**Cálculo Numérico**

**Lista de Exercícios 02**

**Entrega: 30/03/2023**

**Discente: Jheickson Felipe Sousa Santos**

1. Implemente um algoritmo que resolva o sistema triangular superior abaixo e informe a solução:

Resposta (Gauss.java):

Sistema inserido:

1.0x[0] + 2.0x[1] + 1.0x[2] + 2.0x[3] = -2.0

0.0x[0] + -1.0x[1] + -1.0x[2] + -3.0x[3] = 4.0

0.0x[0] + 0.0x[1] + 3.0x[2] + 5.0x[3] = 1.0

0.0x[0] + 0.0x[1] + 0.0x[2] + 3.0x[3] = -3.0

Sistema triangularizado:

1.0x[0] + 2.0x[1] + 1.0x[2] + 2.0x[3] = -2.0

0.0x[0] + -1.0x[1] + -1.0x[2] + -3.0x[3] = 4.0

0.0x[0] + 0.0x[1] + 3.0x[2] + 5.0x[3] = 1.0

0.0x[0] + 0.0x[1] + 0.0x[2] + 3.0x[3] = -3.0

Solucao:

x[0] = 4.0

x[1] = -3.0

x[2] = 2.0

x[3] = -1.0

1. Implemente o algoritmo da Eliminação Gaussiana, resolva o sistema linear e informe a matriz dos coeficientes triangularizada e a solução:

Resposta (Gauss.java):

Sistema inserido:

1.0x[0] + 2.0x[1] + 1.0x[2] + 2.0x[3] = -2.0

2.0x[0] + 3.0x[1] + 1.0x[2] + 1.0x[3] = 0.0

1.0x[0] + 1.0x[1] + 3.0x[2] + 4.0x[3] = 3.0

3.0x[0] + 2.0x[1] + 1.0x[2] + 1.0x[3] = 7.0

Sistema triangularizado:

3.0x[0] + 2.0x[1] + 1.0x[2] + 1.0x[3] = -2.0

0.0x[0] + 1.6666666666666667x[1] + 0.33333333333333337x[2] + 0.33333333333333337x[3] = 0.0

0.0x[0] + 0.0x[1] + 2.5999999999999996x[2] + 3.5999999999999996x[3] = 3.0

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.846153846153846x[3] = 7.0

Solucao:

x[0] = -0.2610722610722611

x[1] = 0.40559440559440557

x[2] = -10.300699300699302

x[3] = 8.272727272727273

1. Seja o sistema linear na forma matricial A x = b



Qual a solução obtida pela Eliminação Gaussiana? Explique os seus resultados.

Resposta (Gauss.java):

Sistema inserido:

2.0x[0] + 1.0x[1] + 7.0x[2] + 4.0x[3] + -3.0x[4] + -1.0x[5] + 4.0x[6] + 4.0x[7] + 7.0x[8] + 0.0x[9] = 86.0

4.0x[0] + 2.0x[1] + 2.0x[2] + 3.0x[3] + -2.0x[4] + 0.0x[5] + 3.0x[6] + 3.0x[7] + 4.0x[8] + 1.0x[9] = 45.0

3.0x[0] + 4.0x[1] + 4.0x[2] + 2.0x[3] + 1.0x[4] + -2.0x[5] + 2.0x[6] + 1.0x[7] + 9.0x[8] + -3.0x[9] = 52.5

9.0x[0] + 3.0x[1] + 5.0x[2] + 1.0x[3] + 0.0x[4] + 5.0x[5] + 6.0x[6] + -5.0x[7] + -3.0x[8] + 4.0x[9] = 108.0

2.0x[0] + 0.0x[1] + 7.0x[2] + 0.0x[3] + -5.0x[4] + 7.0x[5] + 1.0x[6] + 0.0x[7] + 1.0x[8] + 6.0x[9] = 66.5

1.0x[0] + 9.0x[1] + 8.0x[2] + 0.0x[3] + 3.0x[4] + 9.0x[5] + 9.0x[6] + 0.0x[7] + 0.0x[8] + 5.0x[9] = 90.5

4.0x[0] + 1.0x[1] + 9.0x[2] + 0.0x[3] + 4.0x[4] + 3.0x[5] + 7.0x[6] + -4.0x[7] + 1.0x[8] + 3.0x[9] = 139.0

