**UNIVERSIDADE DE SÃO PAULO - USP**

**INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO - ICMC**

**DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO**

**Primeiro Trabalho Prático**

**Cálculo Numérico**



Guilherme Milan 9012966

Romeu Bertho Junior       7151905

São Carlos

Maio/2018

Sumário

[Introdução 3](#_Toc513046747)

[Implementação 4](#_Toc513046748)

[Gauss-Seidel 4](#_Toc513046749)

[Critério de Sassenfeld 5](#_Toc513046750)

[CreateMatrix: 6](#_Toc513046751)

[Createb: 6](#_Toc513046752)

[Createb2: 7](#_Toc513046753)

[QuestaoA: 7](#_Toc513046754)

[QuestaoB: 8](#_Toc513046755)

[QuestaoC1: 8](#_Toc513046756)

[QuestaoC2: 9](#_Toc513046757)

[Resultados 10](#_Toc513046758)

[Item A 10](#_Toc513046759)

[Item B 10](#_Toc513046760)

[Item C 11](#_Toc513046761)

[Conclusão 12](#_Toc513046762)

# Introdução

Neste trabalho foi implementado o método de Gauss-Seidel para solução iterativa de sistemas lineares utilizando a linguagem MATLAB.

O cálculo da solução de sistemas lineares muito extensos apresenta-se demasiado custoso de se computar por meio dos métodos diretos, como o método de eliminação de Gauss e a decomposição LU. Portanto, para tal proposito, e recomendável a utilização de métodos iterativos como o método de Jacobi-Richardson e o método de Gauss-Seidel.

Dentre essas duas técnicas, o método de Gauss-Seidel e mais eficiente em computação não-concorrente e portanto ele será utilizado para a solução dos problemas apresentados neste trabalho.

Segue a descrição do problema. Dado um inteiro n, define-se uma matriz A de acordo com a Figura 1.

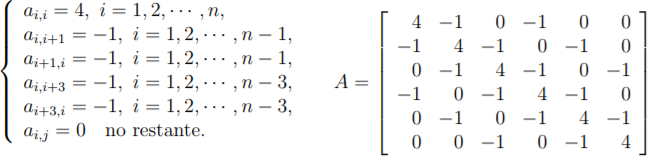


Figura 1. Definição da matriz A. A direita exemplo de matriz A para n=6. Fonte: disponibilizado pelo Professor Murilo F. Tome, ICMC-USP.

A partir da matriz A, busca-se resolver um sistema linear da forma Ax=b, em que b e um vetor de constantes informado a cada item do enunciado.

O método de Gauss consiste na decomposição da matriz A em 3 matrizes: L, D e R. As matrizes L e R são, respectivamente, formadas pelos componentes abaixo e acima da diagonal principal, enquanto D e a própria diagonal. Uma vez que um passo futuro da técnica consiste no cálculo da matriz inversa de D, convém dividir cada linha das matrizes L, D e R e o vetor b pelo componente correspondente da diagonal de A. Denomina-se as matrizes resultantes L\*, I, R\* e b\*. Assim, o sistema adquire o aspecto:

Manipulando esta expressão algebricamente, obtemos a formula do método de Gauss-Seidel:

Em que x(k+1) denota o vetor com o valor da solução do sistema na iteração k+1.

# Implementação

## Gauss-Seidel

**Dados de entrada:**

A: Matriz nxn.

b: Matriz das constantes.

eps: Erro epslon.

itMax: Número de iterações máxima.

**Dados de saída:**

D: Matriz diagonal.

L: Matriz triangular inferior.

U: Matriz triangular superior.

B: Matriz B\* (é a matriz A dividindo cada linha pelo componente correspondente da diagonal).

g: Matriz g\* (é a matriz b dividindo cada linha pelo componente correspondente da diagonal).

Conv: Vetor de erro.

