Aula Prática 2 (ALN)

Raphael F. Levy

April 8, 2021

1 Análise do Método Iterativo de Jacobi

Primeiramente, segue a função criada:

```
function [xk,N,k,rk]=Jacobi_Method(A,b,x0,E,M,norma)

rk=norm(b-A*xk,norma) //Residuo da operacao
D=diag(diag(A)) //Diagonal de A
LPU = A-D //L plus U
invD=diag(1./diag(D))  //Inversa da matriz D
MJ=(-1)*invD*LPU //Matriz de Jacobi
cJ=invD*b //Vetor de Jacobi
k = 0
xk=x0

while k<M | N>E
    xk1=MJ*xk+cJ
    N=norm(xk1-xk,norma) //norma da diferenca entre as aproximacoes
    xk=xk1
    k=k+1
    if k>M
        break
    end
end

endfunction
```

Testando com os valores passados em sala e com número máximo de iterações e tolerância distintos temos:

```

--> A = [3 1 0; -1 -5 2; 1 -2 6]
 A  =

   3.    1.    0.
  -1.   -5.    2.
   1.   -2.    6.

--> b=[2;8;15]
 b  =

   2.
```

```
   8.
   15.

--> A\b
 ans  =

   1.
  -1.
   2.


--> x0=[0;0;0]
 x0  =

   0.
   0.
   0.

--> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-1),10,%inf)
 xk  =

   1.0000793
  -0.9998683
   1.9999686
 N  =

   0.0003696
 k  =

   10.
 rk  =

   9.457D-08

-------------------------------------------------------------------

--> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-2),15,%inf)
 xk  =

   0.9999988
  -1.0000001
   2.000001
 N  =

   0.0000037
 k  =

   15.
 rk  =

   0.0008005

-------------------------------------------------------------------

--> [xk,N,k, rk]= Jacobi_Method(A,b,x0 ,10^(-5),20,%inf)
 xk  =

```

```
   1.
  -1.
   2.
 N  =

   3.398D-08
 k  =

   20.
 rk  =

   15.

-----------------------------------------------------------------

--> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-5),30,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   1.791D-11
 k  =

   30.
 rk  =

   0.0000052

-----------------------------------------------------------------

--> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-5),19,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   0.0000001
 k  =

   19.
 rk  =

   0.0000052

-----------------------------------------------------------------

--> [xk,N,k, rk]= Jacobi_Method(A,b,x0,10^(-5),18,%inf)
 xk  =

   1.
  -0.9999999
   2.0000001
```

```
 N  =

   0.0000002
 k  =

   18.
 rk  =

   15.
```

Fazendo 19 iterações com uma tolerância de 10^{-5} já podemos obter um $x_k = A\backslash b$, logo o método de Jacobi converge.

2 Análise do Primeiro Método de Gauss-Seidel

```
function [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,E,M,norma)

rk=norm(b-A*xk,norma) //Residuo da operacao
D=diag(diag(A)) //Diagonal de A
LPU = A-D //L plus U
invD=diag(1./diag(D))  //Inversa da matriz D
L=tril(A,-1)
U=triu(A,1)
MG=(-1)*(inv(L+D))*(U) //Matriz de Gauss-Seidel
cG=(inv(L+D))*(b) //Gauss-Seidel
k = 0
xk=x0

while k<M | N>E
    xk1=MG*xk+cG
    N=norm(xk1-xk,norma) //norma da diferenca entre as aproximacoes
    xk=xk1
    k=k+1
    if k>M
        break
    end
end

endfunction
```

Testando com os valores passados em sala e com número máximo de iterações e tolerância distintos temos:

```
--> A = [3 1 0; -1 -5 2; 1 -2 6]
 A  =

   3.    1.    0.
  -1.   -5.    2.
   1.   -2.    6.

--> b=[2;8;15]
 b  =

   2.
   8.
   15.
```

```
--> x0=[0;0;0]
 x0  =

   0.
   0.
   0.

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-1),10,%inf)
 xk  =

   1.0000063
  -1.0000053
   1.9999972
 N  =

   0.0000163
 k  =

   10.
 rk  =

   0.0000145

-------------------------------------------------------------------

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-2),15,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   2.779D-08
 k  =

   15.
 rk  =

   0.0000145

-------------------------------------------------------------------

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),20,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   4.741D-11
 k  =

   20.
 rk  =
```

```
   4.231D-11

-------------------------------------------------------------------
--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),14,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   9.944D-08
 k  =

   14.
 rk  =

   15.

-------------------------------------------------------------------
--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),13,%inf)
 xk  =

   1.0000001
  -1.0000001
   1.9999999
 N  =

   0.0000004
 k  =

   13.
 rk  =

   15.
```

Fazendo 14 iterações com uma tolerância de 10^{-5} já podemos obter um $x_k = A\backslash b$, logo o primeiro método de Gauss-Seidel converge.