6.0x[0] + 3.0x[1] + 1.0x[2] + 1.0x[3] + 6.0x[4] + 8.0x[5] + 3.0x[6] + 3.0x[7] + 0.0x[8] + 2.0x[9] = 61.0

6.0x[0] + 5.0x[1] + 0.0x[2] + -7.0x[3] + 7.0x[4] + -7.0x[5] + 6.0x[6] + 2.0x[7] + -6.0x[8] + 1.0x[9] = -43.5

1.0x[0] + 6.0x[1] + 3.0x[2] + 4.0x[3] + 8.0x[4] + 3.0x[5] + -5.0x[6] + 0.0x[7] + -6.0x[8] + 0.0x[9] = 31.0

Sistema triangularizado:

9.0x[0] + 3.0x[1] + 5.0x[2] + 1.0x[3] + 0.0x[4] + 5.0x[5] + 6.0x[6] + -5.0x[7] + -3.0x[8] + 4.0x[9] = 86.0

0.0x[0] + 8.666666666666666x[1] + 7.444444444444445x[2] + -0.1111111111111111x[3] + 3.0x[4] + 8.444444444444445x[5] + 8.333333333333334x[6] + 0.5555555555555556x[7] + 0.3333333333333333x[8] + 4.555555555555555x[9] = 45.0

0.0x[0] + 0.0x[1] + 7.064102564102564x[2] + -0.4487179487179487x[3] + 4.115384615384615x[4] + 1.1025641025641024x[5] + 4.653846153846154x[6] + -1.7564102564102564x[7] + 2.346153846153846x[8] + 1.3974358974358976x[9] = 52.5

0.0x[0] + 0.0x[1] + 0.0x[2] + -8.003629764065336x[3] + 9.404718693284936x[4] + -12.333938294010888x[5] + 3.0090744101633393x[6] + 3.671506352087114x[7] + -2.152450090744101x[8] + -2.0744101633393828x[9] = 108.0

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.0x[3] + 11.924263038548753x[4] + -8.566439909297053x[5] + -8.087528344671203x[6] + 1.336507936507936x[7] + -6.103854875283447x[8] + -3.9306122448979592x[9] = 66.5

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.0x[3] + 2.220446049250313E-16x[4] + -10.776898794355914x[5] + -0.8180123987373076x[6] + 8.550983151409119x[7] + 3.9037386376602132x[8] + -3.7449511276765683x[9] = 90.5

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.0x[3] + 0.0x[4] + 0.0x[5] + -9.471168878834826x[6] + 3.1997568438401034x[7] + -5.025095550842223x[8] + 1.6456148956969474x[9] = 139.0

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.0x[3] + 0.0x[4] + 0.0x[5] + 0.0x[6] + 13.791878592492068x[7] + 7.8665845419160405x[8] + -0.548806471013019x[9] = 61.0

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.0x[3] + 0.0x[4] + 0.0x[5] + 0.0x[6] + 0.0x[7] + 9.652447961785258x[8] + -3.092540281444902x[9] = -43.5

0.0x[0] + 0.0x[1] + 0.0x[2] + 0.0x[3] + 0.0x[4] + 0.0x[5] + 0.0x[6] + 0.0x[7] + 0.0x[8] + 1.5561394746308075x[9] = 31.0

Solucao:

x[0] = 8.676847110775562

x[1] = 4.134834687715406

x[2] = 13.486971261974507

x[3] = -7.748487541462123

x[4] = -2.257762084129581

x[5] = -10.530766275056767

x[6] = -10.809539855421143

x[7] = 4.145635181613729

x[8] = 1.8758748272151753

x[9] = 19.921093517247044

1. Implemente o método iterative de Gauss-Jacobi e Gauss-Sidel, teste para um Sistema 3x3 e compare os resultados.

Resposta:

O Sistema utilizado foi:

Output Jacobi:

Solucao:

x[0] = 0.9999999217195776

x[1] = 2.0000000776570803

x[2] = -0.9999999477768775

Output Seidel:

Solucao:

x[0] = 1.0000000156176152

x[1] = 2.0000000041751704

x[2] = -0.999999998365365

Os dois métodos são bem similares, a diferença é que o método de Gauss-Seidel atualiza cada elemento de x assim que é calculado, enquanto o método de Gauss-Jacobi usa a aproximação anterior de x para calcular todas as novas aproximações antes de atualizar x.