

Aula Prática 2 (ALN)

Raphael F. Levy

April 8, 2021

1 Análise do Método Iterativo de Jacobi

Primeiramente, segue a função criada:

```
1 function [xk,N,k,rk]=Jacobi_Method(A,b,x0,E,M,norma)
2
3 rk=norm(b-A*xk,norma) //Residuo da operacao
4 D=diag(diag(A)) //Diagonal de A
5 LPU = A-D //L plus U
6 invD=diag(1./diag(D)) //Inversa da matriz D
7 MJ=(-1)*invD*LPU //Matriz de Jacobi
8 cJ=invD*b //Vetor de Jacobi
9 k = 0
10 xk=x0
11
12 while k<M | N>E
13     xk1=MJ*xk+cJ
14     N=norm(xk1-xk,norma) //norma da diferenca entre as aproximacoes
15     xk=xk1
16     k=k+1
17     if k>M
18         break
19     end
20 end
21
22 endfunction
```

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

```
1
2 --> A = [3 1 0; -1 -5 2; 1 -2 6]
3 A =
4
5     3.    1.    0.
6    -1.   -5.    2.
7     1.   -2.    6.
8
9 --> b=[2;8;15]
10 b =
11
12     2.
```

```

13      8.
14      15.
15
16 --> A\b
17 ans =
18
19      1.
20     -1.
21      2.
22
23
24 --> x0=[0;0;0]
25 x0 =
26
27      0.
28      0.
29      0.
30
31 --> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-1),10,%inf)
32 xk =
33
34      1.0000793
35     -0.9998683
36      1.9999686
37 N =
38
39      0.0003696
40 k =
41
42      10.
43 rk =
44
45      9.457D-08
46
47 -----
48
49 --> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-2),15,%inf)
50 xk =
51
52      0.9999988
53     -1.0000001
54      2.000001
55 N =
56
57      0.0000037
58 k =
59
60      15.
61 rk =
62
63      0.0008005
64
65 -----
66
67 --> [xk,N,k, rk]= Jacobi_Method(A,b,x0 ,10^(-5),20,%inf)
68 xk =
69

```

```

70      1.
71     -1.
72      2.
73     N   =
74
75     3.398D-08
76     k   =
77
78     20.
79     rk  =
80
81     15.
82
83     -----
84
85     --> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-5),30,%inf)
86     xk   =
87
88     1.
89     -1.
90     2.
91     N   =
92
93     1.791D-11
94     k   =
95
96     30.
97     rk  =
98
99     0.0000052
100
101     -----
102
103     --> [xk,N,k, rk]=Jacobi_Method(A,b,x0,10^(-5),19,%inf)
104     xk   =
105
106     1.
107     -1.
108     2.
109     N   =
110
111     0.0000001
112     k   =
113
114     19.
115     rk  =
116
117     0.0000052
118
119     -----
120
121     --> [xk,N,k, rk]= Jacobi_Method(A,b,x0,10^(-5),18,%inf)
122     xk   =
123
124     1.
125     -0.9999999
126     2.0000001

```

```

127 N =
128
129     0.0000002
130 k =
131
132     18.
133 rk =
134
135     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

```

1 function [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,E,M,norma)
2
3 rk=norm(b-A*xk,norma) //Residuo da operacao
4 D=diag(diag(A)) //Diagonal de A
5 LPU = A-D //L plus U
6 invD=diag(1./diag(D)) //Inversa da matriz D
7 L=tril(A,-1)
8 U=triu(A,1)
9 MG=(-1)*(inv(L+D))*(U) //Matriz de Gauss-Seidel
10 cG=(inv(L+D))*(b) //Gauss-Seidel
11 k = 0
12 xk=x0
13
14 while k<M | N>E
15     xk1=MG*xk+cG
16     N=norm(xk1-xk,norma) //norma da diferenca entre as aproximacoes
17     xk=xk1
18     k=k+1
19     if k>M
20         break
21     end
22 end
23
24 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_1(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
33     10.
34 rk =
35
36     0.0000145
37
38 -----
39
40 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-2),15,%inf)
41 xk =
42
43     1.
44    -1.
45     2.
46 N =
47
48     2.779D-08
49 k =
50
51     15.
52 rk =
53
54     0.0000145
55
56 -----
57
58 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),20,%inf)
59 xk =
60
61     1.
62    -1.
63     2.
64 N =
65
66     4.741D-11
67 k =
68
69     20.
70 rk =

```

