Nota 0,1

Introdução ao MatLab

O Live Editor é uma ferramenta do MatLab que permite a criação e execução de live scripts, ou seja, scripts que combinam códigos, output e texto em um bloco de notas executável. Os live scripts são armazenados em arquivos .mlx.

Podemos observar que existe um menu de abas na parte superior da tela quando estamos trabalhando com o Live Editor. Da esquerda para a direita temos as barras:

- Home - administração do ambiente
- Plots - Criação de gráficos
- Apps - aplicações especiais
- Live Editor - edição e execução de live scripts
- Insert - inserção de objetos especiais
- View - controle de visualização

Vamos utilizar principalmente as abas HOME e LIVE EDITOR.

Um live script é organizado em seções que podem ser editadas e executadas separadamente. As seções podem misturar texto e códigos.

Essa é uma seção de texto e termina aqui.

Operador dois pontos

O operador dois pontos inicio:passo:fim cria um vetor onde o primero elemento tem o valor inicio, o últoo tem o valor fim, e os valores são alterados elemnto a elemento pelo valor em passo. O valor dos passos é opcional, e se não for informado assume-se o valor 1 para os passos.

```
% Cria vetor V1 contendo os números de 1 a 10
V1 = 1:10
```

```
V1 = 1×10
     1     2     3     4     5     6     7     8     9    10
```

```
% Cria o vetor V2 contendo os números de 20 a 15
V2 = 20:-1:15
```

```
V2 = 1×6
    20    19    18    17    16    15
```

```
% Cria o vetor V3 contendo os valores [0, pi/2, pi, 3*pi/2]
V3 = 0:pi/2:3*pi/2
```

```
V3 = 1×4
         0    1.5708    3.1416    4.7124
```

As funções podem ser aplicadas diretamente em vetores

```
sin(V3)
```

```
ans = 1×4
         0    1.0000    0.0000   -1.0000
```

Essa seção termina aqui.

Matrizes

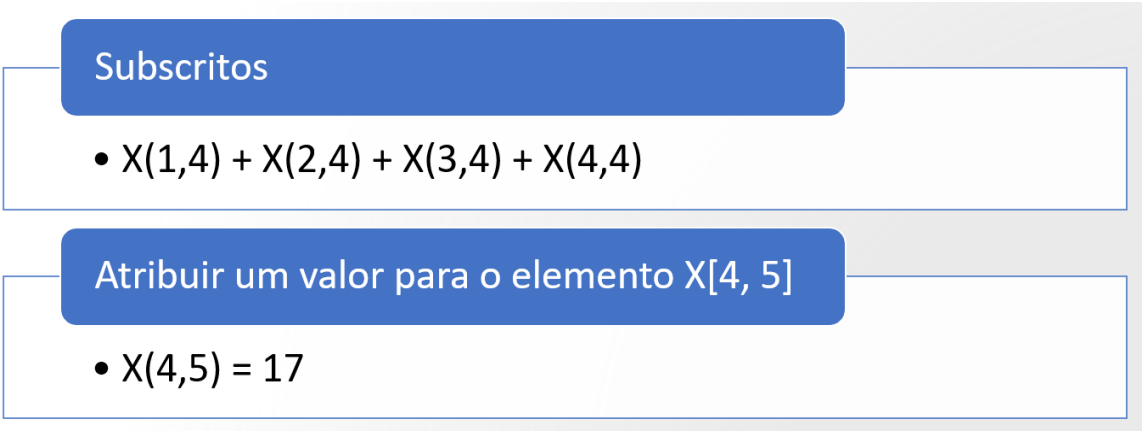
As matrizes são variáveis naturais do MatLab.

Podem ser definidas informando os valores entre colchetes. Os valores das dos elementos de cada linha são separados por espaço em branco ou vírgula. As linhas são separadas entre si por ponto e vírgula.

```
X = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12; 13, 14, 15, 16];
```

Os elementos da matriz podem ser acessados indicando-se entre parenteses o número da linha e da coluna do elemento que se quer acessar.

