MATLAB

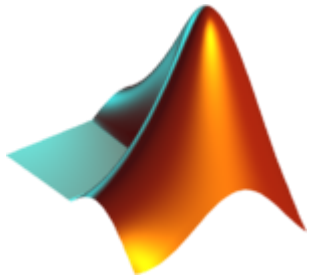
Básico

INTRODUÇÃO – Mathworks e produtos

MathWorks®

Fundada em **1984**
CEO and President: Jack Little
Chief Mathematician: Cleve Moler

Produtos principais:

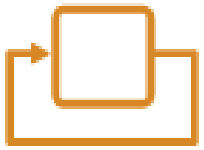


INTRODUÇÃO – Aplicações

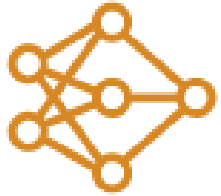
Robótica

Processamento de sinais

Gestão de riscos



Sistemas de controle



Redes neurais

INTRODUÇÃO – Linguagem M

→ Linguagem própria

→ Trabalha com matrizes

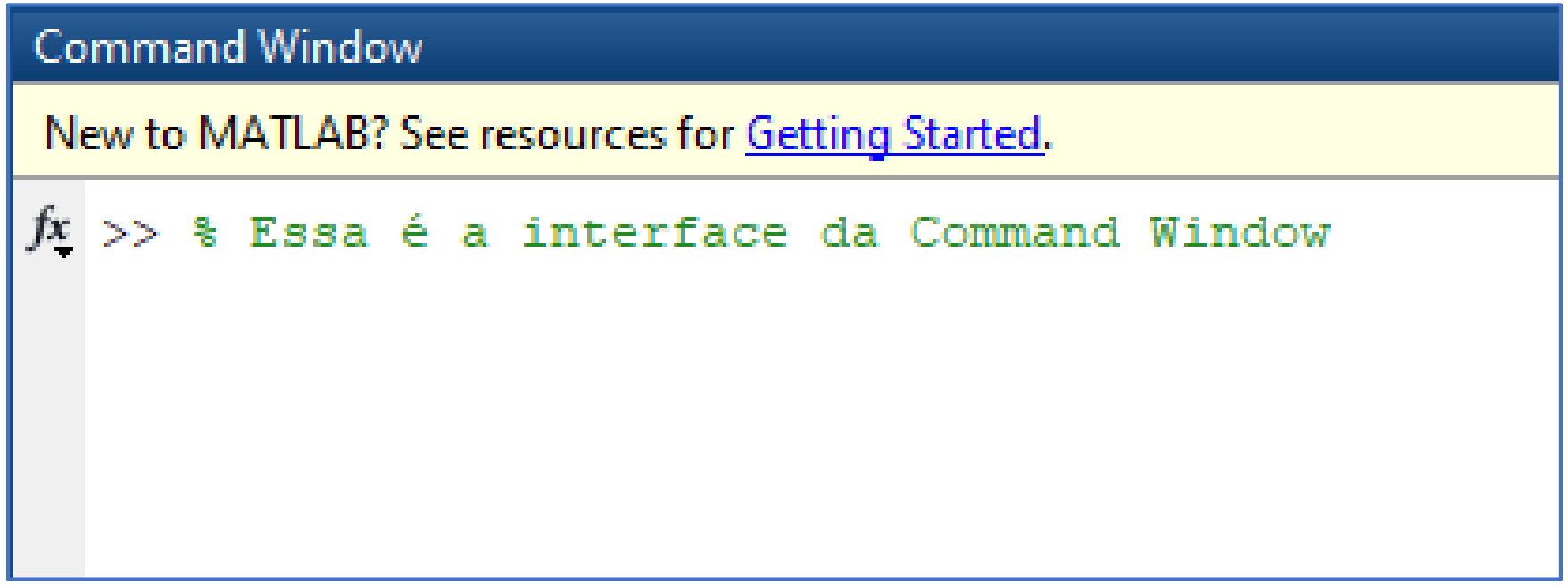
→ A sintaxe dos comandos é simplificada

→ Concentra uma grande variedade de funções matemáticas

→ Permite a inclusão de bibliotecas específicas para trabalho em diversas áreas

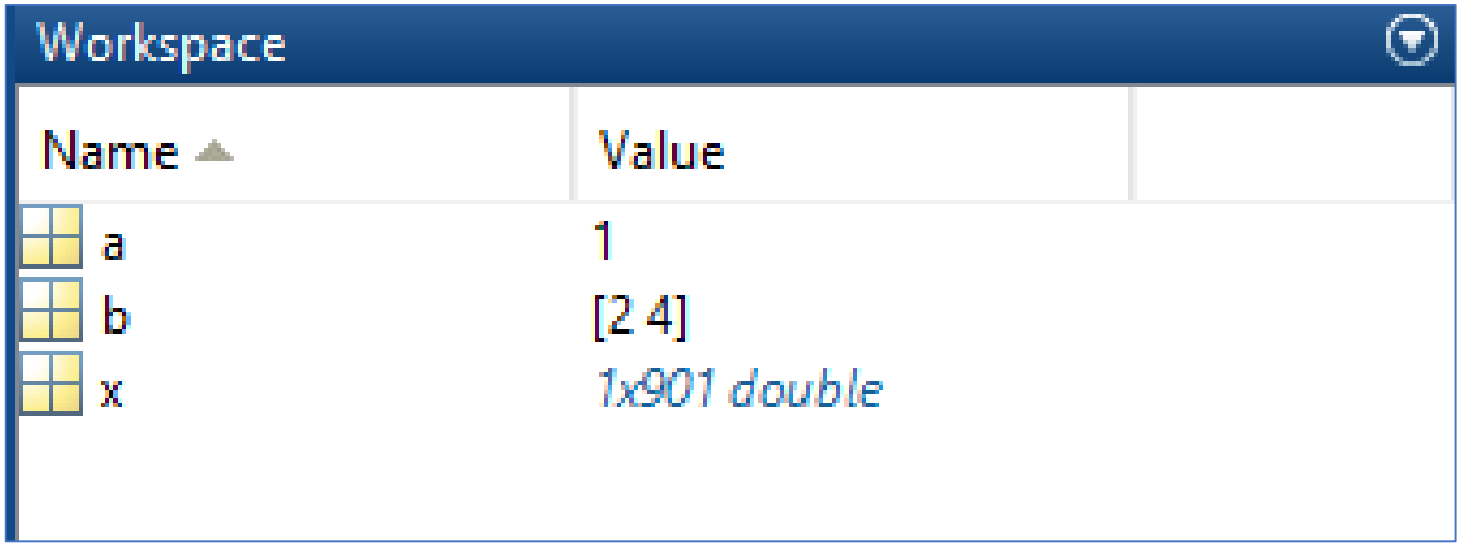
Área de trabalho– Command Window

→ Inserir linhas de comando



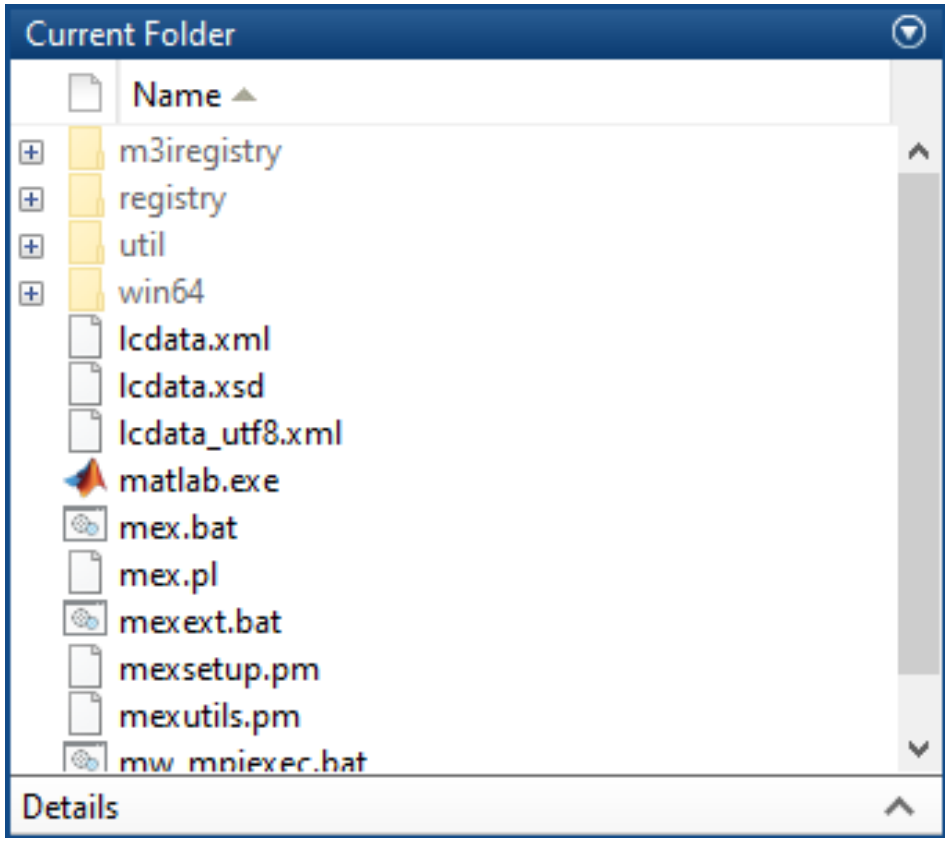
Área de trabalho – Workspace

→ Exibe variáveis e respectivos valores/dimensões



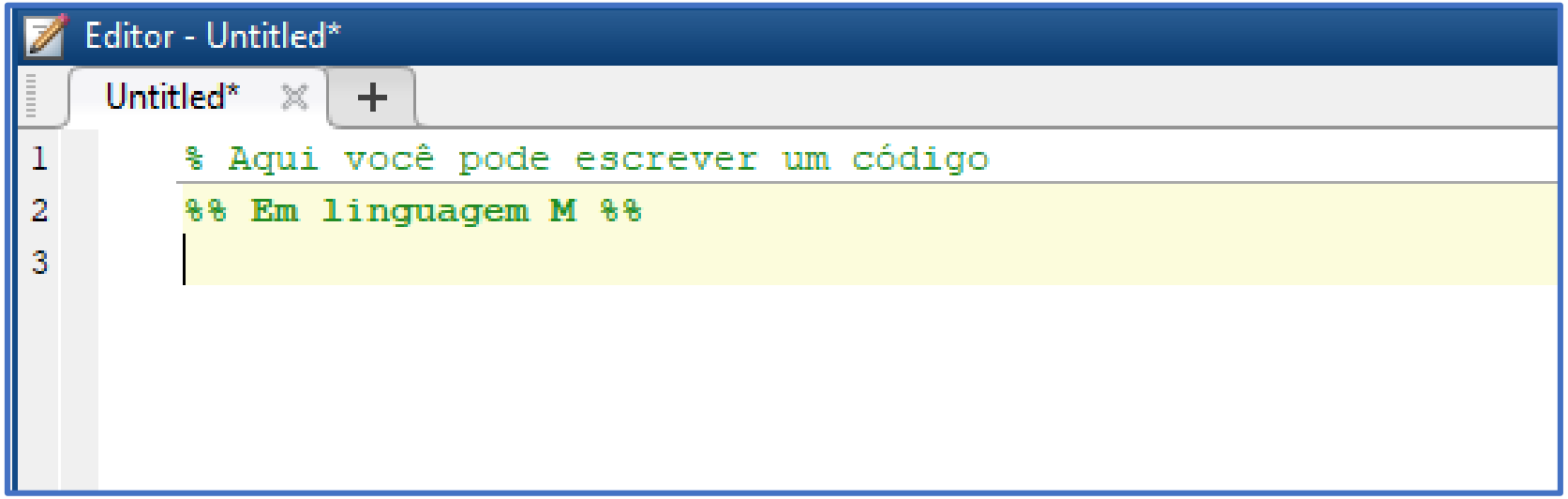
Área de trabalho – Current Folder

→ Exibe pasta predefinida como pasta de trabalho e e seus diretórios

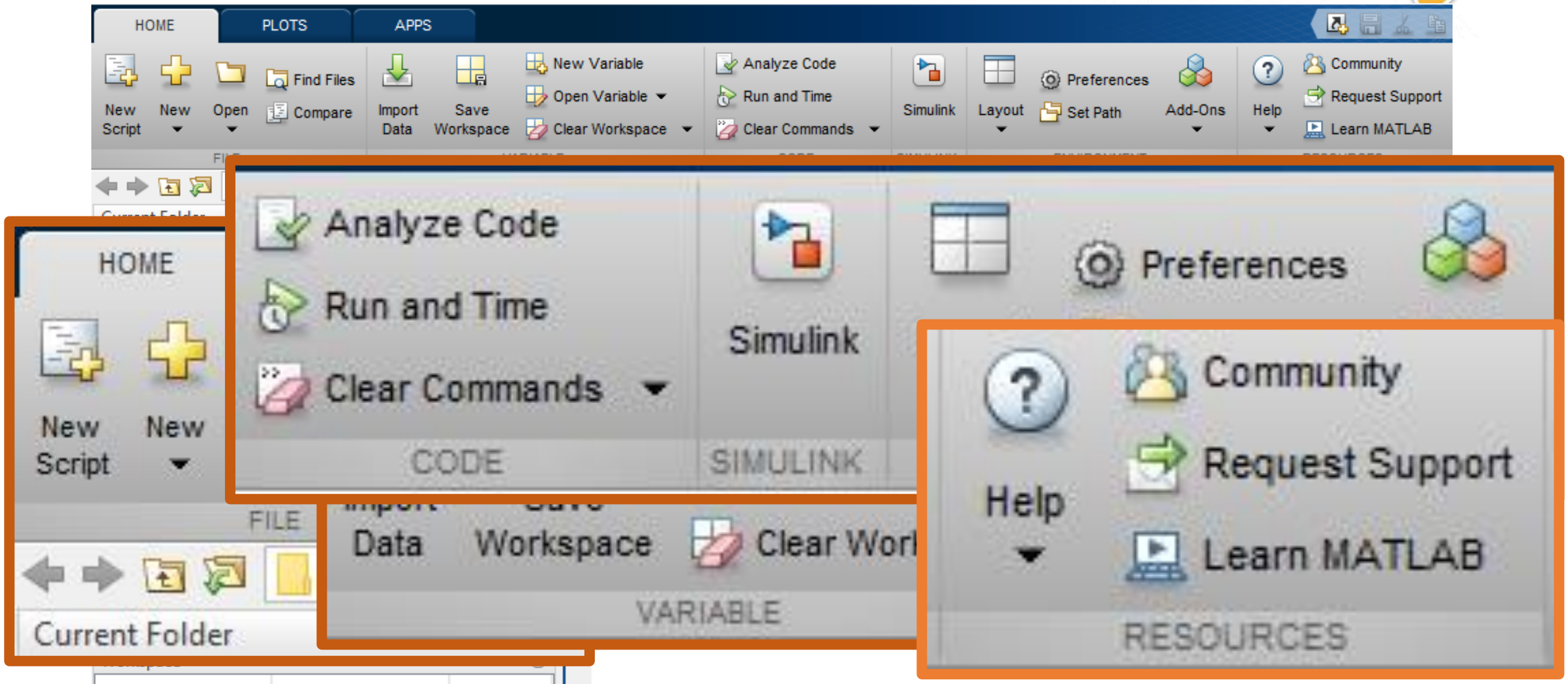


Área de trabalho – Script Editor

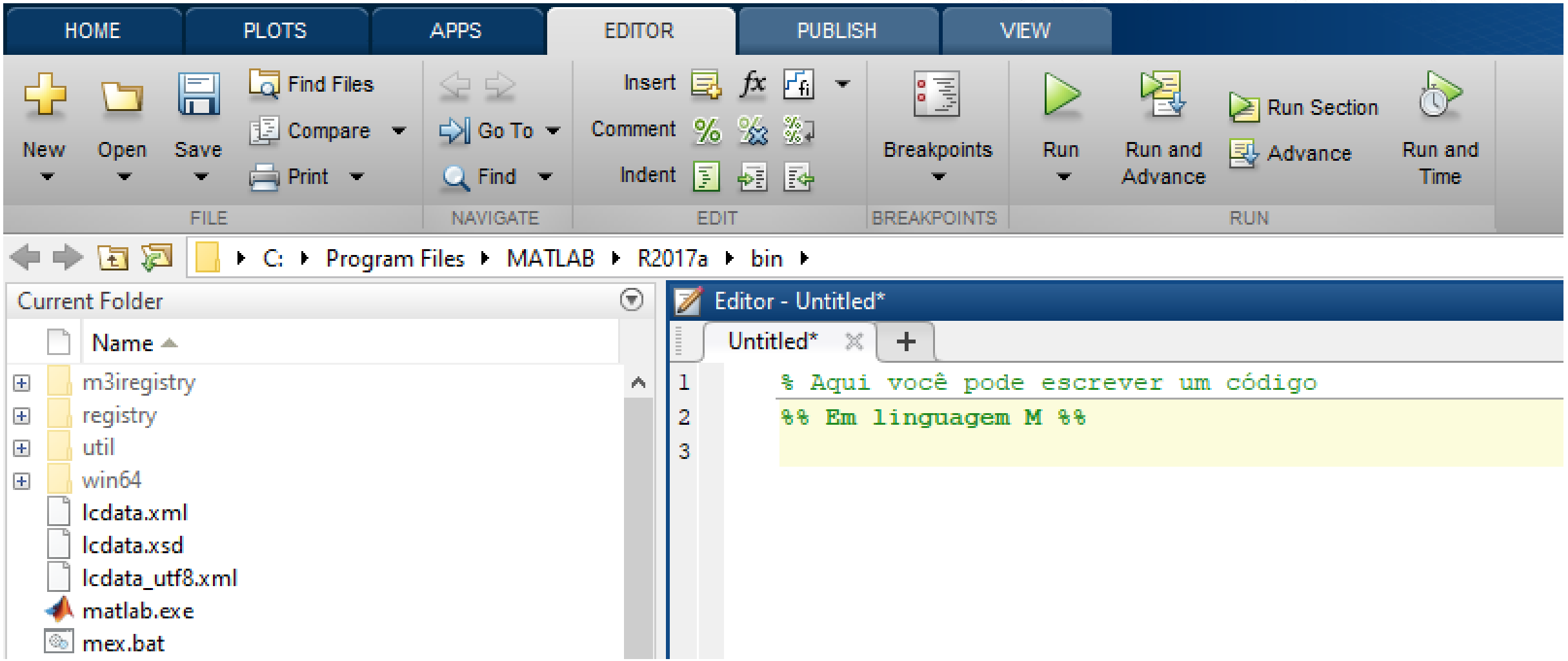
→ Permite edição e criação de Scripts/Algoritmos



Área de trabalho – Barra de Tarefas



Área de trabalho – Barra de Tarefas



Operações Aritméticas

Tabela 1.1 - Operações aritméticas entre dois escalares		
Operação	Forma Algébrica	MATLAB
Adição	a + b	a + b
Subtração	a - b	a - b
Multiplicação	a x b	a*b
Divisão pela Direta	a ÷ b	a/b
Divisão pela Esquerda	b ÷ a	a\b
Exponenciação	a^b	a^b
Radiciação	$\sqrt{a}$	sqrt(a)
Fatorial	a!	factorial(a)

Operações Aritméticas

```
>> x = 5

x =

     5
```

```
>> 2 + 1

ans =

     3
```

```
>> 7 - 9

ans =

    -2
```

```
>> 7*2

ans =

    14
```

```
>> 11/2

ans =

    5.500
```

```
>> 3^3

ans =

    27
```

Hierarquia de Operações

→ Assim como na matemática, no MATLAB existe uma ordem de prioridade para execução das operações aritméticas. Abaixo segue uma tabela com essa ordem.

Hierarquia das Operações Aritméticas	
Prioridade	Operação
1ª	Parênteses
2ª	Exponenciação, esquerda à direita
3ª	Multiplicação e Divisão, esquerda à direita
4ª	Adição e Subtração, esquerda à direita

Hierarquia de Operações

→ Um exemplo é a equação de Bháskara

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
>> a = 5;
>> b = 5;
>> c = 5;
>> x = -b+sqrt(b^2-4*4*a*c)/2*a;
ans =
      25.8388
>> x = (-b+sqrt(b^2-4*4*a*c))/(2*a);
ans =
      0.9136
```

Formatos numéricos

→ format short (padrão)

```
0.3333
```

→ format long

```
0.33333333333333333
```

→ format shorte%

```
3.3333e-01
```

→ format longe%

```
3.33333333333333333e-01
```

→ format hex

```
3fd5555555555555
```

→ format bank

```
0.33
```

Comandos básicos – Limpar e ajuda!

→ clc

Limpa os comandos do Command Window.

→ clear nome_variável

Apaga a variável “nome_variável”.

→ clear all

Apaga todas as variáveis do Workspace.

Comandos básicos – Limpar e ajuda!

→ help nome_comando

Exibe uma explicação do funcionamento do comando

```
>> help sin
sin - Sine of argument in radians
This MATLAB function returns the sine of the
elements of X.
