PETEE UFMG

UFMG

MATLAB

Slide-aula

2020

PROGRAMA DE EDUCAÇÃO TUTORIAL – ENGENHARIA ELÉTRICA – UNIVERSIDADE FEDERAL DE MINAS GERAIS

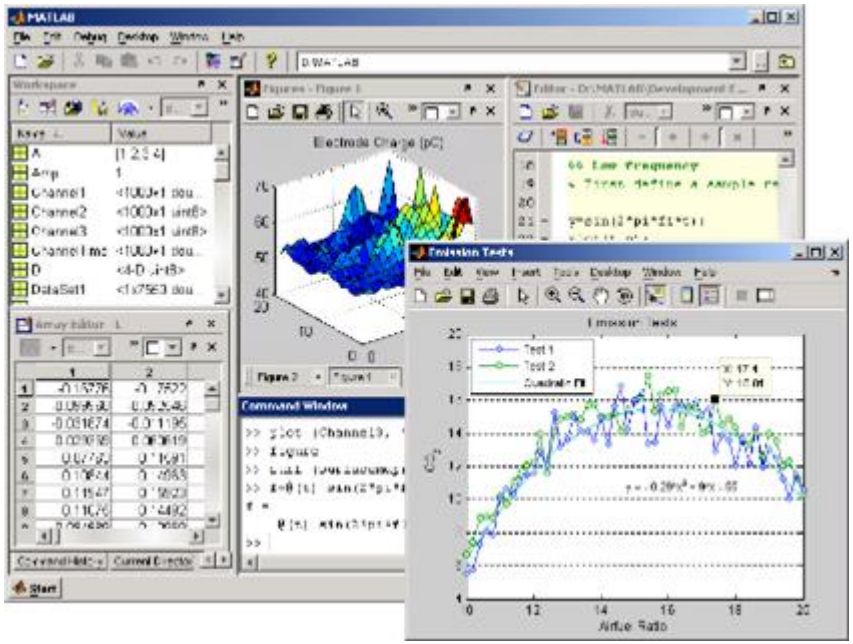
PETEE UFMG

Introdução

Introdução

MATLAB vem de Matrix Laboratory

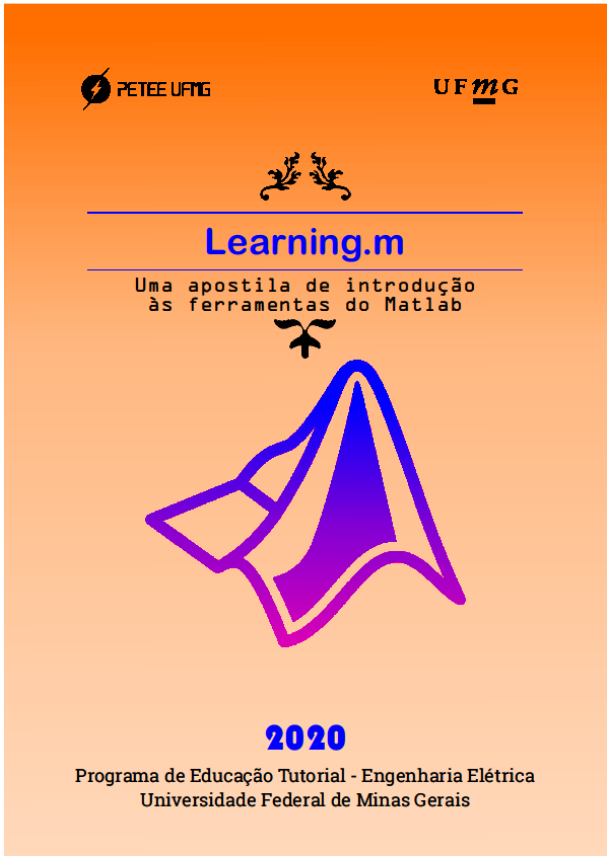
Sistemas de Controle, Processamento de Sinais, Redes Neurais, Wavelets



MINISTÉRIO DA EDUCAÇÃO

UFMG

Acompanhe



Capítulo da Apostila

MINISTÉRIO DA EDUCAÇÃO

UFMG

O que veremos

2 Ambiente de Trabalho

3 Estrutura de Dados

4 Comandos e Funções

5 Operadores

6 Controle de Fluxo

7 Scripts

8 Gráficos

9 Polinômios

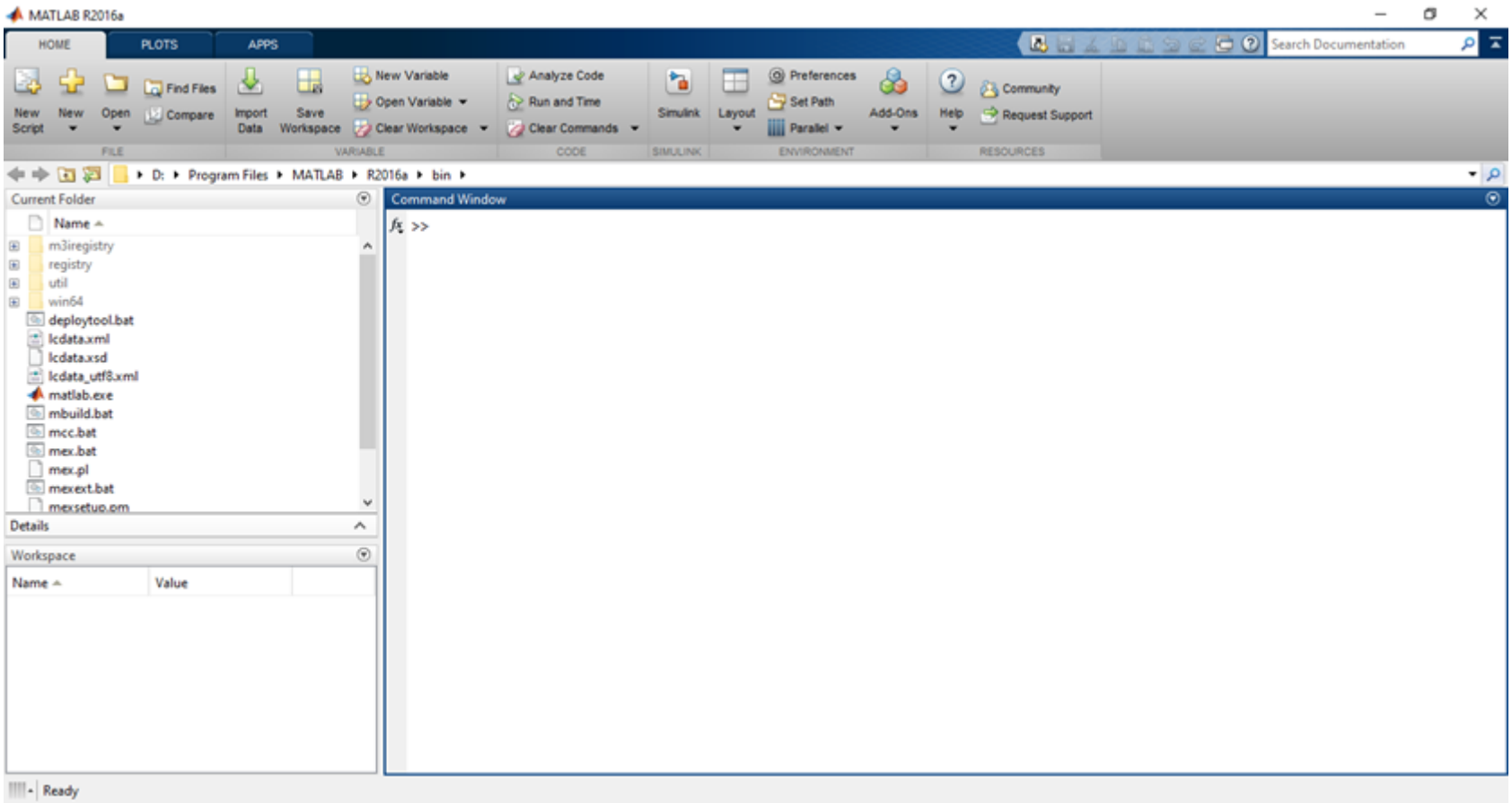
10 Simulink: Introdução

PETEE UFMG

Ambiente de Trabalho

Ambiente de Trabalho

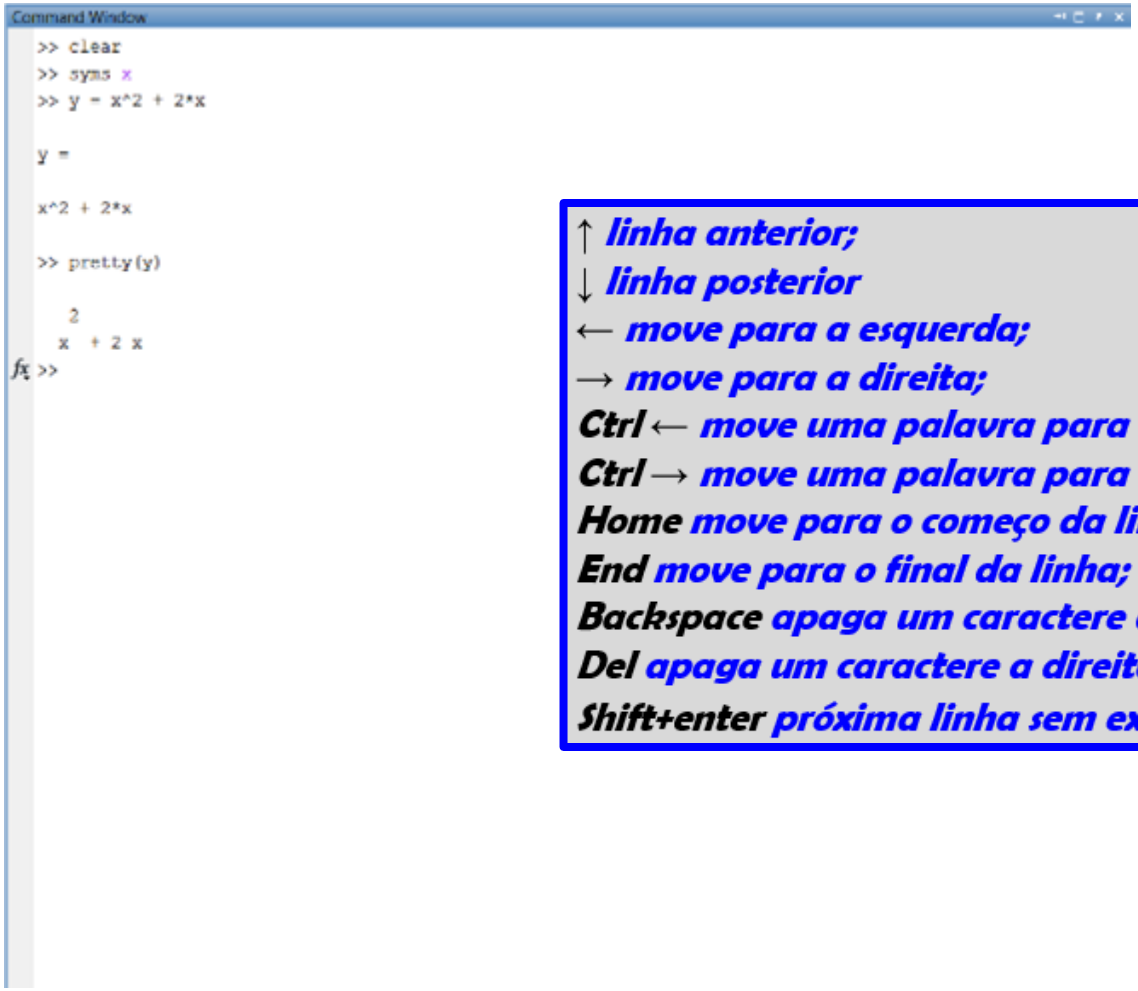
2



MINISTÉRIO DA EDUCAÇÃO

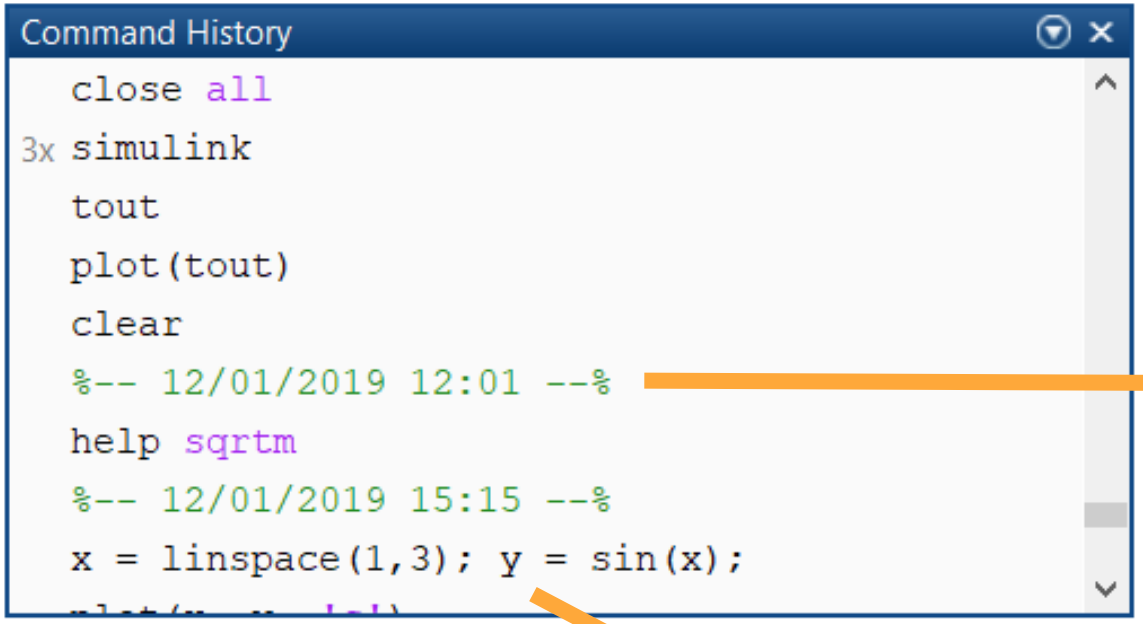
UFMG

Command Window



↑ *linha anterior;*
↓ *linha posterior*
← *move para a esquerda;*
→ *move para a direita;*
Ctrl ← *move uma palavra para a esquerda;*
Ctrl → *move uma palavra para a direita;*
Home *move para o começo da linha;*
End *move para o final da linha;*
Backspace *apaga um caractere a esquerda;*
Del *apaga um caractere a direita;*
Shift+enter *próxima linha sem executar;*

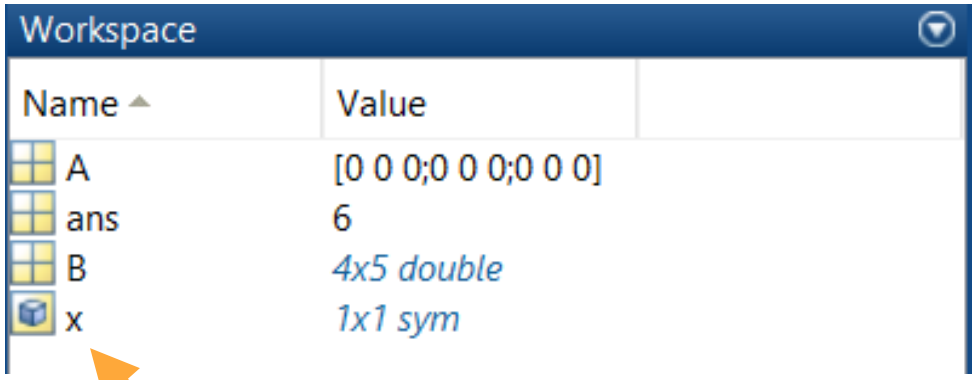
Command History

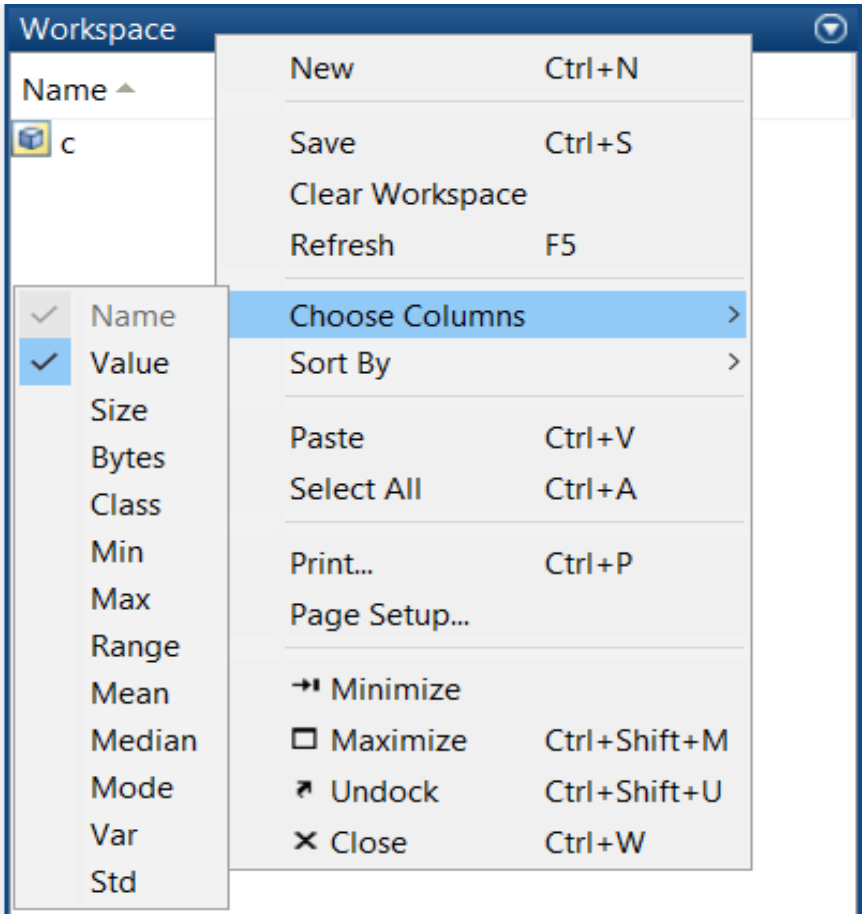


Marcações diárias no início de cada secção

Selecione um ou mais comandos e clique com botão direito para copiar, avaliar ou criar um script

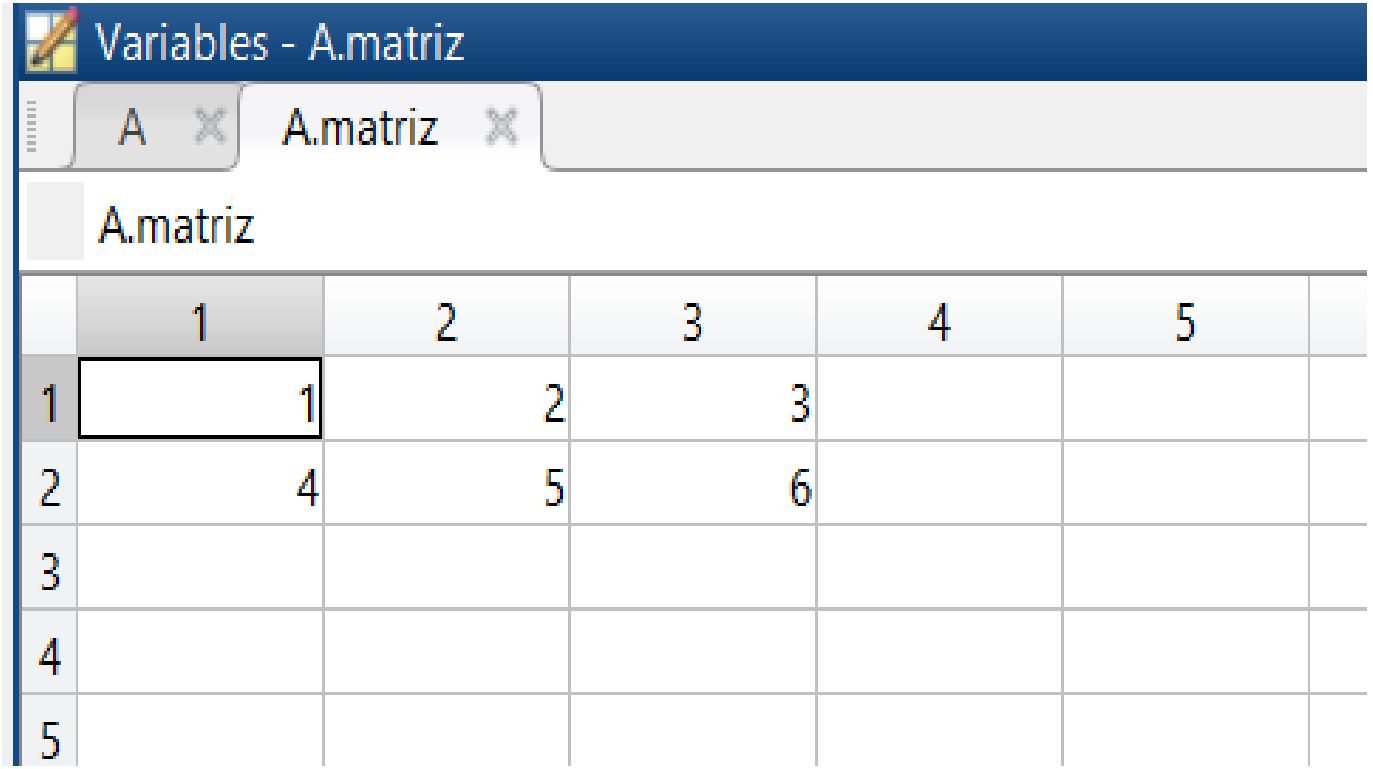
Workspace





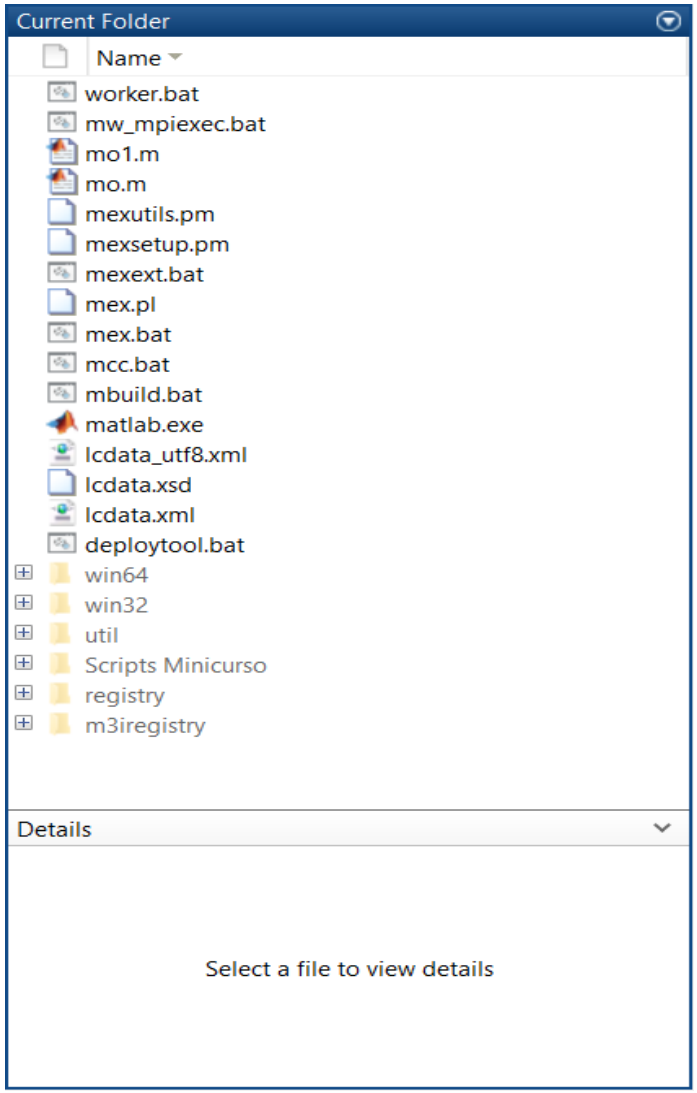
Clique duplo em uma variável para abrir o Variable Editor e editar a variável

Variable Editor



	1	2	3	4	5
1	1	2	3		
2	4	5	6		
3					
4					
5					

Current Folder



cd	troca o diretório de trabalho atual
dir	lista o conteúdo do diretório atual
delete	exclui arquivo
type	mostra o conteúdo do arquivo texto
what	lista arquivos ".m", ".mat" e ".mex".