```

71      4.231D-11
72
73
74 -----
75 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),14,%inf)
76 xk  =
77
78      1.
79     -1.
80      2.
81 N    =
82
83     9.944D-08
84 k    =
85
86     14.
87 rk   =
88
89     15.
90
91 -----
92 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),13,%inf)
93 xk  =
94
95     1.0000001
96    -1.0000001
97     1.9999999
98 N    =
99
100     0.0000004
101 k    =
102
103     13.
104 rk   =
105
106     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

```

1 function [x] = Resolve_Lx(L,b)
2
3 [t]=size(L,1);
4 x=zeros(t,1);
5
6 // Calcula x, sendo Lx=b
7
8 x(1)=b(1)/L(1,1);
9 for i=2:t
10     x(i)=(b(i)-L(i,1:i)*x(1:i))/L(i,i);
11 end
12
13 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

```

```

33     10.
34     rk  =
35
36     2.480D-08
37
38     -----
39
40     --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-2),15,%inf)
41     xk  =
42
43     1.
44     -1.
45     2.
46     N   =
47
48     2.779D-08
49     k   =
50
51     15.
52     rk  =
53
54     0.0000145
55
56     -----
57
58     --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),20,%inf)
59     xk  =
60
61     1.
62     -1.
63     2.
64     N   =
65
66     4.741D-11
67     k   =
68
69     20.
70     rk  =
71
72     2.480D-08
73
74     -----
75     --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),14,%inf)
76     xk  =
77
78     1.
79     -1.
80     2.
81     N   =
82
83     9.944D-08
84     k   =
85
86     14.
87     rk  =
88
89     15.

```



```

90 -----
91
92 --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),13,%inf)
93 xk  =
94
95     1.0000001
96    -1.0000001
97     1.9999999
98 N    =
99
100    0.0000004
101 k     =
102
103     13.
104 rk    =
105
106     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)

```

1 --> A=[1 -4 2; 0 2 4; 6 -1 -2]
2 A  =
3
4     1.    -4.     2.
5     0.     2.     4.
6     6.    -1.    -2.
7
8 --> b=[2;1;1]
9 b  =
10
11     2.
12     1.
13     1.
14
15 --> x0=[0;0;0]
16 x0  =
17
18     0.
19     0.
20     0.
21
22 --> [xk,N,k,rk]= Jacobi_Method(A,b,x0 ,10^(-5),15,%inf)
23 xk  =
24
25    9796164.5
26    30813363.
27   -62520888.
28 N    =
29
30    47114207.
31 k     =
32

```

```

33      16.
34      rk =
35
36      2.
37
38 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0 ,10^(-5),15,%inf)
39      xk =
40
41      3.643D+10
42      9.096D+09
43      1.048D+11
44      N =
45
46      1.093D+11
47      k =
48
49      16.
50      rk =
51
52      2.
53
54 --> [xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^(-5),15,%inf)
55      xk =
56
57      3.643D+10
58      9.096D+09
59      1.048D+11
60      N =
61
62      1.093D+11
63      k =
64
65      16.
66      rk =
67
68      2.
69
70 --> A\b
71      ans =
72
73      0.25
74      -0.25
75      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:

```

2 --> A=[1 -4 2; 0 2 4; 6 -1 -2]
3 A =
4
5     1.    -4.     2.
6     0.     2.     4.
7     6.    -1.    -2.
8
9 --> D=diag(diag(A))
10 D =
11
12     1.     0.     0.
13     0.     2.     0.
14     0.     0.    -2.
15
16 --> L=tril(A,-1)
17 L =
18
19     0.     0.     0.
20     0.     0.     0.
21     6.    -1.     0.
22
23 --> U=triu(A,1)
24 U =
25
26     0.    -4.     2.
27     0.     0.     4.
28     0.     0.     0.
29
30 --> invD=diag(1./diag(D))
31 invD =
32
33     1.     0.     0.
34     0.    0.5     0.
35     0.     0.   -0.5
36
37 --> LPU = A-D
38 LPU =
39
40     0.    -4.     2.
41     0.     0.     4.
42     6.    -1.     0.
43
44 --> MJ=(-1)*invD*LPU
45 MJ =
46
47     0.     4.    -2.
48     0.     0.    -2.
49     3.   -0.5     0.
50
51 --> raiospec=max(abs(spec(MJ)))
52 raiospec =
53
54     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

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

teremos:

