Tiago Correa Guedes-13/0145581

Gabriel Miranda-13/0111350

Fernanda-14/0019855

**Cálculo Numérico**

Turma B

Professora: Luísa Yoko

**METODOLOGIA**

Primeiramente são dadas ao usuário 3 opções: uma para que se resolva o sistema através do método de Gauss-Jacobi, uma para que se resolva o sistema através do método Gauss\_Seidel, e uma que resolve o sistema utilizando os dois métodos e mostra a diferença dos resultados obtidos em cada um. O usuário escolhe qual dos métodos será usado informando o número correspondente a sua opção (1, 2 ou 3).

Uma vez feita a escolha do método usado, inicia-se o processo de leitura do sistema de equações que o programa deve resolver, começando pela matriz formada pelos coeficientes das equações do sistema, que é salva na variável ‘A’, em seguida informa-se os resultados de cada uma das equações do sistema, que são salvos no vetor ‘b’. Tendo sido informado o sistema, o usuário deve agora fornecer a aproximação inicial **X0** a ser usada, que ficará salva no vetor ‘x0’, e por fim, define-se um erro máximo para as aproximações, que fica salvo na variável ‘tol’.

Por exemplo, a leitura do sistema proposto no enunciado:

3,21x + 7,89y + 4,95z - 6,35w = 31,07

8,05x - 4,70y + 5,05z + 1,25w = 38,27

2,02x + 7,77y – 3,04z – 6,72w= 0

0,32x + 9,05y – 4,77z – 4,78w = -20,67

Com aproximação inicial X0=[0 0 0 0] e erro=0.005, seria feita dessa forma no programa:

A=[3.21 7.89 4.95 -6.35;8.05 -4.70 5.05 1.25;2.02 7.77 -3.04 -6.72;0.32 9.05 -4.77 -4.78]

b= [31.07;38.27;0;-20.67]

x0=[0;0;0;0]

tol=0.005

Aqui termina o processo de leitura de dados do programa e inicia-se a resolução aproximada do sistema, para cada método é usada uma função que recebe como parâmetros A, b, x0 e tol. A função Gauss\_Jacobi consistem em usar de funções originais do MatLab para manipular a matriz A para transformar Ax+b em x=Cx+g, e então iniciar uma operação em loop que cessa somente quando é atingido um erro suficientemente pequeno (cada ciclo do loop é contado como uma iteração). A função Gauss\_Seidel temos dois loops que realizam o método, um mais interno, que consiste em calcular os valores aproximados para as incógnitas (**xik**) e salvá-los em um vetor, e um loop mais externo, infinito, que é quebrado quando for verificado que o erro atingido é satisfatório. Uma vez terminados os cálculos para tais funções, o programa mostra na tela os valores encontrados para cada incógnita e o número de iterações usadas em cada método. Em ambos os casos o programa é capaz de verificar, se os valores no vetor x0 são de fato números, ou se estouraram para o infinito, sendo assim capaz de dizer quando o sistema não está convergindo.

**CÓDIGO- FONTE**

**Código principal**

close all;

clc

disp('-----------------TRABALHO COMPUTACIONAL-----------------');

disp('1. Apenas Gauss\_Jacobi');

disp('2. Apenas Gauss\_Seidel');

disp('3. Ambos os métodos');

prompt = 'Faça sua escolha:';

resp = input(prompt);

if(resp~=1 && resp~=2 && resp~=3)

disp('Entrada inválida. Programa encerrado.');

exit();

end

prompt = 'Insira a matriz: ';

A = input(prompt);

prompt = 'Insira o resultado do sistema: ';

b = input(prompt);

prompt = 'Insira a aproximacao inicial: ';

x0 = input(prompt);

prompt = 'Insira o erro maximo: ';

tol = input(prompt);

if(resp==1 || resp==3)

[t\_1,iter\_1,x\_1,converge\_1] = gauss\_Jacobi\_func(A,b,x0,tol);

if(converge\_1==0)

fprintf('\nO sistema escolhido não converge com Gauss Jacobi\n\n');

end

end

if(resp==2 || resp==3)

[t\_2,iter\_2,x\_2,converge\_2] = gauss\_Seidel\_func(A,b,x0,tol,500);

if(converge\_2==0)

fprintf('\nO sistema escolhido não converge com Gauss Seidel\n\n');

end

end

if(resp==1)

printa\_resultados(t\_1,iter\_1,x\_1);

elseif(resp==2)

printa\_resultados(t\_2,iter\_2,x\_2);

elseif(resp==3)

fprintf('\t\tMétodo\t\t\tTempo\t\tIterações\tResultado\n');

fprintf('\t\tGauss Jacobi\t%f\t%d\t\t\t%s\n',t\_1,iter\_1, mat2str(x\_1'));

fprintf('\t\tGauss Seidel\t%f\t%d\t\t\t%s\n',t\_2,iter\_2, mat2str(x\_2'));

fprintf('\t\tDiferença\t\t\t\t\t\t\t\t%s\n',mat2str(abs(x\_1-x\_2)'));

end

**Função gauss\_jacobi\_func**

function [t,iter,x,converge] = gauss\_Jacobi\_func(A,b,x0,tol)

tic;

n = length(A);