PETEE UFMG

Estrutura de Dados

Tipos de Dados

Tipos de Dados → Variáveis

Tipos primitivos:

- Ponto Flutuante
- Inteiro
- Lógico
- Caractere

Tipos de Dados

Não é necessário declarar variáveis no MATLAB.

Por padrão as variáveis numéricas são de precisão dupla (double).

Comando class() informa o tipo da variável.

ans, de answer, é a estrutura padrão para armazenar respostas não atribuídas a variáveis.

```
Command Window
>> x = 1

x =

     1

>> y = x+1

y =

     2
```

Tipos de Dados

double	Tipo padrão, com precisão dupla	8
single	Metade da precisão double	4
int8	Inteiro com sinal, de 8 bits	1
int16	Inteiro com sinal, de 16 bits	2
int32	Inteiro com sinal, de 32 bits	4
int64	Inteiro com sinal, de 64 bits	8
uint8	Inteiro não negativo, de 8 bits	1
uint16	Inteiro não negativo, de 16 bits	2
uint32	Inteiro não negativo, de 32 bits	4
uint64	Inteiro não negativo, de 64 bits	8
logical	Tipo de dado booleano	1
sym	Tipo de dado simbólico	8
char	Caractere único	2

Command Window

```
>> x = 3

x =

     3

>> class(x)

ans =

double

>> int8(x)

ans =

     3

>> class(ans)

ans =

int8
```

Tipos de Números

O MatLab é capaz de reconhecer diversos tipos de números:

Type	Examples
Integer	$1362, -217897$
Real	$1.234, -10.76$
Complex	$3.21 - 4.3i \quad (i = \sqrt{-1})$
Inf	Infinity (result of dividing by 0)
NaN	Not a Number, 0/0

A notação **e** pode ser utilizada para representar expoentes na base 10:

`-1.3412e+03` $= -1.3412 \times 10^{3} = -1341.2$

`-1.3412e-01` $= -1.3412 \times 10^{-1} = -0.13412$

Atribuição

Para associar valores a variáveis simplesmente utilizamos o operador = no prompt de comando:

```
>> x=3

x =

     3
```

Podemos permitir ou não a visualização do comando anterior com o operador:

```
>> x=3;
>>
```


Nomes de Variáveis

O MATLAB Permite qualquer combinação entre letras e dígitos. O primeiro caractere deve ser uma letra.

Permitido: NetCost, Left2Pay, x3, X3, z25c5

Não Permitido: Net-Cost, 2pay, %x, @sign,

Case sensitive

Constantes

Constante	Valor	Tamanho	Bytes	Classe	Descrição
i	0+1i	1x1	16	double(complex)	complexo raiz(-1)
j	0+1i	1x1	16	double(complex)	complexo raiz(-1)
eps	2,22e-16	1x1	8	double	Menor valor ao somar a 1 para que se torne o próximo ponto flutuante maior que 1
realmin	2,23e-308	1x1	8	double	Menor valor de ponto flutuante
realmax	1.8e+308	1x1	8	double	Maior valor de ponto flutuante
true	1	1x1	1	logical	Booleano verdadeiro
false	0	1x1	1	logical	Booleano falso
NaN	NaN	1x1	8	double	Valor não numérico
pi	3.1416	1x1	8	double	Valor numérico de pi
inf	inf	1x1	8	double	infinito

Exercício

Calcule as seguintes expressões:

$$x = 3 \times 2^2 - (3 + 4)^3$$

$$y = x - \frac{2}{3 \times x} + x^{\frac{1}{2}} + 11$$

$$z = y * x - \frac{y}{x + 3 * y} + x^y$$

Obs:

$\frac{a}{b}$ → a/b

ab → a*b

a^b → a^b

Exercício


```
>> x = 3*2^2 - (3+4)^3

x =

   -331

>> y = x - 2/(3*x) + x^(1/2) + 11

y =

   -3.2000e+02 + 1.8193e+01i

>> z = y*x - y/(x + 3*y) + x^y

z =

    1.0592e+05 - 6.0220e+03i
```


PETEE UFMG

Estrutura de Dados

Vetores e Matrizes

Vetores


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

v =

     2     3     5

>> v = [3,7,1]

v =

     3     7     1
```

```
>> length(v)

ans =

     3
```


Vetores

Podemos usar linspace e : para criação de vetores;

```
X = linspace(1, 10);
Y = 10:0.5:14;
```

Vetores

Para vetores do mesmo tamanho podemos realizar certas operações aritméticas:

```
>> v1 = [2 4 1];
>> v2 = [1 1 9];
>> v3 = [ 1 2 5 1];
>> v1 + v2

ans =

     3     5    10

>> v1 + v3
??? Error using ==> plus
Matrix dimensions must agree.

>> v4 = 3*v1

v4 =

     6    12     3
```

```
>> v2*v1
Error using  *
Inner matrix dimensions must agree.

>> v2/v1

ans =

    0.7143
```

Vetores

Podemos ainda criar vetores a partir de vetores pré-existentes:

```
>> w = [1 2 3]; z = [8 9];
>> wz = [2*w, -z]

wz =

     2     4     6    -8    -9
```

Podemos manipular elementos em particular definindo sua posição no vetor utilizando o() :

```
>> w(2) = -10

w =

     1   -10     3
```

```
>> w(3)

ans =

     3
```

Vetores

A construção de vetores coluna é similar a de vetores linhas

Separação dos elementos realizada por ; ou **“novas linhas”**:

Operações aritméticas também podem ser realizadas respeitando os precedentes matemáticos:

```
>> c = [ 1; 3; sqrt(5)]
c =
    1.0000
    3.0000
    2.2361
```

```
>> c3 = 2*c - 3*c2
c3 =
   -7.0000
   -6.0000
  -10.5279
```

```
>> c2 = [3
4
5]
c2 =
     3
     4
     5
```

Vetores

Podemos converter um vetor linha em um vetor coluna através do processo de transposição.

No Matlab este processo é denotado pelo operador ‘

```
>> w,w',c,c'

w =

     1   -10     3


ans =

     1
   -10
     3


c =

    1.0000
    3.0000
    2.2361


ans =

    1.0000    3.0000    2.2361
```

Vetores

O operador de transposição pode ser utilizado em conjunto com sentenças matemáticas de forma livre:

```
>> t = w + 2*c'

t =

    3.0000   -4.0000    7.4721

>> T = v1*v2'

T =

    32
```

Matrizes


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

A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
>> A(8) %valor armazenado em A(4,2)

ans =

    15
```

```
>> A(1,2)

ans =

     3

>> A(3,4)

ans =

    12
```

Matrizes

Se buscarmos uma posição fora das dimensões da matriz especificada teremos o seguinte erro:

```
>> A(4,5)
??? Index exceeds matrix dimensions.
```

No entanto se atribuirmos um valor a uma posição anteriormente inexistente a característica dinâmica das matrizes no Matlab modificará a matriz para acomodar a nova entrada:

```
>> X(4,5)=17

X =

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


Matrizes

Cinco matrizes básicas para uso no MatLab:

- **Zeros:** Matrizes formadas apenas de zeros;
- **Ones:** Matrizes formadas apenas por 1´s;
- **Eye:** Matriz identidade;
- **Rand:** matriz composta de forma randômica uniformemente distribuída;
- **Randn:** matriz composta de forma randômica com distribuição normal.

Matrizes

Cinco matrizes básicas para uso no Matlab:

```
>> Z = zeros(2,4)

Z =

     0     0     0     0
     0     0     0     0

>> u = ones(2,4)

u =

     1     1     1     1
     1     1     1     1
```

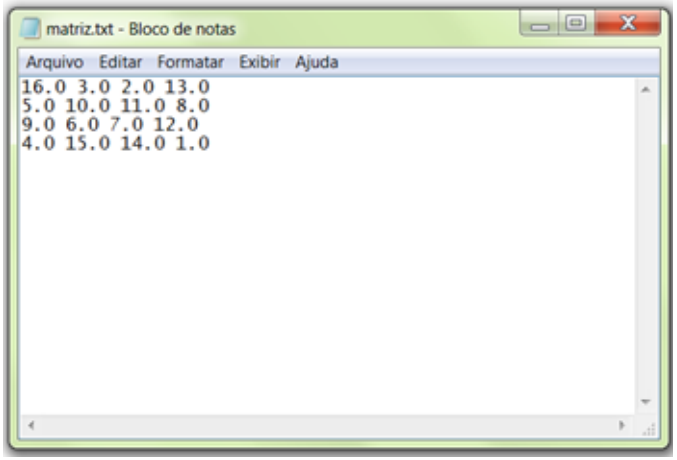
```
>> R = rand(3)

R =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> E = eye(4,4)

E =

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

```
>> Rn = randn(3)

Rn =

    2.7694    0.7254   -0.2050
   -1.3499   -0.0631   -0.1241
    3.0349    0.7147    1.4897
```

Matrizes

Outra forma de carregar uma matriz é através de um arquivo externo contendo dados no formato numérico:

```
16     3     2    13
 5    10    11     8
 9     6     7    12
 4    15    14     1
```



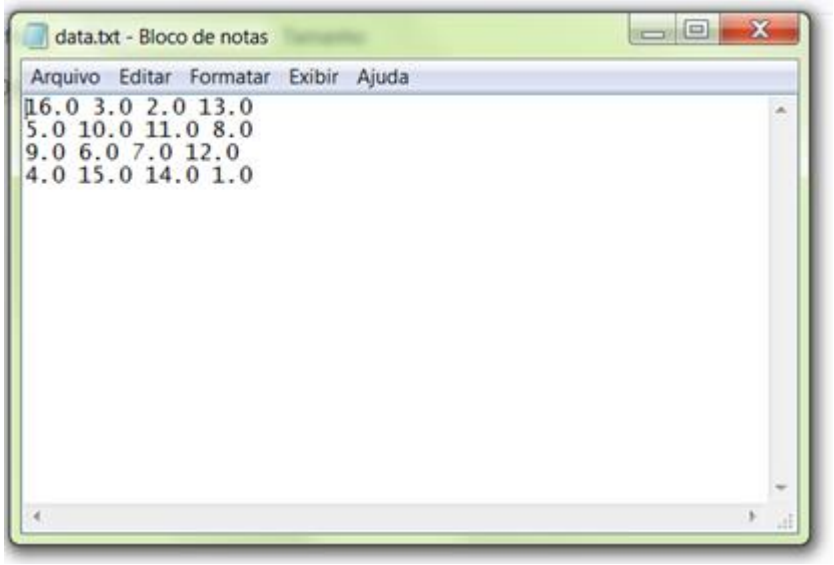
Salve o arquivo no atual diretório corrente do MATLAB.

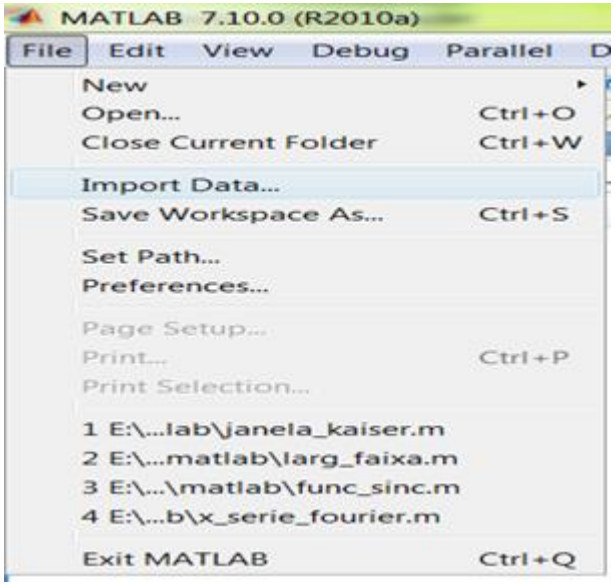
Utilizando o Comando **load** podemos carregar esta matriz montada através de uma outra fonte ou em uma sessão anterior do MATLAB:

```
>> load matriz.dat
```

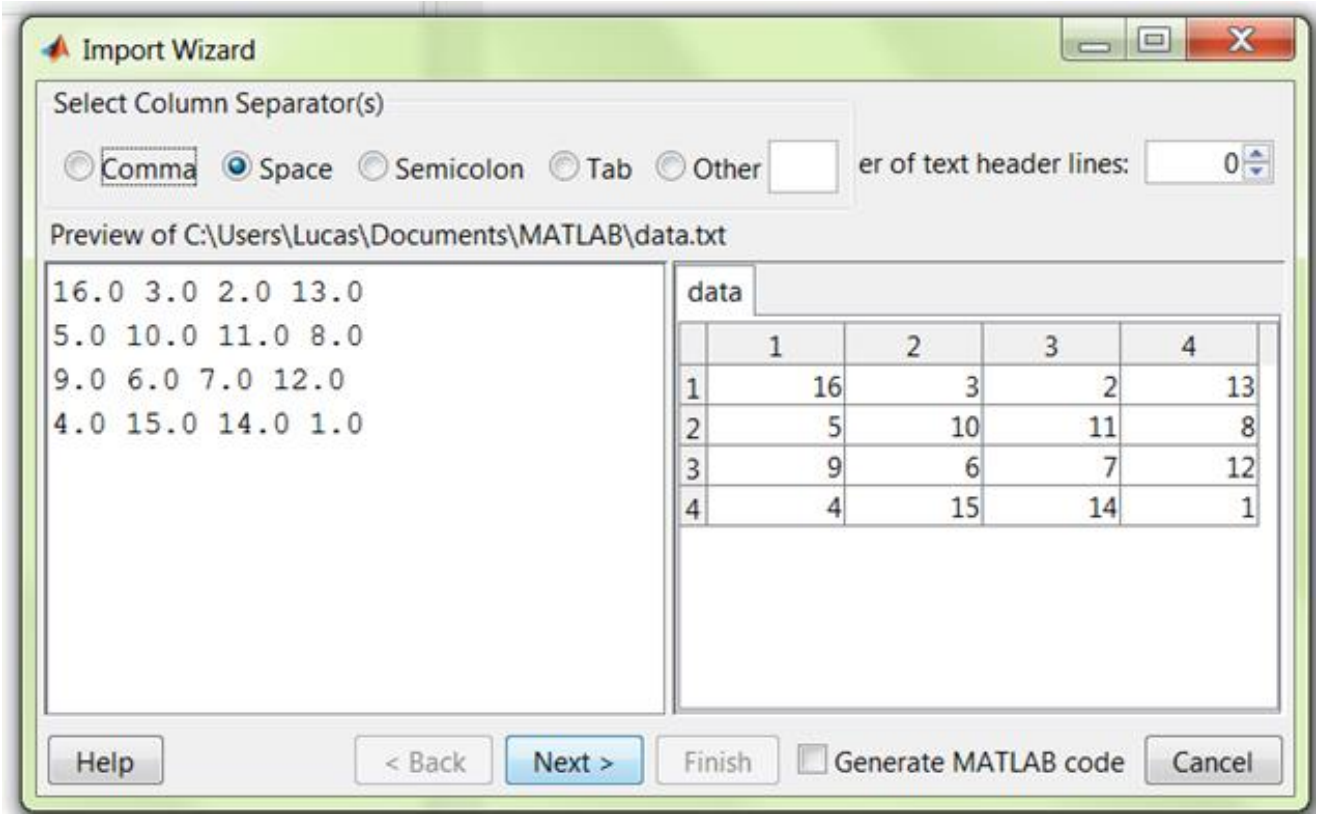
Matrizes

Outra forma de usar os dados de um arquivo qualquer é usando o menu **Import Data...**





Matrizes


Import Wizard
Select Column Separator(s)
Comma
Space
Semicolon
Tab
Other
er of text header lines:
0
Preview of C:\Users\Lucas\Documents\MATLAB\data.txt
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0 6.0 7.0 12.0
4.0 15.0 14.0 1.0
data
1 2 3 4
1 16 3 2 13
2 5 10 11 8
3 9 6 7 12
4 4 15 14 1
Help
< Back
Next >
Finish
Generate MATLAB code
Cancel

Matrizes

Concatenação

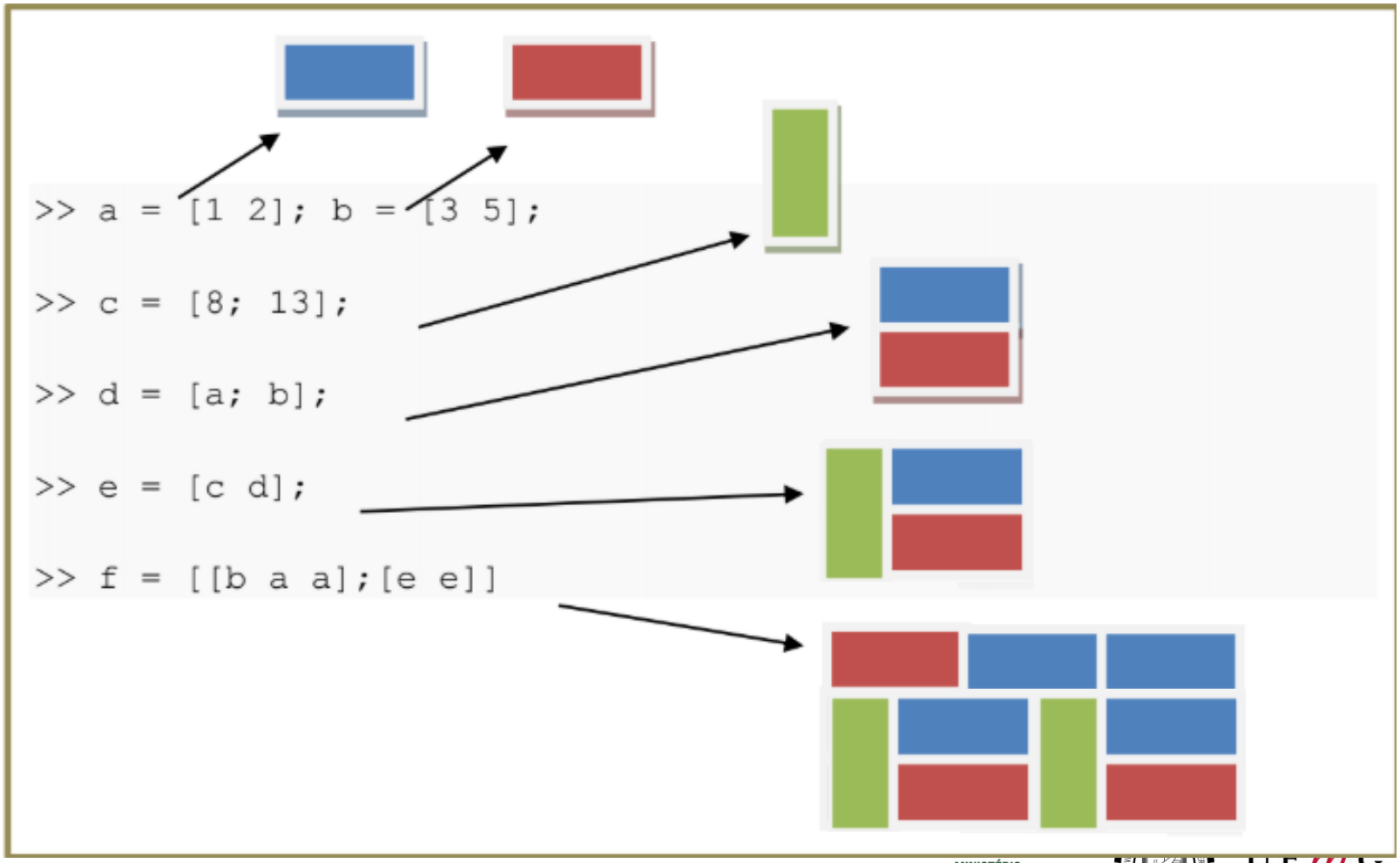
Trata-se do processo de formar matrizes maiores a partir de matrizes menores já existentes.

Para isto utilizamos o operador matricial [] juntamente com as matrizes já declaradas:

```
B = [A  A+32; A+48  A+16]
B =

    16     3     2    13    48    35    34    45
     5    10    11     8    37    42    43    40
     9     6     7    12    41    38    39    44
     4    15    14     1    36    47    46    33
    64    51    50    61    32    19    18    29
    53    58    59    56    21    26    27    24
    57    54    55    60    25    22    23    28
    52    63    62    49    20    31    30    17
```


Matrizes

Concatenação



Matrizes

Deletando linhas e colunas

Utilizamos ainda o operador matricial [] da seguinte forma quando desejamos eliminar linhas ou colunas de matrizes:

`W(:,2)=[]` deleta a segunda coluna da matriz W

```
w =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1

>> w(:,2) = []

w =

    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

Matrizes

Se tentarmos deletar um único elemento de uma matriz o resultado não mais seria uma matriz e desta forma o matlab responderia da seguinte forma:

```
x(2,2) = []
Subscripted assignment dimension mismatch.
```

No entanto se utilizarmos a especificação única para elementos de uma matriz podemos deletar apena um único elemento e o conjunto de dados resultantes toma a forma de um vetor:

```
A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
>> A(2) = []

A =

    16     9     4     2    11     7    14     3    10     6    15    13     8    12     1
```

Matrizes

O comando sum(x) soma as colunas da matriz:

```
x =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
>> sum(x)

ans =

    34    34    34    34
```

Matrizes

O comando diag(A) oferece a diagonal principal de A;

A matriz não precisa ser quadrada.