```
1 --> A=[1 -4 2; 0 2 4; 6 -1 -2]
2 A =
3
4     1.    -4.     2.
5     0.     2.     4.
6     6.    -1.    -2.
7
8 --> P=[0 0 1; 1 0 0; 0 1 0]
9 P =
10
11     0.     0.     1.
12     1.     0.     0.
13     0.     1.     0.
14
15 --> A=P*A
16 A =
17
18     6.    -1.    -2.
19     1.    -4.     2.
20     0.     2.     4.
21
22 --> D=diag(diag(A))
23 D =
24
25     6.     0.     0.
26     0.    -4.     0.
27     0.     0.     4.
28
29 --> L=tril(A,-1)
30 L =
31
32     0.     0.     0.
33     1.     0.     0.
34     0.     2.     0.
35
36 --> U=triu(A,1)
37 U =
38
39     0.    -1.    -2.
40     0.     0.     2.
41     0.     0.     0.
42
43 --> invD=diag(1./diag(D))
44 invD =
45
46     0.1666667     0.     0.
47     0.          -0.25     0.
48     0.           0.     0.25
```

```

49
50 --> LPU = A-D
51 LPU =
52
53     0.   -1.   -2.
54     1.    0.    2.
55     0.    2.    0.
56
57 --> MJ=(-1)*invD*LPU
58 MJ =
59
60     0.         0.1666667   0.3333333
61     0.25      0.         0.5
62     0.        -0.5        0.
63
64 --> raiospec=max(abs(spec(MJ)))
65 raiospec =
66
67     0.4886536

```

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

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

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

```

1 --> A=[6   -1   -2; 1   -4   2; 0   2   4]
2 A =
3
4     6.   -1.   -2.
5     1.   -4.    2.
6     0.    2.    4.
7
8
9 --> b=[1;2;1]
10 b =
11
12     1.
13     2.
14     1.
15
16 --> x0=[0;0;0]
17 x0 =
18
19     0.
20     0.
21     0.
22
23 --> A\b
24 ans =
25
26     0.25
27    -0.25
28     0.375
29
30 --> [xk,N,k,rk]=Jacobi_Method(A,b,x0,10^(-5),15,%inf)

```

```

31 xk =
32
33     0.2499992
34    -0.2500004
35     0.375006
36 N =
37
38     0.0000116
39 k =
40
41     16.
42 rk =
43
44     2.
45
46 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-5),15,%inf)
47 xk =
48
49     0.25
50    -0.25
51     0.375
52 N =
53
54     3.881D-09
55 k =
56
57     15.
58 rk =
59
60     2.
61
62 --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-5),15,%inf)
63 xk =
64
65     0.25
66    -0.25
67     0.375
68 N =
69
70     3.881D-09
71 k =
72
73     15.
74 rk =
75
76     2.

```

Sabendo que $A \setminus b = [0.25; -0.25; 0.375]$, 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)

```

1 --> A=[2 -1 1; 2 2 2; -1 -1 2]
2   A  =
3
4       2.   -1.   1.
5       2.    2.   2.
6      -1.   -1.   2.
7
8 --> b=[-1;4;-5]
9   b  =
10
11      -1.
12       4.
13      -5.
14
15 --> x0=[0;0;0]
16   x0  =
17
18       0.
19       0.
20       0.
21
22 --> [xk,N,k,rk]= Jacobi_Method(A,b,x0 ,10^(-5) ,25,%inf)
23   xk  =
24
25      11.913936
26      45.655746
27     -11.913936
28   N   =
29
30      43.655746
31   k   =
32
33      26.
34   rk  =
35
36       5.
37
38 -----
39
40 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0 ,10^(-5) ,25,%inf)
41   xk  =
42
43      1.0000006
44      1.9999993
45      -1.
46   N   =
47
48      0.0000020
49   k   =
50
51      25.
52   rk  =
53
54       5.
55
56 --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0 ,10^(-5) ,25,%inf)
57   xk  =

```

