv =



>> xgauss = Aginv\*b

**xgauss =**

**1.0e+03 \***

**2.5622**

**2.5622**

**2.5622**

**1.8117**

**-1.2811**

**-1.2811**

**-2.5622**

**0**

**-1.8117**

**0**

**2.2189**

**1.2811**

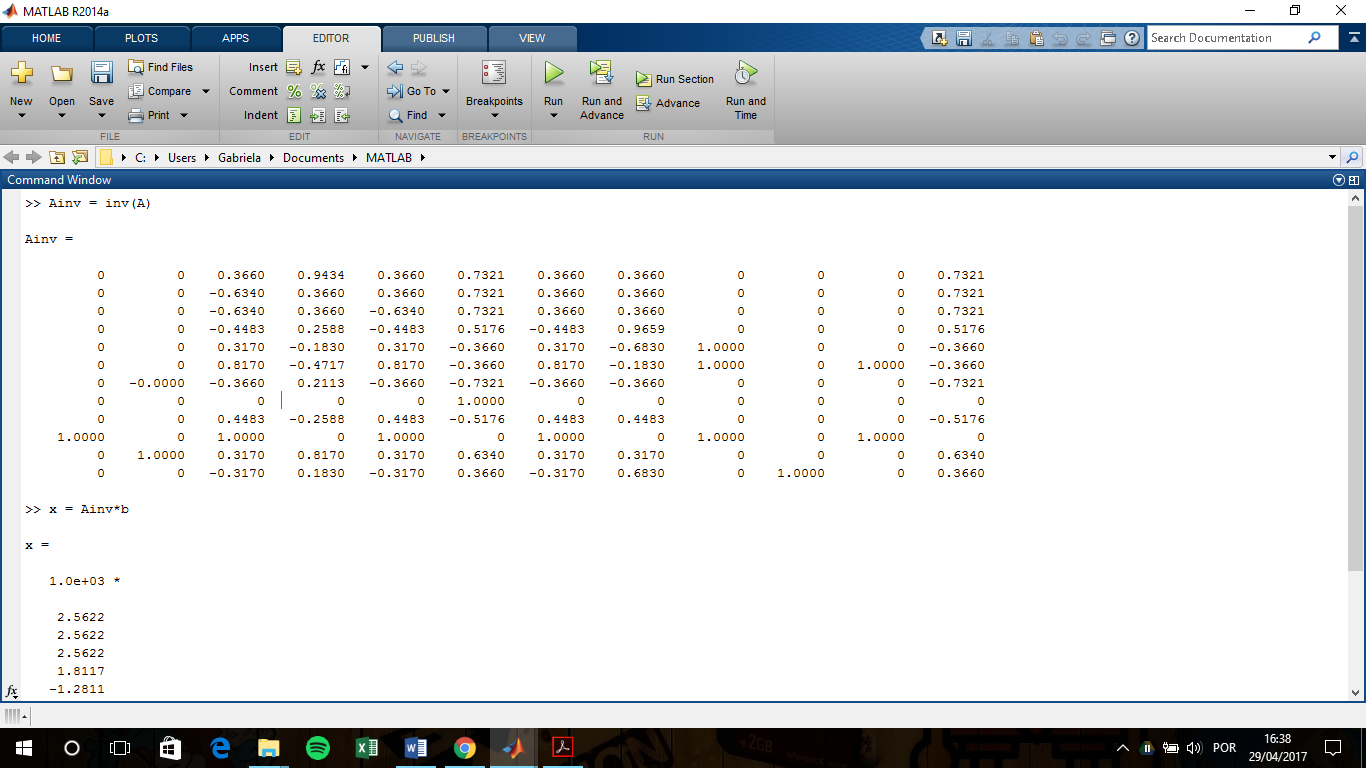
%Comparando com a funcao 'inv' do MATLAB

Ainv = inv(A);

xinv = Ainv\*b;

>> Ainv = inv(A)

Ainv =



>> x = Ainv\*b

**x =**

**1.0e+03 \***

**2.5622**

**2.5622**

**2.5622**

**1.8117**

**-1.2811**

**-1.2811**

**-2.5622**

**0**

**-1.8117**

**0**

**2.2189**

**1.2811**

Para b(12) = 5000, tem-se que os novos valores para as forças e reações da treliça são dadas por:

**X2 =**

**1.0e+03 \***

**3.6603**

**3.6603**

**3.6603**

**2.5882**

**-1.8301**

**-1.8301**

**-3.6603**

**0**

**-2.5882**

**0**

**3.1699**

**1.8301**

Pode-se concluir que os resultados se encontram coerentes em ordem de grandeza e em termos dos esforços de compressão e tração em cada barra, em virtude da aplicação da carga vertical. Soma-se a isso, o fato da reação horizontal ser nula, devido à ausência de cargas externas horizontais e as reações verticais serem positivas (para cima).