Se incluimos um elemento não existente, o MatLab completa matriz com zeros.




```
% Acessa os elementos da coluna 4 e soma
X(1,4) + X(2,4) + X(3,4) + X(4,4)
```

```
ans = 40
```

```
% Atribui um valor para o elemento da linha 4, coluna 5
% Esse elemento não existe
% O Matlab atribui o valor e completa os outros elementos da coluna 5
X(4,5)=17
```

```
X = 4×5
     1     2     3     4     0
     5     6     7     8     0
     9    10    11    12     0
    13    14    15    16    17
```

As funções podem ser aplicadas diretamente às matrizes

```
sqrt(X)
```

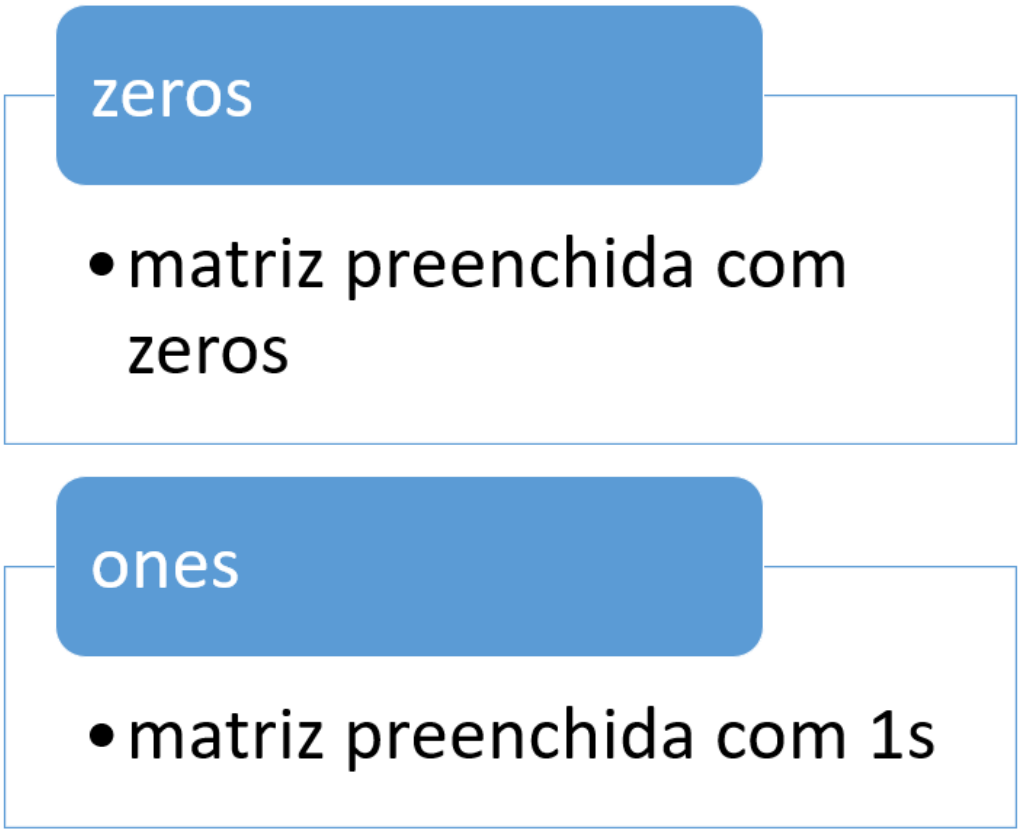
```
ans = 4×5
```

```
    1.0000    1.4142    1.7321    2.0000         0
    2.2361    2.4495    2.6458    2.8284         0
    3.0000    3.1623    3.3166    3.4641         0
    3.6056    3.7417    3.8730    4.0000    4.1231
```

Essa seção termina aqui.

Funções para geração de matrizes

O Matlab dispõe de diversas funções para geração de matrizes.




```
% Cria a matriz Z de diemnsão 3x5 preenchida com zeros
Z = zeros(3,5)
```

```
Z = 3×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
% Cria a matriz O de dimensão 3x2 preenchida com 1s
O = ones(3,2)
```

```
O = 3×2
     1     1
     1     1
     1     1
```

Essa seção termina aqui.