Y=sin(X)
Reference page for sin
See also asin, asind, sind, sinh.
Other uses of sin
fixedpoint/sin, symbolic/sin
```

Comandos básicos – Exibir, salvar e carregar

→ who

Exibe quem são as variáveis atualmente no Workspace.

```
>> who
your variables are:
a b
```

→ whos

Além de exibir as variáveis mostra também a dimensão, número de bytes e a classe da variável.

```
>> whos
Name   Size   Bytes   Class
a      1x1        8   double
b      2x3       48   double
```

Comandos básicos – Exibir, salvar e carregar

→ save nome_ficheiro

Salva as variáveis do Workspace em formato binário ou

formato ascii.

→ load nome_ficheiro

Abri no Workspace variáveis salvas em um arquivo.

Funções f(x)

→ trigonométricas

sinh(x) cosh(x) tanh(x)

asinh(x) acosh(x) atanh(x)

csch(x) sech(x) coth(x)

acsch(x) asech(x) acoth(x)

Funções f(x)

→ Exemplos

```
>> x = pi/3;
>> sin(x)
ans =
      0.8660
```

```
>> x = pi/3;
>> sinh(x)
ans =
      1.2494
```

```
>> x = 0.8860;
>> asin(x)
ans =
      1.0471
```

Funções f(x)

→ conversão (ângulo)

```
degtorad(180)
radtodeg(pi/3)
```

Funções f(x)

→ conversão (ângulo)

```
>> degtorad(180)

ans =

       3.1416
```

```
>> radtodeg(pi/3)

ans =

      60.000
```

Funções f(x)

→ Exponencial e Logarítmica

`exp(x)`
Potenciação com o número de Euler como base

`sqrt(x)`
Raiz quadrada de x

`log(x)`
Logaritmo neperiano

`log10(x)`
Logaritmo de x na base 10

Funções f(x)

→ Exponencial e Logarítmica

```
>> exp(2)

ans =

       7.3891
```

```
>> log(exp(2))

ans =

       2
```

```
>> log10(1000)

ans =

       3
```

```
>> sqrt(64)

ans =

       8
```

Funções f(x)

→ Arredondamento

round(x)
Arredonda um número decimal para o inteiro mais próximo.

floor(x)
Arredonda um número decimal para o inteiro anterior.

ceil(x)
Arredonda um número decimal para o inteiro posterior.

rem(x,y)
Obtém-se o valor do resto da divisão de x por y.

Funções f(x)

→ Arredondamento

```
>> round(1.43)

ans =

     1
```

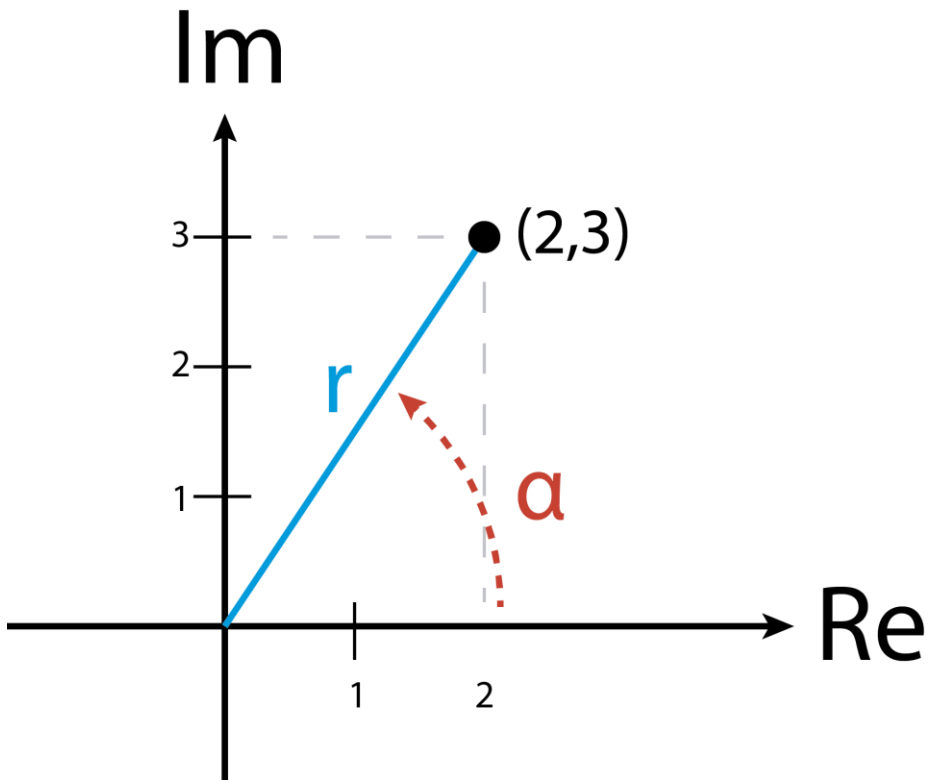
```
>> floor(1.5)

ans =

     1
```

```
>> ceil(1.5)

ans =

     2
```

```
>> rem(8,5)

ans =

     3
```

Números Complexos

→ Números Complexos




```
x=2+3i
```

→ Números Complexos

Coordenadas retangulares para polares:

$$r = \sqrt{x^2 + y^2} \qquad \alpha = tan^{-1}\left(\frac{y}{x}\right)$$

Coordenadas polares para retangulares:

$$x = r \cdot \cos(\alpha) \qquad y = r \cdot sen(\alpha)$$

Números Complexos

→ Números Complexos

```
>> x = 3+4i

X =

       3.0000 + 4.0000i

>> y = 5+3j

y =

       5.0000 + 3.0000i
```

```
>> z = 3+sqrt(-1)

z =

       3.0000 + 1.0000i
```

Números Complexos

→ Operação Aritmética

```
>> c1 = 3+2i

c1 =

        3.0000 + 2.0000i

>> c2 = 5-i

c2 =

        5.0000 - 1.0000i

>> c1+c2

ans =

        8.0000 + 1.0000i
```

```
>> c3 = 4

c3 =

        4

>> c1-c2/c3

ans =

        1.7500 + 2.2500i

>> i^2

ans =

        -1
```

Números Complexos

→ abs(x)

Calcula o valor absoluto

```
>> x = [1+2i 2+3i]

x =

      1.0000 + 2.0000i    2.0000+3.0000i

>> abs(x)

ans =

      2.2361    3.6056
```

Números Complexos

→ angle(x)

Retorna o valor do ângulo de fase (em radianos) de um número complexo.