Xi: Matriz solução de cada iteração.

function [ D,L,U,B,g,Conv,Xi ] = GaussSeidel( A,b,eps,itMax )

%GAUSSSEIDEL Summary of this function goes here

% Detailed explanation goes here

tic

% Obtém o tamanho max(row,col) de A de uma matriz NxN

n=length(A);

% Diagonal de A

D = diag(diag(A));

% Matriz inferior de A

L = tril(A,-1);

% Matriz superior de A

U = triu(A,1);

% Calcula B

B=(-D\(L+U));

% Calcula g

g=D\b;

% Verifica o critério de convergência se a Matriz é estritamente

% diagonalmente dominante

one=ones(n,1);

if max(abs(L+U)\*one-abs(D)\*one) < 0

disp('Passou no teste de convergencia, Matriz é estritamente diagonalmente dominante');

% Verifica o critério de convergência das linhas

elseif max(abs(B)\*one) < 1

disp('Passou no teste de convergencia, norma linha');

% Verifica o critério de convergência das colunas

%elseif max(abs(transpose(B))\*one) < 1

% disp('Passou no teste de convergencia, norma coluna');

elseif SassenfeldCriteria(B)<1

disp('Passou no teste de convergencia, critério de Sassenfeld');

else

disp('Não passou no teste de convergencia!');

end

% Iterando....

% Xi recebe matriz de 0s

Xi=zeros(n,1);

% Matriz de convergencia é incializada com inf

Conv=inf;

% Calcula Xi(k)

for k=1:itMax+1

for i=1:n

Xi(i,k)=B(i,:)\*Xi(:,k)+g(i);

end

% Concatena a k-ésima iteração em Xi

Xi=[Xi Xi(:,k)];

% Calcula a convergência entre X(k) e X(k-1) e concatena na matriz Conv

if(k>1)

Conv=[Conv norm(Xi(:,k)-Xi(:,k-1),inf)/norm(Xi(:,k),inf)];

end

% Verifica se convergiu para o eps esperado

if(Conv(1,k)<eps)

break

end

end

toc

end

## Critério de Sassenfeld

Um dos métodos responsáveis para a verificação de convergência do método de Gauss-Seidel para a matriz A.

**Dados de entrada:**

B: Matriz B\*.

**Dados de saída:**

maxValue: valor máximo do vetor Xi.

function [ maxValue ] = SassenfeldCriteria( B )

%SASSENFELDCRITERIA Summary of this function goes here

% Detailed explanation goes here

n=length(B);

% Xi recebe matriz de 0s

Xi=ones(n,1);

for i=1:n

Xi(i,1)=abs(B(i,:))\*Xi(:,1);

end

maxValue=max(Xi);

end

## CreateMatrix:

Cria a matriz A especificada na descrição do problema.

**Dados de entrada:**

n: Tamanho da Matriz nxn.

**Dados de saída:**

A: Matriz nxn.

function [ A ] = CreateMatrix( n )

%CREATEMATRIX Summary of this function goes here

% Detailed explanation goes here

A=eye(n)\*4;

for i=1:n

if(i<n)

A(i,i+1)=-1;

A(i+1,i)=-1;

end

if(i<n-2)

A(i,i+3)=-1;

A(i+3,i)=-1;

end

end

end

## Createb:

Cria a matriz b de constantes para realizar o teste da questão b).

**Dados de entrada:**

A: Matriz nxn.

**Dados de saída:**

b: Matriz de constantes.

function [ b ] = Createb( A )

%CREATEB Summary of this function goes here

% Detailed explanation goes here

n=length(A);

one=ones(n,1);

b=zeros(n,1);

for i=1:n

b(i)=A(i,:)\*one;

end

end

## Createb2:

Cria a matriz b de constantes para realizar a resolução dos sistemas lineares nas questões a) e c).

**Dados de entrada:**

A: Matriz nxn.

**Dados de saída:**

b: Matriz de constantes.

function [ b ] = Createb2( A )

%CREATEB2 Summary of this function goes here

% Detailed explanation goes here

n=length(A);

b=zeros(n,1);

for i=1:n

b(i)=1/i;

end

end

## QuestaoA:

Programa para a resolução da questão A:

**Dados de entrada:**

m: Tamanho da matriz mxm.

eps: Erro epslon.

itMax: Número de iterações máxima.