3 Análise do Segundo Método de Gauss-Seidel

```
function [x] = Resolve_Lx(L,b)

[t]=size(L,1);
x=zeros(t,1);

// Calcula x, sendo Lx=b

x(1)=b(1)/L(1,1);
for i=2:t
    x(i)=(b(i)-L(i,1:i)*x(1:i))/L(i,i);
end

endfunction
```

```
14
15 function [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,E,M,norma)
16
17 rk=norm(b-A*xk,norma) //Residuo da operacao
18 D=diag(diag(A)) //Diagonal de A
19 LPU = A-D //L plus U
20 L=tril(A,-1)
21 U=triu(A,1)
22 k = 0
23 xk=x0
24
25 while k<M | N>E
26     xk1=Resolve_Lx(L+D,b-U*xk)
27     N=norm(xk1-xk,norma) //norma da diferenca entre as aproximacoes
28     xk=xk1
29     k=k+1
30     if k>M
31         break
32     end
33 end
34
35 endfunction
```

Testando com os valores passados em sala e com número máximo de iterações e tolerância distintos temos:

```
1  --> A = [3 1 0; -1 -5 2; 1 -2 6]
2   A  =
3
4     3.    1.    0.
5    -1.   -5.    2.
6     1.   -2.    6.
7
8  --> b=[2;8;15]
9   b  =
10
11    2.
12    8.
13    15.
14
15 --> x0=[0;0;0]
16  x0  =
17
18    0.
19    0.
20    0.
21
22 --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-1),10,%inf)
23  xk  =
24
25    1.0000063
26   -1.0000053
27    1.9999972
28  N  =
29
30    0.0000163
31  k  =
32
```

```
   10.
 rk  =

   2.480D-08

-----------------------------------------------------------------

--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-2),15,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   2.779D-08
 k  =

   15.
 rk  =

   0.0000145

-----------------------------------------------------------------

--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),20,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   4.741D-11
 k  =

   20.
 rk  =

   2.480D-08

-----------------------------------------------------------------
--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),14,%inf)
 xk  =

   1.
  -1.
   2.
 N  =

   9.944D-08
 k  =

   14.
 rk  =

   15.
```

```
 
-----------------------------------------------------------------
--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),13,%inf)
 xk  =

   1.0000001
  -1.0000001
   1.9999999
 N  =

   0.0000004
 k  =

   13.
 rk  =

   15.
```

Fazendo 14 iterações com uma tolerância de 10^{-5} já podemos obter um $x_k = A\backslash b$, logo o segundo método de Gauss-Seidel converge.

4 Testando o sistema 1 (Questão 3)

```
--> A=[1 -4 2; 0 2 4; 6 -1 -2]
 A  =

   1.   -4.    2.
   0.    2.    4.
   6.   -1.   -2.

--> b=[2;1;1]
 b  =

   2.
   1.
   1.

--> x0=[0;0;0]
 x0  =

   0.
   0.
   0.

--> [xk,N,k,rk]= Jacobi_Method(A,b,x0 ,10^(-5),15,%inf)
 xk  =

   9796164.5
   30813363.
  -62520888.
 N  =

   47114207.
 k  =

```

```
   16.
 rk  =

   2.

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0 ,10^(-5),15,%inf)
 xk  =

   3.643D+10
   9.096D+09
   1.048D+11
 N  =

   1.093D+11
 k  =

   16.
 rk  =

   2.

--> [xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^(-5),15,%inf)
 xk  =

   3.643D+10
   9.096D+09
   1.048D+11
 N  =

   1.093D+11
 k  =

   16.
 rk  =

   2.

--> A\b
 ans  =

   0.25
  -0.25
   0.375
```