```
>> x = [3 2i]

x =

       3.0000 + 2.0000i

>> angle(x)

ans =

       0.5880
```

Números Complexos

→ conj(x)

Retorna o valor do complexo conjugado de um dado número complexo x.

```
>> conj(1+4i)

ans =

      1.0000 - 4.0000i
```

Números Complexos

→ imag(x)

Retorna o valor da parte imaginária de um dado número complexo x.

```
>> imag(4+5i)

ans =

     5
```

Números Complexos

→ real(x)

Retorna o valor da parte real de um dado número complexo x.

```
>> real(4+5i)

ans =

     4
```

Rotinas ou Arquivos M-Files - Scripts

Conhecidos como:

- → M-Files
- → Rotinas
- → Scripts Files

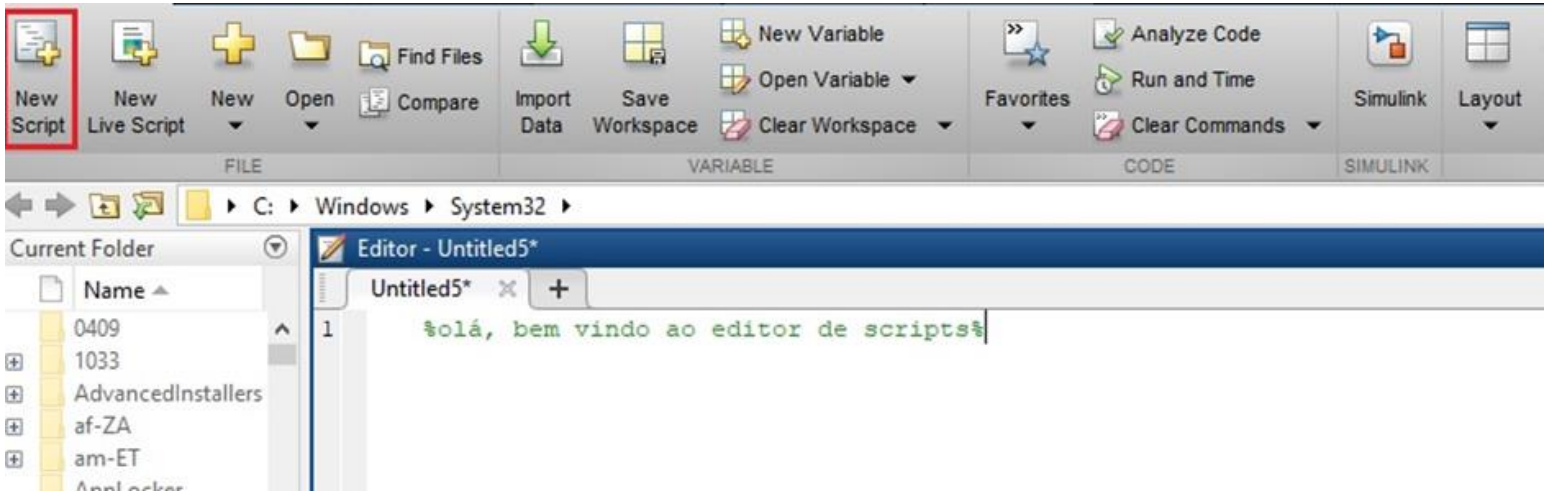
Rotinas ou Arquivos M-Files - Scripts

Características dos Scripts:

- → Os comandos são executados na ordem em que eles foram escritos.
- → Em scripts que possuem comandos de saída, essa saída é mostrada no Command Window
- → Podem ser editados e também serem executados várias vezes
- → Devem ser salvos antes de serem executados

Rotinas ou Arquivos M-Files - Scripts

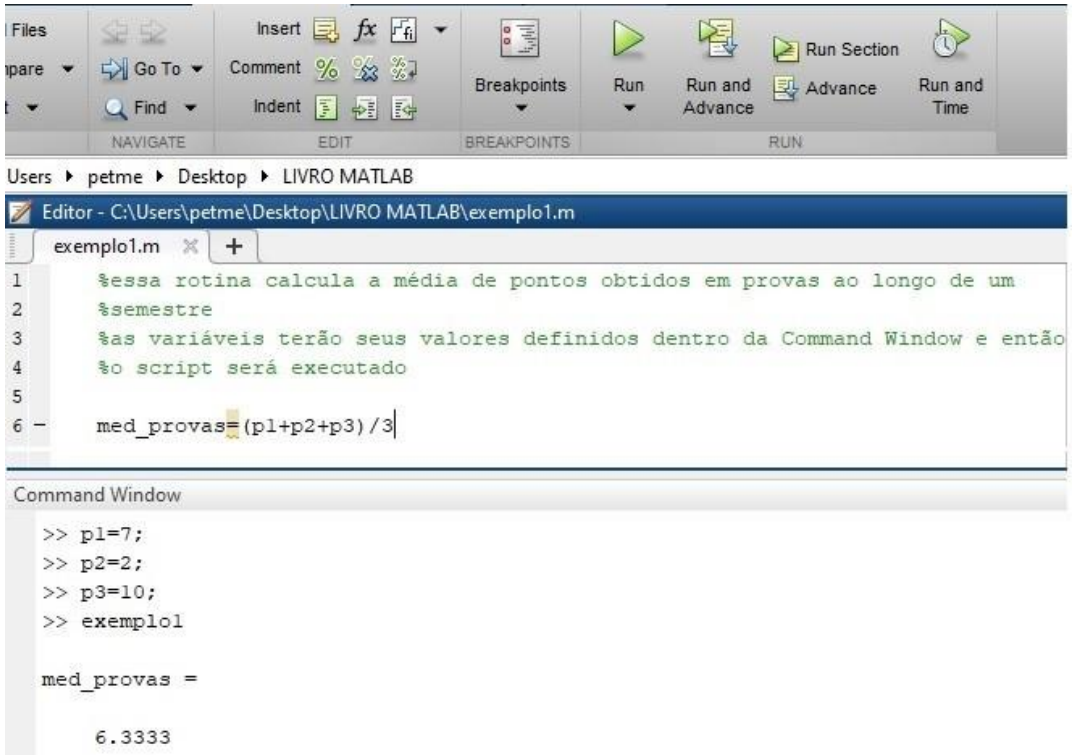
Criando e salvando um script:



Rotinas ou Arquivos M-Files - Scripts

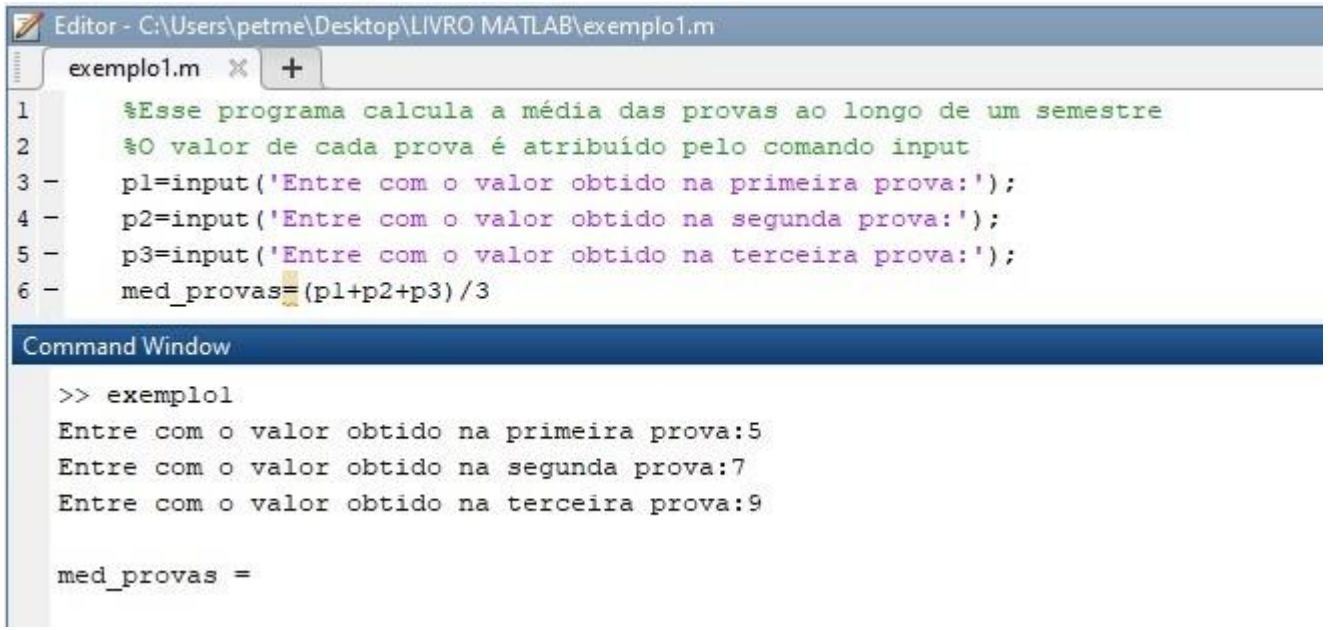
Definindo variáveis

Podemos usar variáveis que foram definidas no Command Window



Rotinas ou Arquivos M-Files - Scripts

As variáveis podem também serem declaradas no script e terem os seus valores atribuídos no Command Window.




```
a = input('Mensagem');
fprintf('Mensagem');
disp('Mensagem');
```

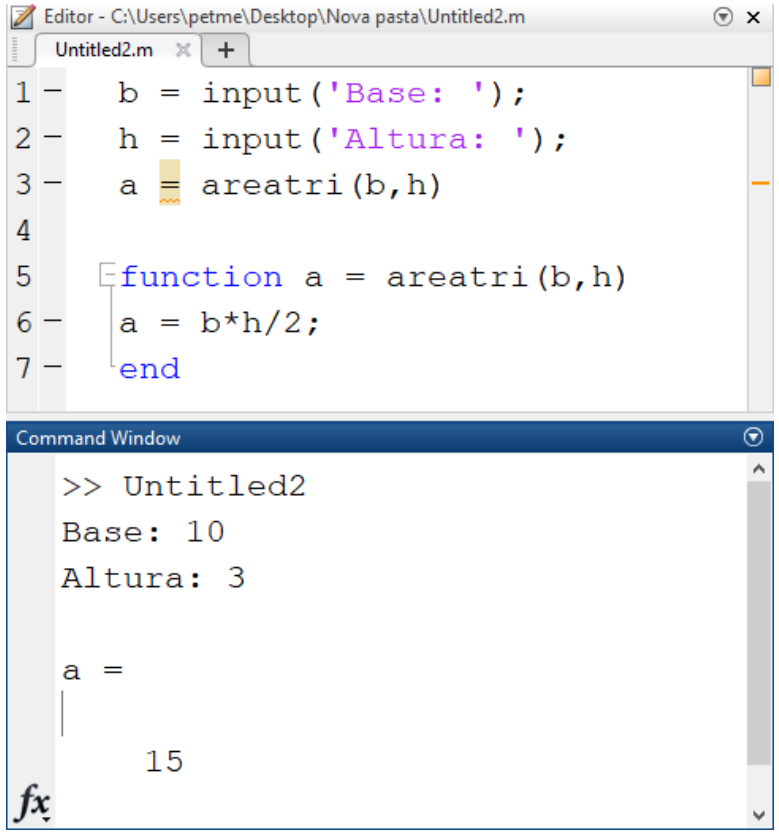
Rotinas ou Arquivos M-Files - Scripts

Funções

Assim como em outras linguagens de programação, o MATLAB permite a criação de funções. Podem ser declaradas no fim do script ou em um script à parte.

```
function saida = nome (parametros)
<comandos>
end
```

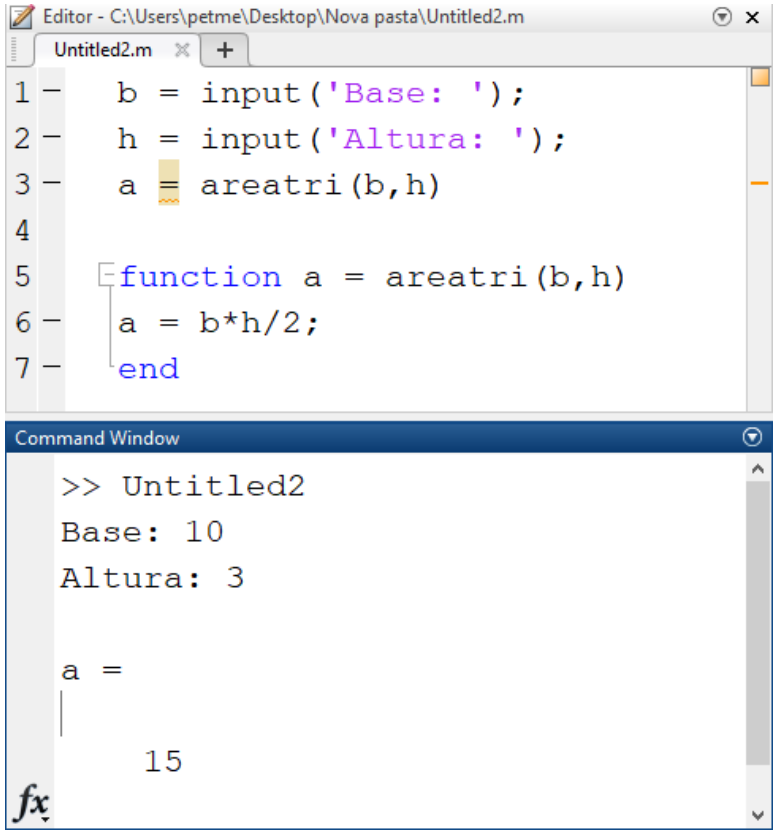
Para executar a função, chame o nome da função e os parâmetros.



Rotinas ou Arquivos M-Files - Scripts

Funções

Para salvar em um arquivo externo, salve o arquivo com o mesmo nome da função.



Operadores lógicos e Relacionais – Operadores relacionais

Os operadores relacionais servem para fazer a comparação entre dois dados. No caso do MATLAB, é possível realizar a comparação de duas matrizes de mesmo número de linhas e colunas ou para comparar uma matriz e um escalar.

Operador	Significado
<	menor que
<=	menor ou igual
>	maior que
>=	maior ou igual
==	igual
~=	diferente (não igual)

Operadores lógicos e Relacionais – Operadores relacionais

→ Exemplo

Se a comparação for verdadeira o Matlab irá retornar 1. Caso não seja, o programa irá retornar 0.