Funções matriciais

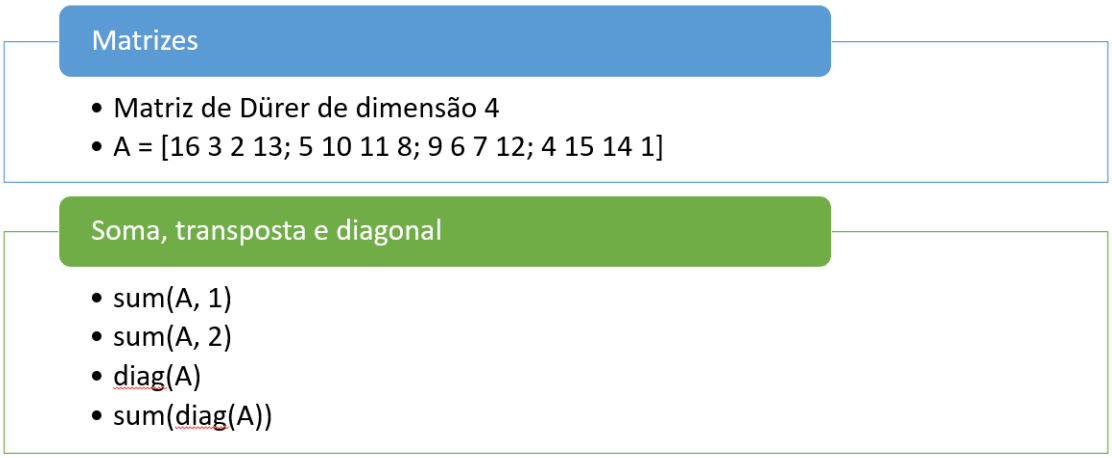
A maior parte das funções do MatLab opera com matrizes, ou seja, receb como argumento uma matriz, retornando outra matriz.

Vamos experimentar algumas funções sobre matrizes utilizando a matriz de Dürer, que é uma matriz quadrada (mesmo número de linhas e de colunas) que tem algumas propriedades:

- A soma dos elementos de cada coluna é o mesmo valor
- A soma dos elementos de cada linha é o mesmo valor
- A soma dos elementos da diagonal é o mesmo valor

Para somar as colunas ou as linhas de uma matriz usamos a função sum, que recebe como primeiro argumento a matriz, e como segundo argumento se devem ser somadas as colunas (valor do argumento igual a 1), ou se devem ser somadas as colunas (valor dos argumentos igual a 2). O resultado é um vetor com as somas de cala coluna ou de cada linha.

Para extrair a diagonal principal de uma matriz usamos a função diag, que recebe como argumento a matriz. O resultado é um vetor contendo os elementos da diagonal principal.



Coloque abaixo o código para salvar a matriz de Dürer na matriz A, e as funções para somar as colunas, as linhas, para extrair a diagonal principal, e para somar os elementos da diagonal principal.

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1];
sum(A,1)
```

```
ans = 1×4
    34    34    34    34
```

```
sum(A,2)
```

```
ans = 4×1
    34
    34
    34
    34
```

```
diag(A)
```

```
ans = 4×1
    16
    10
     7
     1
```

```
sum(diag(A))
```

```
ans = 34
```

Manipulação de matrizes

Função magic

A matriz de Dürer pode ser criada pela função magic do Matlab usado como argumento a dimensão da matriz quadrada.

```
% Esse comando cria a matriz de Durer de dimensão 5 e salva na variável B
B = magic(5);

% Coloque comandos que verificam as propriedades da matriz B
sum(B,1)
```

```
ans = 1×5
    65    65    65    65    65
```

```
sum(B,2)
```

```
ans = 5×1
    65
    65
    65
    65
    65
```

```
diag(B)
```

```
ans = 5×1
    17
     5
    13
    21
     9
```

```
sum(diag(B))
```

```
ans = 65
```

Trocar as posições das colunas de uma matriz

Inverter a ordem das linhas ou das colunas da matriz B

```
% Ver a matriz B
B
```

```
B = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
% Trocar de posição as linhas 3 e 4
C = B([1 2 4 3 5],:)
```

```
C = 5×5
    17    24     1     8    15
    23     5     7    14    16
    10    12    19    21     3
     4     6    13    20    22
    11    18    25     2     9
```

```
% Inverter a ordem das colunas
D = B(:,[5 4 3 2 1])
```

```
D = 5×5
    15     8     1    24    17
    16    14     7     5    23
    22    20    13     6     4
     3    21    19    12    10
     9     2    25    18    11
```

Calcular a transposta de uma matriz

Na matriz trnsposta, as linhas se transformam em colunas e vice-versa.