```
>> x

x =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1

>> diag(x)

ans =

    16
    10
     7
     1
```

Matrizes

Determinante de uma matriz: det(A)

Inversa de uma Matriz: inv(A)

Autovalores (raízes do polinômio característico): eig(A)

Posto de A (numero de linhas ou colunas LI): rank(A)

Polinômio Característico de A : Poly(A)

$$\det(A - \lambda I)$$

Matrizes


```
>> inv(x)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 9.796086e-018.

ans =

  1.0e+015 *

    0.1251    0.3753   -0.3753   -0.1251
   -0.3753   -1.1259    1.1259    0.3753
    0.3753    1.1259   -1.1259   -0.3753
   -0.1251   -0.3753    0.3753    0.1251

>> eig(x)

ans =

   34.0000
    8.0000
    0.0000
   -8.0000

>> rank(x)

ans =

     3

>> det(x)

ans =

  1.0871e-012
```

```
>> rank(x)

ans =

     3

>> poly(x)

ans =

  1.0e+003 *

    0.0010   -0.0340   -0.0640    2.1760   -0.0000
```

Matrizes

Exemplos: Vamos determinar os autovalores e autovetores da matriz:

$$A = \begin{bmatrix} 4 & 2 & 2 \\ 2 & 4 & 2 \\ 2 & 2 & 4 \end{bmatrix}$$

```
>> A = [4 2 2; 2 4 2; 2 2 4]

A =

     4     2     2
     2     4     2
     2     2     4

>> rank(A)

ans =

     3
>> poly(A)

ans =

    1.0000   -12.0000    36.0000   -32.0000
```

```
>> [Autovetores,Autovalores] = eig(A)

Autovetores =

    0.4082    0.7071    0.5774
    0.4082   -0.7071    0.5774
   -0.8165         0    0.5774


Autovalores =

    2.0000         0         0
         0    2.0000         0
         0         0    8.0000
```

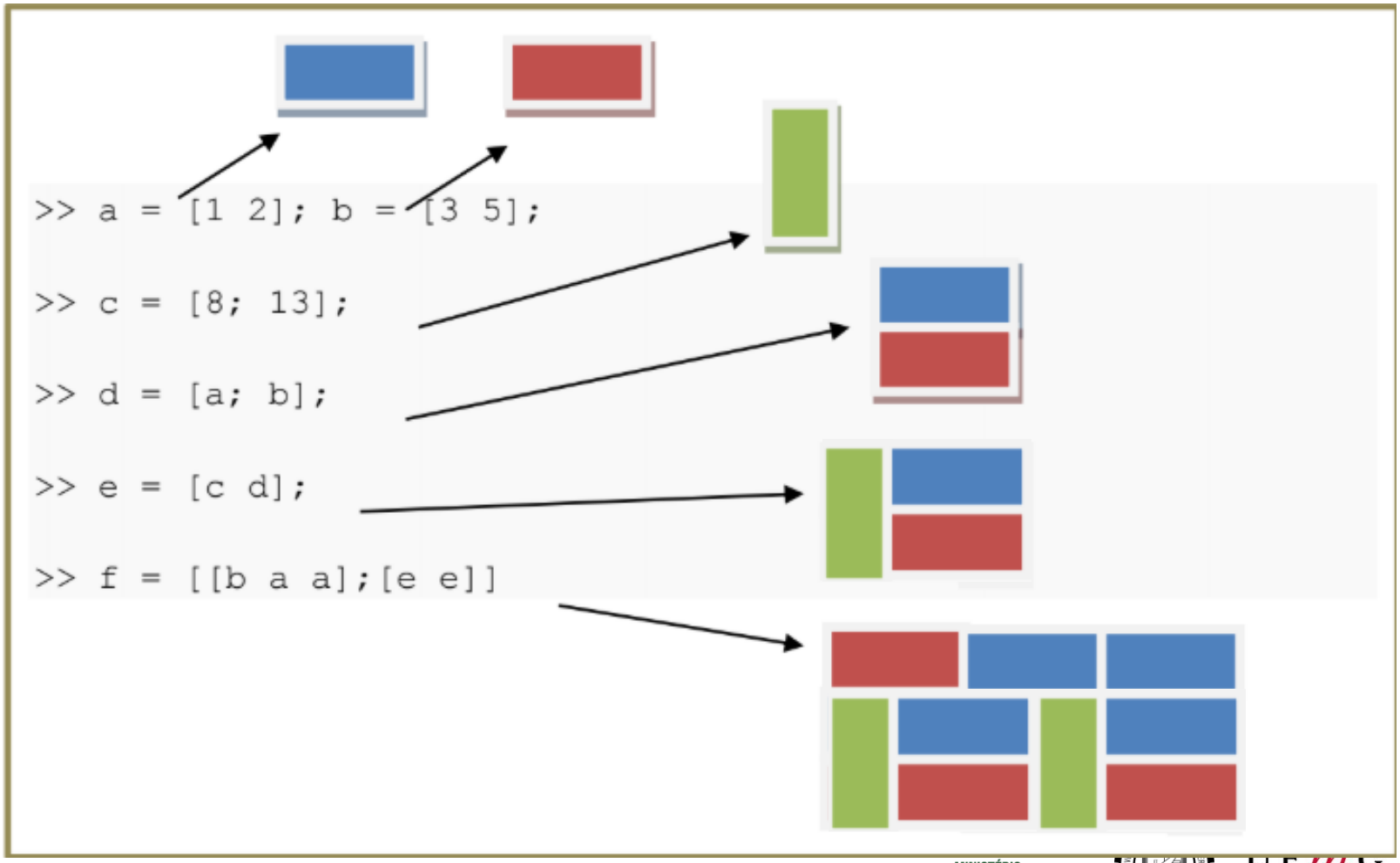
Matrizes

Comando **find**: Retorna uma lista de posições (índices) de elementos de um vetor ou uma matriz que satisfazem determinada condição:

```
>> A = [ -2 3 4 4; 0 5 -1 6; 6 8 0 1]
A =
    -2     3     4     4
     0     5    -1     6
     6     8     0     1
>> k = find(A==0)
k =
     2
     9
```

Exercício

Realize a concatenação do exemplo a



PETEE UFMG

Estrutura de Dados

Expressões Simbólicas

Expressões Simbólicas


```
>> syms x y
>> z = x*y + x^2;
```

```
>> w = x+z

w =

x + x*y + x^2

>> p = w*y

p =

y*(x + x*y + x^2)
```

Expressões Simbólicas

Funções úteis: “expand”, “solve”, “factor”, “subs”

Command Window

```
>> w = (x+3)^3

w =

(x + 3)^3

>> expand(w)

ans =

x^3 + 9*x^2 + 27*x + 27
```

Command Window

```
>> z = expand(w)

z =

x^3 + 9*x^2 + 27*x + 27

>> solve(z)

ans =

 -3
 -3
 -3
```

```
>> factor(z)

ans =

(x + 3)^3
```

```
>> subs(z,2)

ans =

   125

>> w = x*y + x^2 +y

w =

x^2 + y*x + y

>> subs(w,5)

ans =

6*y + 25
```

Expressões Simbólicas

Diferenciação

Command Window

```
>> clear
>> syms x
>> y = x^4

y =

x^4

>> diff(y)

ans =

4*x^3

>> diff(y,x)

ans =

4*x^3
```

Command Window

```
>> syms x y
>> z = x*y +x^2 + y

z =

x^2 + y*x + y

>> diff(z,x)

ans =

2*x + y

>> diff(z,y)

ans =

x + 1
```

```
>> y = x^n

y =

x^n

>> diff(y,x)

ans =

n*x^(n - 1)

>> diff(ans,x)

ans =

n*x^(n - 2)*(n - 1)

>> diff(ans,x)

ans =

n*x^(n - 3)*(n - 1)*(n - 2)
```

Expressões Simbólicas

Diferenciação

Command Window

```
>> y = log(x^3*sin(x)^2)

y =

log(x^3*sin(x)^2)

>> diff(y,x)

ans =

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

Expressões Simbólicas

Integração

```
Command Window
>> y = x^3

y =

x^3

>> int(y,x)

ans =

x^4/4
```

```
Command Window
>> y = x^3

y =

x^3

>> int(y,x,0,3)

ans =

81/4
```

Expressões Simbólicas


```
>> syms x
>> y = x^2

y =

x^2

>> subs(y,2)

ans =

4

>> subs(y,[3 4 5])

ans =

[ 9, 16, 25]
```

```
>> y = (x-3)^3

y =

(x - 3)^3

>> expand(y)

ans =

x^3 - 9*x^2 + 27*x - 27

>> simplify(ans)

ans =

(x - 3)^3
```

```
>> y = 2*x*(x+2) - 4*(x-4)

y =

2*x*(x + 2) - 4*x + 16

>> expand(y)

ans =

2*x^2 + 16

>> factor(ans)

ans =

2*(x^2 + 8)
```

Expressões Simbólicas

Solve

$y = 3x^3 - 5x$

OBS.: Também resolve expressões:

```
>> y = 3*x^3-5*x

y =

3*x^3 - 5*x

>> solve(y)

ans =

             0
  15^(1/2)/3
 -15^(1/2)/3
```

Expressões Simbólicas

Solve

Exemplo: Num deserto, existem apenas camelos e dromedários, num total de 64 animais. O número de camelos e igual ao triplo do número de dromedários. Quantos animais de cada espécie existem nesse deserto?

x+3∗x=64, x = número de dromedários.

```
>> solve(x+3*x == 64)

ans =

16
```

Note que usamos ==.

Expressões Simbólicas

Limites

```
>> limit(sin(x)/x, x, 0)

ans =

1

>> limit((1-1/x)^x, x, inf)

ans =

exp(-1)
```

```
>>limit(f(x),x,a,'left')

ou

>>limit(f(x),x,a,'right')
```


Expressões Simbólicas

Soma de Séries

```
>> syms n
>> symsum((4*n+1)/(n+3),n,1,100)

ans =

  362.7820
```

```
>> symsum(1/n^2,n,1,inf)

ans =

pi^2/6
```

PETEE UFMG

Estrutura de Dados

Números Complexos

Números Complexos

$\sqrt{(-1)} = i = j$

Command Window

```
>> sqrt(-1)

ans =

        0 + 1.0000i

>> i

ans =

        0 + 1.0000i

>> j

ans =

        0 + 1.0000i
```

```
>> z1 = 3 + i*2

z1 =

   3.0000 + 2.0000i

>> z2 = 2 - i*sqrt(5)

z2 =

   2.0000 - 2.2361i

>> z3 = z1 +z2

z3 =

   5.0000 - 0.2361i
```


Números Complexos

$M\angle\theta \equiv M*e^{(j*\theta)} = a+j*b$

Onde:

$M=\sqrt{(a^2+b^2)}$

$\theta = \tan^{(-1)}(b/a)$

$a=M*\cos(\theta)$

$b=M*\sen(\theta)$

Números Complexos

M∠θ≡M*e^(j*θ) =a+j*b

Command Window

```
>> z = 1- 4j

z =

   1.0000 - 4.0000i

>> a = real(z)

a =

     1

>> b = imag(z)

b =

    -4
```

```
>> M = abs(z)

M =

    4.1231

>> theta = angle(z)*180/pi

theta =

  -75.9638
```

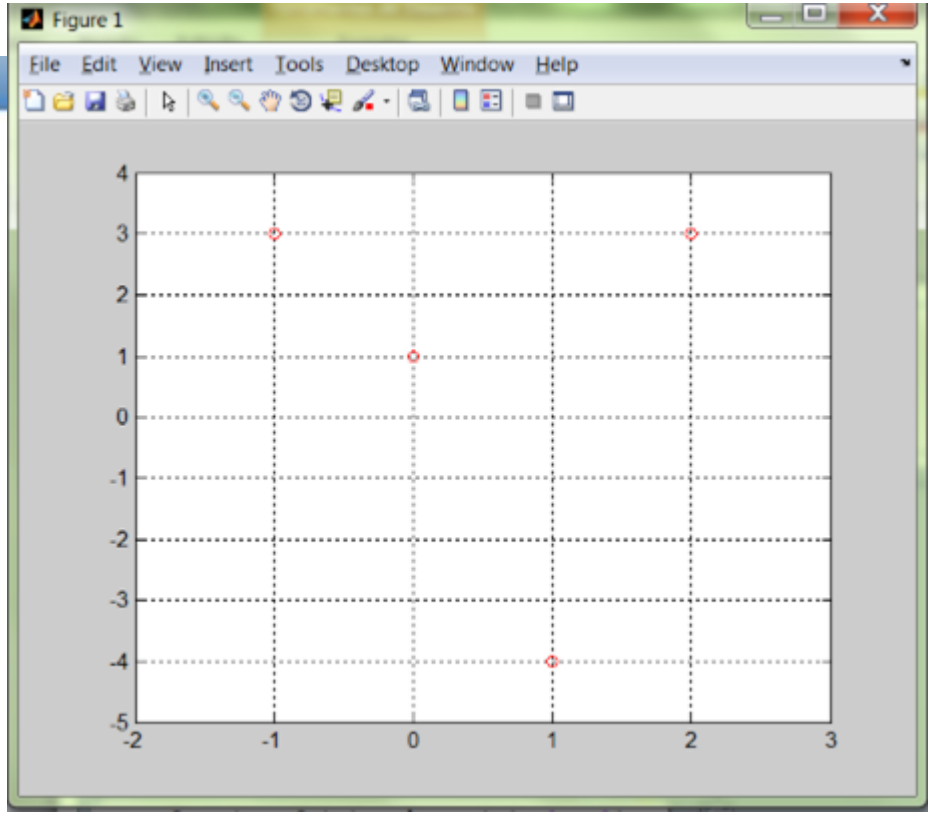
```
>> x = M*exp(j*theta*pi/180)

x =

   1.0000 - 4.0000i
```

Números Complexos

Command Window

```
>> z1 = 1-4j; z2 = 2+3j;
>> z3 = 1j; z4 = -1+3j;
>> z = [z1 z2 z3 z4];
```

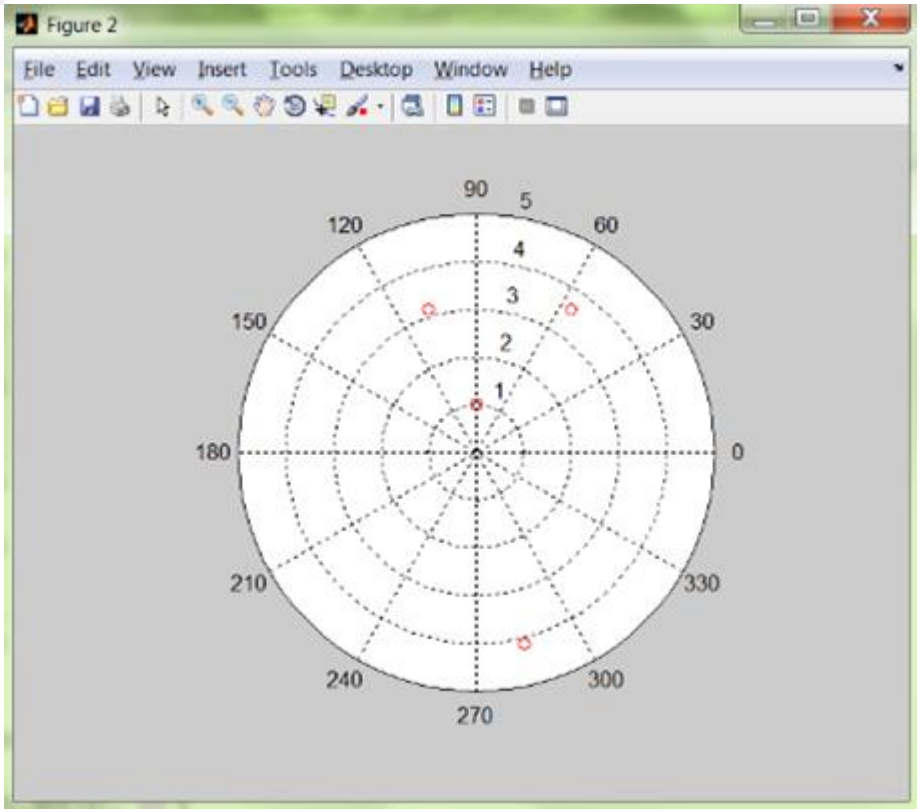
```
>> plot(real(z), imag(z),'ro')
>> axis([-2 3 -5 4])
>> grid
```



OBS: Melhores noções de plotagem serão dadas na parte de gráficos!

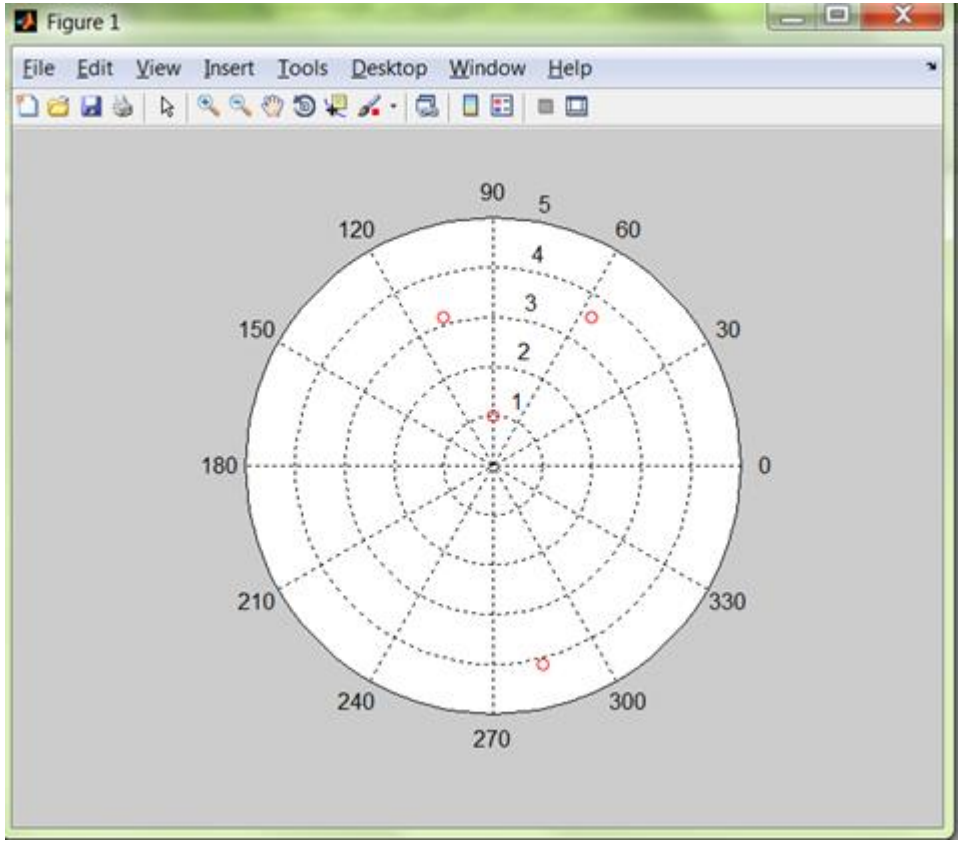
Números Complexos


```
>> figure(2)
>> polar(angle(z),abs(z),'ro')
```



Números Complexos


```
>> [theta M] = cart2pol(real(z),imag(z));
>> polar(theta,M,'ro')
```



Números Complexos

Em Números Complexos o operador ‘ tem outra função: complexo conjugado.

Portanto, se usarmos ‘ em uma matriz de complexos, ela será transposta e cada complexo conjugado, simultaneamente.

Exercício

3

Crie uma matriz Z não quadrada, formada por alguns números complexos de sua escolha. Faça, utilizando Z:

a) Matriz A = transposta e conjugada de Z.

b) Matriz B = conjugada de Z.

c) Matriz C = transposta de Z.

Exercício


```
>> A = Z‘;

>> B = [ Z(1)', Z(2)', Z(3)'];

>> C = [Z(1); Z(2); Z(3)];
```

PETEE UFMG

Estrutura de Dados

Caracteres

Caracteres

Strings são tratadas normalmente como arranjos no MATLAB.

```
Nome = 'Julia'
Nome(5) = 'a' + 14
```

Caracteres

Strings podem ser matrizes:

```
Texto = [‘SATOR’
‘AREPO’
‘TENET’
‘OPERA’
ROTAS’];
```

PETEE UFMG

Comandos e Funções

Introdução

O que são?

Palavras chaves que realizam ações pré-determinadas. Ex: clear.

Conjunto A Conjunto B

Variáveis Retornadas = Função (Parâmetros)

Comando help

Command Window

```
>> help laplace
 --- help for sym/laplace ---

 LAPLACE Laplace transform.
    L = LAPLACE(F) is the Laplace transform of the scalar sym F with
    default independent variable t.  The default return is a function
    of s.  If F = F(s), then LAPLACE returns a function of t:  L = L(t).
    By definition L(s) = int(F(t)*exp(-s*t),0,inf), where integration
    occurs with respect to t.

    L = LAPLACE(F,t) makes L a function of t instead of the default s:
    LAPLACE(F,t) <=> L(t) = int(F(x)*exp(-t*x),0,inf).

    L = LAPLACE(F,w,z) makes L a function of z instead of the
    default s (integration with respect to w).
    LAPLACE(F,w,z) <=> L(z) = int(F(w)*exp(-z*w),0,inf).

    Examples:
       syms a s t w x
       laplace(t^5)            returns   120/s^6
       laplace(exp(a*s))       returns   1/(t-a)
       laplace(sin(w*x),t)     returns   w/(t^2+w^2)
       laplace(cos(x*w),w,t)   returns   t/(t^2+x^2)
       laplace(x^sym(3/2),t)   returns   3/4*pi^(1/2)/t^(5/2)
       laplace(diff(sym('F(t)')))   returns   laplace(F(t),t,s)*s-F(0)

    See also sym/ilaplace, sym/fourier, sym/ztrans.

    Reference page in Help browser
       doc sym/laplace

fx >>
```

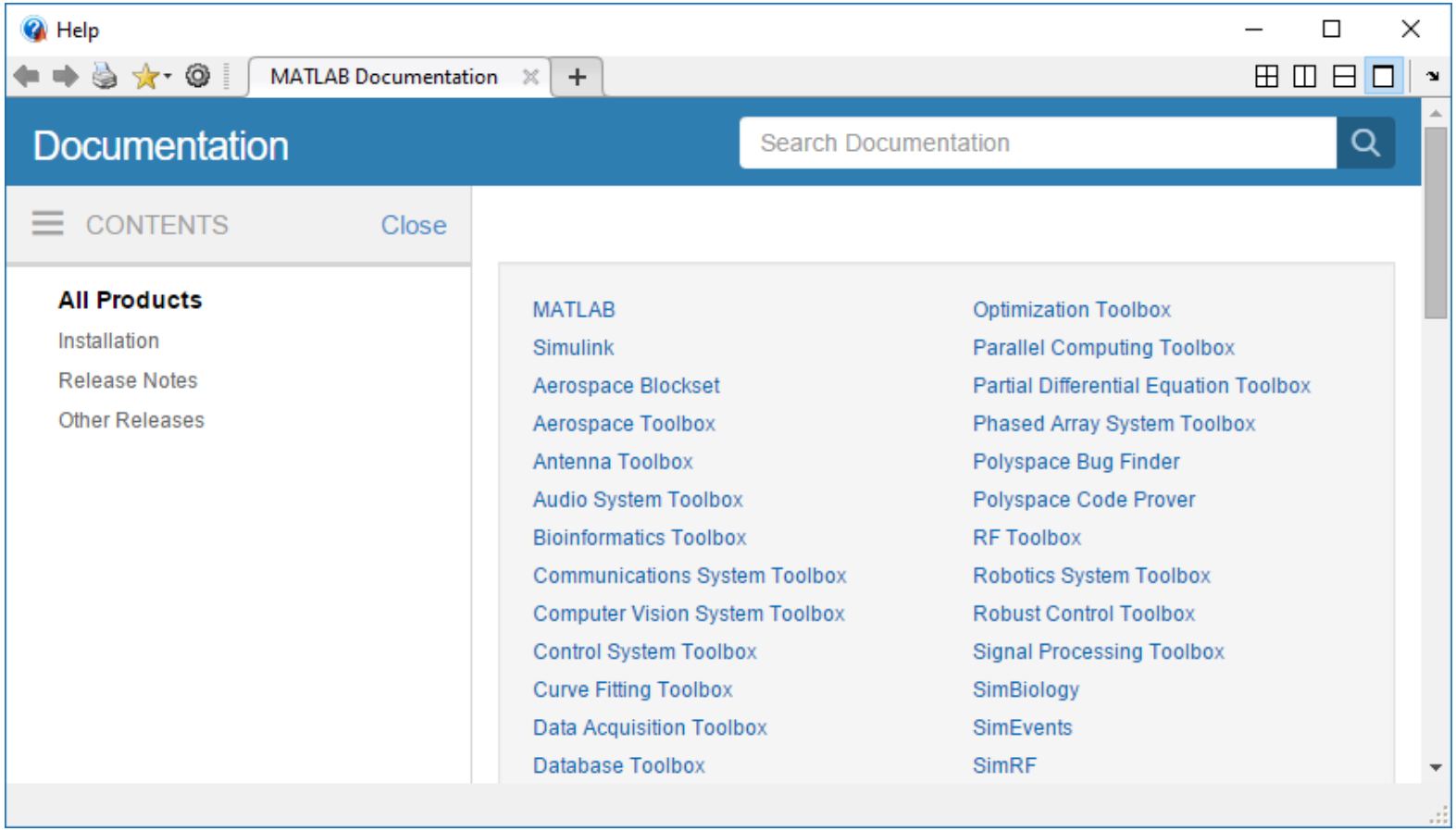
Comando lookfor é similar, e também ajuda.