**Dados de saída:**

A: Matriz A gerada.

b: Matriz de constantes gerada.

D: Matriz diagonal.

L: Matriz triangular inferior.

U: Matriz triangular superior.

B: Matriz B\* (é a matriz A dividindo cada linha pelo componente correspondente da diagonal).

g: Matriz g\* (é a matriz b dividindo cada linha pelo componente correspondente da diagonal).

Conv: Vetor de erro.

Xi: Matriz solução de cada iteração.

function [ A,b,D,L,U,B,g,Conv,Xi ] = QuestaoA(m,eps,itMax)

%QUESTAOA Summary of this function goes here

% Detailed explanation goes here

tic

A=CreateMatrix(m);

b=Createb2(A);

[D,L,U,B,g,Conv,Xi]=GaussSeidel(A,b,eps,itMax);

toc

end

## QuestaoB:

Programa para a resolução da questão B

**Dados de saída:**

A: Matriz A gerada.

b: Matriz de constantes gerada.

D: Matriz diagonal.

L: Matriz triangular inferior.

U: Matriz triangular superior.

B: Matriz B\* (é a matriz A dividindo cada linha pelo componente correspondente da diagonal).

g: Matriz g\* (é a matriz b dividindo cada linha pelo componente correspondente da diagonal).

Conv: Vetor de erro.

Xi: Matriz solução de cada iteração.

function [ A,b,D,L,U,B,g,Conv,Xi ] = QuestaoB()

%QUESTAOB Summary of this function goes here

% Detailed explanation goes here

tic

A=CreateMatrix(100);

b=Createb(A);

[D,L,U,B,g,Conv,Xi]=GaussSeidel(A,b,1e-10,10000);

toc

end

## QuestaoC1:

Programa para a resolução da questão C1.

**Dados de saída:**

A: Matriz A gerada.

b: Matriz de constantes gerada.

D: Matriz diagonal.

L: Matriz triangular inferior.

U: Matriz triangular superior.

B: Matriz B\* (é a matriz A dividindo cada linha pelo componente correspondente da diagonal).

g: Matriz g\* (é a matriz b dividindo cada linha pelo componente correspondente da diagonal).

Conv: Vetor de erro.

Xi: Matriz solução de cada iteração.

function [ A,b,D,L,U,B,g,Conv,Xi ] = QuestaoC1()

%QUESTAOC1 Summary of this function goes here

% Detailed explanation goes here

tic

A=CreateMatrix(6);

b=Createb2(A);

[D,L,U,B,g,Conv,Xi]=GaussSeidel(A,b,1e-10,10000);

toc

end

## QuestaoC2:

Programa para a resolução da questão C2.

**Dados de saída:**

A: Matriz A gerada.

b: Matriz de constantes gerada.

D: Matriz diagonal.

L: Matriz triangular inferior.

U: Matriz triangular superior.

B: Matriz B\* (é a matriz A dividindo cada linha pelo componente correspondente da diagonal).

g: Matriz g\* (é a matriz b dividindo cada linha pelo componente correspondente da diagonal).

Conv: Vetor de erro.

Xi: Matriz solução de cada iteração.

function [ A,b,D,L,U,B,g,Conv,Xi ] = QuestaoC2()

%QUESTAOC2 Summary of this function goes here

% Detailed explanation goes here

tic

A=CreateMatrix(100);

b=Createb2(A);

[D,L,U,B,g,Conv,Xi]=GaussSeidel(A,b,1e-10,10000);

toc

end

# Resultados

## Item A

Neste item, o grupo detectou uma possível ambiguidade na redação do enunciado. No texto, e afirmado que o programa deve receber como entrada uma matriz A, o vetor de constantes b, o erro máximo épsilon e a constante itmax, que determina o número máximo de iterações.

Disto, o grupo entendeu que a matriz A recebida deve ser uma matriz qualquer que será populada de acordo com a formula fornecida no enunciado. A outra interpretação considerada possível e de a matriz A, apenas nesta questão, conter números informados pelo usuário ao invés de utilizar a formula. Foi escolhida a primeira interpretação.