```
E = transpose(B)
```

```
E = 5×5
    17    23     4    10    11
    24     5     6    12    18
     1     7    13    19    25
     8    14    20    21     2
    15    16    22     3     9
```

```
F = B'
```

```
F = 5×5
    17    23     4    10    11
    24     5     6    12    18
     1     7    13    19    25
     8    14    20    21     2
    15    16    22     3     9
```

Concatenar horizontalmente duas matrizes

As matrizes precisam ter o mesmo número de linhas

```
% Mostrar a matriz Z
Z
```

```
Z = 3×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
% Mostrar a matriz O
O
```

```
O = 3×2
     1     1
     1     1
     1     1
```

```
% Concatenar horizontalmente as matrizes
ZO = [Z O]
```

```
ZO = 3×7
     0     0     0     0     0     1     1
     0     0     0     0     0     1     1
     0     0     0     0     0     1     1
```

Concatenar verticalmente duas matrizes

As matrizes precisam ter o memo número de colunas

```
% Concatenar verticalmente as matrizes
G = [Z; B]
```

```
G = 8×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

Essa seção termina aqui.

Operações com matrizes

Multiplicação por escalar

Uma matriz pode ser multiplicada por um valor não matricial. O resultado é que cada elemento da matriz será multiplicao pelo valor.

```
A = [5 6; 4 3]
```

```
A = 2×2
     5     6
     4     3
```

```
C = [4 -3; 7 8]
```

```
C = 2×2
     4    -3
     7     8
```

```
valor = 4
```

```
valor = 4
```

```
A+valor
```

```
ans = 2×2
     9    10
     8     7
```

```
valor*A
```

```
ans = 2×2
    20    24
    16    12
```

```
C-valor
```

```
ans = 2×2
     0    -7
     3     4
```

```
C/sqrt(valor)
```

```
ans = 2×2
    2.0000   -1.5000
    3.5000    4.0000
```

Multiplicação e exponenciação

Nos casos de compatibilidade das dimensões, as operações de multiplicação (*) e exponenciação (^) de matrizes pode ser realizada elemento a elemento, ou com as regras da operação matricial. Para operações elemento a elemento colocamos um ponto antes do operador.

Para multiplicação elemento a elemento, as duas matrizes devem ter as mesmas dimensões.

Para a operação matricial, o número de colunas da primeira matriz de ser igual ao número de linhas da segunda matriz. Segue o procedimento visto em algebra linear.

A

$$\begin{vmatrix} 5 & 6 \\ 4 & 3 \end{vmatrix}$$

C

$$\begin{vmatrix} 4 & -3 \\ 7 & 8 \end{vmatrix}$$

A.*C

$$\begin{vmatrix} 5*4 & 6*-3 \\ 4*7 & 3*8 \end{vmatrix} \qquad \begin{vmatrix} 20 & -18 \\ 28 & 24 \end{vmatrix}$$

A*C

$$\begin{vmatrix} 5*4+6*7 & 5*-3+6*8 \\ 4*4+3*7 & 4*-3+3*8 \end{vmatrix} \qquad \begin{vmatrix} 62 & 33 \\ 37 & 12 \end{vmatrix}$$

C.^2

$$\begin{vmatrix} 4*4 & -3*-3 \\ 7*7 & 8*8 \end{vmatrix} \qquad \begin{vmatrix} 16 & 9 \\ 49 & 64 \end{vmatrix}$$

C^2 C*C

$$\begin{vmatrix} (4*4)+(-3*7) & (4*-3)+(-3*8) \\ 7*4+8*7 & 7*-3+8*8 \end{vmatrix} \qquad \begin{vmatrix} -5 & -36 \\ 84 & 43 \end{vmatrix}$$

```
% Multiplicação elemento a elemento
A.*C
```

```
ans = 2×2
    20   -18
    28    24
```

```
% Multiplicação matricial
A*C
```

```
ans = 2×2
    62    33
    37    12
```

```
% Exponenciação elemento a elemento
C.^2
```

```
ans = 2×2
    16     9
    49    64
```

```
% Exponenciação matricial
C^2
```

```
ans = 2×2
    -5   -36
    84    43
```

