Atividade 3
Lucca Ferreira Paiva 240229

Questão-1)

```
>> Ex01

Letra A
num =  4054
N =  1138
p =  0.31304

Letra B
O tipo da matriz é: ans = Positive Definite
È simetrica

Letra C
c =  8572645.58671
Tempo =  0.0012531
Residuo1 =    1.5635e-16
Erro1 =    4.0011e-12
ans = 1

Letra D
Tempo1 =  0.00047398
Tempo2 =  0.098533
Residuo2 =    2.4892e-16
Erro2 =    2.7303e-12
ans = 1
```

A) a porcentagem de elementos não nulos em A é de 0,313%
B) A matriz A é definida positiva, além de ser simétrica. Este último pode ser verificado pela igualdade tranp(A) = A;

D) Pode-se ver que o tempo pelo algoritmo da fatoração de Cholesky é maior que o pelo comando “\”, se não tivermos a matriz G já pronta. Porém, no caso de ja se ter a matriz G, o tempo é muito menor.

Além disso, em ambos os casos a inequação Erro <= cond(A)*Resíduo é válida. É importante também notar que o erro e o resíduo para ambos os casos é são da mesma ordem, o que mostra que a diferença entre eles é muito pequena.

```
load("1138_bus.mat");
A = Problem.A;
#letra A
printf("Letra A\n")
num = nnz(A)
N = length(A)
p = 100*num/(N*N)

#Letra B
printf("\nLetra B\nO tipo da matriz é: ")
matrix_type(A)
if(A' == A)
  printf("È simetrica\n")
else
  printf("Nao é simetrica\n")
endif

#Letra C
printf("\nLetra C\n")
xsol = 0.5 + sin( 2*pi*(0:N-1)'/(N-1));
b = A*xsol;
c = cond(A)
tic; xn = A\b; Tempo = toc
Residuo1 = norm(b-A*xn)/norm(b)
Erro1 = norm(xsol-xn)/norm(xsol)

Erro1 < c*Residuo1

#Letra D
printf("\nLetra D\n")
G = chol(A);
Gt = G';
tic; xm = G\(Gt\b); Tempo1 = toc
tic; G = chol(A); xm = G\(G'\b); Tempo2 = toc
Residuo2 = norm(b-A*xm)/norm(b)
Erro2 = norm(xsol-xm)/norm(xsol)

Erro2 < c*Residuo2
```

Código utilizado na questão 1

Questao-2)

A) Dependa da quantidade de elementos necessários para se considerar uma matriz esparsa, se for de menos de 1%, somente a matriz 2 nao entra. Mas agora, se considerarmos 2%, a matriz 2 ja entra.

B) Para que um sistema possa ser resolvido por um desses métodos são necessárias duas condições, que a sua determinante seja diferente de zero, e que ele não possua nenhum elemento neutro na sua diagonal principal.

```
Caso:1
T = 1
N =  1000
p =  0.37500
nz =  1000
k = Compressed Column Sparse (rows = 1, cols = 1, nnz = 0 [0%])

d = 0
r =  1000
>> ex2

Caso:2
T = 1
N =  1080
p =  1.9799
nz =  1080
k = Compressed Column Sparse (rows = 1, cols = 1, nnz = 1 [100%])

  (1, 1) ->  Inf
d =  Inf
r =  1080
>> ex2

Caso:3
T = 1
N =  5005
p =  0.079972
nz =  5005
k = Compressed Column Sparse (rows = 1, cols = 1, nnz = 0 [0%])

d = 0
r =  2898
>> ex2

Caso:4
T = 1
N =  1104
p =  0.31063
nz =  1104
k = Compressed Column Sparse (rows = 1, cols = 1, nnz = 1 [100%])

  (1, 1) ->  Inf
d =  Inf
r =  1104
>> ex2

Caso:5
T = 1
N =  3312
p =  0.18956
nz =  3312
k = Compressed Column Sparse (rows = 1, cols = 1, nnz = 1 [100%])

  (1, 1) ->  Inf
d =  Inf
r =  3312
```

Pode-se ver que tanto a matriz 1 quanto a 3 possuem a determinante igual a 0, e portanto não podem ser usadas no método. E a matriz 2 tambem nao é valida por nao ser esparsa.