```

58      1.0000006
59      1.9999993
60      -1.
61      N =
62
63      0.0000020
64      k =
65
66      25.
67      rk =
68
69      5.
70

```

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)

```

1
2 --> A=[1 0 -1; -0.5 1 -0.25; 1 -0.5 1]
3 A =
4
5      1.      0.     -1.
6     -0.5      1.    -0.25
7      1.     -0.5      1.
8
9 --> b=[0.2; -1.425; 2]
10 b =
11
12      0.2
13     -1.425
14      2.
15
16 --> x0=[0;0;0]
17 x0 =
18
19      0.
20      0.
21      0.
22
23 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-2),300,%inf)
24 xk =
25
26      0.9
27     -0.8
28      0.7
29 N =
30
31      0.
32 k =
33
34     300.

```



```

35 rk =
36
37 2.
38
39 --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-2),300,%inf)
40 xk =
41
42 0.9
43 -0.8
44 0.7
45 N =
46
47 2.220D-16
48 k =
49
50 300.
51 rk =
52
53 2.
54
55 -----
56
57 --> A=[1 0 -2; -0.5 1 -0.25; 1 -0.5 1]
58 A =
59
60 1. 0. -2.
61 -0.5 1. -0.25
62 1. -0.5 1.
63
64 --> [xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-2),300,%inf)
65 xk =
66
67 -2.966D+41
68 -1.854D+41
69 2.039D+41
70 N =
71
72 5.123D+41
73 k =
74
75 301.
76 rk =
77
78 2.
79
80 --> [xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-2),300,%inf)
81 xk =
82
83 -2.966D+41
84 -1.854D+41
85 2.039D+41
86 N =
87
88 5.123D+41
89 k =
90
91 301.

```

```

92 rk =
93
94 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 converjam.

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.

```

1 --> A=[1 0 -1; -0.5 1 -0.25; 1 -0.5 1]
2 A =
3
4      1.      0.     -1.
5     -0.5      1.    -0.25
6      1.    -0.5      1.
7
8 --> D=diag(diag(A))
9 D =
10
11      1.      0.      0.
12      0.      1.      0.
13      0.      0.      1.
14
15 --> L=tril(A,-1)
16 L =
17
18      0.      0.      0.
19     -0.5      0.      0.
20      1.    -0.5      0.
21
22 --> U=triu(A,1)
23 U =
24
25      0.      0.     -1.
26      0.      0.    -0.25
27      0.      0.      0.
28
29 --> MG=(-1)*(inv(L+D))*(U)
30 MG =
31
32      0.      0.      1.
33      0.      0.      0.75
34      0.      0.    -0.625
35
36 --> raiospec=max(abs(spec(MG)))
37 raiospec =
38
39      0.625
40
41 -----
42
43 --> A=[1 0 -2; -0.5 1 -0.25; 1 -0.5 1]

```

```

44 A =
45
46     1.     0.    -2.
47    -0.5     1.   -0.25
48     1.   -0.5     1.
49
50 --> D=diag(diag(A))
51 D =
52
53     1.     0.     0.
54     0.     1.     0.
55     0.     0.     1.
56
57 --> L=tril(A,-1)
58 L =
59
60     0.     0.     0.
61    -0.5     0.     0.
62     1.   -0.5     0.
63
64 --> U=triu(A,1)
65 U =
66
67     0.     0.    -2.
68     0.     0.   -0.25
69     0.     0.     0.
70
71 --> MG=(-1)*(inv(L+D))*(U)
72 MG =
73
74     0.     0.     2.
75     0.     0.     1.25
76     0.     0.    -1.375
77
78 --> raiospec=max(abs(spec(MG)))
79 raiospec =
80
81     1.375

```

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

```

1 --> A=rand(10,10);D=diag(diag(A)+10);A=A+D;
2
3 --> b=rand(10,1);
4
5 --> x0=zeros(10,1);
6
7 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf);t1=toc()
8 t1 =
9
10     0.0314419
11