Exemplo de multiplicação matricial com matrizes não quadradas

```
% Matrix 3x2
D = [1 2; 3 4; 5 6]
```

```
D = 3×2
     1     2
     3     4
     5     6
```

```
% Matriz 2x4
E = [1 2 3 4; 5 6 7 8]
```

```
E = 2×4
     1     2     3     4
     5     6     7     8
```

```
% Matrix 3x4
D*E
```

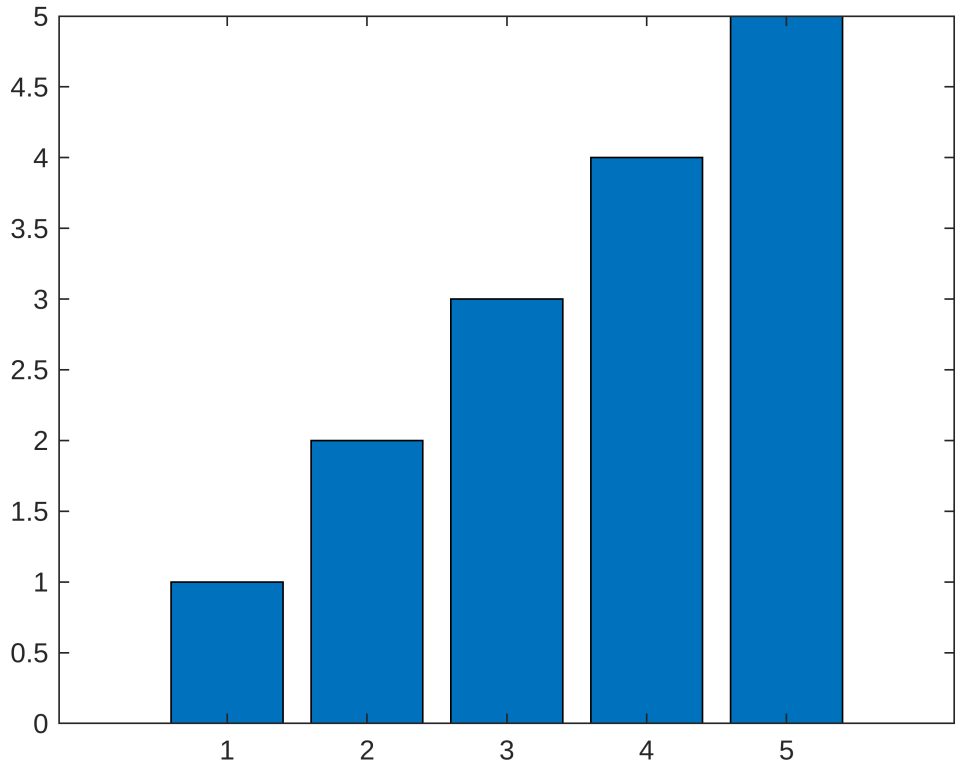
```
ans = 3×4
    11    14    17    20
    23    30    37    44
    35    46    57    68
```