É possível ver que, usando o sistema dado, nenhum dos métodos convergiu. Pelas propriedades de convergência de ambos os métodos, tem-se que independente do vetor x_0 passado, a solução sempre irá convergir para a solução de $Ax = b$ quando o raio espectral da matriz de iteração for menor que 1, ou ainda, é possível garantir que irá convergir se a matriz A for uma matriz com diagonal estritamente dominante ($|a_{ii}| > \sum_{j \neq i} |a_{ij}|$), o que significa que o módulo do elemento de uma linha presente na diagonal deve ser maior que a soma dos módulos dos demais elementos na mesma linha, e é visível que nenhum dos dois ocorre na matriz A passada:

```

```

```
--> A=[1  -4 2; 0 2 4; 6  -1  -2]
 A  =

   1.  -4.   2.
   0.   2.   4.
   6.  -1.  -2.

--> D=diag(diag(A))
 D  =

   1.   0.   0.
   0.   2.   0.
   0.   0.  -2.

--> L=tril(A,-1)
 L  =

   0.   0.   0.
   0.   0.   0.
   6.  -1.   0.

--> U=triu(A,1)
 U  =

   0.  -4.   2.
   0.   0.   4.
   0.   0.   0.

--> invD=diag(1./diag(D))
 invD  =

   1.   0.    0.
   0.   0.5   0.
   0.   0.   -0.5

--> LPU = A-D
 LPU  =

   0.  -4.   2.
   0.   0.   4.
   6.  -1.   0.

--> MJ=(-1)*invD*LPU
 MJ  =

   0.   4.   -2.
   0.   0.   -2.
   3.  -0.5   0.

--> raiospec=max(abs(spec(MJ)))
 raiospec  =

   3.2192026
```

$$|a_{11}| = 1 < \sum_{j \neq i} |a_{1j}| = |a_{12}| + |a_{13}| = 4 + 2 = 6$$

$|a_{22}| = 2 < \sum_{j \neq i} |a_{2j}| = |a_{21}| + |a_{23}| = 0 + 4 = 4$

$|a_{33}| = 2 < \sum_{j \neq i} |a_{3j}| = |a_{31}| + |a_{32}| = 6 + 1 = 7$

Agora, se multiplicarmos A à esquerda por

```
P=[0 0 1; 1 0 0; 0 1 0]
```

teremos:

```
--> A=[1  -4 2; 0 2 4; 6  -1  -2]
 A  =

   1.  -4.   2.
   0.   2.   4.
   6.  -1.  -2.

--> P=[0 0 1; 1 0 0; 0 1 0]
 P  =

   0.   0.   1.
   1.   0.   0.
   0.   1.   0.

--> A=P*A
 A  =

   6.  -1.  -2.
   1.  -4.   2.
   0.   2.   4.

--> D=diag(diag(A))
 D  =

   6.   0.   0.
   0.  -4.   0.
   0.   0.   4.

--> L=tril(A,-1)
 L  =

   0.   0.   0.
   1.   0.   0.
   0.   2.   0.

--> U=triu(A,1)
 U  =

   0.  -1.  -2.
   0.   0.   2.
   0.   0.   0.

--> invD=diag(1./diag(D))
 invD  =

   0.1666667   0.      0.
   0.         -0.25    0.
   0.          0.      0.25
```

```
--> LPU = A-D
 LPU  =

   0.   -1.   -2.
   1.    0.    2.
   0.    2.    0.

--> MJ=(-1)*invD*LPU
 MJ  =

   0.       0.1666667   0.3333333
   0.25     0.          0.5
   0.      -0.5         0.

--> raiospec=max(abs(spec(MJ)))
 raiospec  =

   0.4886536
```

