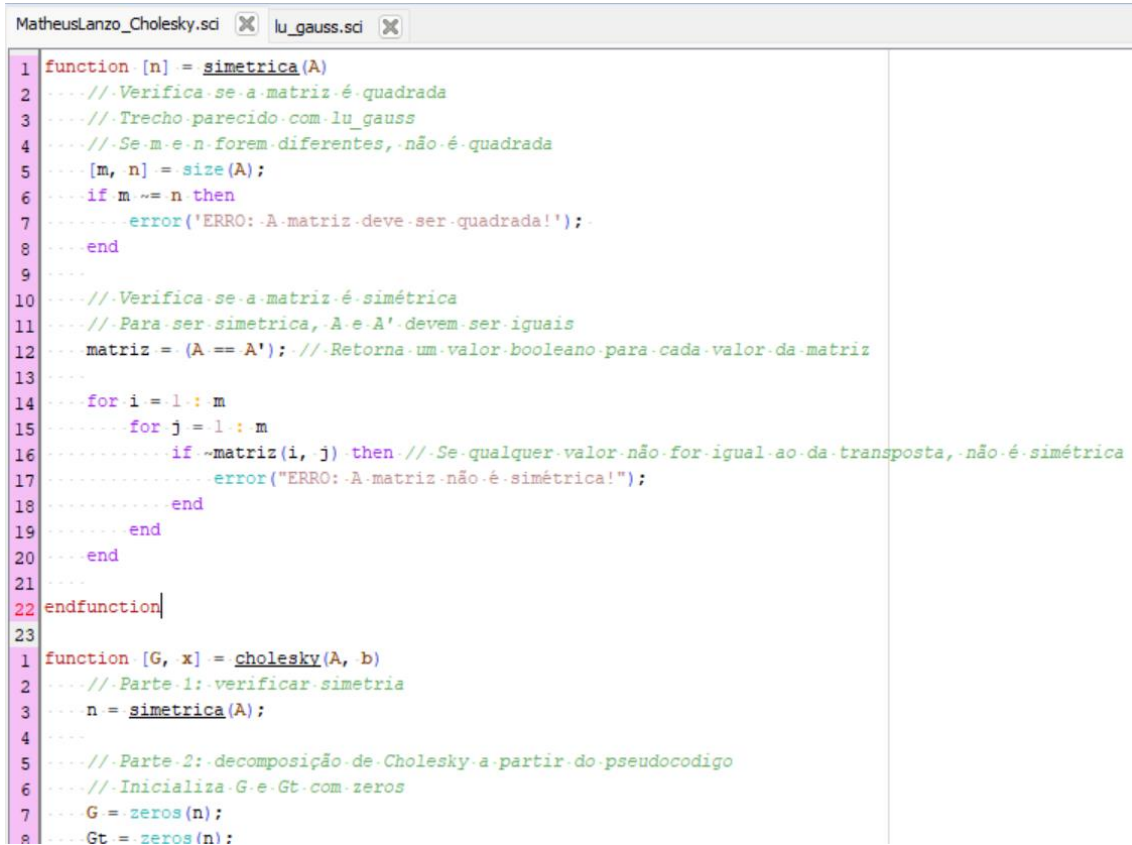


Função que verifica a simetria da matriz.

Caso não seja quadrada ou simétrica, interrompe a execução e acusa o erro.

Caso seja simétrica, não imprime nada, segue a execução normalmente e retorna o número de colunas da matriz. Por se tratar de uma matriz quadrada, o número de linhas e colunas é o mesmo.