L=-tril(A,-1); % extrai a parte triangular inferior da matriz

U=-triu(A,1); % extrai a parte triangular superior da matriz

D=diag(diag(A)); % extrai os elementos da diagonal principal da matriz

N=L+U;

C=D\N;

G=D\b;

iter=1;

x = C\*x0 + G;

converge = 1;

while norm(x-x0)>tol\*norm(x0)

x0 = x;

x = C\*x0 + G;

iter = iter + 1;

end

t = toc;

for i = 1:n

if isnan(x(i)) || isinf(x(i)) || iter>1000

converge = 0;

end

end

**Função gauss\_seidel\_func**

function [t,iter,x,converge] = gauss\_Seidel\_func(A,b,x0,tol,maximo)

n = length(A);

x = x0;

converge = 1;

tic;

for iter=1:maximo %entra num loop que vai até o maximo caso esteja divergindo

xk=x;

for i = 1:n %método propriamente dito

x(i) = (1/A(i, i))\*(b(i) - A(i, 1:n)\*x + A(i, i)\*x(i));

end

if(abs((max(x-xk)/max(x))< tol)) %critério de parada

break; %Se ele for atingido, sai do loop infinito

end

end

t = toc;

for i = 1:n

if isnan(x(i)) || isinf(x(i))

converge = 0;

end

end

**Função printa\_resultados**

function printa\_resultados(t,iter,x)

disp('Tempo de processamento(segundos):');

disp(t);

disp('Iteracoes:');

disp(iter);

disp('Resultado:');

disp(x);

**RESULTADOS ENCONTRADOS**

Como o sistema não converge da forma que foi apresentado, devemos permutar algumas linhas e colunas dele para que seja possível a convergência, de tal forma que a matriz inicial A seja [9.05 0.32 4.78 -4.77;-4.70 8.05 1.25 5.05;7.77 2.02 -6.72 -3.04;7.89 3.21 -6.35 4.95].

Para a aproximação inicial X0=[0 0 0 0] os valores encontrados foram os seguintes:

**Método Tempo Iterações**

Gauss Jacobi 0.964567 2424

Gauss Seidel 0.122973 23

**Resultados:** Gaus-Jacobi [-Inf Inf Inf NaN]

Gaus-Seidel [-0.0636324093186352 2.63630330437982 -0.877641132437941 3.54272826494624]

Vale lembrar que devido a manipulação das linhas e colunas do sistema, o resultado está na forma [y x w z], ou seja, o resultado encontrado pelo método Gauss-Seidel são:

x= 2.63630330437982

y= -0.0636324093186352

z= 3.54272826494624

w= -0.877641132437941

Podemos notar que mesmo com as permutações de linhas e colunas das equações do sistema, não houve convergência para o método de Gauss-Jacobi.

**TESTES ADICIONAIS**

Aqui vamos observar diferenças encontradas nos resultados de acordo com a aproximação inicial x0 usada. Para tais testes, usamos o 4 digitos finais das matriculas dos componentes do grupo e erro máximo de 0,005:

-Com x0=[5 5 8 1]

**Método Tempo Iterações**

Gauss Jacobi 0.070945 2419

Gauss Seidel 0.004288 33

**Resultados**

Gaus-Jacobi [1.09177481933362e+308 4.69826729759195e+307 5.73813958272374e+307 -Inf]

Gaus-Seidel [-0.0624176144236736 2.6366178351339 -0.876574434516082 3.54195637734259]

**-**Com x0=[1 3 5 0]

**Método Tempo Iterações**

Gauss Jacobi 0.065222 2420

Gauss Seidel 0.003251 31

**Resultados**

Gaus-Jacobi [-Inf Inf Inf NaN]

Gaus-Seidel [-0.0628388462092829 2.63650841653355 -0.876944628881748 3.54222385578171]

-Com x0=[9 8 5 5]

**Método Tempo Iterações**

Gauss Jacobi 0.064621 2419

Gauss Seidel 0.000253 1

**Resultados**

Gaus-Jacobi [Inf -Inf -Inf NaN]

Gaus-Seidel [-2.57237569060773 -0.660890154764763 -5.43487458759627 3.83354504332396]