Funções gráficas

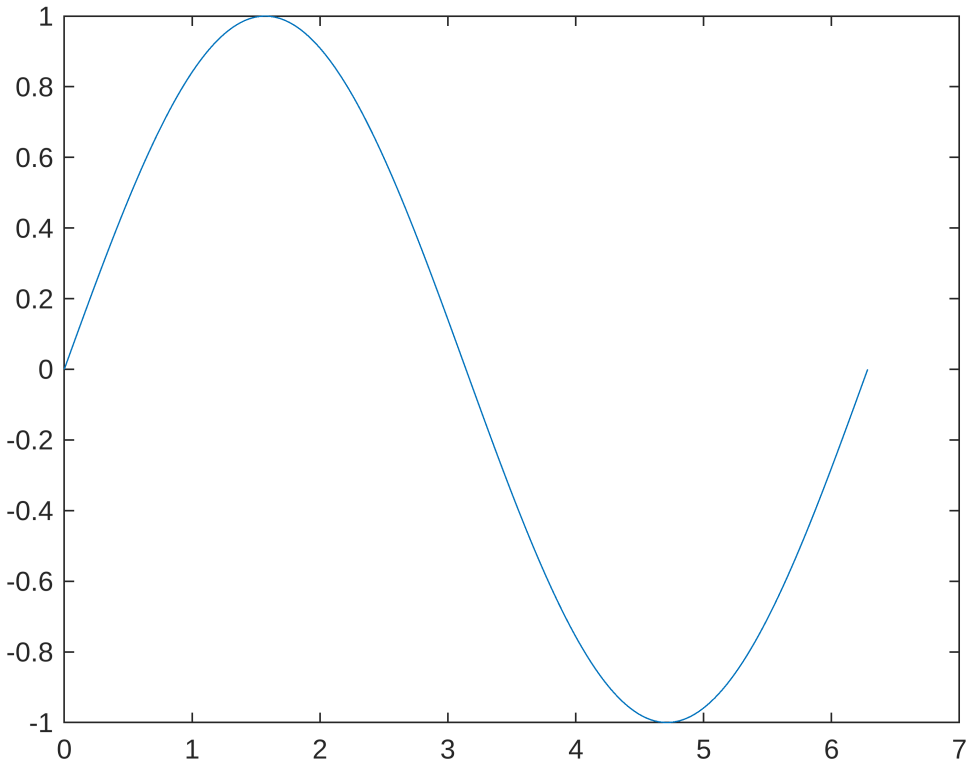
```
X = [1 2 3 4 5]
```

```
X = 1×5
     1     2     3     4     5
```

```
bar(X)
```



```
% Observe que tem muito valores no vetor x
% Isso é para a curva ter uma aparência contínua
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```



Principais comandos de programação

IF

if<condição>

<comandos>

end

- Se o resultado da expressão lógica for verdadeiro então os comandos serão executados
- Se o resultado for falso, os não serão executados

if<condição>

<comandos1>

else<comandos2>

end

- Se a expressão lógica for verdadeira então os comandos1 serão executados
- Se for falsa então os comandos2 serão executados

FOR

for <variavel>=<passos>

<comandos>

end

- variavel é a variável-de-controle que recebe o valor de cada elemento do vetor <passos>
- para cada conteúdo que receba, executa o corpo do for
- o número de repetições dos <comandos > é igual ao número de elementos no vetor <passos>
- a variável-de-controle não pode ser redefinida dentro da estrutura for

FUNCTION

function retorno = nome_da_função(argumentos)

- function: Palavra chave.
- nome_da_função: Deve ser o mesmo nome que o arquivo .m
- retorno: Variável de retorno. Pode ser uma matriz.
- argumentos: Lista de variáveis passadas como argumento.

O código da função pode estar no LiveScript ou em um arquivo externo .m.

No exemplo a seguir, o código da função FatorialI está no arquivo FatorialI.m e o código da função FatorialR está no livescript.

Quando a função está no livescript o código da função deve vir DEPOIS da chamada da função, no final do LivScript

FORMATIVA

Alterar o código da função FatorialV na próxima célula e executar o código a seguir.

ENTREGA

Colocar a opção Output Inline na aba VIEW.

Selecionar a opção Save Export to PDF na ava LIVE EDITOR.

Fazer o upload no AVA.

```
% Função do matlab
factorial(5)
```

```
ans = 120
```

```
% Fatorial interativo
% Ver o código no arquivo FatorialI.m
FatorialI(5)
```

```
ans = 120
```

```
% Fatorial recursivo
% Ver o código no final do LiveScript
FatorialR(5)
```

```
ans = 120
```

```
% Fatorial vetorial
% Modificar o código no final do LiveScript
FatorialV(5)
```

```
ans = 120
```

Código das funções

```
function fat = FatorialI(n)
  fat = 1;
  if (n>0)
    for k = n:-1:1
      fat = fat*k;
    end
  end
end

function fat = FatorialR(n)
  if (n==0)
      fat = 1;
  else
      fat = n*FatorialR(n-1);
  end
end

function fat = FatorialV(n)
  fat = 1;
  if (n>1)
      fat = n*FatorialV(n-1);
```

Esse é um código recursivo

fat = prod(1:n)

```
      % substituir o valor zero no comando acima
      % pelo produto dos termos de um vetor com valores de 1 até n
  end
end
```