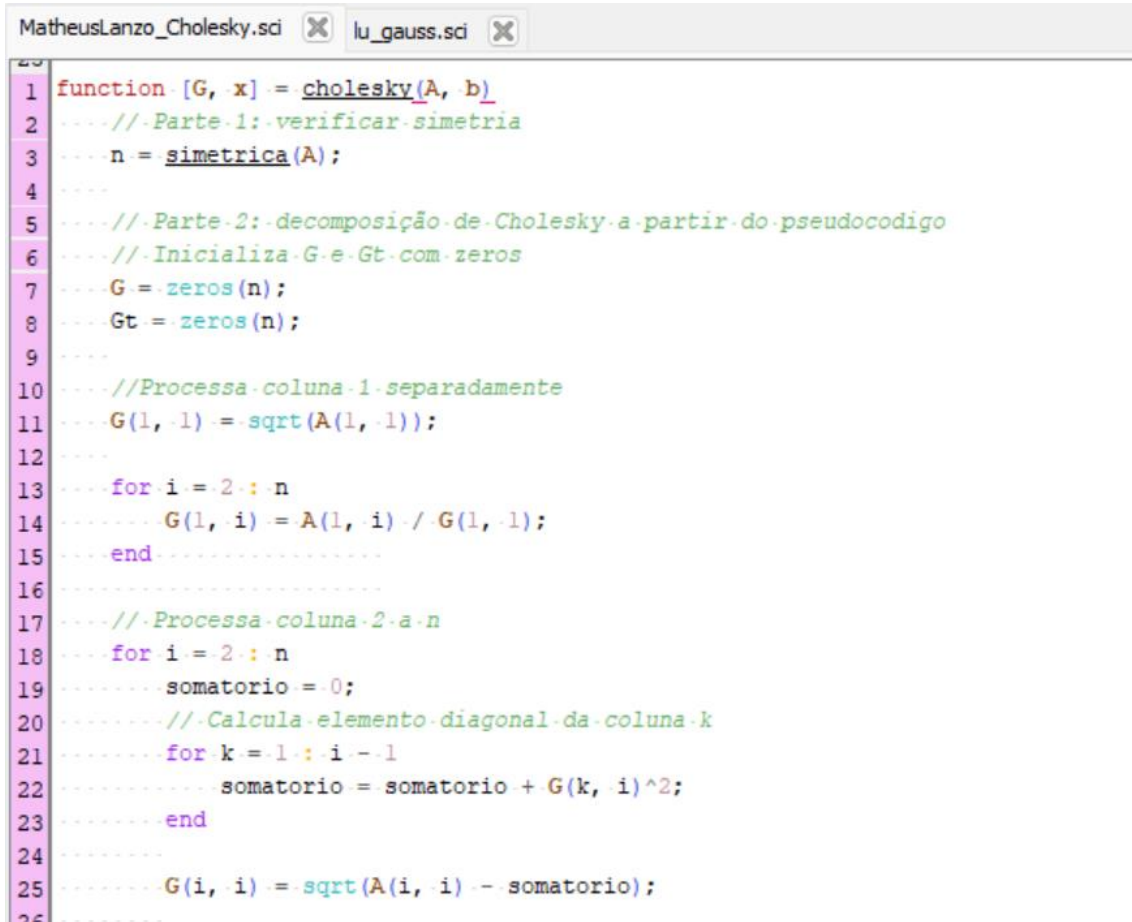
```

MatheusLanzo_Cholesky.sci | lu_gauss.sci
1 function [n] = simetrica(A)
2     %//Verifica se a matriz é quadrada
3     %//Trecho parecido com lu_gauss
4     %//Se m e n forem diferentes, não é quadrada
5     [m, n] = size(A);
6     if m ~= n then
7         error('ERRO: A matriz deve ser quadrada!');
8     end
9
10    %//Verifica se a matriz é simétrica
11    %//Para ser simétrica, A e A' devem ser iguais
12    matriz = (A == A'); %//Retorna um valor booleano para cada valor da matriz
13
14    for i = 1:m
15        for j = 1:m
16            if ~matriz(i, j) then %//Se qualquer valor não for igual ao da transposta, não é simétrica
17                error("ERRO: A matriz não é simétrica!");
18            end
19        end
20    end
21
22 endfunction
23
24 function [G, x] = cholesky(A, b)
25     %//Parte 1: verificar simetria
26     n = simetrica(A);
27
28     %//Parte 2: decomposição de Cholesky a partir do pseudocódigo
29     %//Inicializa G e Gt com zeros
30     G = zeros(n);
31     Gt = zeros(n);
  
```

Função que calcula segundo a decomposição de Cholesky.

Recebe o sistema linear simétrico em forma de matriz A, e o resultado em vetor B.