**QUESTÃO 05:** (Ver Anexos)

>> A = [200 0 -50 0; -50 150 -50 0; 150 150 -390 90; 0 0 -240 240]

A =

200 0 -50 0

-50 150 -50 0

150 150 -390 90

0 0 -240 240

>> b = [150; 2000; 0; 5000]

b =

150

2000

0

5000

(a) Utilizando novamente o método de Crout:

>> Ainv = Crout(A)

Ainv =

0.0063 0.0013 -0.0013 0.0005

0.0037 0.0088 -0.0021 0.0008

0.0050 0.0050 -0.0050 0.0019

0.0050 0.0050 -0.0050 0.0060

>> conc = Ainv\*b

**conc =**

**5.7813**

**21.9688**

**20.1250**

**40.9583**

(b) Pode-se calcular a redução da carga na sala 4 proporcional à variação da concentração na sala 2, a partir do coeficiente de proporcionalidade correspondente ao elemento Ainv(2,4) da matriz inversa.

>> Dconc2 = conc(2) - 20

Dconc2 =

1.9688

>> RedCarga4 = Dconc2/Ainv(2,4)

**RedCarga4 =**

**2520**

**QUESTÃO 06:**

>> A = [4 1 1 4; 2 -8 1 -1; 1 2 -5 1; 1 1 1 -4]

A =

4 1 1 4

2 -8 1 -1

1 2 -5 1

1 1 1 -4

>> b = [7; -6; -1; -1]

b =

7

-6

-1

-1

Pelo método de Jacobi, tem-se que:

>> x0 = [1 1 1 1];

>> x = Jacobi(x0,A,b,20,1)

x =

1.0000 1.0000 1.0000 1.0000

0.2500 1.0000 1.0000 1.0000

0.2500 0.8125 0.8500 0.8125

0.5219 0.8172 0.7375 0.7281

0.6332 0.8816 0.7769 0.7691

0.5662 0.9093 0.8331 0.8229

0.4915 0.8928 0.8415 0.8272

0.4893 0.8747 0.8209 0.8065

0.5197 0.8741 0.8090 0.7962

0.5330 0.8815 0.8128 0.8007

0.5257 0.8848 0.8194 0.8068

0.5171 0.8830 0.8204 0.8075

**0.5167 0.8809 0.8181 0.8051**

Por Gauss-Seidel:

>> x = GaussSeidel(A,b,1,20)

**x =**

**0.5211**

**0.8819**

**0.8178**

**0.8052**

**QUESTÃO 07:**

>> A = [10 4 -1 3 0; 0 -8 -2 1 -3; 2 -4 7 0 0;

-1 2 -3 -10 2; 2 -1 -1 1 -7]

A =

10 4 -1 3 0

0 -8 -2 1 -3

2 -4 7 0 0

-1 2 -3 -10 2

2 -1 -1 1 -7

>> b = [2; 5; 13; 4; 7]

b =

2

5

13

4

7

>> x = GaussSeidel(A,b,1)

**x =**

**0.9591**

**-0.7070**

**1.1791**

**-1.1840**

**-0.9626**

**QUESTÃO 08:**

>> A = [2 -6 -1; -3 -1 -7; -8 1 -2]

A =

2 -6 -1

-3 -1 -7

-8 1 -2

>> b = [-38; -34; -20]

b =

-38

-34

-20

(a) Pelo método de Gauss-Seidel sem relaxação:

>> x = GaussSeidel(A,b,2,50)

**x =**

**1.0e+65 \***

**0.3153**

**-1.7026**

**-2.1127**

(b) Pelo método de Gauss-Seidel com relaxação (=1,2):

>> x = GaussSeidelrelax(A,b,1.2,2,50)

**x =**

**1.0e+71 \***

**-0.4451**

**2.7787**

**3.8314**

**QUESTÃO 09:**

(a) Pelo gráfico, pode-se observar que a solução do sistema, representada pela interseção entre as curvas, encontra-se no intervalo de 0 e 2, para valores positivos tanto de x quanto de y.



(b) Pelo método da substituição sucessiva, com aproximações iniciais de 1,5 para x e y, e adotando uma tolerância de 0.001 ou no máximo 100 iterações:

>> [x,y] = subsucess(1.5,1.5,0.001,100)

ii x y eax eay

1 1.581139 1.581139 5.131670 5.131670

2 1.606592 1.555269 1.584314 1.663377

3 1.598521 1.563564 0.504934 0.530504

4 1.601113 1.560909 0.161913 0.170079

5 1.600284 1.561759 0.051819 0.054436

6 1.600550 1.561487 0.016595 0.017432

7 1.600465 1.561574 0.005313 0.005582

8 1.600492 1.561546 0.001701 0.001787

**x =**

**1.6005**

**y =**

**1.5616**

(c) Pelo método de Newton-Raphson, com aproximações iniciais de 1,5 para x e y e adotando uma tolerância de 0.001 ou no máximo 100 iterações:

function [J,f] = f1(x0)

x = x0(1);

y = x0(2);

f = [x^2 + y^2 - 5; x^2 - y - 1];

J = [2\*x 2\*y; 2\*x -1];

end

>> x0 = [1.5; 1.5];

>> [x,f,ea,iter] = newton\_mult('f1',x0,0.01,1000)

matriz ampliada após eliminação progressiva:

3.0000 3.0000 -0.5000

0 -4.0000 0.2500

matriz ampliada após eliminação progressiva:

3.2083 3.1250 0.0148

0 -4.1250 -0.0039

matriz ampliada após eliminação progressiva:

3.2010 3.1231 0.0000

0 -4.1231 -0.0000

**x =**

**1.6005**

**1.5616**

f =

1.0e-04 \*

0.1442

0.1352

ea =

2.6818e-04

iter =

3