MINISTÉRIO DA EDUCAÇÃO

UFMG

Comando doc

Reference page for help



Funções elementares

Trigonométricas: **sin(x), cos(x), tan(x)**;

Hiperbólicas: **cosh(x), sinh(x), tanh(x)**;

Raiz quadrada: **sqrt(x)**;

Exponencial: **exp(x)**;

Inverso da função exponencial (exp(x)): **log(x)**;

Logaritmo na base 10: **log10(x)**;

Resto da divisão: **rem(X, Y), mod(X, Y)**;

Funções elementares


```
>> x = 9

x =

     9

>> sqrt(x)

ans =

     3

>> exp(x)

ans =

   8.1031e+03
```

```
>> log(x)

ans =

    2.1972

>> log10(x)

ans =

    0.9542

>> sin(2*pi/3)

ans =

    0.8660
```

Exercício

4

Calcule as seguintes expressões:

$$x = sen(1.2 \times \pi) * e^3 + \ln(pi)$$

$$y = \frac{\cosh(x \times (3+x))}{sen(pi \times x)}$$

$$z = log_{10}(x * y)$$

Exercício

Calcule as seguintes expressões:

$$x = sen(1.2 \times \pi) * e^3 + \ln(pi)$$

$$y = \frac{\cosh(x \times (3+x))}{sen(pi \times x)}$$

$$z = log_{10}(x * y)$$

```
>> x = sin(1.2*pi)*exp(3) + log(pi)

x =

  -10.6613

>> y = cosh(x*(3+x))/sin(pi*x)

y =

  -1.6975e+35

>> z = log10(x*y)

z =

   36.2576
```

PETEE UFMG

Comandos e Funções

Salvar e Carregar o Workspace

Salvar e Carregar

Função Save

```
>> x = pi

x =

    3.1416

>> y = exp(1)

y =

    2.7183

>> z = pi*exp(pi)

z =

   72.6986
```

```
>> save('myfile','x','y','z')
>> save('myfile2.txt','x','y','z', '-ascii')
```

Salvar e Carregar

Função

```
>> load('myfile','x','y','z')
>> load myfile x y z
```

```
>> V = load('myfile','x','z')

V =

    x: 3.1416
    z: 72.6986
```

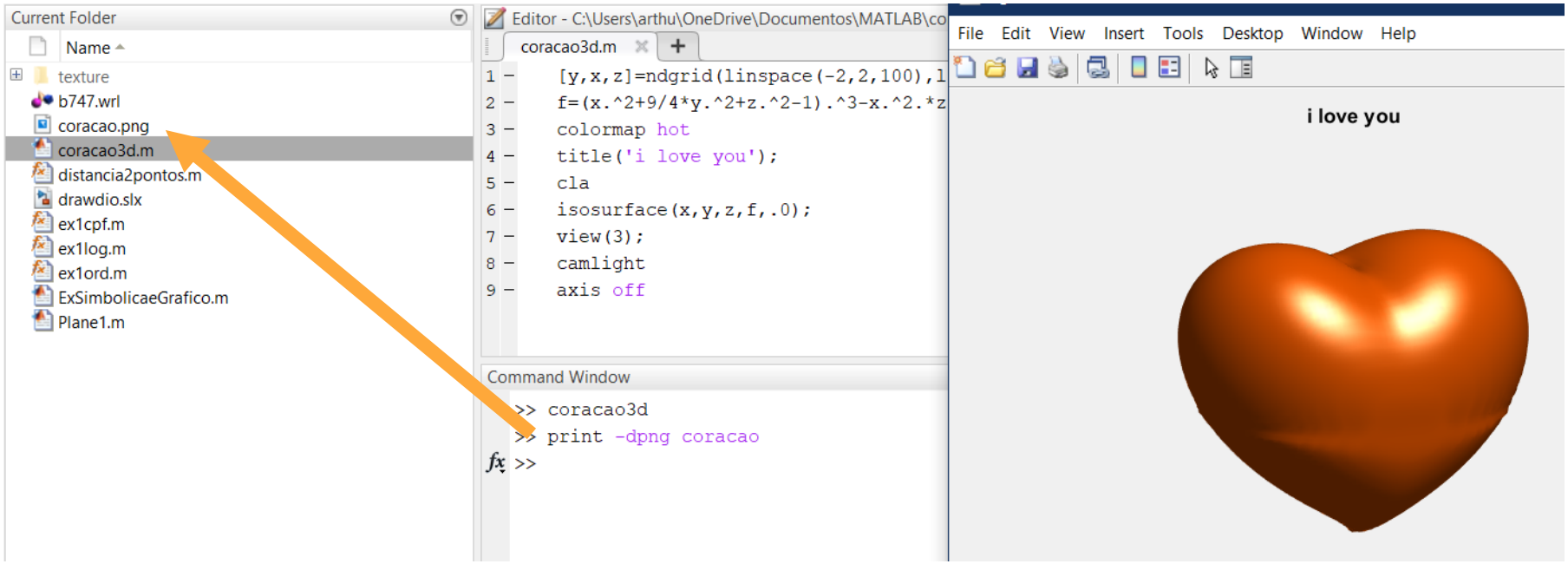
```
>> load('myfile.mat')
>> teste = load('myfile2.txt')

teste =

    3.1416
    2.7183
   72.6986
```

- Quando salvo em ASCII não é possível escolher a variável:

Salvar e Carregar

Função print

```
print(fig, '-dpdf', 'myfigure.pdf');
print -dpng coração;
```



PETEE UFMG

Comandos e Funções

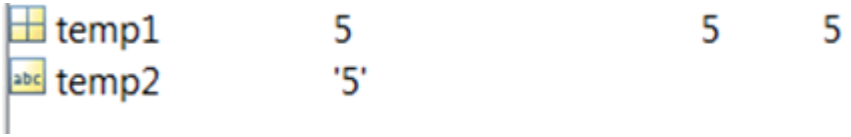
Entrada e Saída de Dados

Entrada

Insere um valor digitado por um usuário em uma variável;

Também pode apresentar um texto.

```
>> temp1 = input('Digite um valor: ')
Digite um valor: 5

temp1 =

     5
```




```
>> temp2 = input('Digite um valor: ','s')
Digite um valor: 5

temp2 =

5
```

Entrada

Converte número para string e vice-versa

- str2num
- num2str

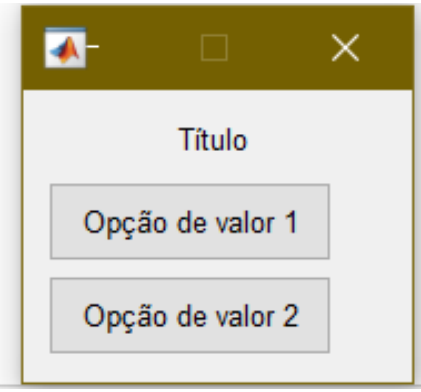
```
>> a1 = 9; a2 = '13';
>> whos
  Name        Size              Bytes  Class     Attributes

  a1          1x1                   8  double
  a2          1x2                   4  char
```

```
>> b2 = str2num(a2); b1 = num2str(a1);
>> whos
  Name        Size              Bytes  Class     Attributes

  a1          1x1                   8  double
  a2          1x2                   4  char
  b1          1x1                   2  char
  b2          1x1                   8  double
```

Entrada

Menu:

```
>> clear
>> X = menu('Título', 'Opção de valor 1', 'Opção de valor 2')
```



Saída

Command Window

```
>> disp('\n***PET-EE***\n')
\n***PET-EE***\n
>> disp(13)
    13

>> fprintf('\n***PET-EE***\n')

***PET-EE***
```

Command Window

```
>> x = 12.0

x =

    12

>> fprintf('%i\n',x)
12
>> fprintf('%f\n',x)
12.000000
>> fprintf('%.2f\n',x)
12.00
```

Command Window

```
>> fprintf('\nA soma é %.2f\n !!!\n PET-EE\n UFMG\n', x);

A soma é 12.00
 !!!
 PET-EE
 UFMG
```

Saída

Converte número para string

```
>> n = 13;
>> ns = num2str(n);
>> frase = ['sabado, ' ns ' de agosto']

frase =

sabado, 13 de agosto

>> disp(frase)
sabado, 13 de agosto
>>
```

Saída

`fprintf`

Code	Conversion instruction
`%s`	format as a string
`%d`	format with no fractional part (integer format)
`%f`	format as a floating-point value
`%e`	format as a floating-point value in scientific notation
`%g`	format in the most compact form of either `%f` or `%e`
`\n`	insert newline in output string
`\t`	insert tab in output string

PETEE UFMG

Operadores

Operador “:”

Operador “:”

Em sua forma mais simples é utilizado para criar um vetor de passo unitário delimitados pelo menor e maior elemento:

```
>> VEC=1:10

VEC =

     1     2     3     4     5     6     7     8     9    10
```

Configurado na forma x:y temos que o segundo elemento deve ser maior que o primeiro pois o comando busca alcançar y através do incremento de x:

```
>> K=100:4

K =

   Empty matrix: 1-by-0
```

Operador “:”

Operador “:”

Podemos também utilizar o operador : da forma **a:b:c**

```
>> x = 1:0.2:2

x =

    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

O operador : é ainda utilizado para indicar porções de uma matriz

```
>> A(3:5)

ans =

     9     4     3

>> A(2,2:end)

ans =

    10    11     8
```

Operador “ ’ ”

Operador aspas simples

Realiza a transposição matricial e conjugação complexa simultaneamente.

PETEE UFMG

Operadores

Aritméticos

Aritméticos

As operações básicas no MatLab são + - * / ^

Os mesmos devem ser utilizados em conjunto com ( )

$$2+\frac{3}{4}\times 5=5.7500$$

```
>> 2 + 3/4*5

ans =

    5.7500

>>
```

Aritméticos

Operadores do elemento “.”

.*	Element-by-element multiplication
./	Element-by-element division
.\	Element-by-element left division
.^	Element-by-element power
.'	Unconjugated array transpose

Aritméticos

Operadores do elemento “.”

```
>> A^2
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.

>> A.^2

ans =

     4    49     1
```

```
>> A = [ 2 7 1]; B = [9 5 2];
>> A*B
Error using  *
Inner matrix dimensions must agree.

>> A.*B

ans =

    18    35     2
```

Aritméticos

Resumindo:

1) Operações ponto a ponto;
2) Diferença entre a/b e a\b (barra e barra invertida);
3) Operador aspas simples;

PETEE UFMG

Operadores

Expressões Lógicas

Expressões Lógicas

As expressões lógicas avaliam termos falsos e verdadeiros e retornam uma resposta falsa ou verdadeira.

0 = falso
1 = verdadeiro

Obs: Por padrão, 1 é verdadeiro, mas no MATLAB, como em muitas linguagens de programação, qualquer valor diferente de 0 é também verdadeiro.

Expressões Lógicas

Exemplo:
<expressão> **e** <expressão>
O operador “e” retorna verdadeiro somente se ambas as expressões forem verdadeiras:

Você cursa GAAL **e** Cálculo?
Sim. Ou seja, as duas são verdadeiras.

Expressões Lógicas

O MATLAB, também pode ser usado como uma calculadora lógica. Exemplo:

Pergunta: 2 e 10?
Resposta: Veradeira

```
>> 2 & 10

ans =

  logical

   1
```


PETEE UFMG

Operadores

Lógicos

Lógicos


```
&    - e
&&   - e curto circuito
|    - ou
||   - ou curto circuito
~    - não
```

Obs: “Ou-exclusivo” não possui operador, é representado pela função xor(A,B)

```
Command Window
>> 1 & 0

ans =

  logical

   0

>> 0 | 5

ans =

  logical

   1
```

Lógicos

Curto circuito só avalia a segunda expressão se necessário.

```
>> x = [2 2]; y = [1 2 3];
>> a = 1; x = [2 2]; y = [1 2 3];
>> a == 0 && x+y == 1 % e curto circuito
ans =
  logical
    0
>> a == 0 & x+y == 1  % e
Matrix dimensions must agree.
```

PETEE UFMG

Operadores

Relacionais

Relacionais

`x == 2`	is x equal to 2?
`x ~= 2`	is x **not** equal to 2?
`x > 2`	is x greater than 2?
`x < 2`	is x less than 2?
`x >= 2`	is x greater than or equal to 2?
`x <= 2`	is x less than or equal to 2?

```
>> x = pi
x =
    3.1416
>> x ~= 3,  x ~= pi
ans =
     1
ans =
     0
```

Relacionais

Quando utilizamos matrizes ou vetores os testes são realizados em cada elemento

```
x =
   -2.0000    3.1416    5.0000
   -1.0000         0    1.0000
>> x == 0
ans =
     0     0     0
     0     1     0
```

```
>> x > 1,  x >=-1
ans =
     0     1     1
     0     0     0
ans =
     0     1     1
     1     1     1
```

Relacionais

Podemos ainda combinar expressões lógicas da seguinte forma:

```
>> x
x =
   -2.0000    3.1416    5.0000
   -5.0000   -3.0000   -1.0000
>> x > 3 & x < 4
ans =
     0     1     0
     0     0     0
>> x > 3 | x == -3
ans =
     0     1     1
     0     1     0
```

```
>> x > 3 | x == -3 | x <= -5
ans =
     0     1     1
     1     1     0
```

PETEE UFMG

Operadores

Prioridades

Prioridades

Lista de prioridades:

```
()
.'      .^      '       ^
.^-     .^+     .^~     ^-      ^+
+(unário)       -(unário)       ~
.*      ./      .\      *       /       \
+ -
:
<       <=      >       >=      ==      ~=
&
|
&&
||
```

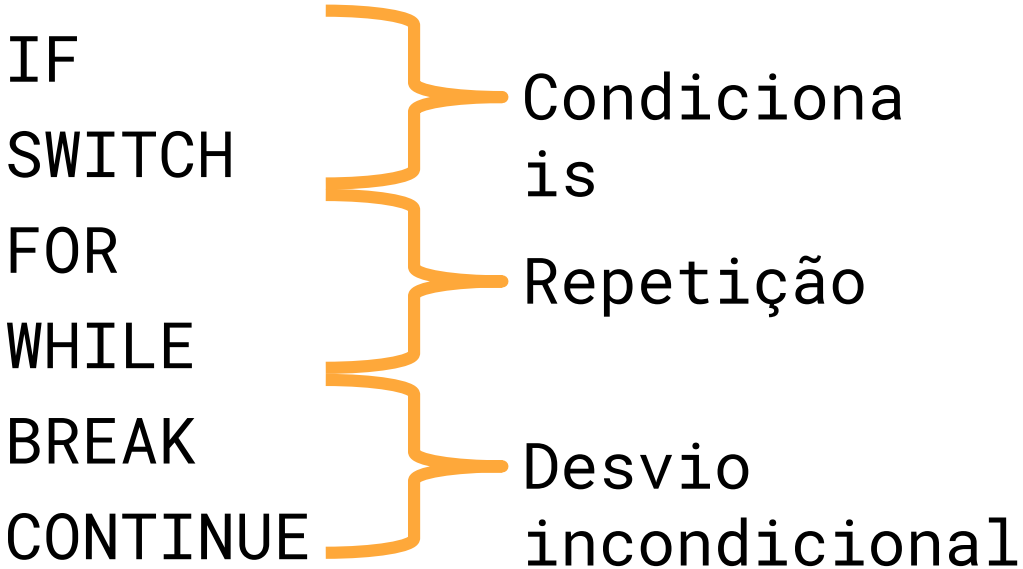
Resumindo: na dúvida, coloque parênteses...

PETEE UFMG

Controle de Fluxo

Controle de Fluxo

Rotinas utilizadas para criar laços de repetição ou tomar decisões



Condicionais

If:

```
1       %Exemplo do uso do IF
2
3 -     a = 4;
4 -     b = 3;
5
6 -     if a > b
7 -         t = 2;
8 -     elseif a < b
9 -         t = pi;
10 -    else
11 -        t = b-a;
12 -    end
```

Condicionais

Switch:

```
1     %exemplo do Switch e Case
2
3 -   x = 4;
4
5 -   switch x
6 -       case 1
7 -           a = x - 1;
8 -       case 2
9 -           a = x^2;
10 -      case 4
11 -          a = x;
12 -      otherwise
13 -          a = 0;
14 -  end
```

```
1     %exemplo do Switch e Case
2
3 -   x = [12, 64, 24];
4 -   plottype = 'pie3';
5
6 -   switch plottype
7 -       case 'bar'
8 -           bar(x)
9 -           title('Bar Graph')
10 -      case {'pie','pie3'}
11 -          pie3(x)
12 -          title('Pie Chart')
13 -          legend('First','Second','Third')
14 -      otherwise
15 -          warning('Unexpected plot type. No plot created.');
16 -  end
```

Repetição

For:

```
for i = 1:m
    for j = 1:n
        H(i,j) = 1/(i+j);
    end
end
```

Fibonacci

```
>> F(1) = 0; F(2) = 1;
>> for i = 3:20
     F(i) = F(i-1) + F(i-2);
   end
```

```
for k = [ 2 6 11 3]
    fprintf('numero %f\n', k);
end
```


Repetição

While:

```
>> S = 1; n = 1;
>> while S+ (n+1)^2 < 100
    n = n+1;    S = S + n^2;
   end
>> [n, S]
ans =
     6    91
```

Desvio Incondicional

Break: possibilita uma saída antecipada de um laço for ou while.

Em laços concatenados o break possibilita a saída do laço mais interno.

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if fx == 0
        break
    elseif sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x
```

Desvio Incondicional

Continue: passa para a próxima iteração do laço.

```
for i = 1:10
    if i == 3
        continue
    end
    fprintf('%i ', i)
end
```

1 2 4 5 6 7 8 9 10

Exercício

Calcule 9! de dois modos: utilizando o laço **for** e o laço **while**.

Exercício

Calcule 9! de dois modos: utilizando o laço **for** e o laço **while**.

```
5       %primeira opção
6 -     n = 9;
7 -     fat1 = n;
8 -     while n > 1
9 -         fat1 = fat1*(n-1);
10 -        n = n-1;
11 -    end
12
```

```
13      %segunda opção
14 -    n = 9;
15 -    fat2 = 1;
16 -    for ii = 1:n
17 -        fat2 = fat2*ii;
18 -    end
```


PETEE UFMG

Scripts

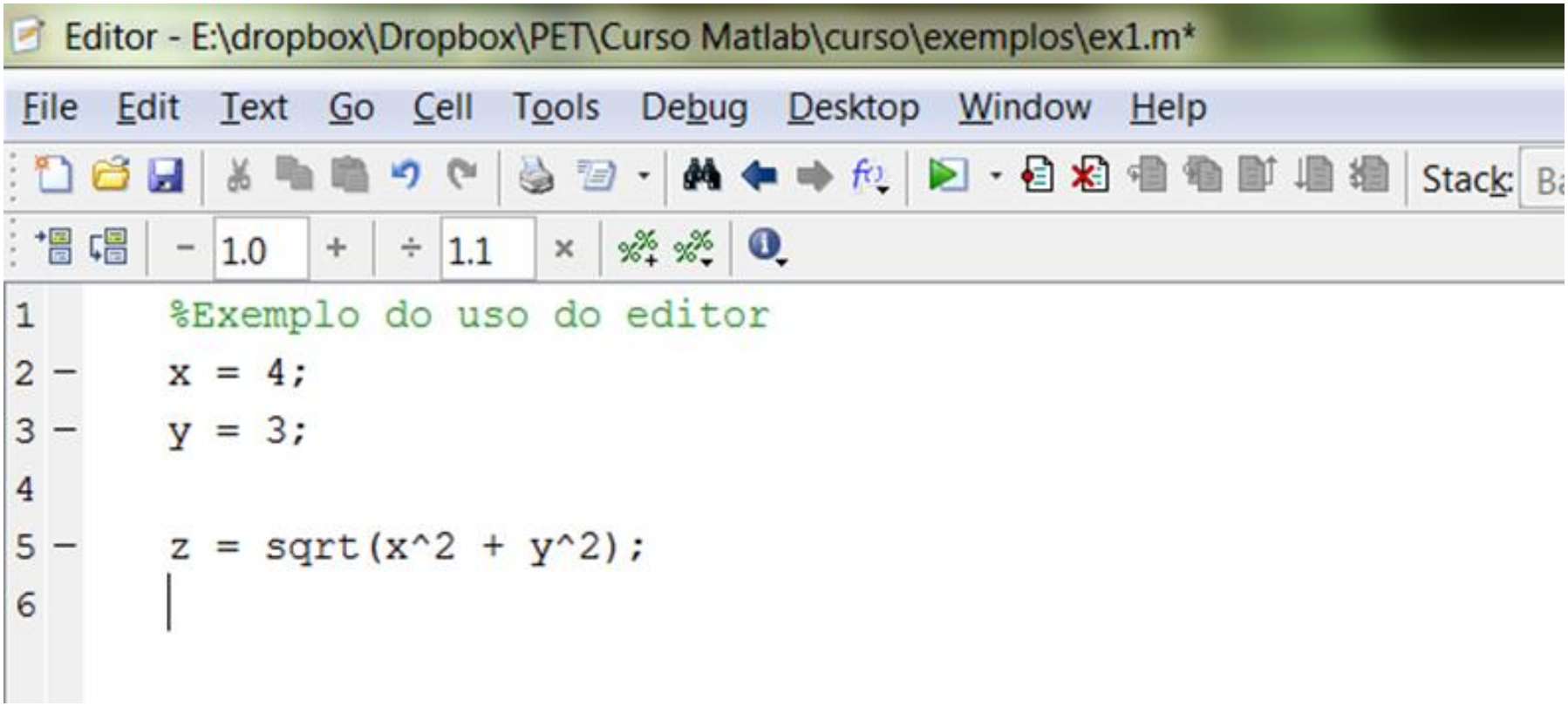
Editor/Debugger

Utilizado para executar uma sequência de comandos pré-determinada;

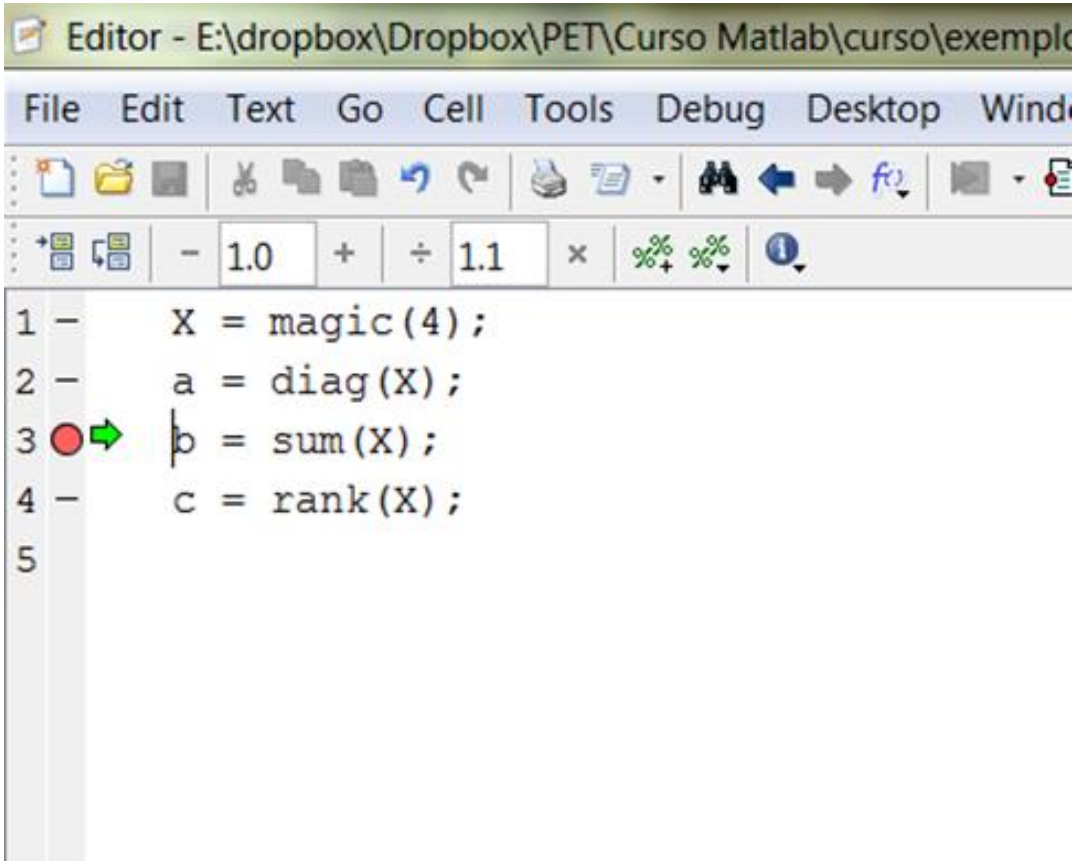
Diversas opções para execução dos comandos;

Opção para seleção de breakpoints.