Retorna a matriz triangular superior de A e o vetor solução encontrado.



```

1 function [G, x] = cholesky(A, b)
2 ....//Parte 1: verificar simetria
3 ....n = simetrica(A);
4 ....
5 ....//Parte 2: decomposição de Cholesky a partir do pseudocódigo
6 ....//Inicializa G e Gt com zeros
7 ....G = zeros(n);
8 ....Gt = zeros(n);
9 ....
10 ....//Processa coluna 1 separadamente
11 ....G(1, 1) = sqrt(A(1, 1));
12 ....
13 ....for i = 2:n
14 ....    G(1, i) = A(1, i) / G(1, 1);
15 ....end
16 ....
17 ....//Processa coluna 2 a n
18 ....for i = 2:n
19 ....    somatorio = 0;
20 ....    ....//Calcula elemento diagonal da coluna k
21 ....    ....for k = 1:i-1
22 ....    ....    somatorio = somatorio + G(k, i)^2;
23 ....    ....end
24 ....    ....
25 ....    G(i, i) = sqrt(A(i, i) - somatorio);
26 ....

```

```

MatheusLanzo_Cholesky.sci  lu_gauss.sci
24 .....
25 .....G(i, i) = sqrt(A(i, i) - somatorio);
26 .....
27 .....//Calcula elementos não diagonais da coluna k
28 .....for j = i+1:n
29 .....    somatorio = 0;
30 .....    for k = 1:i-1
31 .....        somatorio = somatorio + G(k, i) * G(k, j);
32 .....    end
33 .....    G(i, j) = (A(i, j) - somatorio) / G(i, i);
34 .....end
35 ---end
36 ---
37 ---Gt = G';
38 ---
39 ---//Resolve sistema (trecho parecido com lu_gauss)
40 ---//Solução do Sistema Triangular Inferior
41 ---d = zeros(n, 1);
42 ---d(1) = b(1) / Gt(1, 1);
43 ---for i = 2:n
44 .....d(i) = (b(i) - Gt(i, 1:i-1) * d(1:i-1)) / Gt(i, i);
45 ---end
46 ---
47 ---//Solução do Sistema Triangular Superior
48 ---x = zeros(n, 1);
49 ---x(n) = d(n) / G(n, n);
50 ---for (i = n-1:-1:1)
51 .....x(i) = (d(i) - G(i, i+1:n) * x(i+1:n)) / G(i, i);
52 ---end
53 endfunction
77

```

Agora, a verificação das funções no console:

Primeiro inserimos a matriz requisitada pelo trabalho e seu vetor. A função cholesky feita, retorna a matriz triangular superior e o vetor solução encontrado.

```
--> A = [1, 2, 3; 2, 8, 22; 3, 22, 82]
A =

    1.    2.    3.
    2.    8.   22.
    3.   22.   82.

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

    1.
    1.
    1.

--> [G, x] = cholesky(A, b)
G =

    1.    2.    3.
    0.    2.    8.
    0.    0.    3.
x =

    2.6111111
   -1.1388889
    0.2222222
```

Podemos então, usar as funções do próprio SciLab para verificar a validade do nossa própria decomposição de Cholesky.

```
--> x = inv(A) * b
x =

    2.6111111
   -1.1388889
    0.2222222

--> chol(A)
ans =

    1.    2.    3.
    0.    2.    8.
    0.    0.    3.

-->
```

Agora vamos fazer o teste da nossa função para uma matriz que nós mesmos criamos.

Temos novamente a matriz triangular superior e o vetor solução encontrados.

```
--> A = [4, 2, -4; 2, 10, 4; -4, 4, 9]
A =

    4.    2.   -4.
    2.   10.    4.
   -4.    4.    9.

--> b = [2; 2; 3]
b =

    2.
    2.
    3.

--> exec('C:\Users\Matheus\Desktop\Semestre-IV\Calculo-Numerico\MatheusYo:

--> [G, x] = cholesky(A, b)
G =

    2.    1.   -2.
    0.    3.    2.
    0.    0.    1.
x =

    6.2222222
   -2.7777778
    4.3333333

-->
```

Por fim, o teste com as funções próprias do SciLab.

```
--> x = inv(A) * b
x =

    6.2222222
   -2.7777778
    4.3333333

--> chol(A)
ans =

    2.    1.   -2.
    0.    3.    2.
    0.    0.    1.

--> |
```