```

```

12 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
13 t2 =
14
15     0.0026701
16
17 --> A\b
18 ans =
19
20     0.0207198
21     0.0129233
22     0.0442663
23     0.0001204
24     0.046471
25     0.0544607
26     0.0168831
27    -0.0084929
28     0.039264
29     0.0230345
30
31 -----
32
33 --> A=rand(100,100);D=diag(diag(A)+100);A=A+D;
34
35 --> b=rand(100,1);
36
37 --> x0=zeros(100,1);
38
39 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
    );t1=toc()
40 t1 =
41
42     0.002301
43
44 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
45 t2 =
46
47     0.0222825
48
49 -----
50
51 --> A=rand(1000,1000);D=diag(diag(A)+1000);A=A+D;
52
53 --> b=rand(1000,1);
54
55 --> x0=zeros(1000,1);
56
57 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
    );t1=toc()
58 t1 =
59
60     0.1222208
61
62 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
    );t2=toc()
63 t2 =

```

```

64      0.5862368
65
66
67 -----
68
69 --> A=rand(2000,2000);D=diag(diag(A)+2000);A=A+D;
70
71 --> b=rand(2000,1);
72
73 --> x0=zeros(2000,1);
74
75 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_1(A,b,x0,10^(-10),25,%inf
76     );t1=toc()
77
78     1.6976052
79
80 --> tic();[xk,N,k,rk]=Gauss_Seidel_Method_2(A,b,x0,10^(-10),25,%inf
81     );t2=toc()
82
83     1.9643495
84
85 -----
86
87 --> A=rand(3000,3000);D=diag(diag(A)+3000);A=A+D;
88
89 --> b=rand(3000,1);
90
91 --> x0=zeros(3000,1);
92
93 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
94     ,25,%inf); t1=toc()
95
96     6.1454899
97
98 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
99     ,25,%inf); t2=toc()
100
101     4.3633564
102
103 -----
104
105 --> A=rand(5000,5000);D=diag(diag(A)+5000);A=A+D;
106
107 --> b=rand(5000,1);
108
109 --> x0=zeros(5000,1);
110
111 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
112     ,25,%inf); t1=toc()
113
114     38.044012
115

```

```

116 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
      ,25,%inf); t2=toc()
117 t2 =
118
119 11.643595

```

Para garantir que as matrizes tivessem a diagonal estritamente dominante, foram criadas matrizes $n \times n$ 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.

```

1 --> A=rand(10,10);D=diag(diag(A)+10);A=A+D;
2
3 --> b=rand(10,1);
4
5 --> x0=zeros(10,1);
6
7 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
      ,100,%inf); t1=toc()
8 t1 =
9
10 0.0011312
11
12 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
      ,100,%inf); t2=toc()
13 t2 =
14
15 0.0113768
16
17 -----
18
19 --> A=rand(100,100);D=diag(diag(A)+100);A=A+D;
20
21 --> b=rand(100,1);
22
23 --> x0=zeros(100,1);
24

```

```

25 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
26 t1 =
27
28 0.0046774
29
30 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
31 t2 =
32
33 0.0838892
34
35 -----
36
37 --> A=rand(1000,1000);D=diag(diag(A)+1000);A=A+D;
38
39 --> b=rand(1000,1);
40
41 --> x0=zeros(1000,1);
42
43 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
44 t1 =
45
46 0.1363375
47
48 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
49 t2 =
50
51 2.2565777
52
53 -----
54
55 --> A=rand(2000,2000);D=diag(diag(A)+2000);A=A+D;
56
57 --> b=rand(2000,1);
58
59 --> x0=zeros(2000,1);
60
61 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
    ,100,%inf); t1=toc()
62 t1 =
63
64 1.9136224
65
66 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
    ,100,%inf); t2=toc()
67 t2 =
68
69 7.6570308
70
71 -----
72
73 --> A=rand(3000,3000);D=diag(diag(A)+3000);A=A+D;
74
75 --> 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

```



```

127 --> A=rand(8000,8000);D=diag(diag(A)+8000);A=A+D;
128
129 --> b=rand(8000,1);
130
131 --> x0=zeros(8000,1);
132
133 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_1(A,b,x0 ,10^( -10)
134         ,100,%inf); t1=toc()
135
136         136.45289
137
138 --> tic();[xk,N,k,rk]= Gauss_Seidel_Method_2(A,b,x0 ,10^( -10)
139         ,100,%inf); t2=toc()
140
141         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.