O programa implementado apresentou os resultados esperados. Demonstra-se a seguir a resposta obtida para N= 6, utilizando .

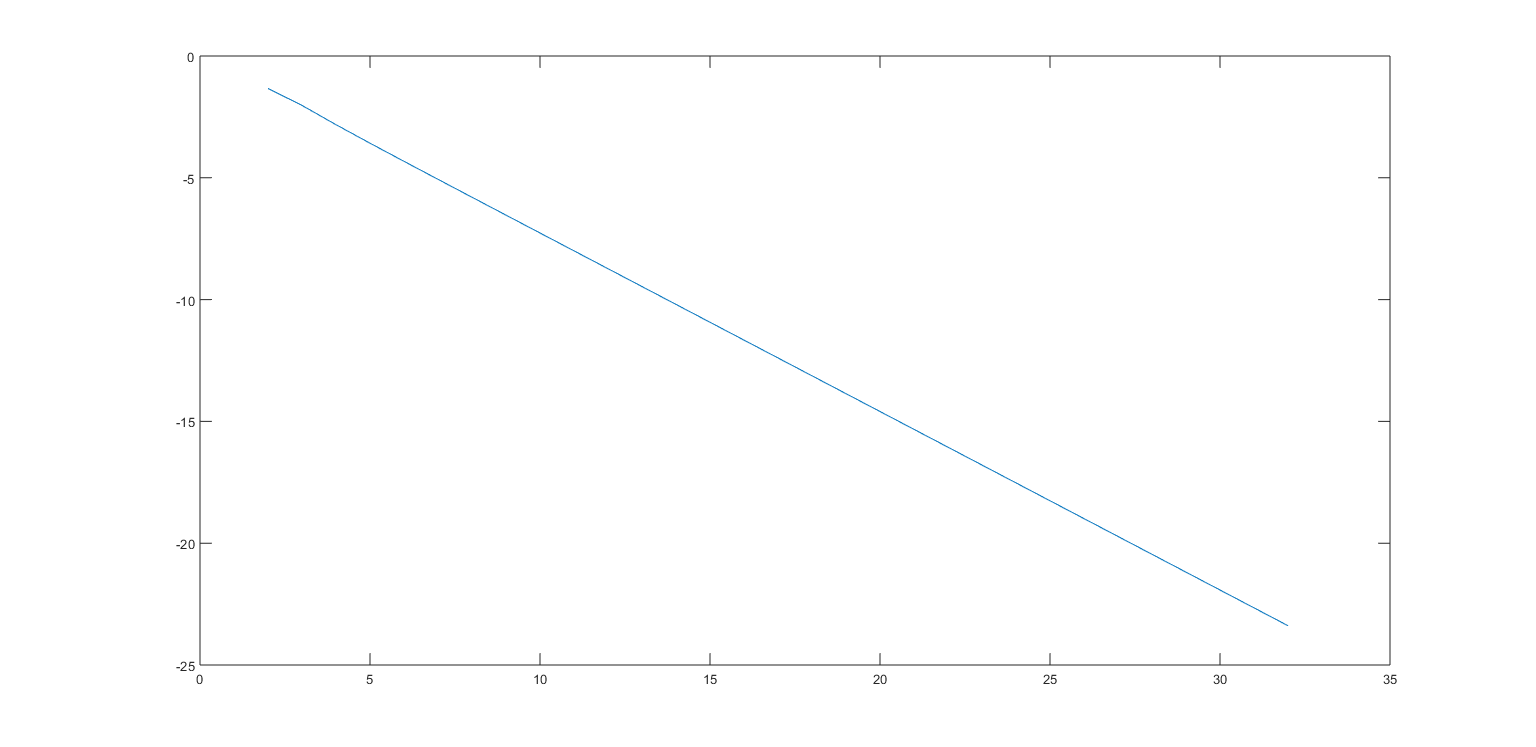
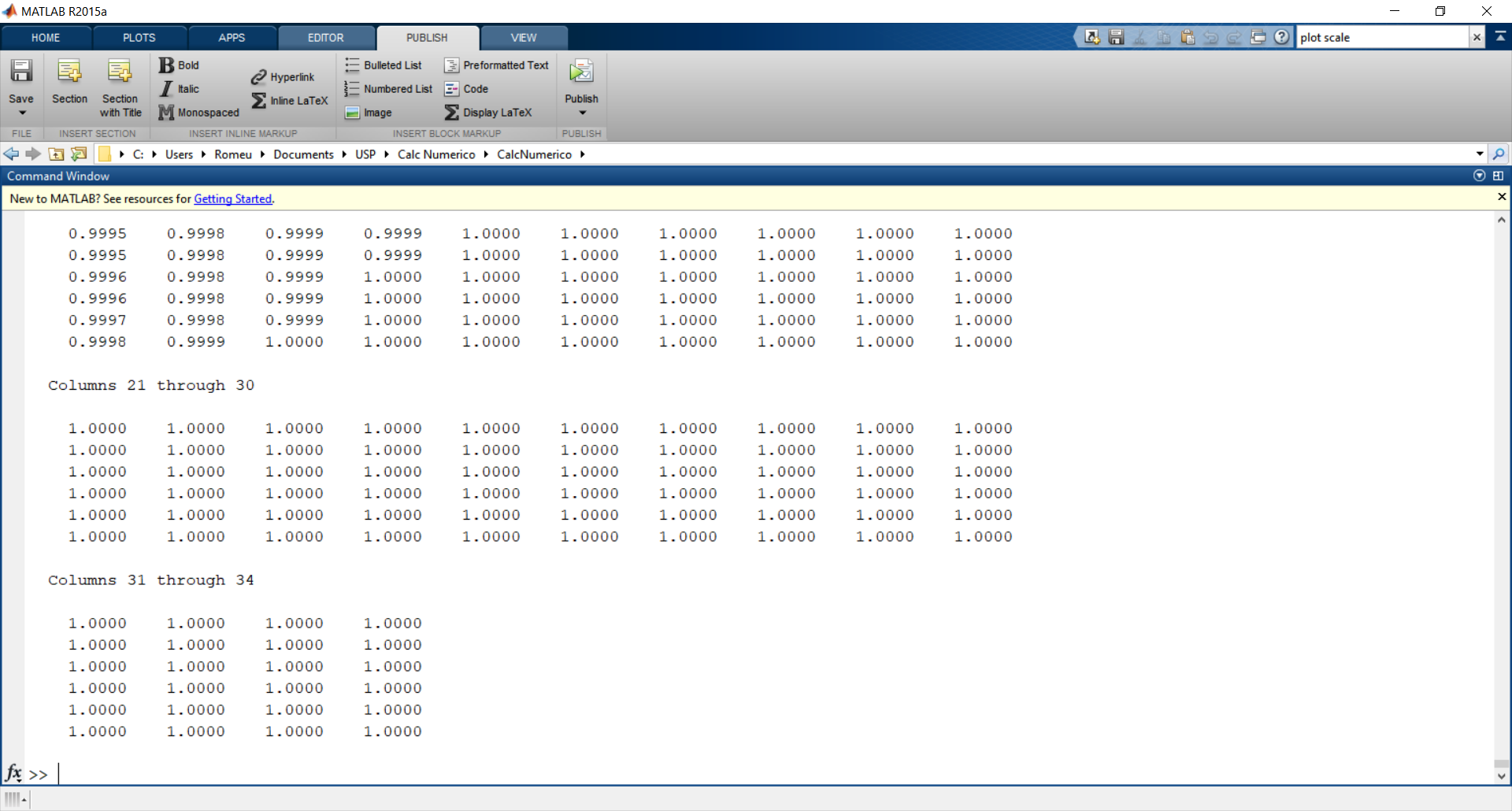