$$|a_{11}| = 6 > \sum_{j \neq i} |a_{1j}| = |a_{12}| + |a_{13}| = 1 + 2 = 3$$

$$|a_{22}| = 4 > \sum_{j \neq i} |a_{2j}| = |a_{21}| + |a_{23}| = 1 + 2 = 3$$

$$|a_{33}| = 4 > \sum_{j \neq i} |a_{3j}| = |a_{31}| + |a_{32}| = 0 + 2 = 2$$

```
--> A=[6   -1   -2; 1   -4 2; 0 2 4]
 A  =

   6.   -1.   -2.
   1.   -4.    2.
   0.    2.    4.


--> b=[1;2;1]
 b  =

   1.
   2.
   1.

--> x0=[0;0;0]
 x0  =

   0.
   0.
   0.

--> A\b
 ans  =

   0.25
  -0.25
   0.375

--> [xk,N,k,rk]=Jacobi_Method(A,b,x0,10^(-5),15,%inf)
```

```
 xk  =

   0.2499992
  -0.2500004
   0.375006
 N  =

   0.0000116
 k  =

   16.
 rk  =

   2.

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),15,%inf)
 xk  =

   0.25
  -0.25
   0.375
 N  =

   3.881D-09
 k  =

   15.
 rk  =

   2.

--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),15,%inf)
 xk  =

   0.25
  -0.25
   0.375
 N  =

   3.881D-09
 k  =

   15.
 rk  =

   2.
```

Sabendo que $A\backslash b = [0.25; -0.25; 0.375]$, temos que para a matriz A com diagonal estritamente dominante e com raio espectral menor que 1, os métodos convergem para a solução de $Ax = b$, embora o Método de Jacobi não consiga uma precisão tão exata quanto os Métodos de Gauss-Seidel usando o mesmo número máximo de iterações e tolerância.

5 Testando o sistema 2 (Questão 4)

```
--> A=[2 -1 1; 2 2 2; -1 -1 2]
 A  =

   2.  -1.   1.
   2.   2.   2.
  -1.  -1.   2.

--> b=[-1;4;-5]
 b  =

  -1.
   4.
  -5.

--> x0=[0;0;0]
 x0  =

   0.
   0.
   0.

--> [xk,N,k,rk]= Jacobi_Method(A,b,x0 ,10^( -5) ,25,%inf)
 xk  =

   11.913936
   45.655746
  -11.913936
 N  =

   43.655746
 k  =

   26.
 rk  =

   5.

-------------------------------------------------------------------

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0 ,10^(-5),25,%inf)
 xk  =

   1.0000006
   1.9999993
  -1.
 N  =

   0.0000020
 k  =

   25.
 rk  =

   5.

--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0 ,10^(-5),25,%inf)
 xk  =
```

```
 
   1.0000006
   1.9999993
  -1.
 N  =

   0.0000020
 k  =

   25.
 rk  =

   5.
```

Sabendo que a solução x para o sistema $Ax = b$ é $[1; 2; -1]$, é possível ver que os métodos de Gauss-Seidel foram boas aproximações para a resolução do sistema, com uma margem de erro de 10^{-7} casas decimais para a solução correta, enquanto que o método de Jacobi teve um resultado bem diferente do esperado.

6 Testando o sistema 3 (Questão 5)

```

--> A=[1 0 -1; -0.5 1 -0.25; 1 -0.5 1]
 A  =

   1.     0.    -1.
  -0.5    1.    -0.25
   1.    -0.5    1.

--> b=[0.2; -1.425; 2]
 b  =

   0.2
  -1.425
   2.

--> x0=[0;0;0]
 x0  =

   0.
   0.
   0.

--> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-2),300,%inf)
 xk  =

   0.9
  -0.8
   0.7
 N  =

   0.
 k  =

   300.
```

```
 rk  =

   2.

--> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-2),300,%inf)
 xk  =

   0.9
  -0.8
   0.7
 N  =

   2.220D-16
 k  =

   300.
 rk  =

   2.

-----------------------------------------------------------------

--> A=[1 0 -2; -0.5 1 -0.25; 1 -0.5 1]
 A  =

   1.    0.   -2.
  -0.5   1.   -0.25
   1.   -0.5   1.

-->  [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-2),300,%inf)
 xk  =

  -2.966D+41
  -1.854D+41
   2.039D+41
 N  =

   5.123D+41
 k  =

   301.
 rk  =

   2.

-->  [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-2),300,%inf)
 xk  =

  -2.966D+41
  -1.854D+41
   2.039D+41
 N  =

   5.123D+41
 k  =

   301.
```

```
 rk  =

   2.
```

Para a matriz A com $a_{13} = -1$, calculando o raio espectral da sua matriz MG obtemos 0.625, então mesmo que a matriz A original não tenha uma diagonal estritamente dominante ($|a_{11}| = |a_{12}| + |a_{13}| = 1$), o método iterativo ainda funciona porque o raio espectral de A é menor que 1, o que é necessário para que as iterações do método convirjam.