Editor/Debugger



Editor/Debugger



Comandos


```
>> edit código.m
```

Abrir o editor

```
>> publish('codigo.m', 'pdf')
```

Documentar

UFMG

Exercício

Faça em um script:

a) Crie o vetor x = [1, 2, 3, 4, 5]

b) Calcule sen(x^2) e cos(x^2)

c) Calcule tan(x^2) através dos resultados anteriores.

Exercício

Exercício:

Faça em um script:

a) Crie o vetor x = [1, 2, 3, 4, 5]

b) Calcule sen(x^2) e cos(x^2)

c) Calcule tan(x^2) através dos resultados anteriores.

```
x = 1:5;
s = sin(x.^2); c =
cos(x.^2);
t = s./c
```

PETEE UFMG

Scripts e Arquivos .m

Funções

Funções

Utilizada quando desejamos preparar uma combinação de operações e ideais em um script que pode ser acessado posteriormente de forma rápida.

Tomaremos como exemplo o seguinte problema:

Desejamos preparar uma função que calcula a área A de um triangulo cujo o comprimento dos lados é a, b e c.

Funções

Nome do arquivo = nome da função

```
function  [Va. Retornadas] = nome(
  Parâmetros)
  ...Procedimentos...
end
```


Exemplo

distancia2pontos.m

```
1   function resposta = distancia2pontos(x1, y1, x2, y2)
2   %distancia2pontos calcula a distância entre dois pontos
3   %Exemplo de descrição da função
4   %conteúdo e etc...
5   resposta = sqrt( (x1-x2)^2 + (y1-y2)^2 );
6   end
```


Funções

Por fim podemos documentar através de comentários o arquivo de função de forma que o usuário tenha acesso ao comando help

distancia2pontos.m

```
1   function resposta = distancia2pontos(x1, y1, x2, y2)
2   %distancia2pontos calcula a distância entre dois pontos
3   %Exemplo de descrição da função
4   %conteúdo e etc...
5   resposta = sqrt( (x1-x2)^2 + (y1-y2)^2 );
6   end
```

Command Window

```
>> help distancia2pontos
 distancia2pontos calcula a distância entre dois pontos
 Exemplo de descrição da função
 conteúdo e etc...
```


Exercício

Crie um arquivo .m chamado ex1log.m que define a função ex1log. A função receberá dois escalares x e y e retornará o $\log_y x$.
Lembre-se que:
$\log_b a = \log_c a / \log_c b$.

Exercício

Crie um arquivo .m chamado ex1log.m que define a função ex1log. A função receberá dois escalares x e y e retornará o $\log_y x$.
Lembre-se que:
$\log_b a = \log_c a / \log_c b$.

```
function c = ex1log(x,y)
c = log(x)/log(y);
end
```


Exercício

Crie um arquivo .m chamado ex1ord que define a função ex1ord. A função deverá receber um arranjo linear e retornar outro arranjo, com os mesmo valores do primeiro ordenados em ordem crescente. Use o método de ordenação que lhe deixar mais confortável.

Exercício

Crie um arquivo .m chamado ex1ord que define a função ex1ord. A função deverá receber um arranjo linear e retornar outro arranjo, com os mesmo valores do primeiro ordenados em ordem crescente. Use o método de ordenação que lhe deixar mais confortável.

```
function ordenado = ex1ord(x)
for i = 1:size(x,2)
    for j = (i+1):size(x,2)
        if x(i) > x(j)
            aux = x(i);
            x(i) = x(j);
            x(j) = aux;
        end
    end
end
ordenado = x;
end
```

PETEE UFMG

Gráficos

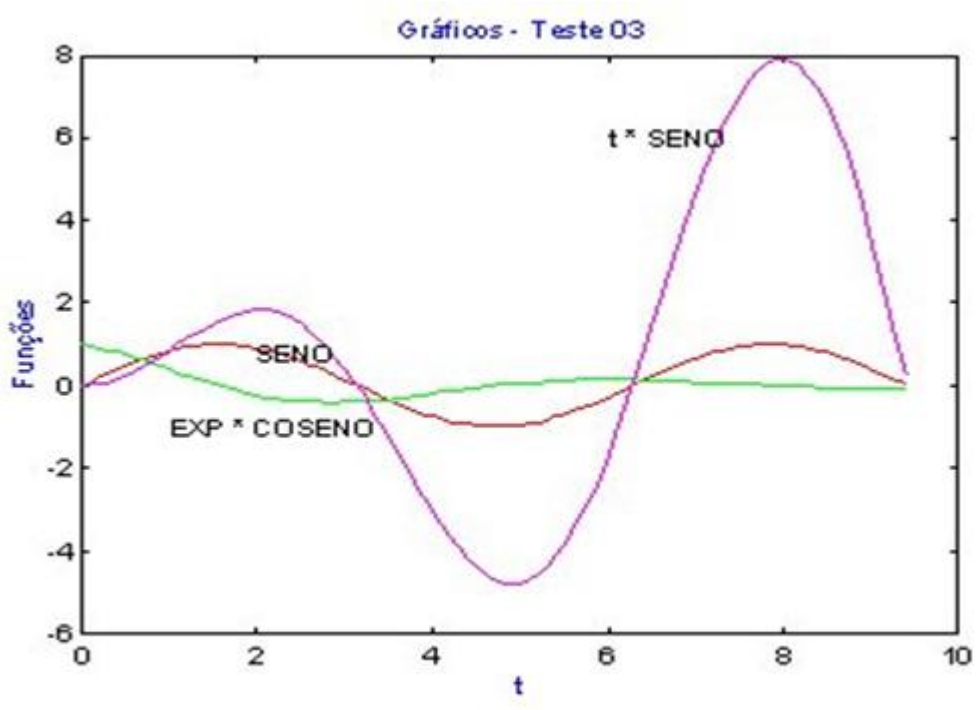
PETEE UFMG

Gráficos

Bidimensionais

Bidimensionais

O Matlab possui uma extensa variedade para representar vetores e matrizes na forma gráfica.



Bidimensionais

Plot

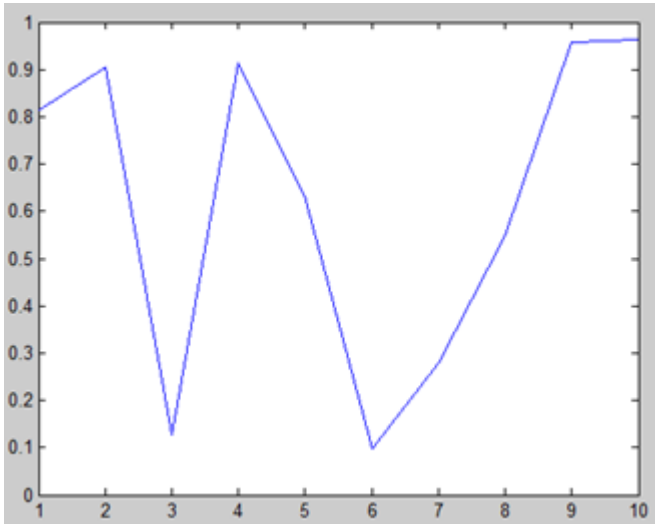
Possui diferentes formas de uso dependendo da quantidade de seus argumentos de entrada.

Em sua forma mais simples, dado um vetor y o comando “plot(y)” produz um gráfico dos elementos de y distribuídos de forma linear ao longo do x.

Bidimensionais

Plot

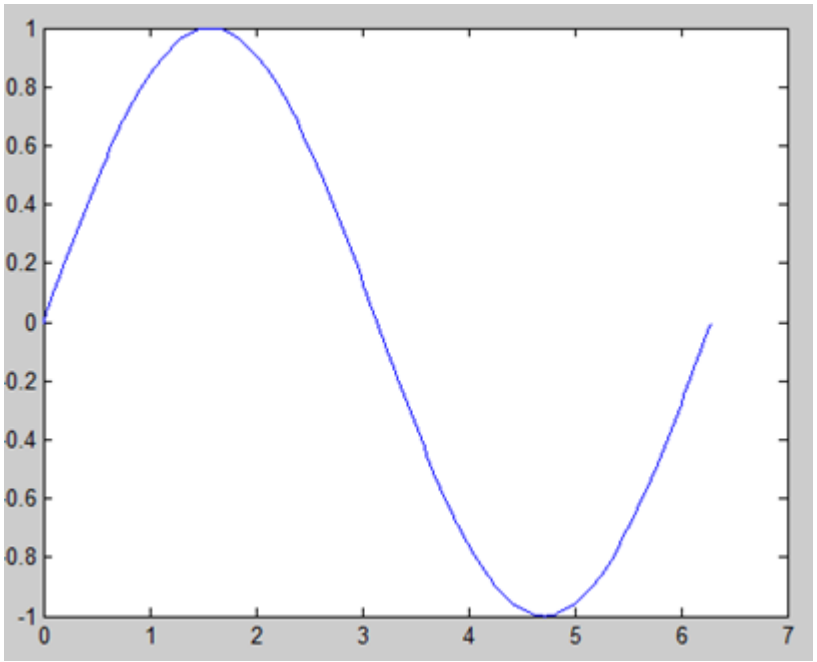
```
>> V=rand(1,10)

V =

  Columns 1 through 9

    0.8147    0.9058    0.1270    0.9134    0.6324    0.0975    0.2785    0.5469    0.9575

  Column 10

    0.9649

>> plot(V)
```



Bidimensionais

Plot

Se utilizarmos 2 vetores x e y como entrada teremos como resultado um gráfico de x vs y.

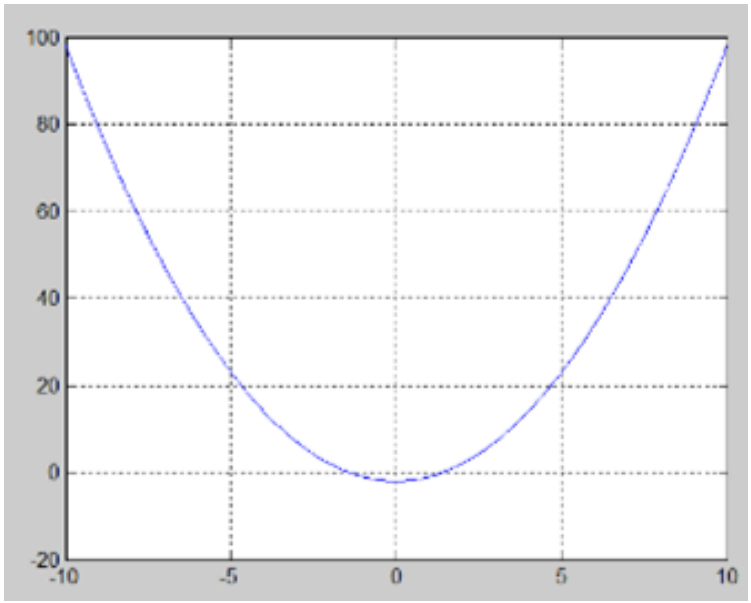



```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

Bidimensionais

Plot

Se utilizarmos 2 vetores x e y como entrada teremos como resultado um gráfico de x vs y.



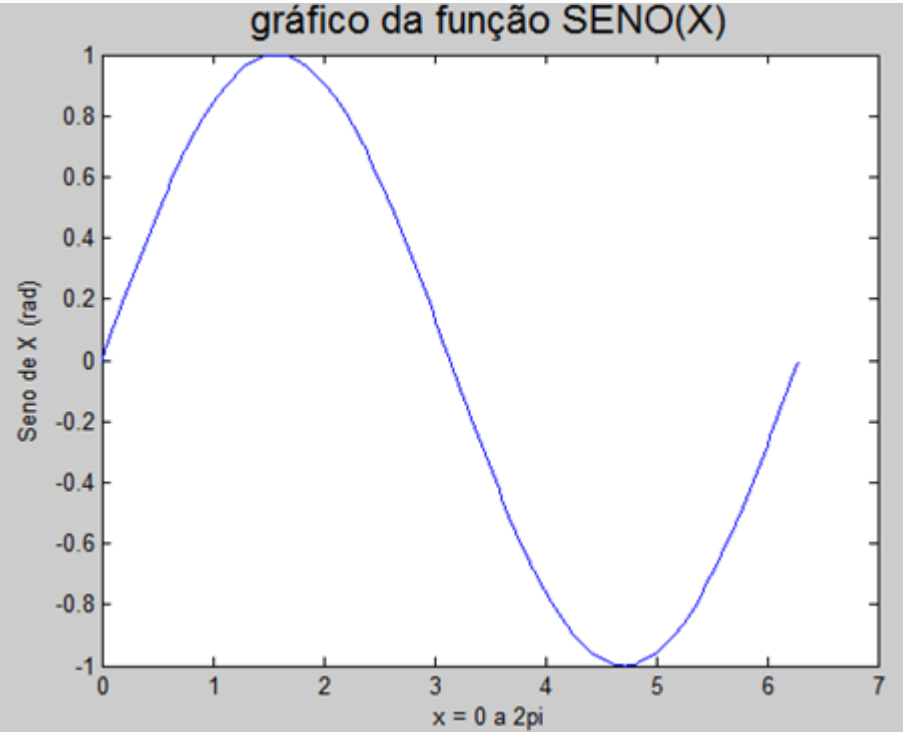
Command Window

```
>> x = -10:0.01:10;
>> y = x.^2 - 2;
>> plot(x,y);
>> grid on;
```

Bidimensionais

Plot

Podemos identificar os eixos presentes no gráfico bem como adicionar um título ao mesmo através dos comandos:

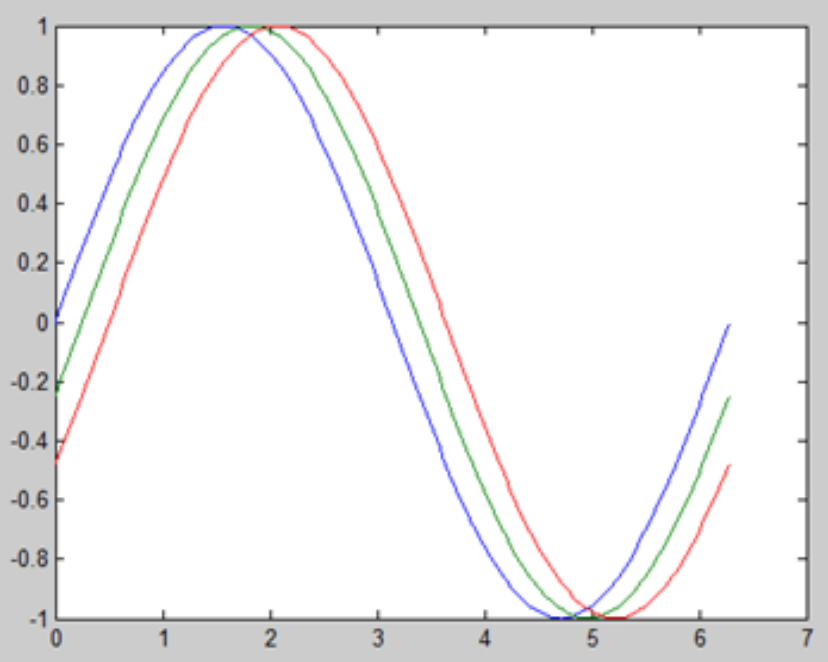



```
>> xlabel('x = 0 a 2pi')
>> ylabel('Seno de X (rad)')
>> title('grafico da função SENO(X)', 'FontSize', 16)
```

Bidimensionais

Plot

Múltiplos pares de dados z-y como argumentos de entrada produzem múltiplos gráficos em apenas uma chamada do “plot”

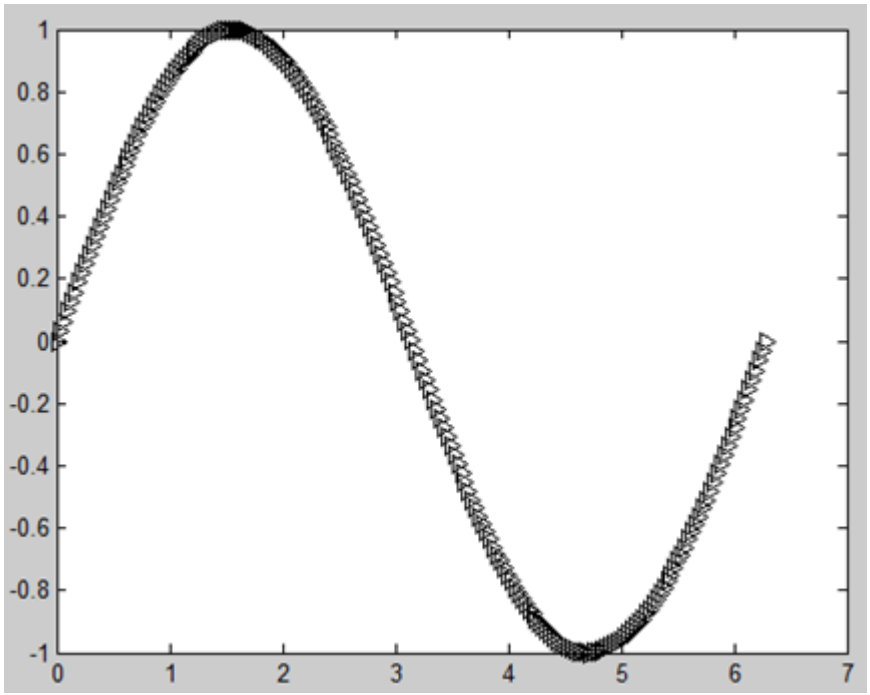



```
y2 = sin(x-.25);
y3 = sin(x-.5);
plot(x,y,x,y2,x,y3)
```

Bidimensionais - estilos

Plot

É possível ainda especificarmos a cor e o estilo de marcador do gráfico utilizando o “plot” com a seguinte sintaxe



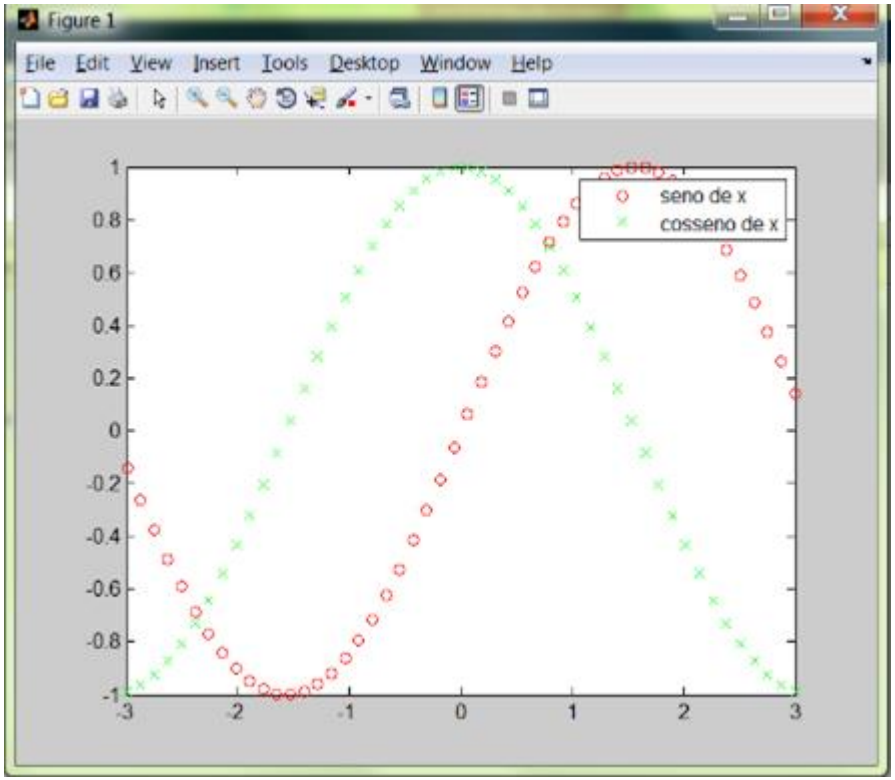

```
plot(x,y,'color_style_marker')
>> plot(x,y,'k>')
```

Colours		Line Styles	
y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green	-	solid
b	blue	*	star
w	white	:	dotted
k	black	-.	dashdot
		--	dashed

Bidimensionais - estilos

Plot

Exemplo