Gráfico do erro . Xi=1, i=1,2,...,6. Grafico do erro relativo linearizado utilizando o logaritmo dos dados.

## Item B

O programa apresentou o resultado esperado de acordo com o enunciado.



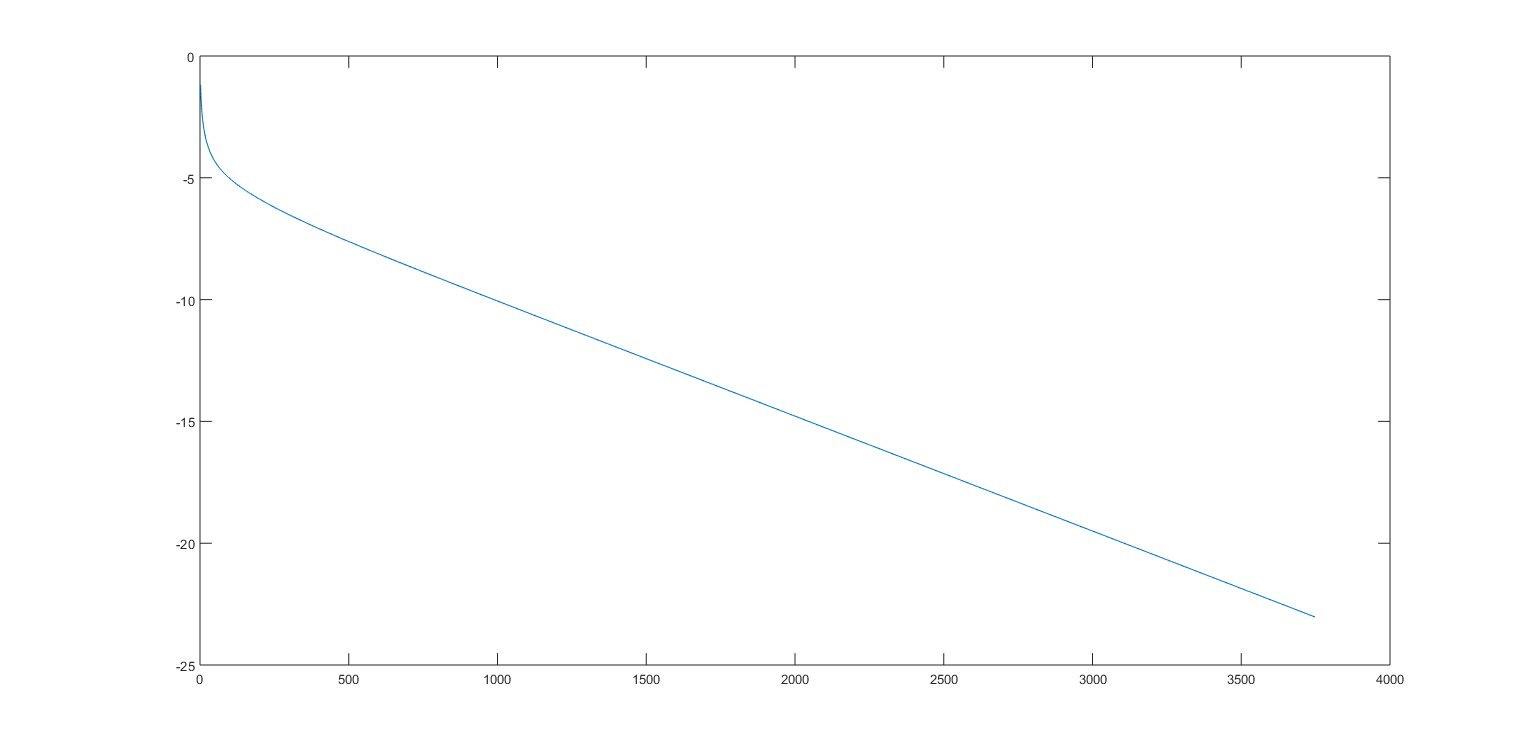
Captura de tela da saída do programa

Gráfico do erro . Xi=1, i=1,2,...,100. Grafico do erro relativo linearizado utilizando o logaritmo dos dados.

## Item C

Neste item, resolveu-se o sistema para n=100, obtendo o resultado desejado para .