Fazendo com que a_{13} passe de -1 para -2 por outro lado, o raio espectral da matriz fica maior que 1 (1.375), e portanto os métodos não vão convergir.

```
--> A=[1 0 -1; -0.5 1 -0.25; 1 -0.5 1]
 A  =

   1.     0.    -1.
  -0.5    1.    -0.25
   1.    -0.5    1.

--> D=diag(diag(A))
 D  =

   1.    0.    0.
   0.    1.    0.
   0.    0.    1.

--> L=tril(A,-1)
 L  =

   0.     0.     0.
  -0.5    0.     0.
   1.    -0.5    0.

--> U=triu(A,1)
 U  =

   0.    0.   -1.
   0.    0.   -0.25
   0.    0.    0.

--> MG=(-1)*(inv(L+D))*(U)
 MG  =

   0.    0.    1.
   0.    0.    0.75
   0.    0.   -0.625

--> raiospec=max(abs(spec(MG)))
 raiospec  =

   0.625

-------------------------------------------------------------------

--> A=[1 0 -2; -0.5 1 -0.25; 1 -0.5 1]
```

```
 A  =

   1.     0.    -2.
  -0.5    1.    -0.25
   1.    -0.5    1.

--> D=diag(diag(A))
 D  =

   1.    0.    0.
   0.    1.    0.
   0.    0.    1.

--> L=tril(A,-1)
 L  =

   0.     0.     0.
  -0.5    0.     0.
   1.    -0.5    0.

--> U=triu(A,1)
 U  =

   0.    0.   -2.
   0.    0.   -0.25
   0.    0.    0.

--> MG=(-1)*(inv(L+D))*(U)
 MG  =

   0.    0.    2.
   0.    0.    1.25
   0.    0.   -1.375

--> raiospec=max(abs(spec(MG)))
 raiospec  =

   1.375
```

7 Testando matrizes aleatórias com diagonal estritamente dominante (Questão 6)

```
--> A=rand(10,10);D=diag(diag(A)+10);A=A+D;

--> b=rand(10,1);

--> x0=zeros(10,1);

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
    );t1=toc()
 t1  =

   0.0314419

```

```
--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
 t2  =

   0.0026701

--> A\b
 ans  =

   0.0207198
   0.0129233
   0.0442663
   0.0001204
   0.046471
   0.0544607
   0.0168831
  -0.0084929
   0.039264
   0.0230345

-------------------------------------------------------------

--> A=rand(100,100);D=diag(diag(A)+100);A=A+D;

--> b=rand(100,1);

--> x0=zeros(100,1);

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
    );t1=toc()
 t1  =

   0.002301

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
 t2  =

   0.0222825

-------------------------------------------------------------

--> A=rand(1000,1000);D=diag(diag(A)+1000);A=A+D;

--> b=rand(1000,1);

--> x0=zeros(1000,1);

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
    );t1=toc()
 t1  =

   0.1222208

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
 t2  =
```

```
   0.5862368

-------------------------------------------------------------------

--> A=rand(2000,2000);D=diag(diag(A)+2000);A=A+D;

--> b=rand(2000,1);

--> x0=zeros(2000,1);

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
    );t1=toc()
 t1  =

   1.6976052

--> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
 t2  =

   1.9643495

-------------------------------------------------------------------

--> A=rand(3000,3000);D=diag(diag(A)+3000);A=A+D;

--> b=rand(3000,1);

--> x0=zeros(3000,1);

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,25,%inf); t1=toc()
 t1  =

   6.1454899

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,25,%inf); t2=toc()
 t2  =

   4.3633564

-------------------------------------------------------------------

--> A=rand(5000,5000);D=diag(diag(A)+5000);A=A+D;

--> b=rand(5000,1);

--> x0=zeros(5000,1);

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,25,%inf); t1=toc()
 t1  =

   38.044012
```

```
--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,25,%inf); t2=toc()
 t2  =

   11.643595
```

Para garantir que as matrizes tivessem a diagonal estritamente dominante, foram criadas matrizes nxn aleatórias usando a função *rand* do Scilab, o que faz com que cada elemento na matriz tenha um valor entre 0 e 1. Assim, foi criada uma matriz diagonal D que incrementa os valores da diagonal de A por n já que, tendo uma matriz aleatória, a soma em cada linha será menor que n, dado que a função *rand* cria números em um intervalo $[0, 1)$. Dessa forma, aumentando o valor do elemento na diagonal principal em n, teremos garantia que esse elemento é maior que a soma dos módulos dos demais elementos na linha, visto que essa soma será menor que n, enquanto que o elemento na diagonal principal estará em um intervalo $n \leq x < n + 1$.