```
>> y1 = sin(x);
>> y2 = cos(x);
>> plot(x,y1,'ro')
>> hold on
>> plot(x,y2,'gx')
>> legend('seno de x','cosseno de x')
```

Bidimensionais

Plot

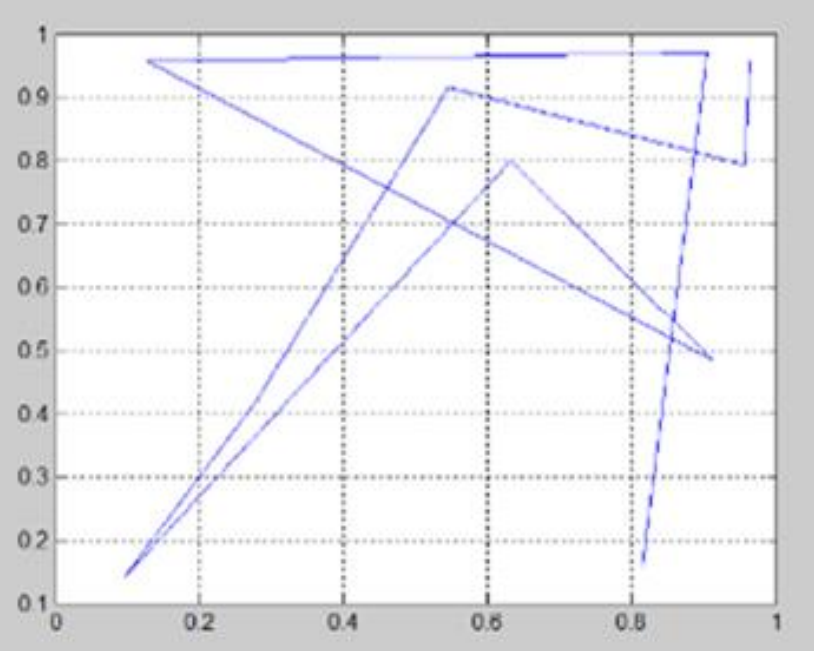
Números Complexos: Ao trabalharmos com números complexos o comando plot necessita apenas de um argumento para entrada já que os mesmo já estão representados em suas partes reais e imaginárias:

```
Command Window
>> r = rand(1,10);
>> I = rand(1,10);
>> Z = r+j*I;
>> Z

Z =

  Columns 1 through 5

   0.8147 + 0.1576i   0.9058 + 0.9706i   0.1270 + 0.9572i   0.9134 + 0.4854i   0.6324 + 0.8003i

  Columns 6 through 10

   0.0975 + 0.1419i   0.2785 + 0.4218i   0.5469 + 0.9157i   0.9575 + 0.7922i   0.9649 + 0.9595i

>> plot(Z)
>> grid
```

Bidimensionais

Plot

Números Complexos: Ao trabalharmos com números complexos o comando plot necessita apenas de um argumento para entrada dos dos já que os mesmo já estão representados em suas partes reais e imaginárias:

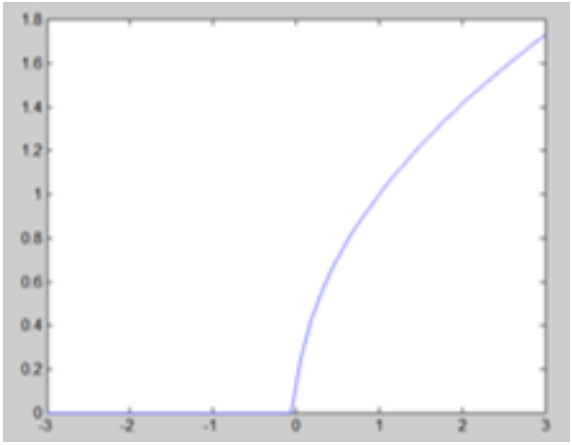


Bidimensionais

Plot

Números Complexos: No entanto se utilizarmos o “plot” com mais de um argumento de entrada o Matlab irá ignorar a parte imaginária dos números na tentativa de adequar o gráfico a apenas 2 dimensões:

```
>> x = linspace(-3,3,50);
>> y = sqrt(x);
>> plot(x,y)
Warning: Imaginary parts of complex X and/or Y arguments ignored
```



Bidimensionais

fplot – plotar funções usando como argumentos funções simbólicas ou strings. Ex:

```
fplot('sin(x)');
syms x;
Y = x^2 + 2;
fplot(Y);
```

Pode receber também, como argumentos, o intervalo e o estilo:

```
fplot('sin(x)', [0, 2*pi], 'ro');
```

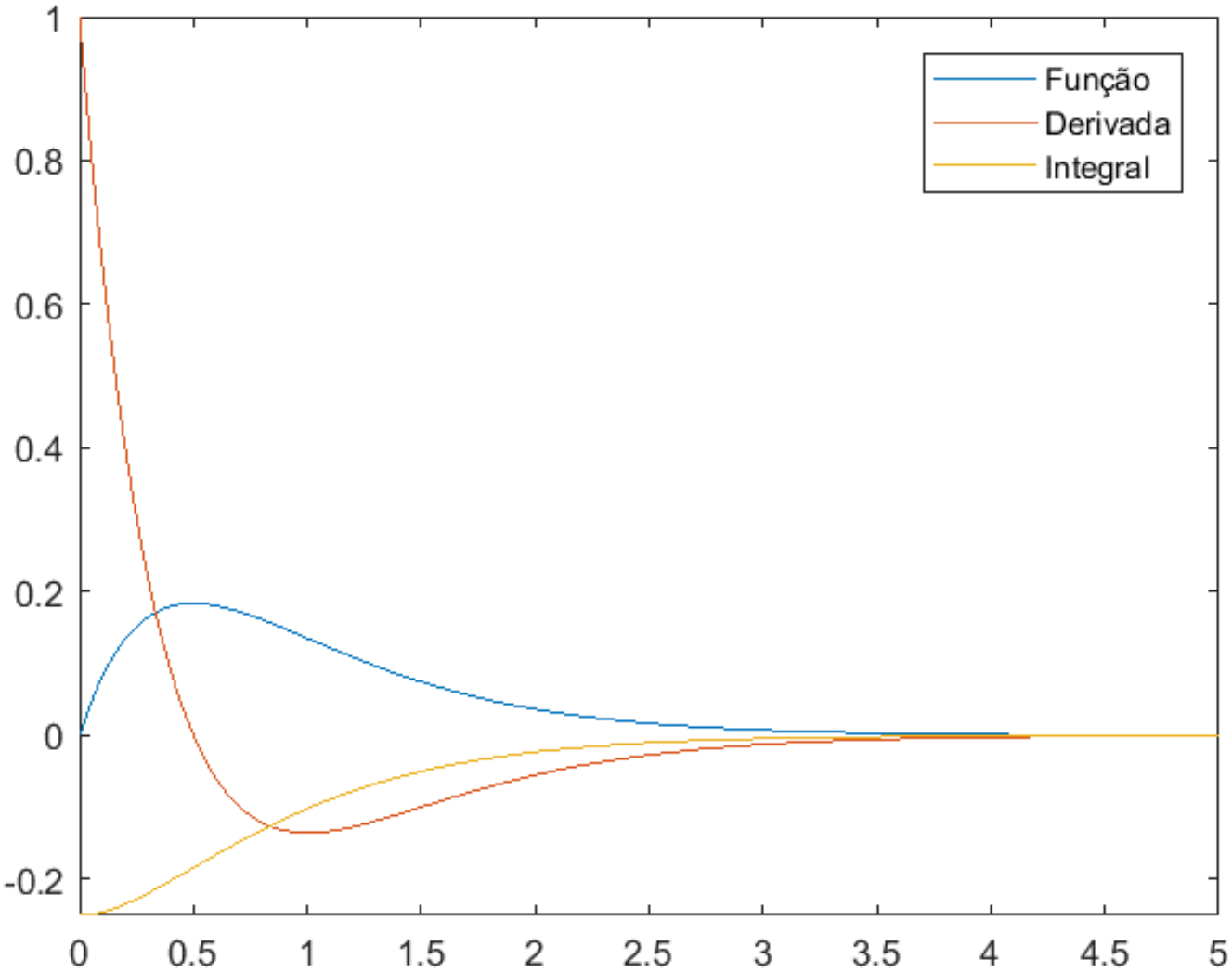
Exercício

Exercício: Dada a função $y = xe^{-2x}$, trace o gráfico da função, da sua derivada e da sua integral.

Exercício


```
syms x;
y = x*exp(-2*x)
dif_y = diff(y, x)
int_y = int(y, x)
fplot(y, [0, 5])
hold on
fplot(dif_y, [0,5])
fplot(int_y, [0,5])
legend('Função', 'Derivada', 'Integral')
clear x; clear y; clear dif_y; clear
int_y;
```

Exercício


Função
Derivada
Integral
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Exercício

Outra forma de solução:

Command Window

```
>> y = x*exp(-2*x)

y =

x/exp(2*x)

>> v = 0:0.01:5;
>> plot(v,subs(y,v))
>> hold on
>> dif_y = diff(y,x)

dif_y =

1/exp(2*x) - (2*x)/exp(2*x)

>> plot(v,subs(dif_y,v),'r')
>> grid on
>> int_y = int(y,x)

int_y =

-(2*x + 1)/(4*exp(2*x))

>> plot(v,subs(int_y,v),'g')
>> legend('função','Derivada da função', 'Integral da função')
```

Exercício

8

A função exponencial e^x pode ser interpretada como um somatório de termos infinitos $\sum_{n=0}^{\infty} \frac{x^n}{n!}$. Esse processo é chamado de expansão em série de Taylor. Utilizando linhas tracejadas e o intervalo de -2 a 5, para facilitar a visualização, plote, em uma mesma figura o somatório com n variando de 0 a:

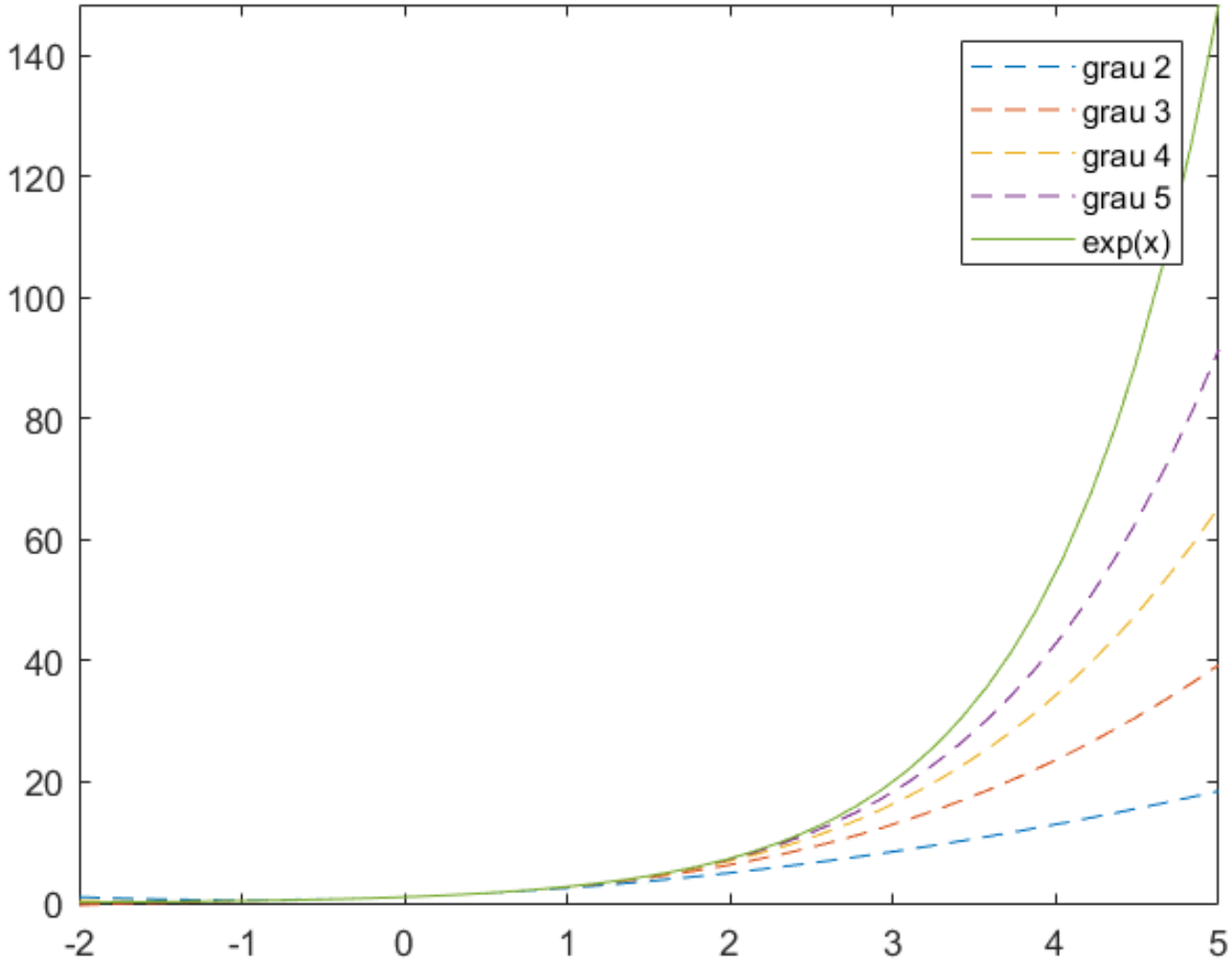
a) 2
b) 3
c) 4
d) 5

Em seguida, plote a própria função exponencial e^x , em linha contínua e no intervalo de -2 a 5, e observe como a função converge a medida em que somamos o próximo termo. Insira legenda.

Exercício


```
>>syms x;
>> s2 = 1 + x + x^2/factorial(2);
>> s3 = s2 + x^3/factorial(3);
>> s4 = s3 + x^4/factorial(4);
>> s5 = s4 + x^5/factorial(5);
>> fplot(s2, [-2, 5], '--'); hold on;
>> fplot(s3, [-2, 5], '--');  fplot(s4, [-2, 5],
'--');  fplot(s5, [-2, 5], '--');
>> fplot(exp(x), [-2, 5]);
>> legend('grau 2', 'grau 3', 'grau 4', 'grau 5',
'exp(x)');
```

Exercício


grau 2
grau 3
grau 4
grau 5
exp(x)
140
120
100
80
60
40
20
0
-2
-1
0
1
2
3
4
5

PETEE UFMG

Gráficos

Tridimensionais

Tridimensionais

Um gráfico bidimensional pode ser transformado em tridimensional pelo comando “view(3)”.

Além disso, uma **curva** pode ser plotada em três dimensões usando “plot3(x,y,z)”, sendo x, y e z 3 vetores.

No entanto, para **superfícies** são necessárias **matrizes**

Tridimensionais

Uma superfície é definida matematicamente como uma função de duas variáveis f(x,y).

Correspondendo a cada valor (x,y) computamos o valor funcional por z= f(x,y).

Desta forma para traçarmos a superfície devemos primeiramente decidir os limites de x e y, por exemplo, $2 \leq x \leq 4$ e $1 \leq y \leq 3$.

Tridimensionais

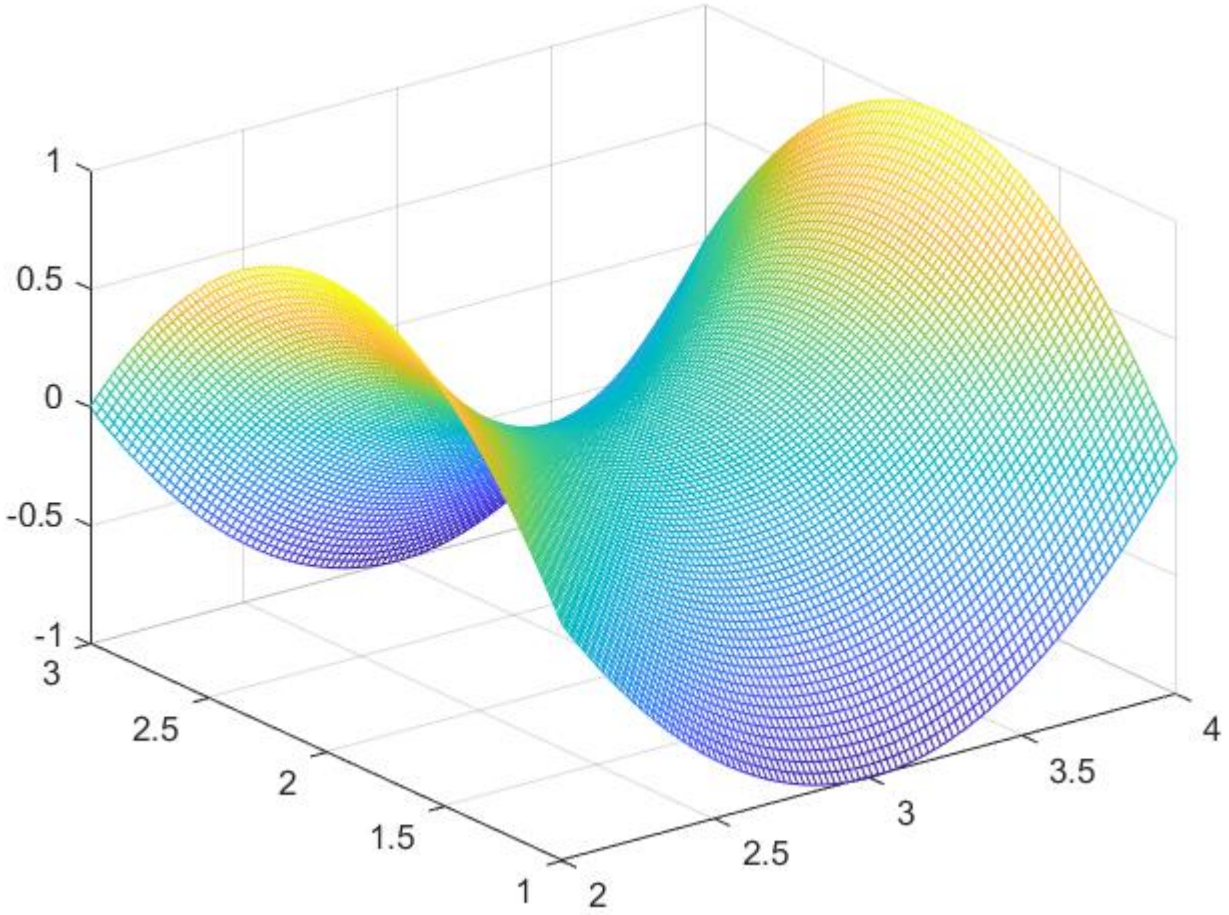
$$f(x,y) = (x-3)^2 - (y-2)^2 \qquad \begin{matrix} 2 \le x \le 4 \\ 1 \le y \le 3. \end{matrix}$$

Command Window

```
>> x = linspace(2,4); y = linspace(1,3);
>> [X, Y] = meshgrid(x, y);
>> Z = (X-3).^2 - (Y-2).^2;
```

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


Tridimensionais



Tridimensionais

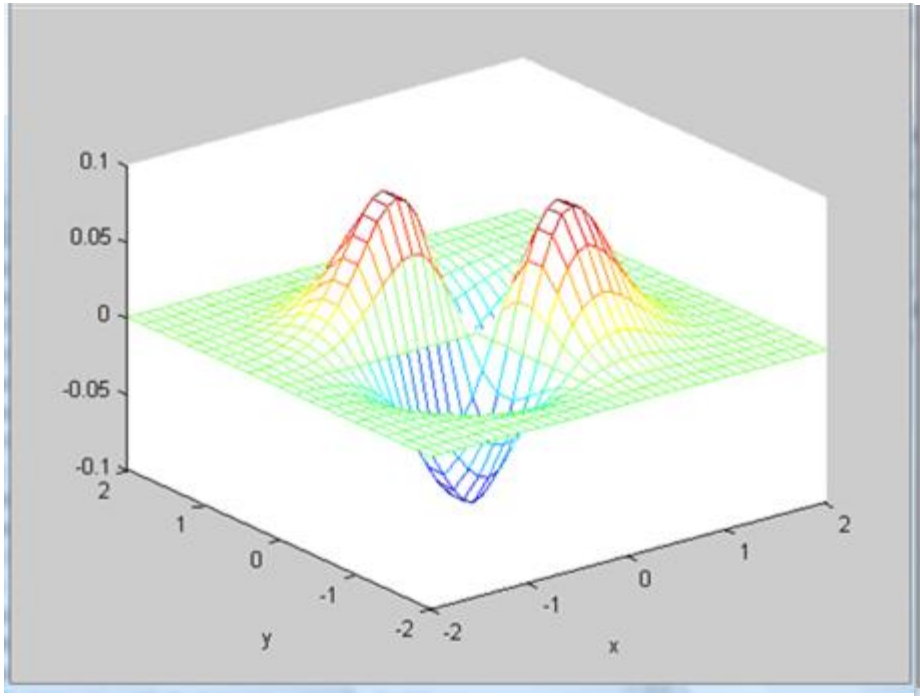
Exemplo

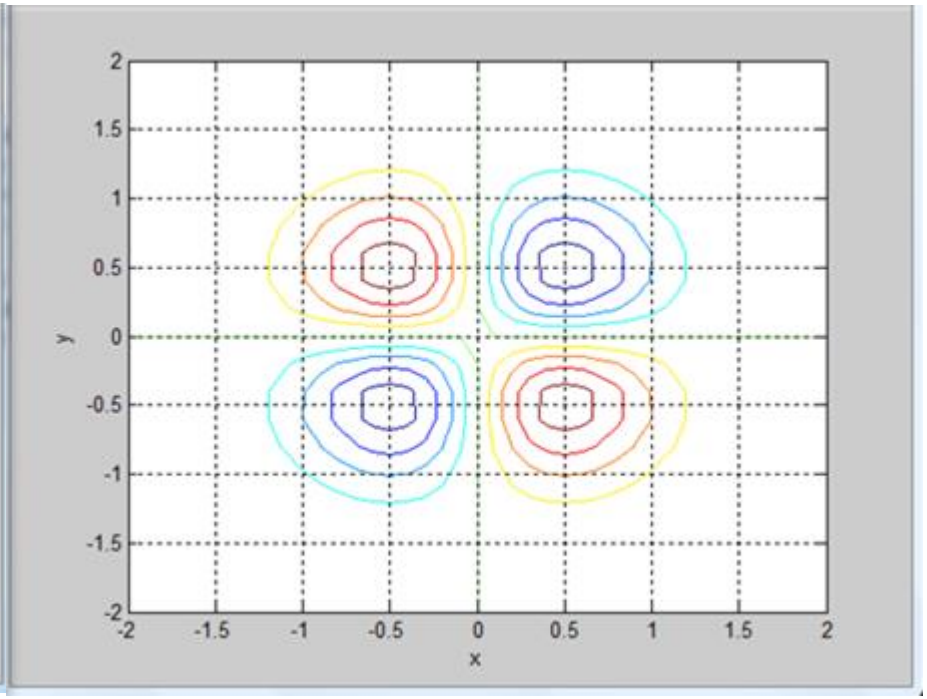
$$f = -xye^{-2(x^2+y^2)} \qquad -2 \leq x \leq 2, -2 \leq y \leq 2.$$