```
>> A = [1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> B = [0 6 3; 7 6 6]

B =

     0     6     3
     7     6     6

>> c = 4

c =

     4
```

```
>> A < B

ans =

     0     1     0
     1     1     0

>> A >= c

ans =

     0     0     0
     1     1     1
```

Operadores lógicos e Relacionais – Operadores lógicos

É possível combinar duas ou mais comparações (operações relacionais) utilizando os operadores lógicos

Operador	Significado
&	E
|	OU
~	NÃO

Operadores lógicos e Relacionais – Operadores lógicos

→ E

Quando duas expressões forem combinadas com E (&) seu resultado só será 1 (*verdadeiro*) se o resultado das duas expressões individualmente for 1 (verdadeiro). Caso uma delas seja 0, o resultado da operação & é 0 (*falso*).

Combinação com E	Falso	Verdadeiro
Falso	0	0
Verdadeiro	0	1

Operadores lógicos e Relacionais – Operadores lógicos

→ OU

O resultado combinação de duas expressões utilizando com OU (|) será 1 se pelo menos o resultado individual de uma das expressões for 1. OU é 0 apenas se as duas expressões combinadas forem 0.

Combinação com OU	Falso	Verdadeiro
Falso	0	1
Verdadeiro	1	1

Operadores lógicos e Relacionais – Operadores lógicos

→ NÃO

O papel da operação NÃO (~) é inverter o resultado de toda a expressão. Ou seja, se o resultado da expressão for 1, utilizar a operação NÃO a transforma para 0, e vice-versa.

```
>> A = 1;
>> B = 2;
>> C = 4;

>> A > B & C > B

ans =

     0

>> A > B | C > B

ans =

     1
```

Declaração de Variáveis

→ Nome de variável:

Letras maiúsculas e minúsculas se diferenciam

Permitido números, letras e underline.

Declaração de Variáveis

→ Variáveis reservadas

Ans	Variável onde são guardados os resultados de operações não atribuídas a uma variável específica - ans, diminutivo de answer.
pi	Valor de π = 3.1416.
Eps	Unidade de arredondamento da máquina, i.e., o menor valor que adicionado a 1 representa um número maior que 1
Flops	Contador do número de operações efetuadas. Estamos a falar de operações em vírgula flutuante.
Inf	Representa

Matrizes e Vetores – Matrizes

→ Escrevendo uma matriz

```
M = [1 2 3; 4 5 6; 7 8 9]
```

```
M =

     1     2     3

     4     5     6

     7     8     9
```

Matrizes e Vetores – Vetores

→ Vetor gerado por incremento

```
v = 1:5
```

```
v =

     1     2     3     4     5
```

Matrizes e Vetores – Vetores

→ Vetor gerado por incremento

```
v = 1:2:10
```

```
v =

     1     3     5     7     9
```

Matrizes e Vetores – Vetores

→ Vetor gerado por incremento

```
v = linspace(início, fim, quantidade)
```

```
>> v=linspace(2,6,10)

v =

    2.0000    2.4444    2.8889    3.3333    3.7778    4.2222
4.6667    5.1111    5.5556    6.0000
```

Matrizes e Vetores – Matrizes

→ Matriz randômica

```
rand(n°deLinhas,n°deColunas)
```

```
rand(2,3)

ans =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975
```

Matrizes e Vetores – Matrizes

→ Matriz quadrada mágica

`magic(dimensão)`

```
>> magic(5)

ans =

    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
```

Matrizes e Vetores – Matrizes

→ zeros, ones, eye

```
>> zeros(2,3)

ans =

     0     0     0
     0     0     0
```

```
>> ones(3,4)

ans =

     1     1     1     1
     1     1     1     1
     1     1     1     1
```

```
>> eye(4)

ans =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Matrizes e Vetores – Matrizes

→ pascal

```
>> pascal(5)

ans =

     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
```

Matrizes e Vetores – Operações

→ Adição (A+B)

→ Subtração (A-B)

→ Multiplicação A*B???

Matrizes e Vetores – Solução de sist. linear

→ A\b

```
>> A=[2 3 5; 1 5 7; 1 1 1];
>> b=[2;4;7];
>> A\b

ans =

    4.0000
   10.5000
   -7.5000
```

Matrizes e Vetores – Solução de sist. linear

→ A\b

```
>> A=[2 3 5; 1 5 7; 1 1 1];
>> b=[2;4;7];
>> A\b

ans =

    4.0000
   10.5000
   -7.5000
```

Matrizes e Vetores – Transposta

→ A’

```
>> A=[2 3 5; 1 5 7; 1 1 1];
>> A'

ans =

     2     1     1
     3     5     1
     5     7     1
```

Matrizes e Vetores – Op. Elemento a elemento

→ A*A x A.*A

```
A =

     2     3     5
     1     5     7
     1     1     1
```

```
>> A*A

ans =

    12    26    36
    14    35    47
     4     9    13
```

```
>> A.*A

ans =

     4     9    25
     1    25    49
     1     1     1
```

Matrizes e Vetores – Op. Elemento a elemento

→ A^2 x A.^2

```
A =

     2     3     5
     1     5     7
     1     1     1
```

```
>> A^2

ans =

    12    26    36
    14    35    47
     4     9    13
```

```
>> A.^2

ans =

     4     9    25
     1    25    49
     1     1     1
```

Matrizes e Vetores – Termos

→ Considere

```
>> B=rand(10)

B =

    0.7577    0.8235    0.4898    0.4984    0.9593    0.3500    0.2858    0.1299    0.6020    0.8258
    0.7431    0.6948    0.4456    0.9597    0.5472    0.1966    0.7572    0.5688    0.2630    0.5383
    0.3922    0.3171    0.6463    0.3404    0.1386    0.2511    0.7537    0.4694    0.6541    0.9961
    0.6555    0.9502    0.7094    0.5853    0.1493    0.6160    0.3804    0.0119    0.6892    0.0782
    0.1712    0.0344    0.7547    0.2238    0.2575    0.4733    0.5678    0.3371    0.7482    0.4427
    0.7060    0.4387    0.2760    0.7513    0.8407    0.3517    0.0759    0.1622    0.4505    0.1067
    0.0318    0.3816    0.6797    0.2551    0.2543    0.8308    0.0540    0.7943    0.0838    0.9619
    0.2769    0.7655    0.6551    0.5060    0.8143    0.5853    0.5308    0.3112    0.2290    0.0046
    0.0462    0.7952    0.1626    0.6991    0.2435    0.5497    0.7792    0.5285    0.9133    0.7749
    0.0971    0.1869    0.1190    0.8909    0.9293    0.9172    0.9340    0.1656    0.1524    0.8173
```

Matrizes e Vetores – Termos

→ Então

```
>> B(1,2)

ans =

    0.8235
```

Matrizes e Vetores – Termos

→ Então

```
>> B(3,:)

ans =

    0.3922    0.3171    0.6463    0.3404    0.1386    0.2511    0.7537
0.4694    0.6541    0.9961
```

Matrizes e Vetores – Termos

→ Então

```
>> B(:,3)

ans =

    0.4898
    0.4456
    0.6463
    0.7094
    0.7547
    0.2760
    0.6797
    0.6551
    0.1626
    0.1190
```

Matrizes e Vetores – Termos

→ Então

```
>> B(2:4,:)

ans =

    0.7431   0.6948   0.456    0.9597   0.5472   0.1966   0.7572   0.5688   0.2630   0.5383
    0.3922   0.3171   0.6463   0.3404   0.1386   0.2511   0.7537   0.4694   0.6541   0.9961
    0.6555   0.9502   0.7094   0.5853   0.1493   0.6160   0.3804   0.0119   0.6892   0.0782
```

Matrizes e Vetores – Determinante

→ det (B)

```
>> det(B)

ans =

    0.0081
```

Matrizes e Vetores – Inversa

→ inv (B)

```
>> inv(B)

ans =

  -17.5686  -22.1791   39.7960   -0.7247  -27.8529   23.4407   -1.9684   14.4951   -2.2404    0.3120
    1.4354    1.2051   -2.2684    0.3217    0.7476   -1.9230    0.1281   -0.2613    0.5317   -0.3175
   14.9218   19.2208  -33.2936    1.2043   24.0248  -19.8941    1.7906  -12.4162    0.7152   -0.4059
   18.2597   24.6938  -42.6279    1.5098   29.6867  -24.1443    2.1766  -16.7602    1.8466   -0.0505
    0.6875   -0.0954   -0.4748   -0.8260    0.4301    0.0705   -0.1887    0.5820    0.0257   -0.0219
  -13.1722  -16.9325   28.4983   -0.1690  -20.2140   16.9968   -0.9622   10.7352   -1.2805    0.7610
   -8.4901  -10.3920   19.1248   -0.5786  -13.0649   10.1955   -1.5218    7.6340   -0.8516    0.4706
  -11.4216  -13.4255   24.2549   -1.5379  -16.9185   14.7944   -0.7385    9.6482   -0.5253   -0.4857
   -0.2780   -0.9368    0.6739   -0.1862    0.1148    1.0544   -0.3348    0.0495    0.6673   -0.3444
    5.7528    6.5262  -11.2355    0.3279    7.8149   -7.1317    0.8168   -4.9833    0.3908    0.1690
```

Matrizes e Vetores – Autovalores

→ eig (A)

```
>> A=[2 3 5; 1 7 5; -2 5 -1]

A =

     2     3     5
     1     7     5
    -2     5    -1
```



```
>> eig(A)

ans =

   -1.0000
   -0.4244
    9.4244
```

Matrizes e Vetores – Autovetores

→ [P,D]=eig (A)

```
[P,D]=eig(A)

P =

    0.7321    0.7998    0.5186
    0.2929    0.2575    0.8055
   -0.6150   -0.5423    0.2868


D =

   -1.0000         0         0
         0   -0.4244         0
         0         0    9.4244
```

Matrizes e Vetores – Eq. Caract.

→ poly(A)

```
>> poly(A)

ans =

    1.0000   -8.0000  -13.0000   -4.0000
```

Matrizes e Vetores – NORMA

→ norm(v)

```
>> v=[2  3  5]

v =

     2     3     5

>> norm(v)

ans =

    6.1644
```

Matrizes e Vetores – NORMA

→ norm(v)

```
>> v=[2  3  5]

v =

     2     3     5

>> norm(v)

ans =

    6.1644
```

Matrizes e Vetores – Produto interno

→ dot(u,v)

```
>> u=[2 3 6]; v=[2 6 7];
>> dot(u,v)

ans =

    64
```

Matrizes e Vetores – Produto vetorial

→ cross(u,v)

```
>> cross(u,v)

ans =

   -15    -2
6
```

Matrizes e Vetores – média e desv. padrão

→ mean(u)

Valor médio dos termos de u

→ std(u)

Desvio padrão dos termos de u

```
>> u=[2 3 6];
>> mean(u)

ans =

    3.6667

>> std(u)

ans =

    2.0817
```

Matrizes e Vetores – soma e mediana

→ median(v)

Mediana dos termos de v

→ sum(v)

Soma dos termos de v

```
v =
     3    5    7    1    9    2    3    24    -2    6    -3     19

>> median(v)
ans =
     4

>> sum(v)
ans =

    74
```

Matrizes e Vetores - extremos

→ max(v)

Maior valor de v

→ min(v)

Menor valor de v

```
v =
     3    5    7    1    9    2    3   24   -2    6   -3   19

>> max(v)
ans =
     24

>> min(v)
ans =

    -3
```

Matrizes e Vetores - ordenar

→ sort(v)

Ordena em ordem crescente

```
v =
     3     5     7     1     9     2     3    24    -2     6    -3    19

>> sort(v)

ans =

    -3    -2     1     2     3     3     5     6     7     9    19    24
```

Matrizes e Vetores - Dimensões

→ length(v) e size(B)

Dão dimensões dos vetores / matrizes

```
>> B = [1 2 3;6 2 1];
>> size(B)
ans =
     2     3

>> size(B,1)
ans =
     2
```

```
>> v = linspace(1,12,12);

>> length(v)
ans =
    12
```

Comandos de fluxo

Os comandos de fluxo são usados para desviar o fluxo natural do código, ou seja, executar ou não uma ação dependendo de uma condição ou executar uma mesma ação repetidas vezes.

Comandos de fluxo – For

→ For

O for é um comando de loop é utilizado quando se quer realizar uma série de comandos por um número de vezes fixo e predefinido.

Estrutura básica do comando

```
for i = 1:10

<comandos>

end
```

Comandos de fluxo – For

→ Exemplo

```
for i=1:5
    for j = 1:5
        A(i,j) = i + j;
    end
end

A =

     2     3     4     5     6
     3     4     5     6     7
     4     5     6     7     8
     5     6     7     8     9
     6     7     8     9    10
```

Comandos de fluxo – While

→ While

O loop while é executado enquanto o resultado de condição predeterminada for verdadeira.

Estrutura básica do comando

```
while <condição>

<comandos>

end
```

Comandos de fluxo – While

→ Exemplo

```
a = 1;
b = 10;
i = 1;
while b >= a
A(i,i) = i^2;
b = b-i;
i = i+1;
end

A =

     1     0     0     0
     0     4     0     0
     0     0     9     0
     0     0     0    16
```

Comandos de fluxo – If-Else

→ If-Else

O operador if-else serve para executar um bloco de comandos se uma condição for verdadeira. Senão, outro bloco de comandos será executado.