C) fas

```
Caso:4
T = 1
N =  1104
p =  0.31063
nz =  1104
d =  Inf
r =  1104
TempoJacobi =  0.012094
TempoGaussSeidel =  0.010600
TempoBarra =  0.0033960
c =  2178.6
Erro1 =  0.0063867
Erro2 =  0.0058608
Residuo1 =  0.67689
Residuo2 =  0.57662
Residuo3 =    6.0970e-14
ans = 1
ans = 1
```

No caso 4, pode-se ver que o tempo para a resolucao com o comando “\” e bem menor que os outros dois, e dentre estes, o metodo de Gauss-Seidel foi mais rapido que o de Jacobi. Alem disso, o erro desses dois metodos foi baixo, e o residuo foi muito menor pelo metodo “\”. É possivel tambem notar que Erro <= cond(A)*Resíduo e valido para ambos os metodos.

```
Caso:5
T = 1
N =  3312
p =  0.18956
nz =  3312
d =  Inf
r =  3312
TempoJacobi =  0.030731
TempoGaussSeidel =  0.025551
TempoBarra =  0.018180
c =  187940.83056
Erro1 =  1.6250
Erro2 =  1.3125
Residuo1 =    4.7460e+30
Residuo2 =    3.7587e+50
Residuo3 =    9.1206e-13
ans = 1
ans = 1
```

No caso 5, pode-se ver que o tempo para a resolução com o comando “\” é bem menor que os outros dois, e dentre estes, o método de Gauss-Seidel foi mais rápido que o de Jacobi. Além disso, o erro desses dois métodos foi baixo, e o resíduo foi muito menor pelo método “\”. É possível também notar que Erro <= cond(A)*Resíduo é válido para ambos os métodos.

```
caso = "5";
printf(strcat("Caso: ", caso, "\n"))
load(strcat("sherman", caso, ".mat"));
A = Problem.A;
b = Problem.b;

#Verificando esprcidade
T = issparse(A)
num = nnz(A);
N = length(A)
p = 100*num/(N*N)
nz = nnz(diag(A))
d = det(A)
#Verificando se pode ser resolvida
r = rank(A) # Se rank = N --> é nao singular(det != 0)


tic;[x1, Dr1] = MetodoJacobi(A, b); TempoJacobi = toc
tic;[x2, Dr2] = MetodoGaussSeidel(A, b); TempoGaussSeidel = toc
tic;x3 = A\b; TempoBarra = toc

#Comparacoes
c = cond(A)
Erro1 = Dr1(length(Dr1))
Erro2 = Dr2(length(Dr2))
Residuo1 = norm(b-A*x1)/norm(b)
Residuo2 = norm(b-A*x2)/norm(b)
Residuo3 = norm(b-A*x3)/norm(b)
Erro1 < c*Residuo1
Erro2 < c*Residuo2
```

Código para a questão 2, note que basta trocar o número do caso para mudar a matriz referente.

questao-3)

```
>> Ex03

indice =  1
m =  0.082343
```

O índice da página com maior relevância é 1. É o sistema foi resolvido pelo comando "\", isso porque como visto no primeiro exercício, o tempo de execução dele é menor que o do algoritmo de fatoração de Cholesky, e o erro e da mesma ordem.

```
function [P] = Prob(A)
  n = length(A);
  S = sum(A);
  f = 1/n;
  #P = (1/n)*ones(n , n);
  for i = 1:n
    for j = 1:n
      if(S(j) >= 1)
        P(i, j) = A(i, j)/S(j);
      else
        P(i, j) = f;
       endif
    endfor
  endfor
endfunction
```

```
load("Harvard500.mat");
A = Problem.A;
P = Prob(A);

alfa = 0.85;
n = length(P);
V = (1 - alfa)*ones(n)*(1/n);
B = (alfa*P - eye(n));
x = -B\V;
S = sum(x, 2);
m = max(S);
indice = find(m == max(S))
m = m/n
```

O primeiro é a função que computa a matriz P, e o segundo é o script para resolver o sistema, e os comandos para achar o máximo.