```
>> [X,Y] = meshgrid(-2:.1:2,-2:.2:2);
f = -X.*Y.*exp(-2*(X.^2+Y.^2));
figure (1)
mesh(X,Y,f), xlabel('x'), ylabel('y'), grid
figure (2), contour(X,Y,f)
xlabel('x'), ylabel('y'), grid, hold on
>>
```

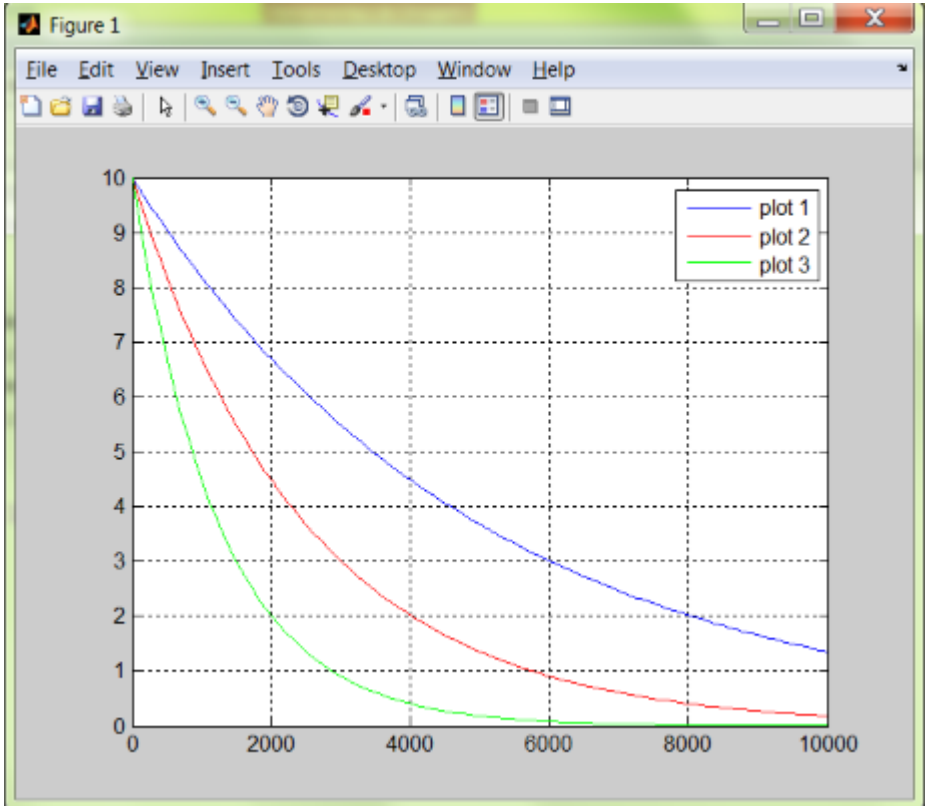

Tridimensionais

Exemplo





Hold



Command Window

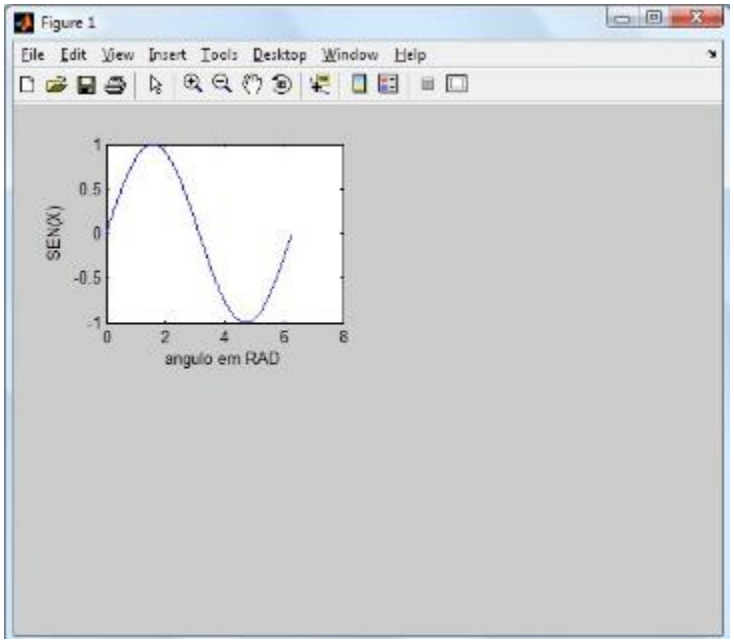
```
>> t = linspace(0,10000,100);
>> y = 10*exp(-2e-4*t);
>> plot(t,y);
>> hold on
>> y = 10*exp(-4e-4*t);
>> plot(t,y,'r');
>> y = 10*exp(-8e-4*t);
>> plot(t,y,'g');
>> grid on
>> hold off
>> legend('plot 1','plot 2','plot 3')
```

Subplot

Subplot

As janelas são numeradas na forma de um vetor coluna iniciando da janela superior esquerda

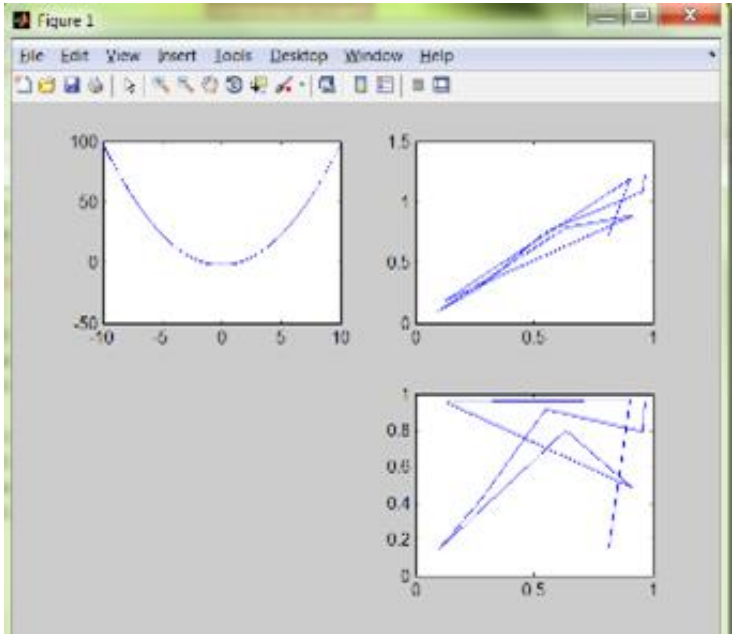
Comandos como “hold” ou “label” também são válidos nestes casos.




```
>> subplot(2,2,1), plot(x,y)
>> xlabel('angulo em RAD'), ylabel('SEN(X)')
```

Subplot


```
>> subplot(2,2,1); plot(x,y)
>> subplot(2,2,2); plot(Z,Sz)
Warning: Imaginary parts of complex X and/or Y arguments ignored
>> subplot(2,2,4); plot(r,I)
```



Exercício

Trace os gráficos das seguintes funções em uma mesma janela:

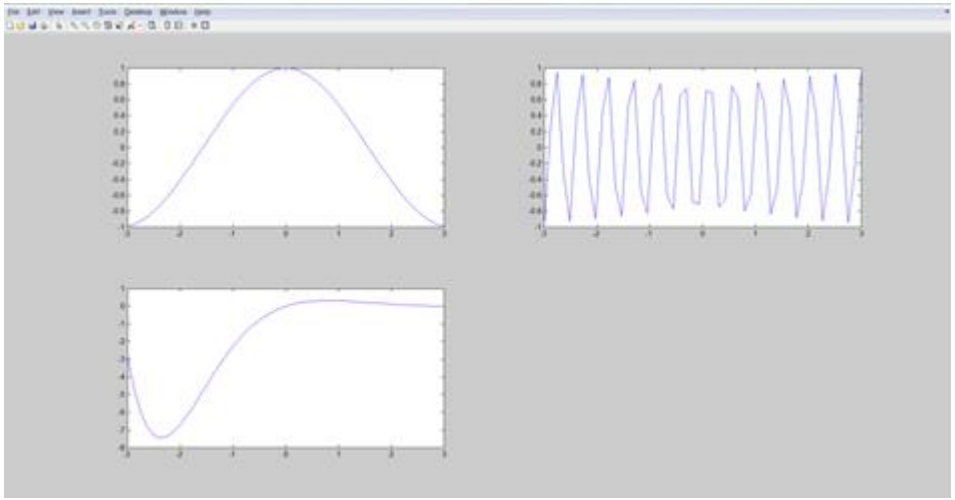
- cos(x)
- "sen(3*x)"
- e^(-x)*sin(x)

Exercício

Trace os gráficos das seguintes funções em uma mesma janela:

- cos(x)
- "sen(3*x)"
- e^(-x)*sin(x)

```
>> x = linspace(-3,3,50);
>> x = linspace(-3,3,50);
>> y1 = cos(x);
>> y2 = sin(13*x);
>> y3 = exp(-x).*sin(x);
>> subplot(2,2,1)
>> plot(x,y1);
>> subplot(2,2,2)
>> plot(x,y2);
>> subplot(2,2,3)
>> plot(x,y3);
```



Exercício

A superfície cone é definida como:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z^2}{c^2}$$

Plote o gráfico dessa função utilizando a função mesh, sendo a=2, b=3 e c=5. Use $x\ e\ y\ entre\ [-10, 10]$.

Adicione ainda, nesse mesmo gráfico, o contorno 2D.

Dica:

1 - Isole z, plote o gráfico positivo, congele a imagem, plote a parte negativa.

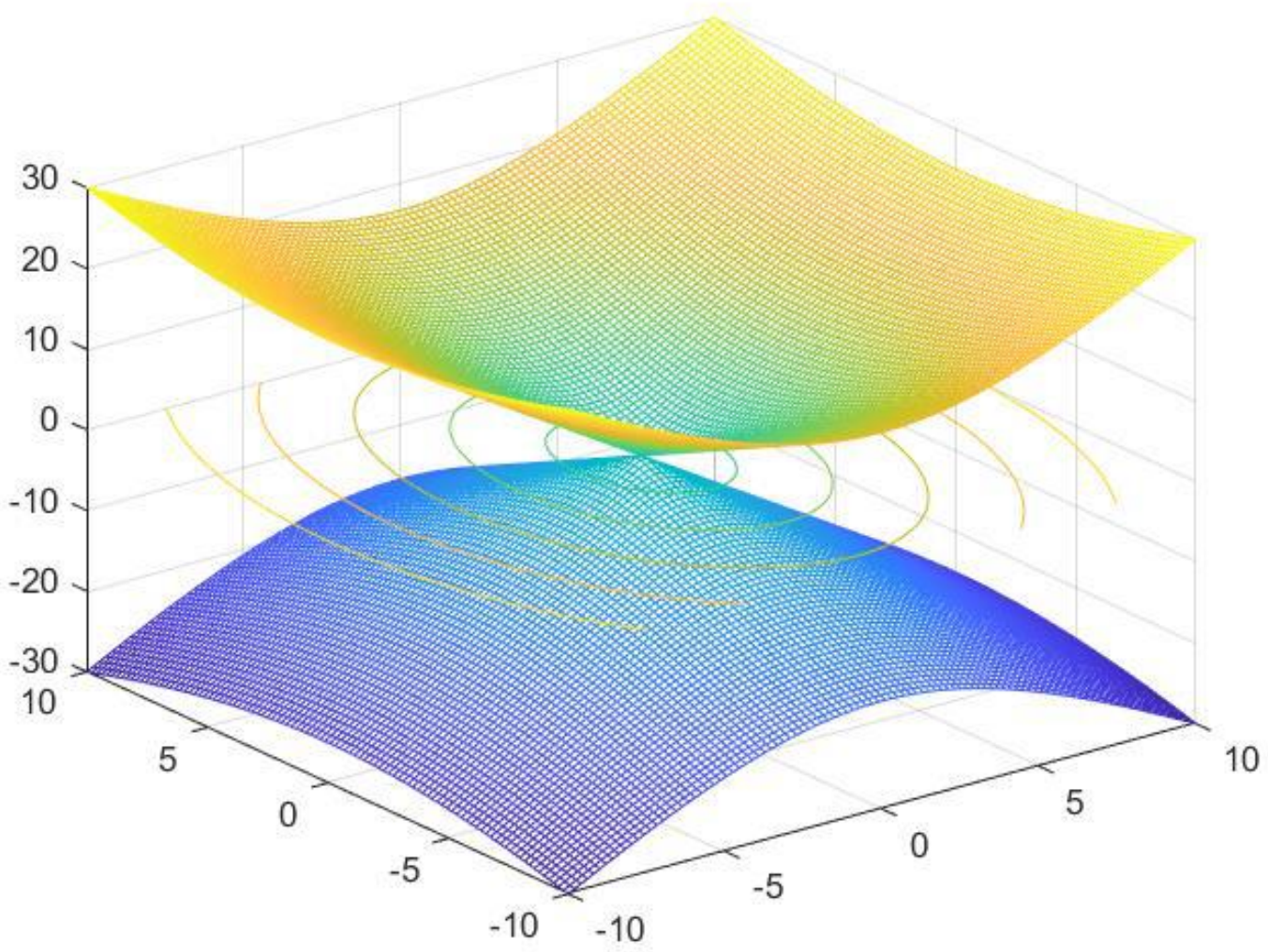
2 - Note que: $z = \pm 5\sqrt{\frac{x^2}{4} + \frac{y^2}{9}}$

3 - Função contour(x,y,z).

Exercício


```
>> x=linspace(-10, 10, 100); [x,
y]=meshgrid(x);
>> z=5*sqrt(x.^2/4 + y.^2/9);
>> mesh(x, y, z); hold on; mesh(x,
y, -z)
>> contour(x, y, z)
```


Exercício



PETEE UFMG

Polinômios

Polinômios

Arranjo, no qual, os elementos são os coeficientes Cn até C0.

$$C_n x^n + C_{n-1} x^{n-1} + \cdots + C_1 x^1 + C_0$$

Polinômio = [C_n C_{n-1} … C_1 C_0];

Polinômios

$y = 4*x^2 + x - 5$

$y = x^4 + 3$

Command Window

```
>> y = [4 1 -5]

y =

     4     1    -5

>> y = [1 0 0 0 3]

y =

     1     0     0     0     3
```

PETEE UFMG

Polinômios

Raízes

Raízes

Raízes de um polinômio

Roots e Poly

Command Window

```
>> y

y =

     4     1    -5

>> roots(y)

ans =

   -1.2500
    1.0000
```

```
>> r = [-3 2];
>> p = poly(r)

p =

     1     1    -6
```

Raízes

$$\det([A]-\lambda[I])=\begin{vmatrix}1-\lambda & 2 & 3\\ 4 & 5-\lambda & 6\\ 7 & 8 & 9-\lambda\end{vmatrix}=\lambda^3-15\lambda^2-18\lambda$$

```
>> B = [2, 2, 2; 0, 0,
0; 1, 2, 3];
>> pp = poly(B)
pp =
     1    -5     4     0
>> roots(pp)
ans =
     0
     4
     1
% Função eig(B);
```

PETEE UFMG

Polinômios

Operações Elementares

Adição e Subtração

Adição e subtração

$f1(x)=4*x^2 + 1*x - 5$

$f2(x)=x^2 + 9$

```
Command Window
>> f1

f1 =

     4     1    -5

>> f2

f2 =

     1     0     9

>> f3 = f1 + f2

f3 =

     5     1     4

>> f4 = f1 - f2

f4 =

     3     1   -14
```

Multiplicação

Multiplicação

$f1(x)=4*x^2 + 1*x - 5$

$f2(x)=x^2 + 9$

conv

Command Window

```
>> f3 = conv(f1,f2)

f3 =

     4     1    31     9   -45

>> f4 = f1.*f2 %Incorreto

f4 =

     4     0   -45

>> f4 = f1*f2 %incorreto
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Divisão

Divisão

$f1(x)=4*x^2 + 1*x - 5$

$f2(x)=x^2 + 9$

deconv

Command Window

```
>> [q r] = deconv(f1,f2)

q =

     4


r =

     0     1   -41
```

Integral e Derivada

Integral

$$\int df/dx\ (x)dx = \int (6*x-2)\ dx$$

$$f(x) = 3*x^2 - 2*x + C$$

```
>> x = polyint(der_y,4)

x =

     3     -2      4
```

Command Window

```
>> y = [3 -2 4]

y =

     3     -2      4

>> der_y = polyder(y)

der_y =

     6     -2

>> x = polyint(der_y)

x =

     3     -2      0
```

PETEE UFMG

Polinômios

Avaliação e Ajuste

Avaliação


```
%polyval(Polinômio, Valor(es))
>> P=[4 4 0 2 -5]; x=0:0.5:2;
>> polyval(P, x)
ans =
    -5.0000   -3.2500    5.0000
31.7500   95.0000
```

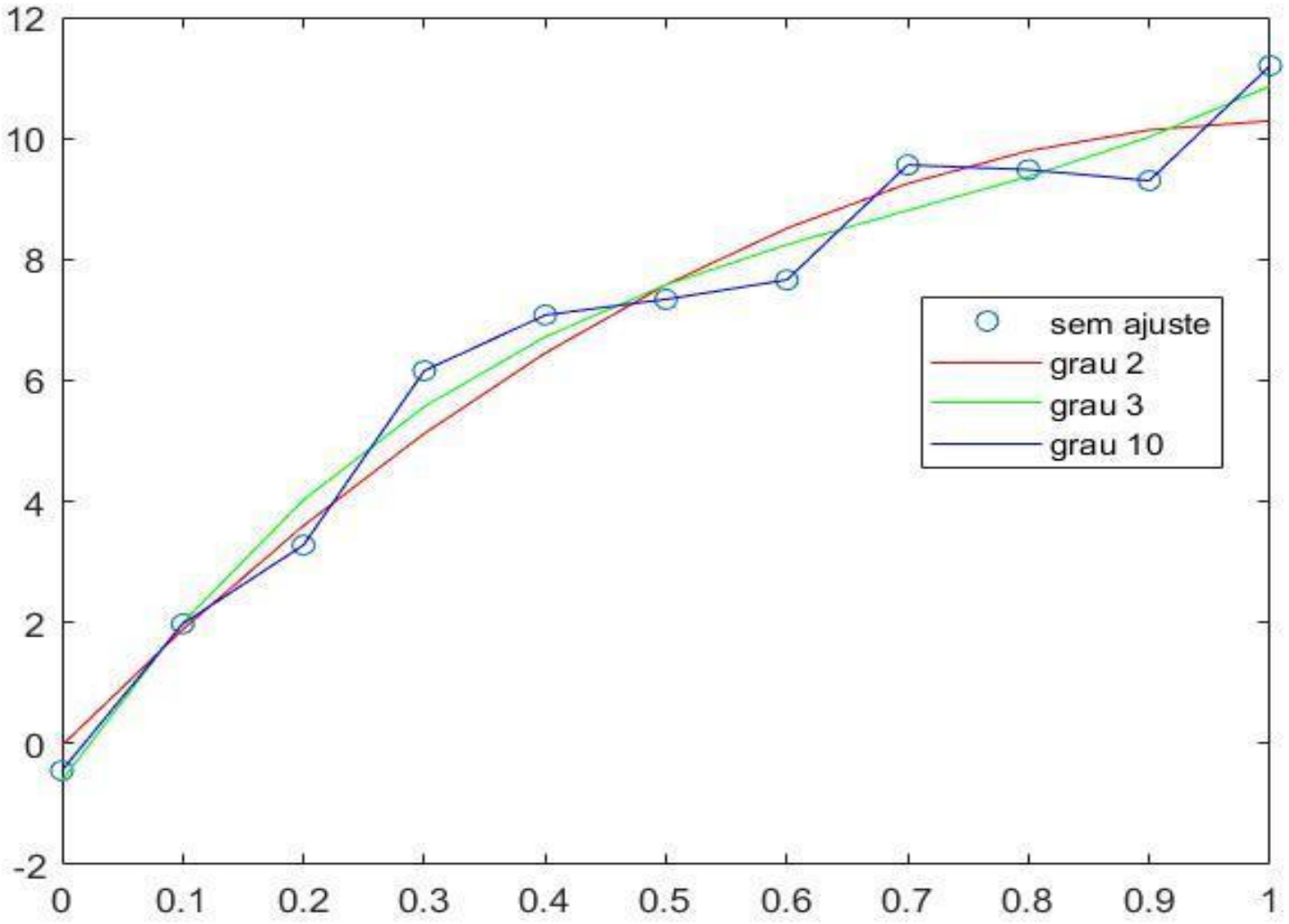
Ajuste

polyfit(*x*, *y*, *n*) retorna os coeficientes do polinômio ajustado de grau n nos valos de x e y;

Ajuste


```
>> x=0:0.1:1; y = [-0.447 1.978 3.28
6.16 7.08 7.34 7.66 9.56 9.48 9.30
11.2];
>>pp2 = polyfit(x,y,2); pp3 =
polyfit(x,y,3); pp10 = polyfit(x,y,10);
>>yp2 = polyval(pp2, x); yp3 =
polyval(pp3, x); yp10 = polyval(pp10,
x);
>>plot(x, y, 'o', x, yp2, 'r', x, yp3,
'g', x, yp10, 'b')
>> legend('sem ajuste', 'grau 2', 'grau
3', 'grau 10')
```

Ajuste


12
10
8
6
4
2
0
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
sem ajuste
grau 2
grau 3
grau 10

PETEE UFMG

Polinômios

Frações Parciais

Frações Parciais

$$\frac{B(s)}{A(s)} = \frac{10s^3 + 80s^2 + 177s + 95}{s^3 + 8s^2 + 19s + 12}$$

$$\frac{B(s)}{A(s)} = \frac{9}{s+4} + \frac{-7}{s+3} + \frac{-2}{s+1} + 10$$

Frações Parciais

`[r, p, k] = residue(n, d)` sendo n e d, o numerador e denominador, respectivamente, a função retorna r=resíduo, p=polo, k=termo direto;

`[n, d] = residue(r, p, k)` sendo r=resíduo, p=polo, k=termo direto, a função retorna n e d, o numerador e denominador, respectivamente;

Frações Parciais


```
>>B=[10 80 177 95]; A=[1 8 19 12];
>> [res, pol, k]= residue(B, A)
res =
    9.0000
   -7.0000
   -2.0000
pol =
   -4.0000
   -3.0000
   -1.0000
k =
    10
```

$$\frac{B(s)}{A(s)} = \frac{9}{s+4} + \frac{-7}{s+3} + \frac{-2}{s+1} + 10$$

Exemplo

Suponha o polinômio:

$$3x^3 + x + 2$$

É descrito pelo vetor

3 0 1 2

Crie-o

Exemplo

Suponha o polinômio:

$$3x^3 + x + 2$$

É descrito pelo vetor

3 0 1 2

Crie-o

Para pensar: Como plotar o gráfico desse polinômio?

Exemplo

$$3x^3 + x + 2$$

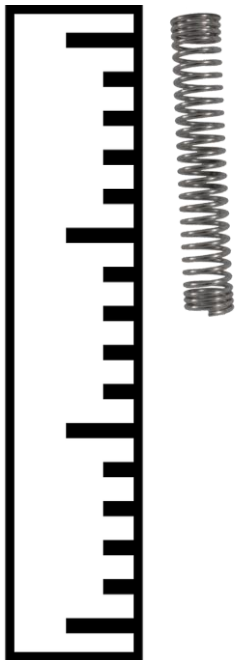
3 0 1 2

Para pensar: Como plotar o gráfico desse polinômio?

```
>> P = [3 0 1 2]

P =

     3     0     1     2

>> x = linspace(-3, 3); y = polyval(P, x);
>> plot(x, y)
```

Exercício

Em um determinado experimento de física, deseja-se medir a constante elástica k de uma certa mola de 10cm.



Seis unidades

0.5

Exercício

O tamanho da mola a cada quantidade de massa colocada foi medida, e foram montados os vetores

```
x = 0    0.5000    1.0000    1.5000    2.0000
2.5000    3.0000
referente a massa em kg

y = 0.0980    0.1530    0.1990    0.2610    0.2980
0.3500    0.3900 referente ao tamanho da mola em m.
```

Criem os vetores x e y em seu workspace.