Estrutura básica do comando

```
if <condição>
<comandos1>

else
<comandos2>
end
```

Comandos de fluxo – If-Else

→ Exemplo

```
a = rand(1,1)
if a < 0.5
    A = linspace(0,70,8)
else
    A = linspace(1,50,8)
end

a =

    0.8491


A =

     1     8    15    22    29    36    43    50
```

Polinômios

Um polinômio é representado por um vetor linha contendo os coeficientes do polinômio em ordem decrescente

Exemplo:

Declaração do polinômio $-x^5 - 5x^4 + 8x^2 + 30$

```
p = [-1 -5 0 8 0 30].
```

Polinômios - Raízes

→ roots()

Calcula a raiz do polinômio.

```
>> r = roots(p)

r =

  -4.5416 + 0.0000i
  -2.2503 + 0.0000i
   1.6741 + 0.0000i
   0.0589 + 1.3229i
   0.0589 - 1.3229i
```

→ poly()

Encontra o polinômio correspondente a uma determinada raiz.

```
>> poly(r)

ans =

    1.0000    5.0000    0.0000   -8.0000    0.0000  -30.0000
```

Polinômios - Produto

→ conv()

Calcula o produto entre dois polinômios

```
>> p1 = [1 2 -3]

p1 =

     1     2    -3

>> p2 = [-1 5]

p2 =

    -1     5

>> conv(p1,p2)

ans =

    -1     3    13   -15
```

Polinômios - Divisão

→ deconv()

Realiza a divisão entre dois polinômios

```
>> [q, r] = deconv(p1,p2)

q =

    -1    -7

r =

     0     0    32
```

Polinômios – Avaliação de polinômios

Polinômios podem ser avaliados de duas formas distintas

1ª Forma:

Se x for um escalar

```
>> x = 2;

>> f = 2*x^4 - 5*x^3 + 8*x^2 - 10*x + 40

f =

44
```

Polinômios - Avaliação de polinômios

1ª Forma:

Se x for um vetor contendo um intervalo de valores, então será necessário utilizar o operador ponto-escalar

```
>> x = 0:0.5:2;

>> f = 2*x.^4 - 5*x.^3 + 8*x.^2 - 10*x + 40

f =

40.0000 36.5000 35.0000 36.2500 44.0000
```

Polinômios - Avaliação de polinômios

2ª Forma:

→ polyval()

Avalia numericamente o polinômio para um dado valor ou conjunto de valores de x

```
>> x = 0:0.5:2;

>> p = [2 -5 8 -10 40];

>> y = polyval(p,x)

y =

40.0000 36.5000 35.0000 36.2500 44.0000
```

Gráficos Bidimensionais

O MatLab se apresenta como uma ferramenta potente e eficaz quanto a geração de informações gráficas de até quatro variáveis

Gráficos Bidimensionais – Criando um gráfico

→ figure

Abre uma nova janela para plotar gráficos

→plot(x,y)

Gera um gráfico que relaciona os vetores x e y.

Os vetores devem ter as mesmas dimensões

Gráficos Bidimensionais – Criando um gráfico

→ xlabel(‘texto_X’) e ylabel(‘texto_Y’)

Nomeia os eixos x e y

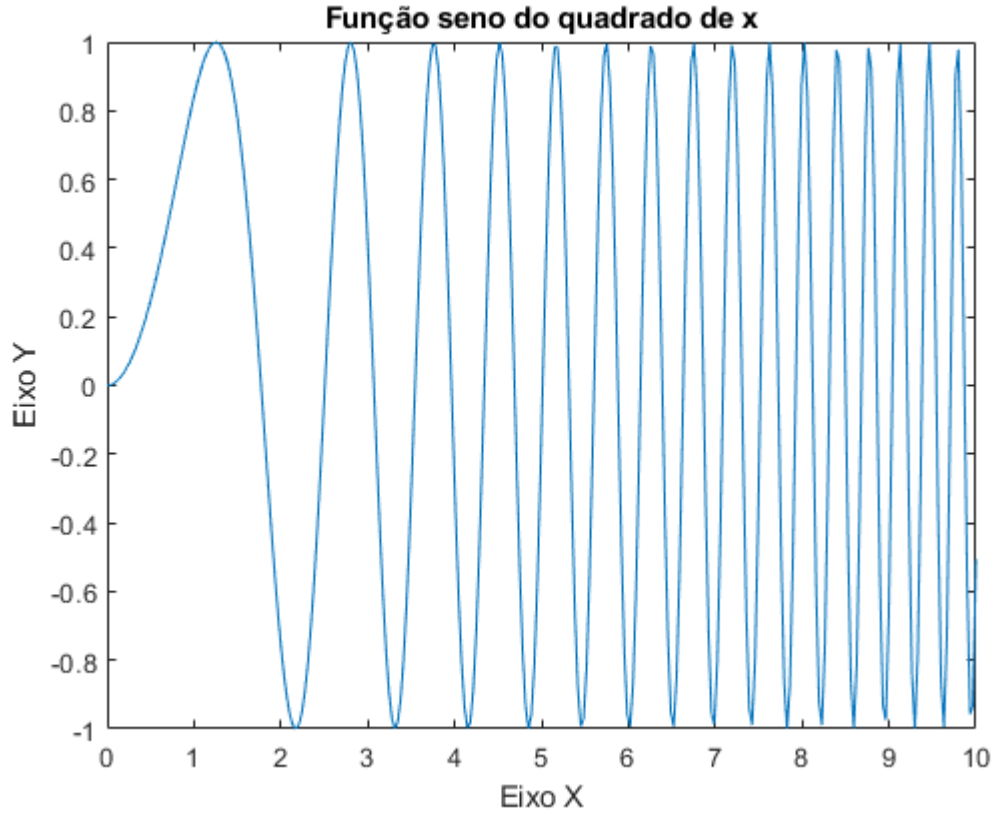
→ title(‘Texto do título’)

Define um título para o gráfico

Gráficos Bidimensionais – Criando um gráfico

→ Exemplo

```
>> x=linspace(0,10,300);
>> y=sin(x.^2);
>> figure
>> plot(x,y)
>> xlabel('Eixo X');
ylabel('Eixo Y');
>> title('Função seno do
quadrado de x');
```



Gráficos Bidimensionais – Criando um gráfico

→ xlim([lim_inferior,lim_superior]);

→ ylim([lim_inferior,lim_superior])

Permite definir os limites dos eixos exibidos na tela do gráfico

Gráficos Bidimensionais – Criando um gráfico

→ grid

Exibe a malha quadriculada no fundo do gráfico, para ajudar a visualizar as retas paralelas aos eixos. Para esconde-la, utilize grid off

Gráficos Bidimensionais – Criando um gráfico

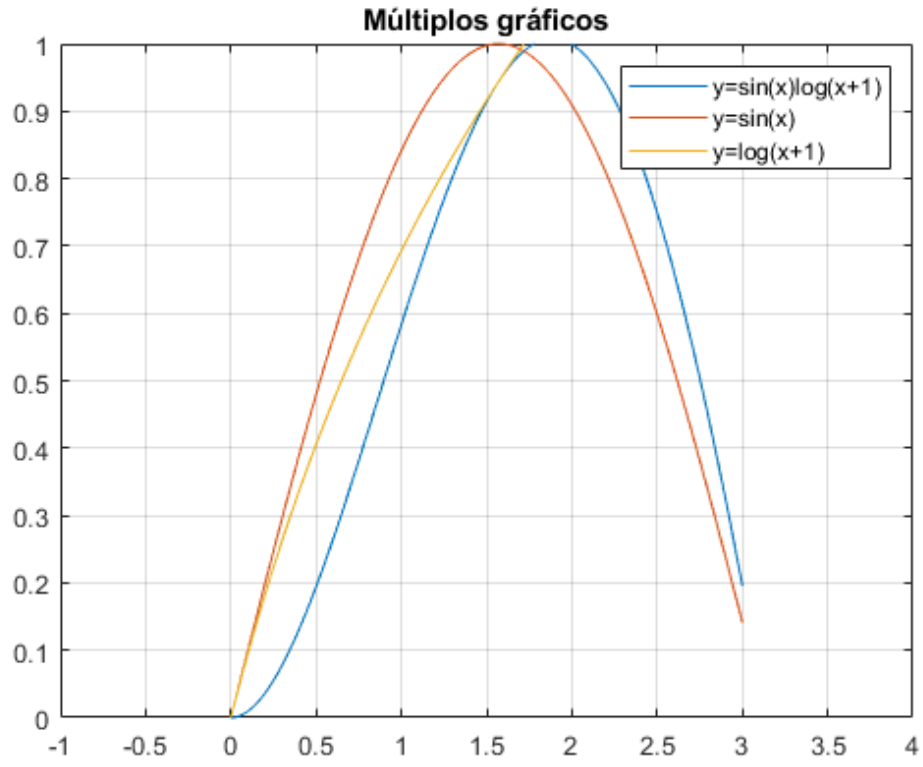
→ hold on

Armazena uma função definida em um plot no próximo comando. Assim é possível plotar mais de uma curva no mesmo gráfico

Gráficos Bidimensionais – Criando um gráfico

→ Exemplo

```
>> x=0:0.01:3;
>> y=sin(x)*log(x+1);
>> y=sin(x).*log(x+1);
>> plot(x,y)
>> grid;
>> xlim([-1 4]);
>> ylim([0 1]);
>> hold on
>> y2=sin(x);
>> y3=log(x+1);
>> plot(x,y2)
>> plot(x,y3)
>> title('Múltiplos gráficos')
>> legend('y=sin(x)log(x+1)','y=sin(x)','y=log(x+1)')
```



Gráficos Bidimensionais – Criando um gráfico

→ ‘Color’,parâmetro

Há três formas de se definir a cor de uma linha, ou de um elemento gráfico, pelo nome, pelo atalho, ou pelo array de fração de RGB.

[1 1 0]	y	yellow
[1 0 1]	m	magenta
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue

Gráficos Bidimensionais – Criando um gráfico

Outros atalhos para cores

cyan(c)
[.3 0 1] = 30% de vermelho, 0% de verde, 100% de azul

white(w)
[.2 .5 1] = 20% de vermelho, 50% de verde, 100% de azul

black(k)
[.9 .42 .2] = 90% de vermelho, 42% de verde, 20% de azul

Gráficos Bidimensionais – Criando um gráfico

→ Estilo da linha

Dentro do comando plot, pode-se atribuir à linha de gráfico um estilo de linha ou marcador para representar os pontos, para alterar o estilo

Gráficos Bidimensionais – Criando um gráfico

→ Estilo da linha

Comandos	Descrição
'-'	Linha cheia
'--'	Linha tracejada
':'	Linha pontilhada
'-.'	Linha traço-ponto
'+'	Marcador em cruz
'o'	Marcador circular
'*'	Marcador em asterisco
'X'	Marcador em x

Gráficos Bidimensionais – Criando um gráfico

→ Estilo da linha

Comandos	Descrição
‘square’ ou ‘s’	Marcador quadrado
‘diamond’ ou ‘d’	Marcador em diamante
‘^’	Marcador triangular para cima
‘v’	Marcador triangular para baixo
‘>’	Marcador triangular para direita
‘<’	Marcador triangular para esquerda
‘pentagram’ ou ‘p’	Marcador estrela 5 pontas
‘hexagram’ ou ‘h’	Marcador estrela de 6 pontas

Gráficos Bidimensionais – Criando um gráfico

→ ‘linewidth’, valor_peso

Altera o peso da linha no gráfico, o número que sucede ‘linewidth’ é o peso associado.

Gráficos Bidimensionais – Criando um gráfico

subplot(m,n,p)

Plota vários gráficos na mesma janela de figura. Os valores de *m* e *n* devem ser fixados, eles designam a quantidade de gráficos que será mostrada na tela em que m é o número de gráficos por linha e n é o número de gráficos por coluna. O valor de p varia de acordo com a posição que é desejada mostrar.