Fazendo 25 iterações, é possível ver que até $n = 2000$, o primeiro método de Gauss-Seidel é mais rápido que o segundo, a não ser para $n = 10$. Para números maiores, contados a partir de $n = 3000$, o segundo método foi mais rápido. Assim, é possível perceber que, tirando para números bem pequenos, o método que usa a própria função *inv* do Scilab é mais rápida que resolvendo o sistema linear, porém o primeiro método perde sua vantagem para números pelo menos maiores que 3000. Isso porque a partir de certo ponto, leva mais tempo para encontrar a inversa da matriz usando a função *inv* do que resolver o sistema a cada iteração.

```
--> A=rand(10,10);D=diag(diag(A)+10);A=A+D;

--> b=rand(10,1);

--> x0=zeros(10,1);

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
 t1  =

   0.0011312

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
 t2  =

   0.0113768

-------------------------------------------------------------------

--> A=rand(100,100);D=diag(diag(A)+100);A=A+D;

--> b=rand(100,1);

--> x0=zeros(100,1);

```

```
--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
 t1  =

   0.0046774

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
 t2  =

   0.0838892

-------------------------------------------------------------------

--> A=rand(1000,1000);D=diag(diag(A)+1000);A=A+D;

--> b=rand(1000,1);

--> x0=zeros(1000,1);

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
 t1  =

   0.1363375

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
 t2  =

   2.2565777

-------------------------------------------------------------------

--> A=rand(2000,2000);D=diag(diag(A)+2000);A=A+D;

--> b=rand(2000,1);

--> x0=zeros(2000,1);

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
 t1  =

   1.9136224

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
 t2  =

   7.6570308

-------------------------------------------------------------------

--> A=rand(3000,3000);D=diag(diag(A)+3000);A=A+D;

--> b=rand(3000,1);
```

```
76
77 --> x0=zeros(3000,1);
78
79 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
       ,100,%inf); t1=toc()
80  t1  =
81
82    6.5221472
83
84 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
       ,100,%inf); t2=toc()
85  t2  =
86
87    16.588227
88
89 ---------------------------------------------------------------
90
91 --> A=rand(5000,5000);D=diag(diag(A)+5000);A=A+D;
92
93 --> b=rand(5000,1);
94
95 --> x0=zeros(5000,1);
96
97 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
       ,100,%inf); t1=toc()
98  t1  =
99
100   30.014090
101
102 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
       ,100,%inf); t2=toc()
103  t2  =
104
105   43.680670
106
107 ---------------------------------------------------------------
108
109 --> A=rand(7000,7000);D=diag(diag(A)+7000);A=A+D;
110
111 --> b=rand(7000,1);
112
113 --> x0=zeros(7000,1);
114
115 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
       ,100,%inf); t1=toc()
116  t1  =
117
118   84.906025
119
120 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
       ,100,%inf); t2=toc()
121  t2  =
122
123   89.345658
124
125 ---------------------------------------------------------------
126
```

```
--> A=rand(8000,8000);D=diag(diag(A)+8000);A=A+D;

--> b=rand(8000,1);

--> x0=zeros(8000,1);

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
 t1  =

   136.45289

--> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
 t2  =

   118.44110
```

Fazendo 100 iterações, é possível ver que até $n = 7000$, o primeiro método de Gauss-Seidel é mais rápido que o segundo, incluindo valores menores como o próprio $n = 10$. Para valores maiores ou iguais a 8000, por outro lado, o segundo método, que precisa resolver o sistema linear para ser calculado, acaba ficando mais rápido. Assim, mesmo que iterando mais vezes a função *inv* ainda seja mais vantajosa até certo n, para matrizes muito grandes é mais benéfico para o programa resolver o sistema ao invés de calcular a sua inversa.