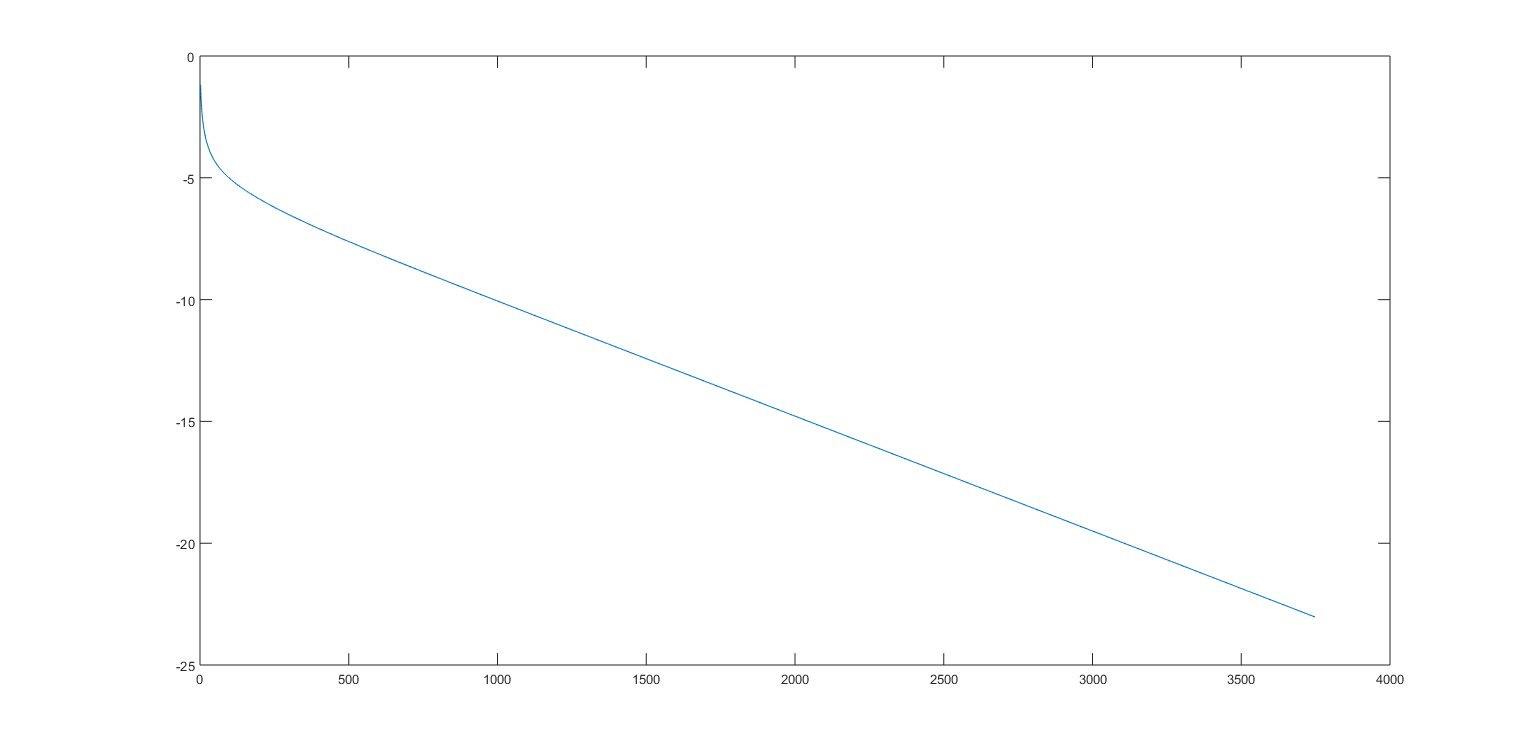
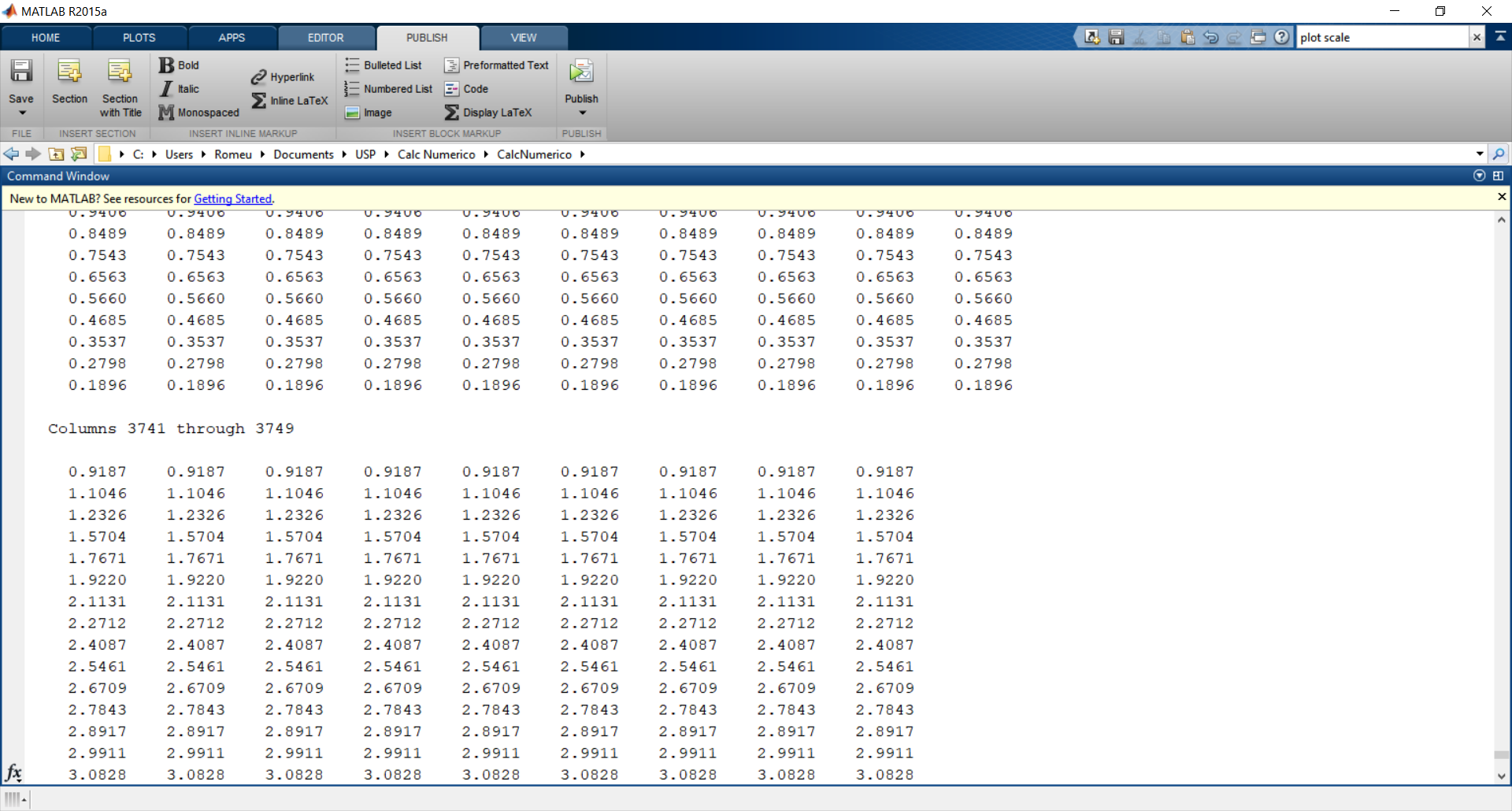


Gráfico do erro Gráfico do erro relativo linearizado utilizando o logaritmo dos dados.



Captura de tela com parte do resultado de Xi

A solução obtida apresentou erro na escala de 10-10, atestando pela alta precisão obtida na resposta.

# Conclusão

O método se demonstrou adequado para o proposito buscado por este trabalho, uma vez que sistemas da ordem de 100 equações lineares foram rapidamente resolvidos com alta precisão. Alem disso, a linguagem Matlab foi de grande ajuda, facilitando a realização das operações exigidas pela técnica e criando uma maneira mais conveniente de representação de expressões matemáticas em código de software.