Gráficos Bidimensionais – Criando um gráfico

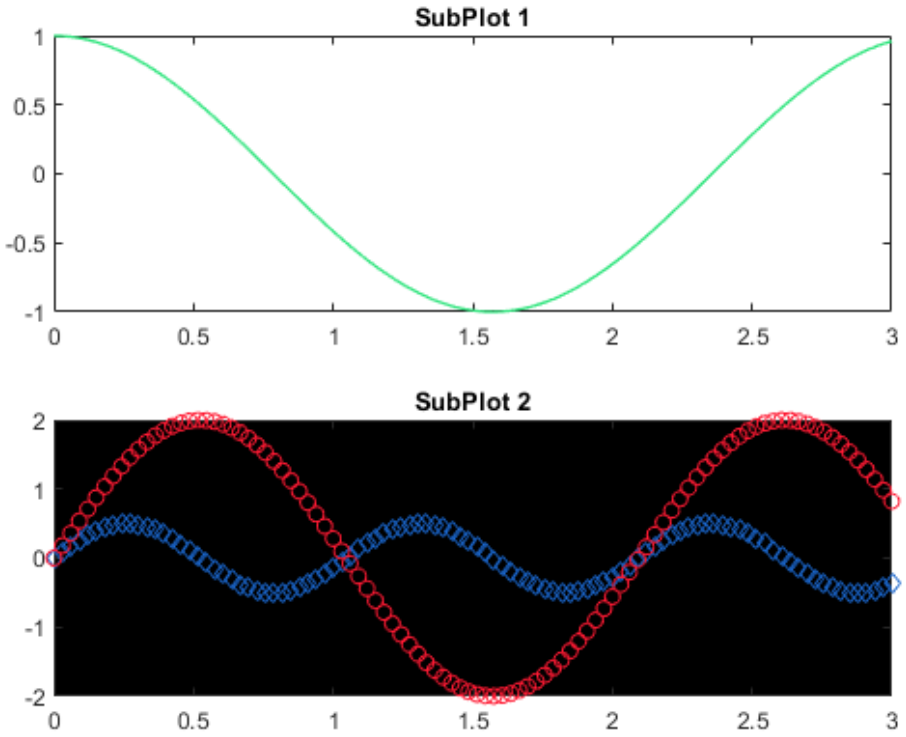
→ set(gca, ‘Color’, parâmetro)

Muda a cor do background

Gráficos Bidimensionais – Criando um gráfico

→ Exemplo

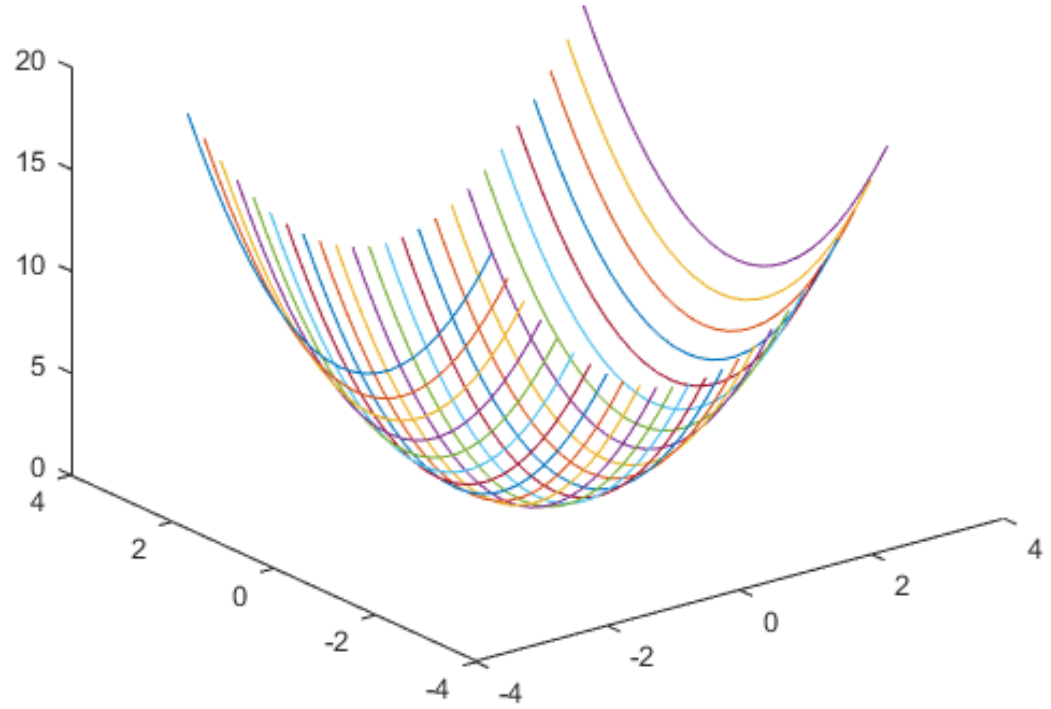
```
x=linspace(0,3,100)
y=sin(3*x)
z=cos(2*x)
w=1/2*sin(6*x)

sub1=subplot(2,1,1)
sub2=subplot(2,1,2)

plot(sub1,x,z,'Linewidth',1,'Color',[.2 .9 .5])
title(sub1,'SubPlot 1')
plot(sub2,x,w,'d','Linewidth',0.5,'Color',[.1 .4 .8])
hold on;
plot(sub2,x,2*y,'o','Linewidth',0.5,'Color',[1 .1 .2])
set(gca,'Color','k')
title(sub2,'SubPlot 2')
```



Gráficos Tridimensionais – Criando um gráfico

→ plot3(x,y,z)

Plota uma curva tridimensional em que os vetores x, y e z devem ser discretizados



Gráficos Tridimensionais – Criando um gráfico

→ [X,Y]= meshgrid(x,y)

Uma vez definida a discretização dos valores de x e y, deseja-se criar uma matriz de valores para gerar um conjunto de pontos em um domínio retangular para associar a uma imagem Z. Para isso é utilizado o comando meshgrid que vai gerar matrizes X e Y que correspondem ao conjunto de pontos neste domínio retangular

Gráficos Tridimensionais – Criando um gráfico

→ Exemplo

```
>> x=linspace(-3,3,10);
>> y=linspace(-3,3,10);
>> [X,Y]=meshgrid(x,y)

X =

   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000


Y =

   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000
   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333
   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667
   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333
    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667
    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333
    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000
```

Gráficos Tridimensionais – Criando um gráfico

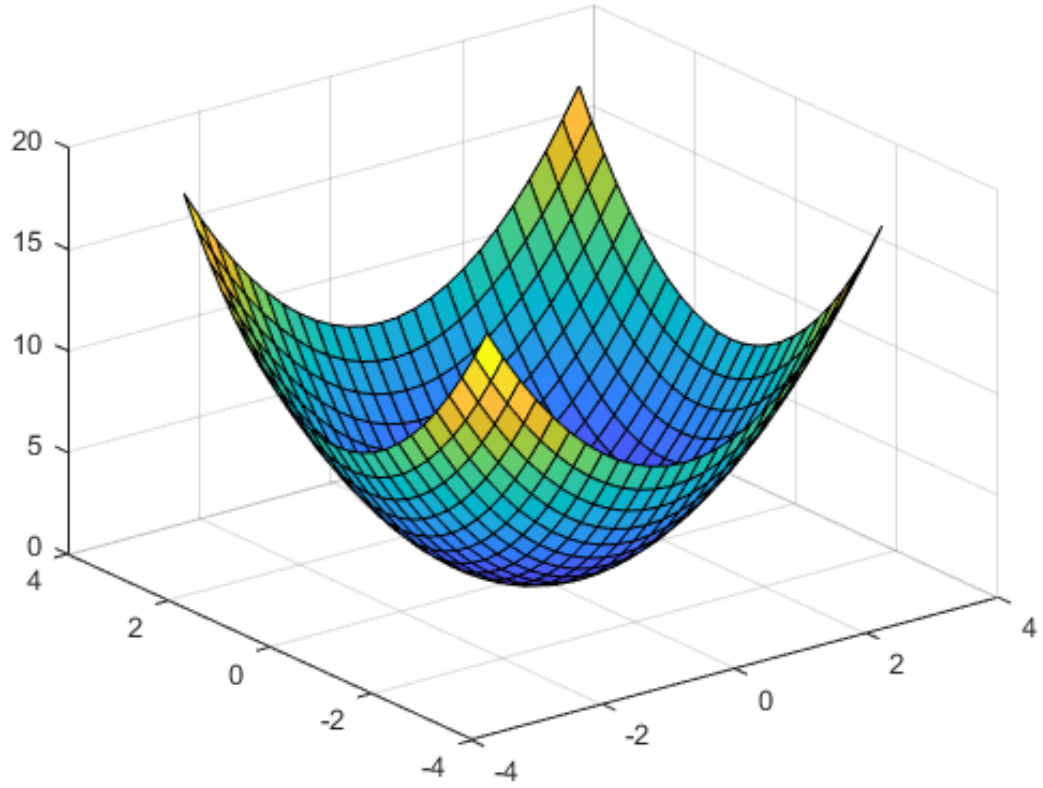
→ surf(X,Y,Z)

Plota uma superfície. As matrizes X, Y e Z devem ter o mesmo tamanho

Gráficos Tridimensionais – Criando um gráfico

→ Exemplo

```
>> Z=X.^2+Y.^2
>> surf(X,Y,Z)
```



Gráficos Tridimensionais – Criando um gráfico

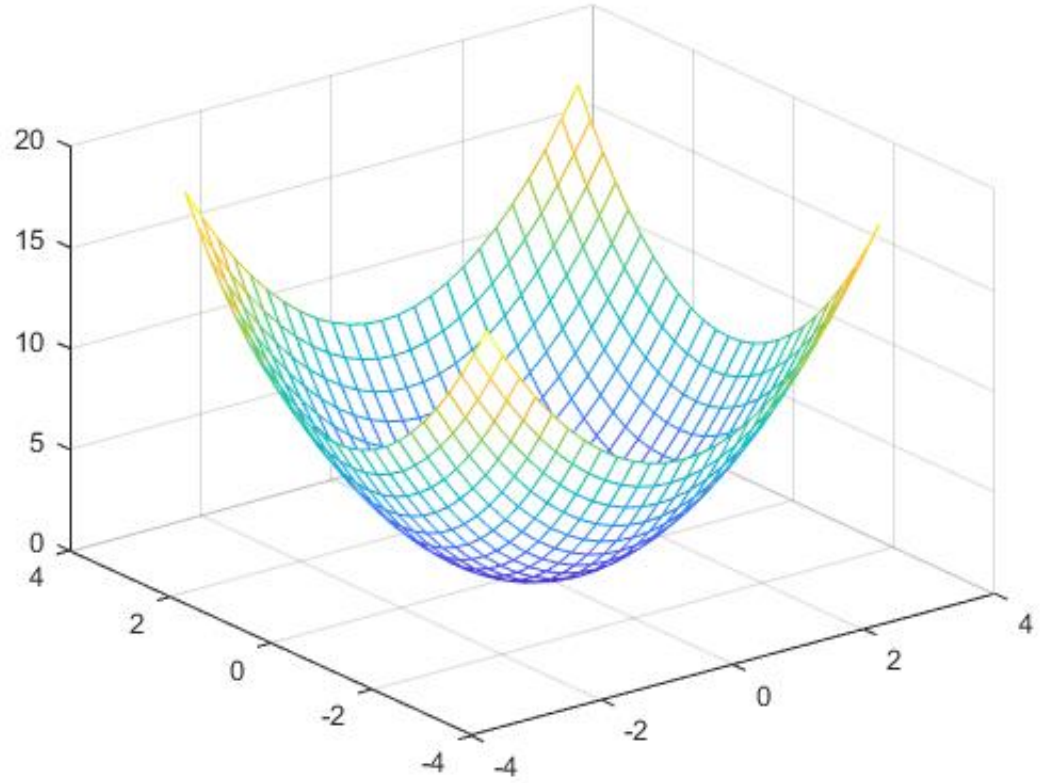
→ mesh(X,Y,Z)

Enquanto o surf gera uma superfície entre os pontos, o comando mesh mostrará os pontos e elementos da função Z(X,Y)

Gráficos Tridimensionais – Criando um gráfico

→ Exemplo

```
>> Z=X.^2+Y.^2
>> plot(X,Y,Z)
```



Gráficos Tridimensionais – Criando um gráfico

→ view(az,el)

Especifica os ângulos associados à vista do gráfico com precisão utilizando os parâmetros *az*(ângulo azimutal) e *el*(ângulo de elevação). Os ângulos são dados em graus.

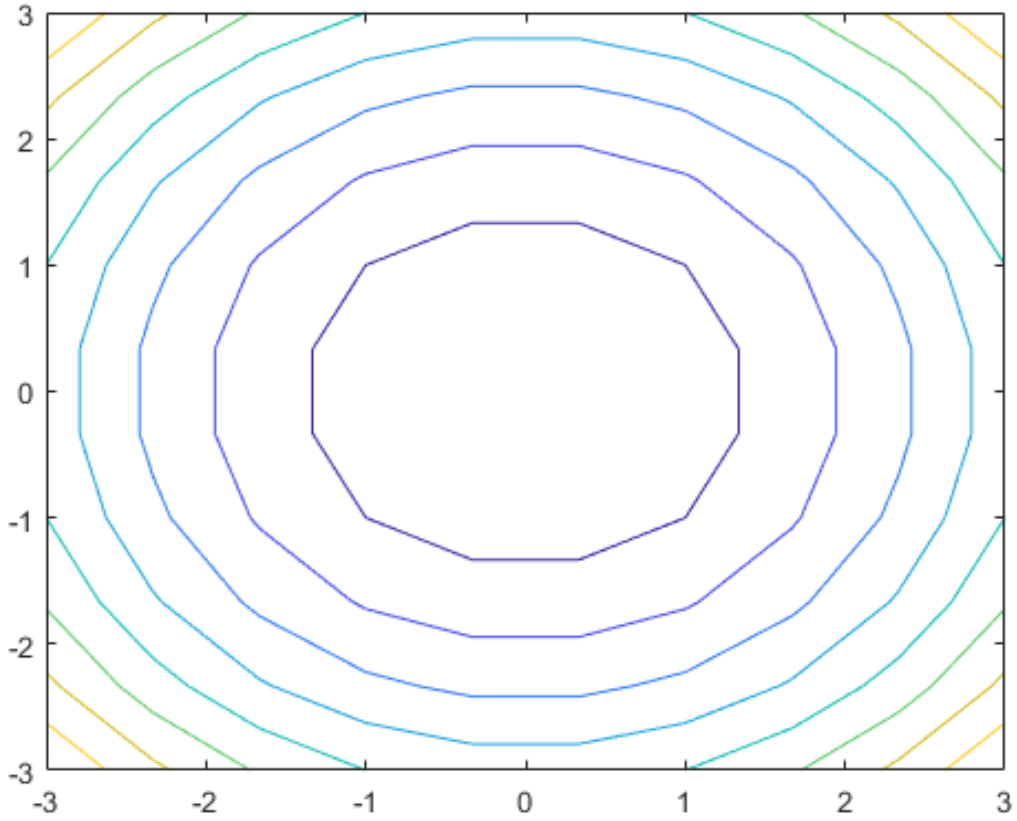
```
>> surf(X,Y,Z)
>> view(60,30)
```

Gráficos Tridimensionais – Criando um gráfico

→ contour(X,Y,Z)

Cria um mapa de curvas de níveis bidimensionais.

```
>> contour(X,Y,Z)
```



Interpolação e Extrapolação

Quando se trabalha com um conjunto de dados coletados de maneira discreta, por vezes, é vantajoso pensar em uma função analítica contínua que representa aquele conjunto de dados da melhor maneira possível. Aproximações podem ser desenvolvidas para se obter estimativas de $f(x)$, em valores de que não constam do conjunto conhecido.

Interpolação e Extrapolação

→ Interpolação de curva

Determinar uma estimativa de um valor x_i pertencente ao intervalo de dados que se localiza entre o valor mínimo e máximo conhecido.

$$X_1 \leq x_i \leq X_N$$

→ Extrapolação

Predizer sobre o valor de um dado x_i fora do conjunto de dados conhecido

$$x_i \leq X_1 \quad \text{ou} \quad x_i \geq X_N$$

→ Interpolação e Extrapolação polinomial

Suponha que tenha sido importada para o MatLab uma tabela contendo os seguintes valores para um dado experimento que relaciona a profundidade de um tubo (P), em metro, com a temperatura(T), em Kelvin. Para realizar uma interpolação unidimensional $y = f(x)$ no nosso conjunto de dados basta utilizar interp1(P,T,p) em que p é um valor de P não contido no conjunto fornecido.

Interpolação e Extrapolação

→ Função Interp1

Sintaxes:

Para interpolação:

```
interp1(x,Y,xi) ou
interp1(x,Y,xi,method);
```

Para extrapolação:

```
interp1(x,Y,xi,method,'extrap')
```

A string ‘extrap’ é adicionada para extrapolação.

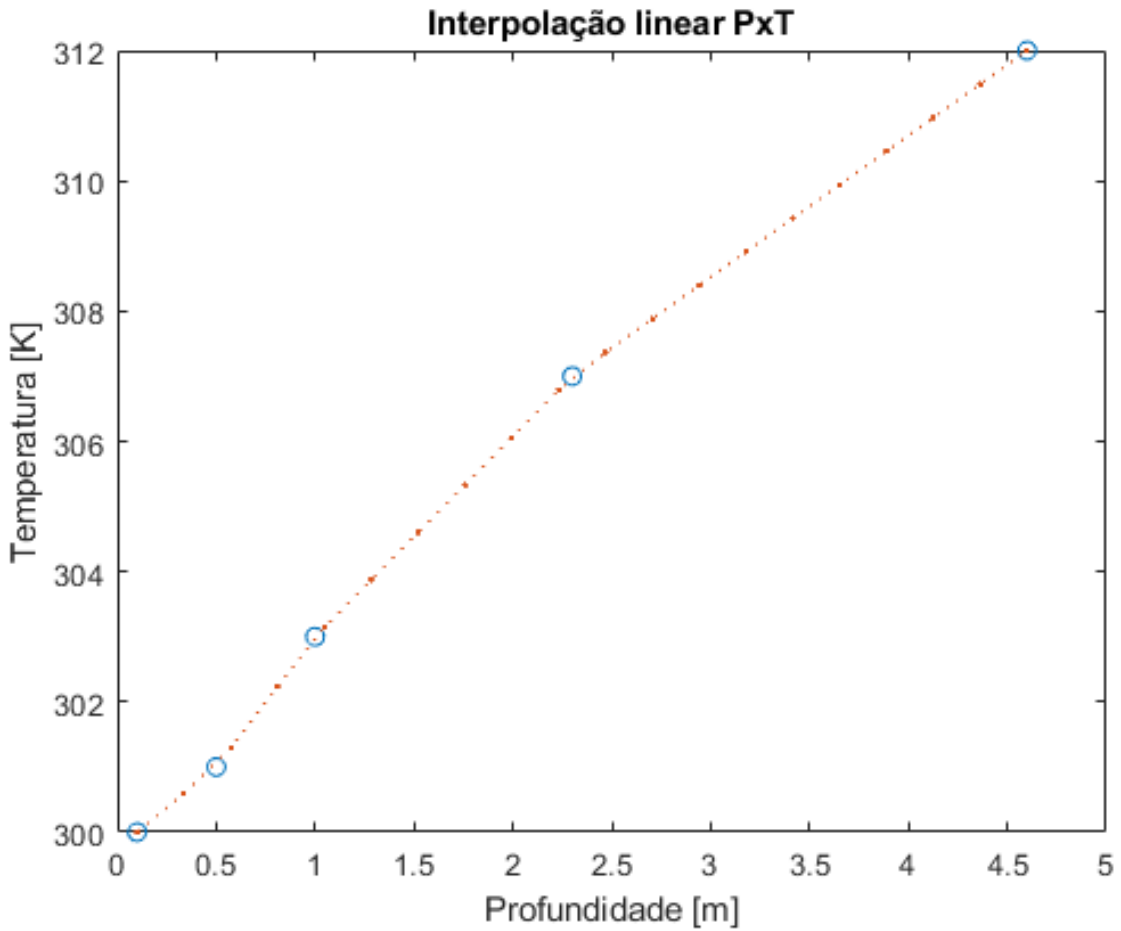
Entrada	Descrição	
x, Y	Vetor contendo os pontos conhecido	
x_i	Pode ser um escalar ou um vetor unidimensional ou bidimensional.	
method	’nearest’	Interpolação vizinha mais próxima
	‘linear’	Interpolação linear
	‘spline’	Spline cúbico
	‘pchip’	Interpolação de Hermite
	‘cubic’	Interpolação cúbica
	‘v5cubic’	Interpolação cúbica matlab

Interpolação e Extrapolação

→ interp1(P,T,p)

```
>> P=[0.1;0.5;1;2.3;4.6]

P =

    0.1000
    0.5000
    1.0000
    2.3000
    4.6000

>> T=[300;301;303;307;312]

T =

   300
   301
   303
   307
   312

>> interp1(P,T,3.5)

ans =

  309.6087

>> interp1(P,T,'pchip',4.7)

ans =

  312.1565
```

Interpolação e ajuste de curvas – Interpolação polinomial



Interpolação e ajuste de curvas – Interpolação polinomial

Na interpolação, normalmente fazemos uma conexão entre alguns pontos, por um tipo específico de função.

Para interpolar um polinômio de grau N a um conjunto de N+1 pontos, resolve-se a equação linear:

$$a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_{n+1} x_i^n = y_i$$

Que é resolvida em um sistema de N+1 equações, uma equação para cada ponto i={1,...N+1}.

Para interpolar um segmento de reta, são necessários dois pontos. Para interpolar uma parábola são necessários 3 pontos

$$y_1 = ax_1^2 + bx_1 + c$$
$$y_2 = ax_2^2 + bx_2 + c$$
$$y_3 = ax_3^2 + bx_3 + c$$

E para polinômios de grau n, são necessários n+1 pontos.

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curvas pelo comando *polyfit*

→ p=polyfit(x,y,n）

p = o vetor que receberá os **coeficientes** do polinômio de ajuste

x = o vetor que contém os valores das abcissas dos pontos que serão ajustados

y = o vetor que contém os valores das ordenadas dos pontos que serão ajustados

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curvas pelo comando *polyfit*

→ p=polyfit(x,y,n）

n = indica o grau do polinômio que se deseja o ajuste (n=1, 2, 3...)

Interpolação e ajuste de curvas – Ajuste de curvas

→ Exemplo

```
>> x = [1 2 3 5];
>> y = [2 4 6 11];
>> p = polyfit(x,y,1)

p =

    2.2571   -0.4571
```

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curva para funções que não são polinomiais

→ Potência: $y = bx^m$

```
>> p=polyfit(log(x),log(y),1)
```

→ Exponencial: $y = be^{mx}$

```
>> p=polyfit(x,log(y),1)
```

$y = b10^{mx}$

```
>> p=polyfit(x,log10(y),1)
```

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curva para funções que não são polinomiais

→ Logarítimica: $y = m \ln x + b$

```
>> p=polyfit(log(x),y,1)
```

$$y = m \log x + b$$

```
>> p=polyfit(log10(x),y,1)
```

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curva para funções que não são polinomiais

→ Hiperbólica: $y = \frac{1}{mx+b}$

```
>> p=polyfit(x,1./y,1)
```

Mínimos e Máximos de uma função

→x = fminbnd(‘função’, x1,x2)

Usado para encontrar o mínimo e o máximo de uma função. X1 e x2 devem ser os limites do intervalo que se deseja avaliar.

→[x valor] = fminbnd(‘função’,x1,x2)

Usado para atribuir o resultado do comando em um valor

Mínimos e Máximos de uma função

→Exemplo

```
>> [x valor]=fminbnd('x^3 -12*x^2 +40.25*x -
36.5',0,8)

x = %valor onde a função apresenta o mínimo valor%

5.6073

valor = %o mínimo valor da função no intervalo
especificado%

-11.8043
```

Mínimos e Máximos de uma função

→Exemplo

Quando se desejar encontrar o máximo valor da função em um dado intervalo, basta multiplicar toda a função desejada por “-1” e manter a mesma estrutura de comando

```
>> [x valor]=fminbnd+('-x^3 12*x^2 -40.25*x +36.5',0,8)

x =

2.3927

valor =

-4.8043
```

Diferenciação

Dados dois pontos muito próximos podemos aproximar a derivada da forma a seguir

$$\frac{dy}{dx} \approx \frac{y_2 - y_1}{x_2 - x_1}$$

→ diff

Calcula a diferença entre dois pontos adjacentes de um vetor e armazena o resultado em um novo vetor

Diferenciação

→ Exemplo

```
>> x = [0 1 2 3 4]

x =

     0     1     2     3     4

>> diff(x)

ans =

     1     1     1     1

>> y = [2 3 5 1 9 0]

y =

     2     3     5     1     9     0

>> diff(y)

ans =

     1     2    -4     8    -9
```

Diferenciação

→ OBS.:

Se *diff* for utilizado com uma matriz, a função tratará cada coluna da matriz como um vetor e calculará as diferenças para as colunas.

Tendo em mãos os vetores $diff(x)$ e $diff(y)$, para calcular a derivada basta realizar o seguinte cálculo

$$diff(y)\ ./diff(x)$$

Integração Num. – Quadratura de Simpson

→ i=quad(@(x)(fun),a,b) **ou** (‘fun’,a,b)

Atente para o fato que a função deve considerar a sintaxe de operações elemento a elemento

```
>> i=quad('x.^2',0,2)

resi =

    2.6667

>> i=quad(@(x)(x.^2),0,2)

resi =

    2.6667
```

Integração Num. – Quadratura de Simpson

→ i=dblquad(@(x,y)(fun),a,b,ay,by) **ou** (‘fun’,a,b,ay,by)

```
>> i=dblquad(@(x,y)(x.^2+y),0,2,0,3)

i =

    17

>> i=dblquad('x.^2+y',0,2,0,3)

i =

    17
```

Integração Num. – Quadratura de Simpson

→ i=triplequad(@(x,y,z)(fun),a,b,ay,by,az,bz) **ou** (‘fun’,a,b,ay,by,az,bz)

```
>> i=triplequad(@(x,y,z)(x.^2+y+z),0,2,0,3,0,1)

i =

    20

>> i=triplequad('x.^2+y +z',0,2,0,3,0,1)

i =

    20
```

Integração Num. – Quadratura de Simpson

→ **Atente** para a ordem de declaração das variáveis, isso também será sua ordem de integração

```
>> i=triplequad(@(y,z,x)(x.^2+y+z),0,2,0,3,0,1)

i =

    17
```

Integração Num. – Integral

→ i=integral(fun,a,b) | fun=@(x) x.^2

```
>> fun = @(x) x.^2;
>> i=integral(fun,0,2)

i =

    2.6667
```

Integração Num. – Integral

→ i=integral2(fun,a,b,ay,by) | fun=@(x,y) x.^2 +y

```
>> fun = @(x,y) x.^2+y;
>> i=integral2(fun,0,2,0,3)

i =

   17.0000
```

Integração Num. – Integral

→ i=integral3(fun,a,b,ay,by,az,bz) | fun=@(x,y,z) x.^2 +y +z

```
>> fun = @(x,y,z) x.^2+y+z;
>> i=integral3(fun,0,2,0,3,0,1)

i =

   20.0000
```

Matemática Simbólica

→ syms

Usado para declarar variáveis simbólicas.

→ pretty(S)

Usado para expressar uma expressão matemática como é escrita por nós humanos

→ S nesse caso é a expressão ou função que desejamos representar

Matemática Simbólica

→ Exemplo

```
>> syms x y z
>> S=x^2 +2*y +1/z

S =

2*y + x^2 + 1/z

>> pretty(S)
        2   1
2 y + x  + -
           z
```

Matemática Simbólica – Eqs. Algébricas

→ `h=solve(eq)` **ou** `h=solve(eq,var)`

Note que quando não declaramos um valor de igualdade o MATLAB atribui zero a esse valor.

solve('eq') => solve('eq' == 0)

```
>> syms x
>>  h = solve(x^2 + 2*x + 1)

h =

 -1
 -1
```

```
>> syms x
>>  h = solve(x^2 + 2*x + 1 == 0)

h =

 -1
 -1
```

Matemática Simbólica – Eqs. Algébricas

Note que devemos atribuir a igualdade com (==)

```
>> syms x
>> h = solve(x^2 + 2*x + 1==2)

h =

 - 2^(1/2) - 1
   2^(1/2) - 1
```

Matemática Simbólica – Eqs. Algébricas

Note que a ‘eq.’ também pode ser expressa como uma variável simbólica

```
>> syms x
>> eq=x^2+2*x+1==2;

>> raizes=solve(eq)

raizes =

 - 2^(1/2) - 1
   2^(1/2) - 1
```

Matemática Simbólica – Sistema de Eqs.

→ (saída)=solve(eq1,eq2,...,eqn) **ou**

→ (saída)=solve(eq1,eq2,...,eqn,var1,var2,...,varn)

Se o número **n** de equações é igual ao número de variáveis nas equações, o MATLAB apresenta uma solução numérica para todas as variáveis

Matemática Simbólica – Sistema de Eqs.

→ (saída)=solve(eq1,eq2,...,eqn) **ou**

→ (saída)=solve(eq1,eq2,...,eqn,var1,var2,...,varn)

Se o número de variáveis for **maior** que a de equações, o MATLAB apresenta uma solução para **n** variáveis em termos do restante das variáveis

Matemática Simbólica – Sistema de Eqs.

→ (saída)=solve(eq1,eq2,...,eqn) **ou**

→ (saída)=solve(eq1,eq2,...,eqn,var1,var2,...,varn)

Quando o número de variáveis supera o número de equações você pode escolher para que variáveis o sistema será resolvido (usa-se a segunda sintaxe)

Matemática Simbólica – Sistema de Eqs.

→ [x1, x2, x3]=solve(eq1,eq2,eq3)

Representação das saídas está esquematizada acima.
Note que x1, x2 e x3 podem ter mais de um valor, sendo assim vetores colunas.

Matemática Simbólica – Sistema de Eqs.

→ Exemplo

A seguir demonstraremos como resolver o seguinte sistema de equações

$$(\mathrm{I})\ 10x + 12y + 6t = 0$$
$$(\mathrm{II})\ 5x - y = 13t$$

Matemática Simbólica – Sistema de Eqs.

→ Exemplo

```
>> syms x y t
>> S=10*x+12*y+16*t;
>> [x1 y1]=solve(S,'5*x-y=13*t')

x1 =

2*t


y1 =

-3*t
```

Matemática Simbólica – Sistema de Eqs.

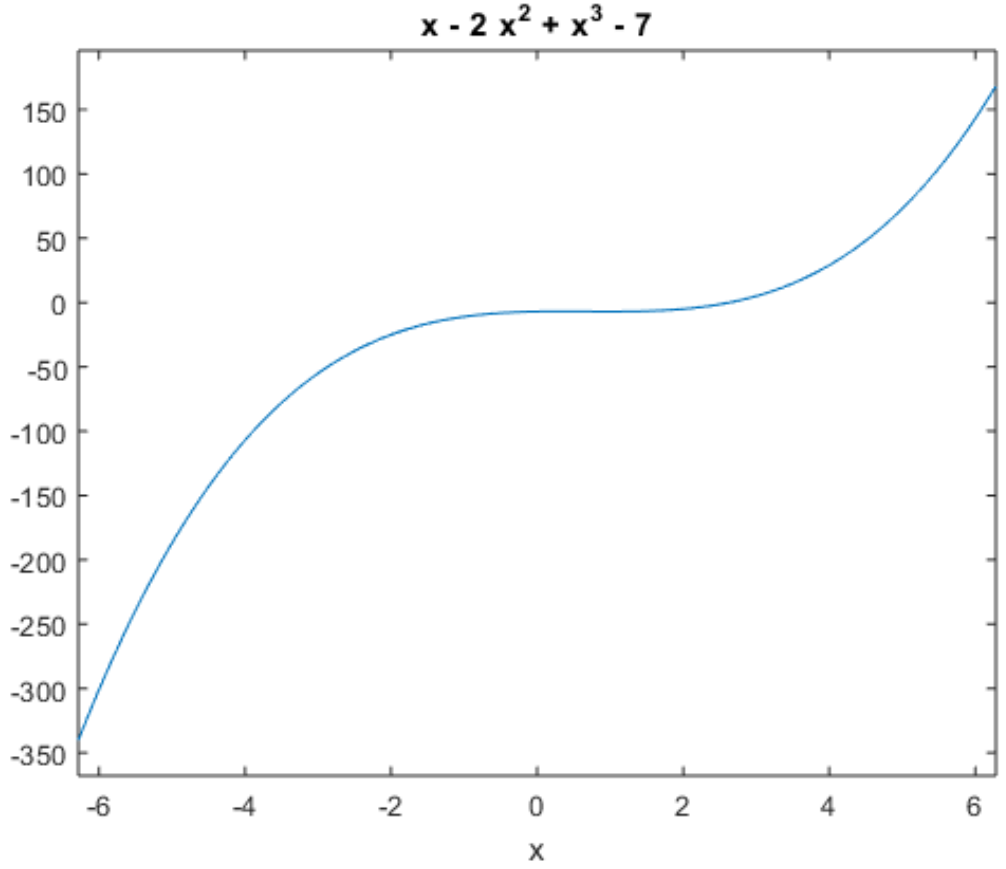
→ Exemplo

```
>> syms x y t
>> S=10*x+12*y+16*t;
>> T=5*x-y-13*t;
>> [x1 y1]=solve(S,T)

x1 =

2*t


y1 =

-3*t
```

Plotando Funções Simbólicas

→ `ezplot(S)`

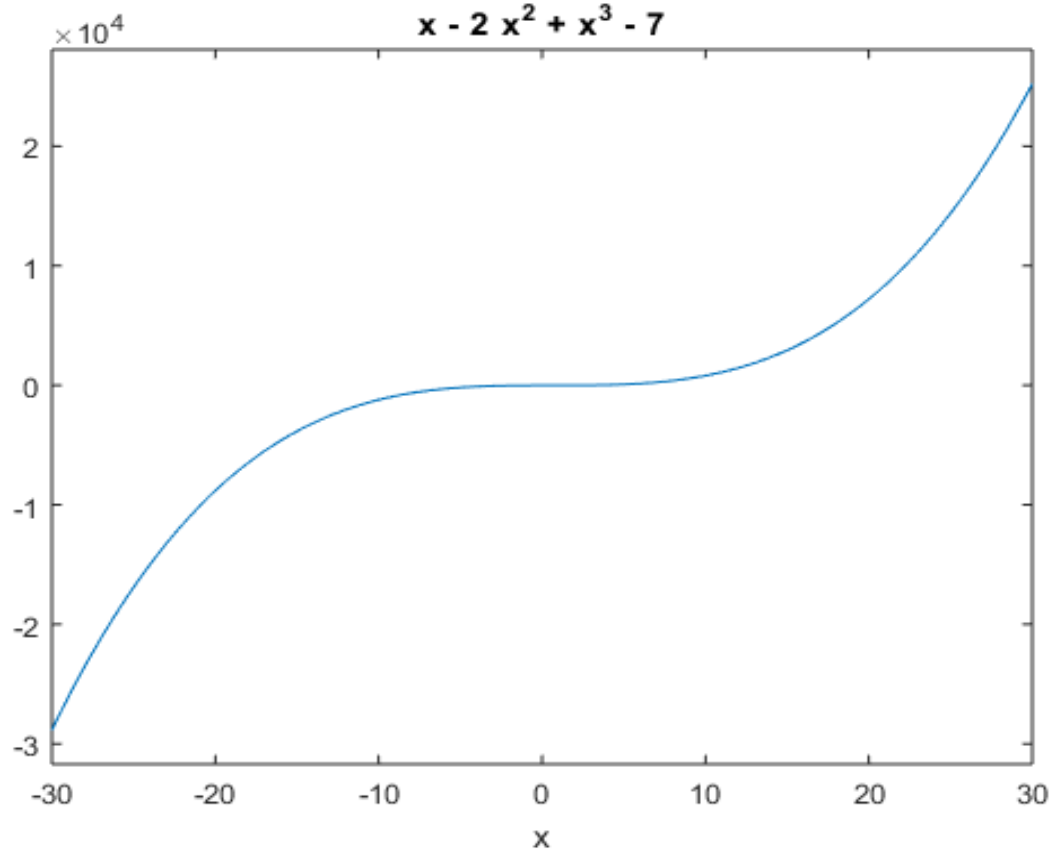
```
>> syms x
>> S=x^3 -2*x^2 +x-7;
>> ezplot(S)
```



Plotando Funções Simbólicas

→ `ezplot(S,[xmin,xmáx])`

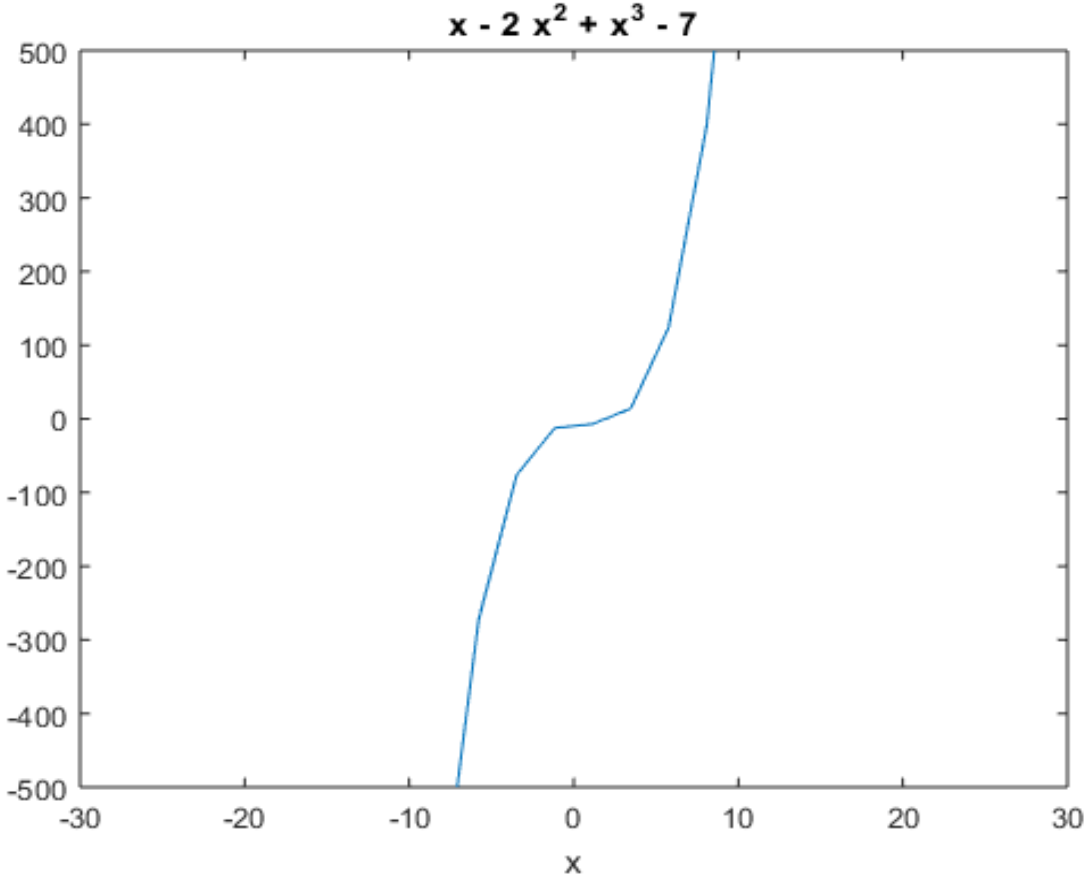
```
>> syms x
>> S=x^3 -2*x^2 +x-7;
>> ezplot(S,[-30,30])
```



Plotando Funções Simbólicas

→ `ezplot(S,[xmin,xmáx,ymin,ymáx])`

```
>> syms x
>> S=x^3 -2*x^2 +x-7;
>> ezplot(S,[-30,30,-500,500])
```



Avaliando Funções Simbólicas

→ res=subs(S,var,número)

```
>> syms x
>> S=x^2;
>> res=subs(S,x,4)

res =

16
```

Avaliando Funções Simbólicas

→ res=subs(S,{var1,var2,...,varn},{n1,n2,...,nn})

```
>> syms x y z t

>> S=x^2+3*t+log(y)+exp(z);
>> res=subs(S,{x,t,y,z},{1,2,5,3})

res =

exp(3) + log(5) + 7
```

Diferenciação – Diferenciação simbólica

→ d=diff(S)

Usado para diferenciar simbolicamente.

```
>> syms x

>> S = 3*x + 4*x^3;

>> d=diff(S)

d =

12*x^2 + 3
```

Diferenciação – Diferenciação simbólica

→ Exemplo

Também é possível fazer a derivação sem declarar vetores

```
>> syms x
>> d=diff(2*x +x^2)

d =

2*x + 2
```

Diferenciação – Diferenciação simbólica

→ d=diff(S,var)

Usado para diferenciar parcialmente em que *var* trata-se da variável que se deseja diferenciar.

```
>> syms x y
>> d=diff(x^2 +2*x +y,y)

d =

1

>> d=diff(x^2 +2*x +y,x)

d =

2*x + 2
```

Diferenciação – Diferenciação simbólica

→ d=diff(S,var)

Usado para diferenciar parcialmente em que *var* trata-se da variável que se deseja diferenciar.

```
>> syms x y
>> d=diff(diff(x^2 +2*x +y,y),x)

d =

0

>> d=diff(diff(x^2 +2*x +y,x),y)

d =

0
```

Integração – Integração simbólica

→ i=int(S)

Usado para integrar simbolicamente

```
>> syms x

>> S=x^2 +1;

>> i=int(S)

i =

(x*(x^2 + 3))/3
```

Integração – Integração simbólica

→ i=int(S, var)

Usado para integrar simbolicamente. Var trata-se da variável em que está integrando

```
>> syms x z
>> resi=int(S,z)

i =

z*(x^2 + 1)
```

Integração – Integração simbólica

→ i=int(S, a, b) **ou** resi=int(S, var, a, b)

```
>> syms x
>> S=x^2;
>> i=int(S,0,2)

i =

8/3
```

Integração – Integração simbólica dupla

→ i=int(int(S, var1, a, b),var2,a1,b1)

```
>> syms x y
>> S=x^2+y;
>> i=int(int(S,x,0,2),y,0,3)

i =

17
```

Integração – Integração simbólica tripla

→ i=int(int(int(S, var1, a, b),var2,a1,b2),var3,a3,b3)

```
>> syms x y z
>> S=x^2+y+z;
>> i=int(int(int(S,x,0,2),y,0,3),z,0,1)

i =

20
```