Obs: x = 0:0.5:3;

Exercício

Sabendo que a força elástica de uma mola é dada segundo a equação $F = kx$, e que a força é o peso $P = mg$, encontre a constante k da mola. Use g = 9.8m/s^2.

Dicas:

1 - O que x e y representa?

2 - Compare a equação da força elástica com a função $y = ax + b$.

3 - Quais valores do ajuste considerar?

Exercício

$$massa = a * deformação + b$$

$$mg = kx$$

$$m = \frac{k}{g}x$$

$$m = a\,x + b, a = \frac{k}{g}$$

Exercício

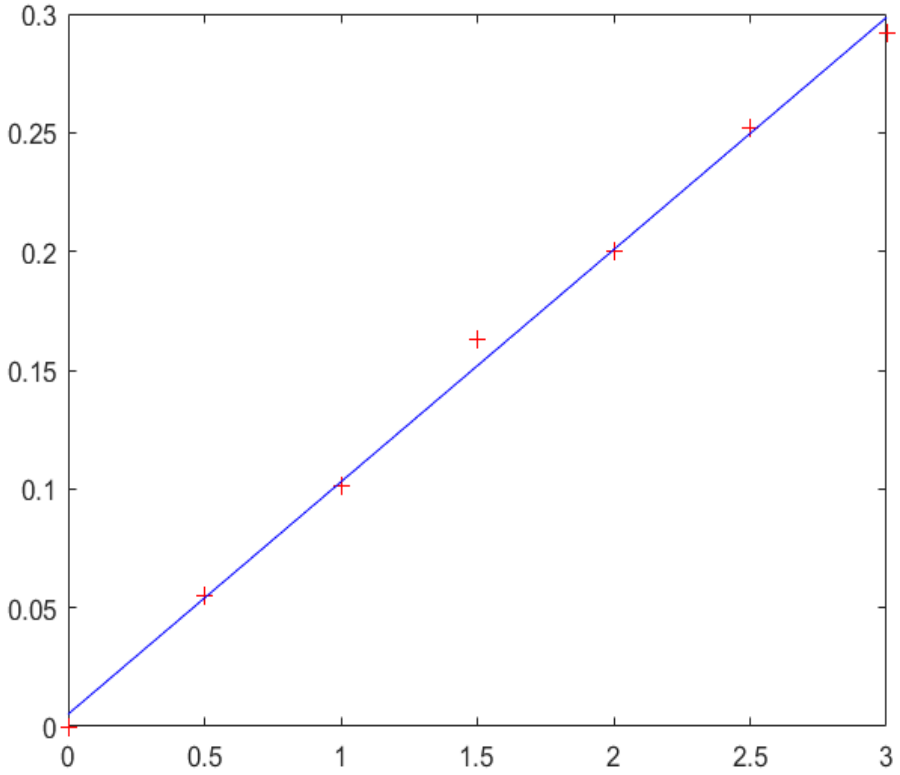

```
>>  x  =  0:0.5:3;    y  =  [0.0980
0.1530          0.1990          0.2610
0.2980    0.3500    0.3900];
>> polyfit(x, y, 1)
ans =
    0.0978    0.1032
>> 9.8/ans(1)
ans =
    100.2191
```

Exercício

Plote em uma mesma imagem os pontos do experimento e a função linear devidamente ajustada. Observe o resultado.

Exercício


```
>> a = polyfit(x,
y, 1);
>> y2 =
polyval(a, x);
>> plot(x, y,
'r+', x, y2, 'b')
```



PETEE UFMG

Simulink: Introdução

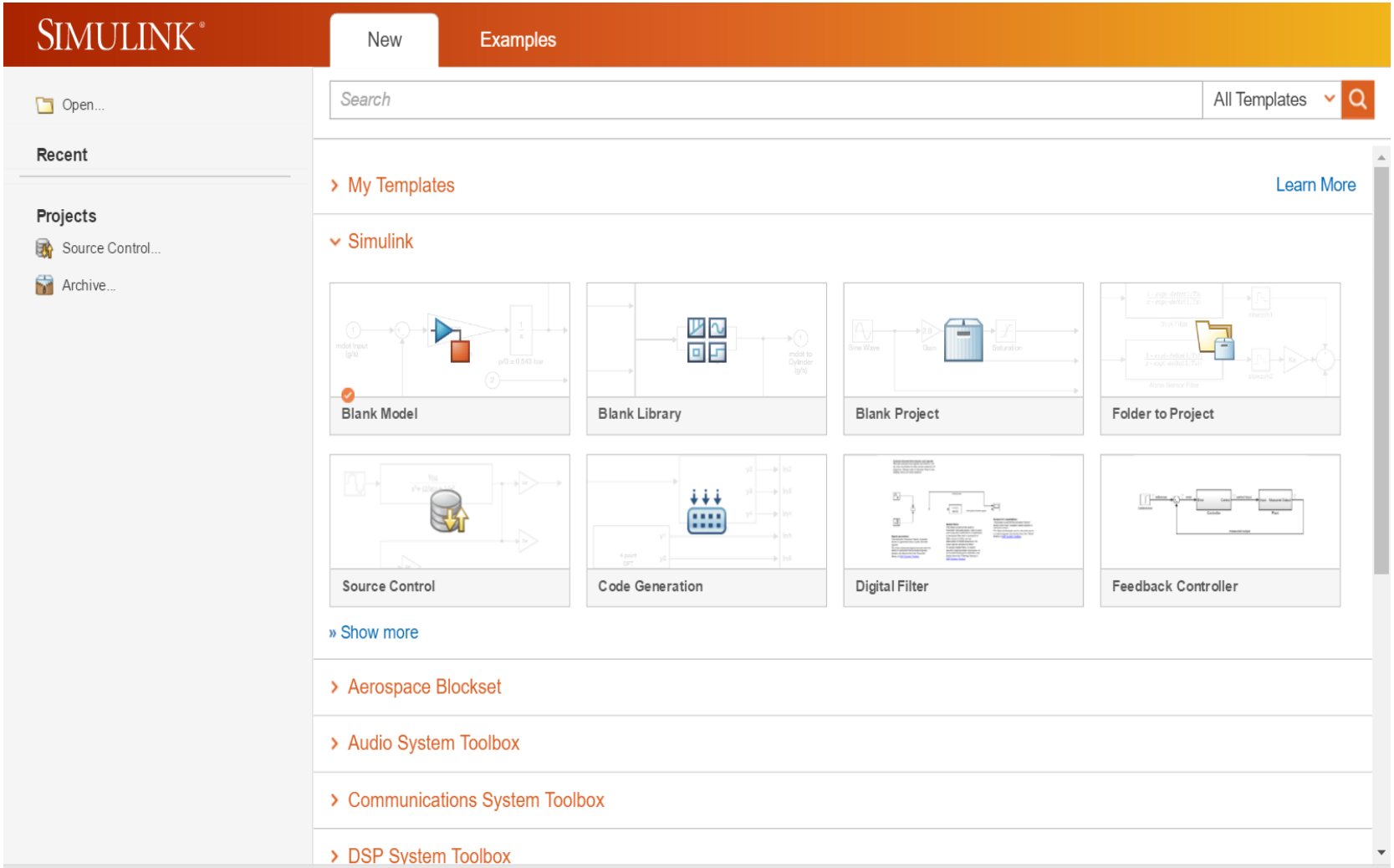
Introdução

Modelagem, simulações, sistemas.

“simulink” na Command Window ou clicar no ícone.

Interface Gráfica do Usuário

Ambiente


SIMULINK
New
Examples
Open...
Recent
Projects
Source Control...
Archive...
Search
All Templates
My Templates
Learn More
Simulink
Blank Model
Blank Library
Blank Project
Folder to Project
Source Control
Code Generation
Digital Filter
Feedback Controller
Show more
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
DSP System Toolbox

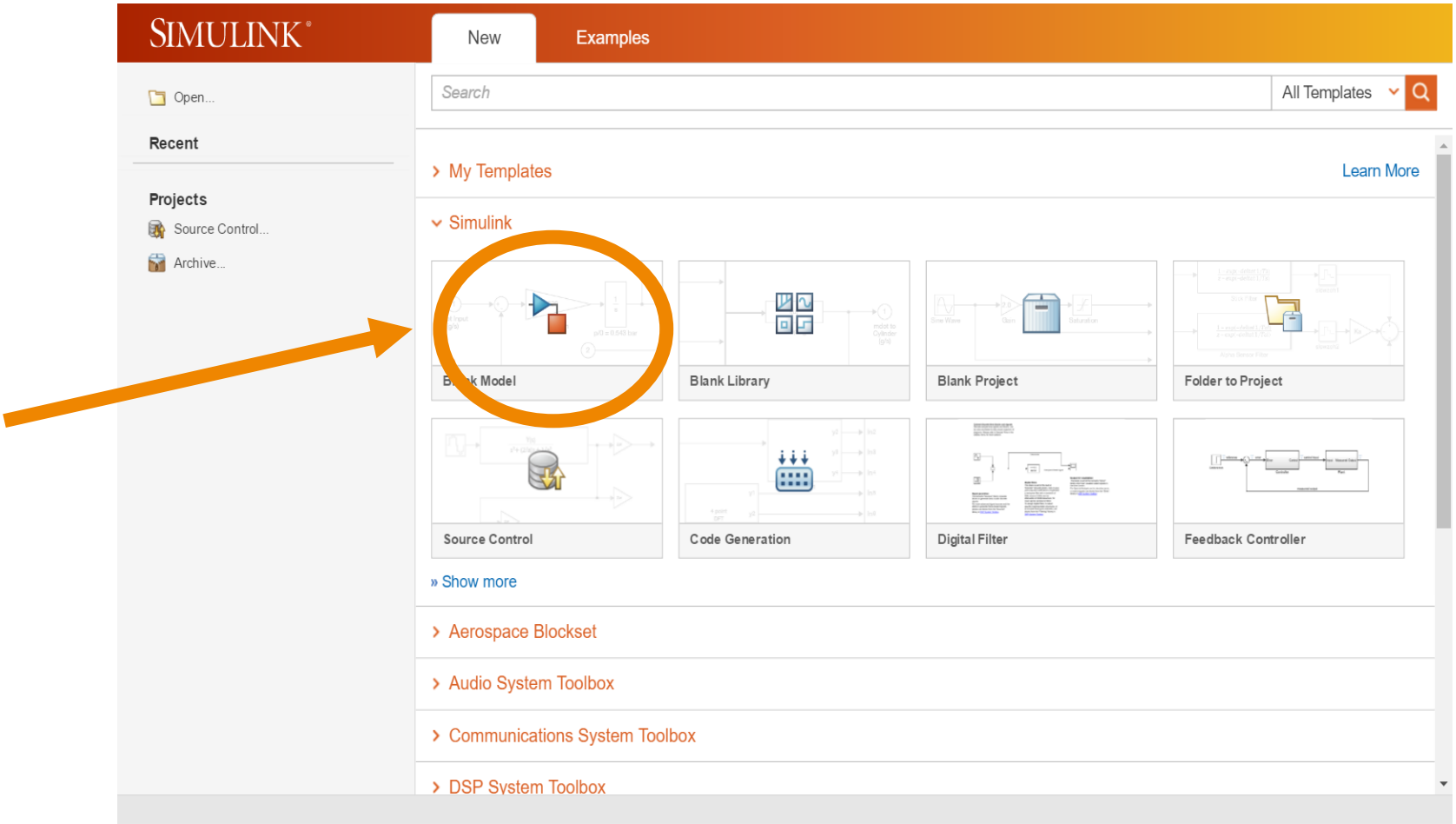
PETEE UFMG

Simulink: Introdução

Modelagem

Modelagem

Modelagem

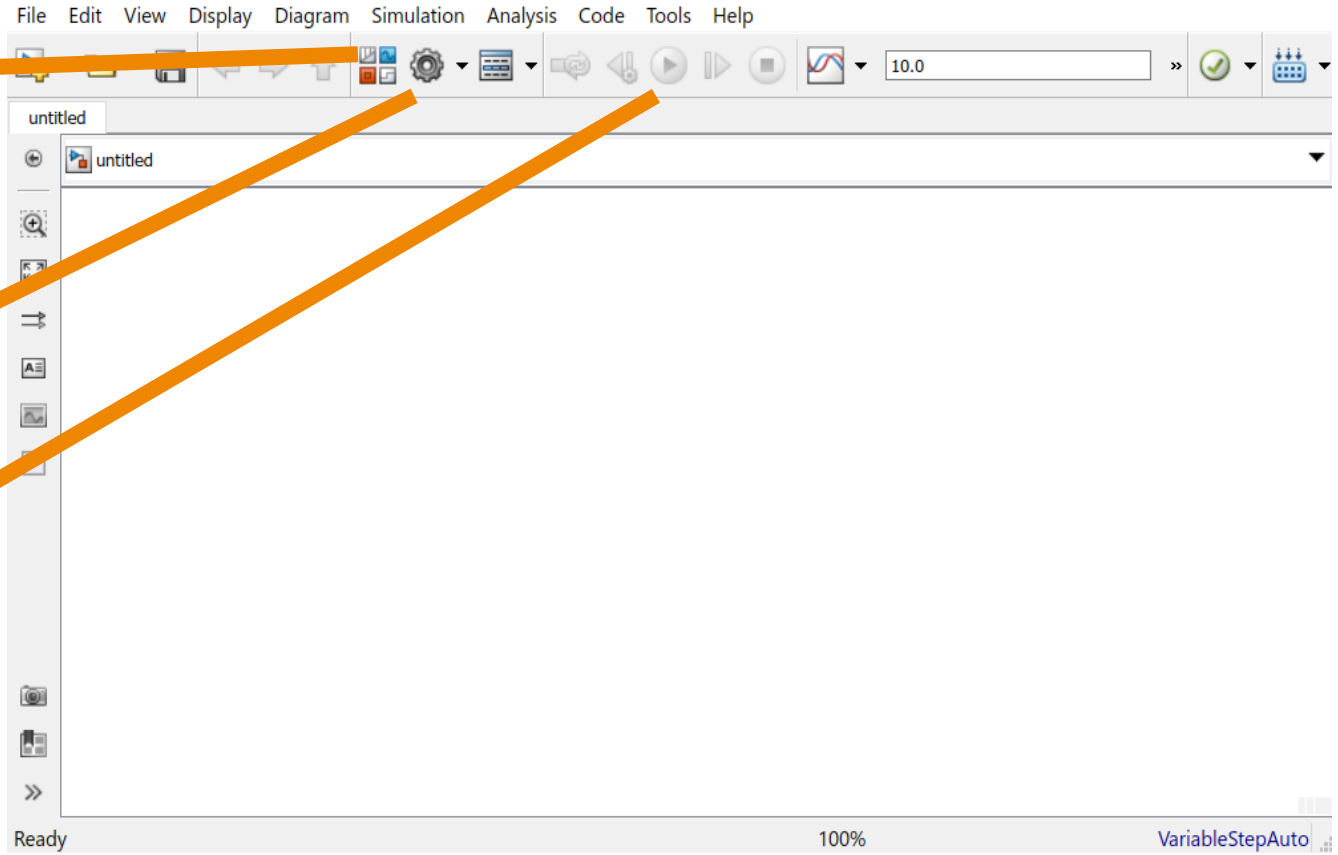

SIMULINK
New
Examples
Open...
Recent
Projects
Source Control...
Archive...
Search
All Templates
My Templates
Learn More
Simulink
Blank Model
Blank Library
Blank Project
Folder to Project
Source Control
Code Generation
Digital Filter
Feedback Controller
Show more
Aerospace Blockset
Audio System Toolbox
Communications System Toolbox
DSP System Toolbox

Modelagem

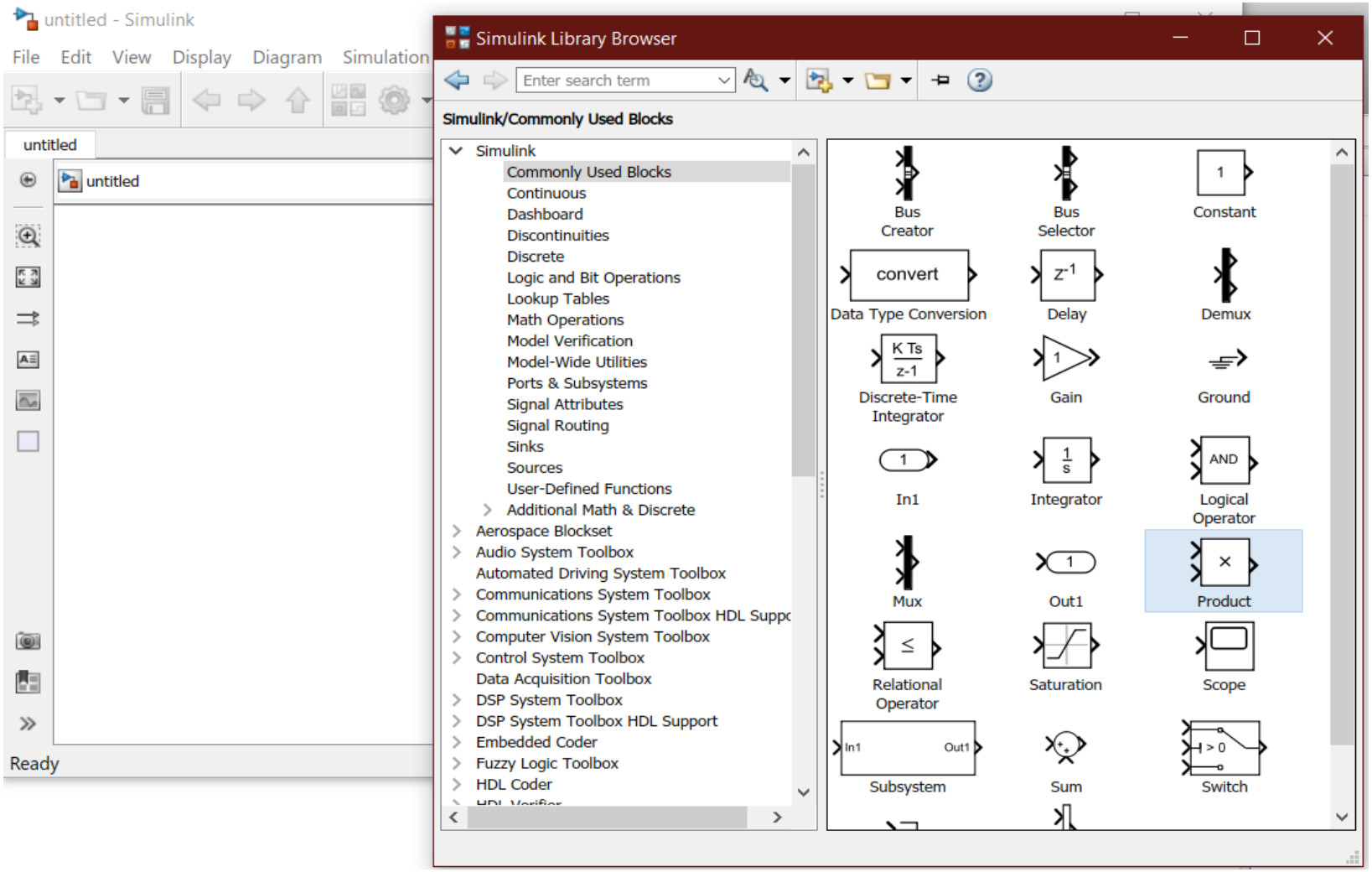
Library
Browser

Model
Configuration
Parameters

Run



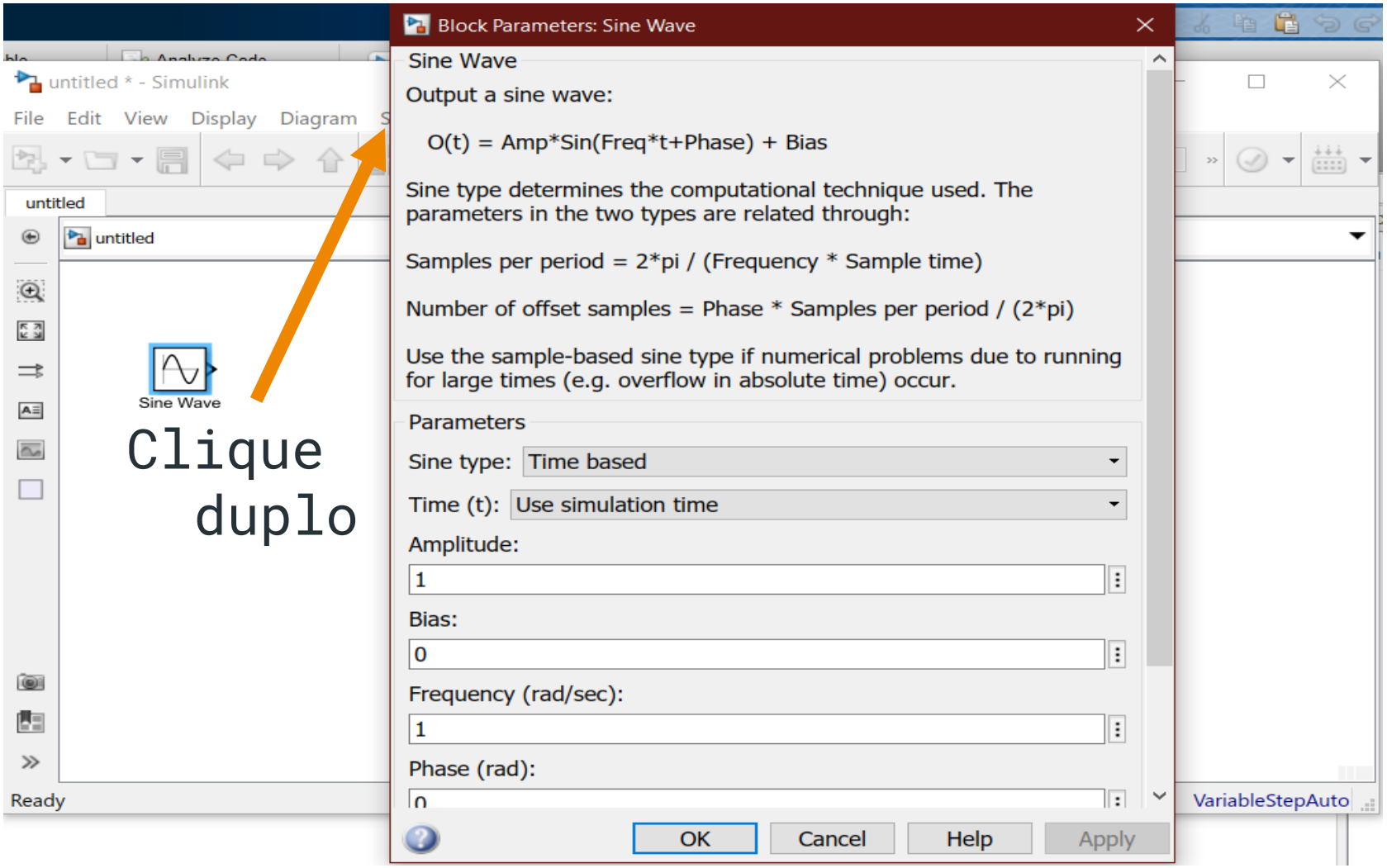
Library Browser



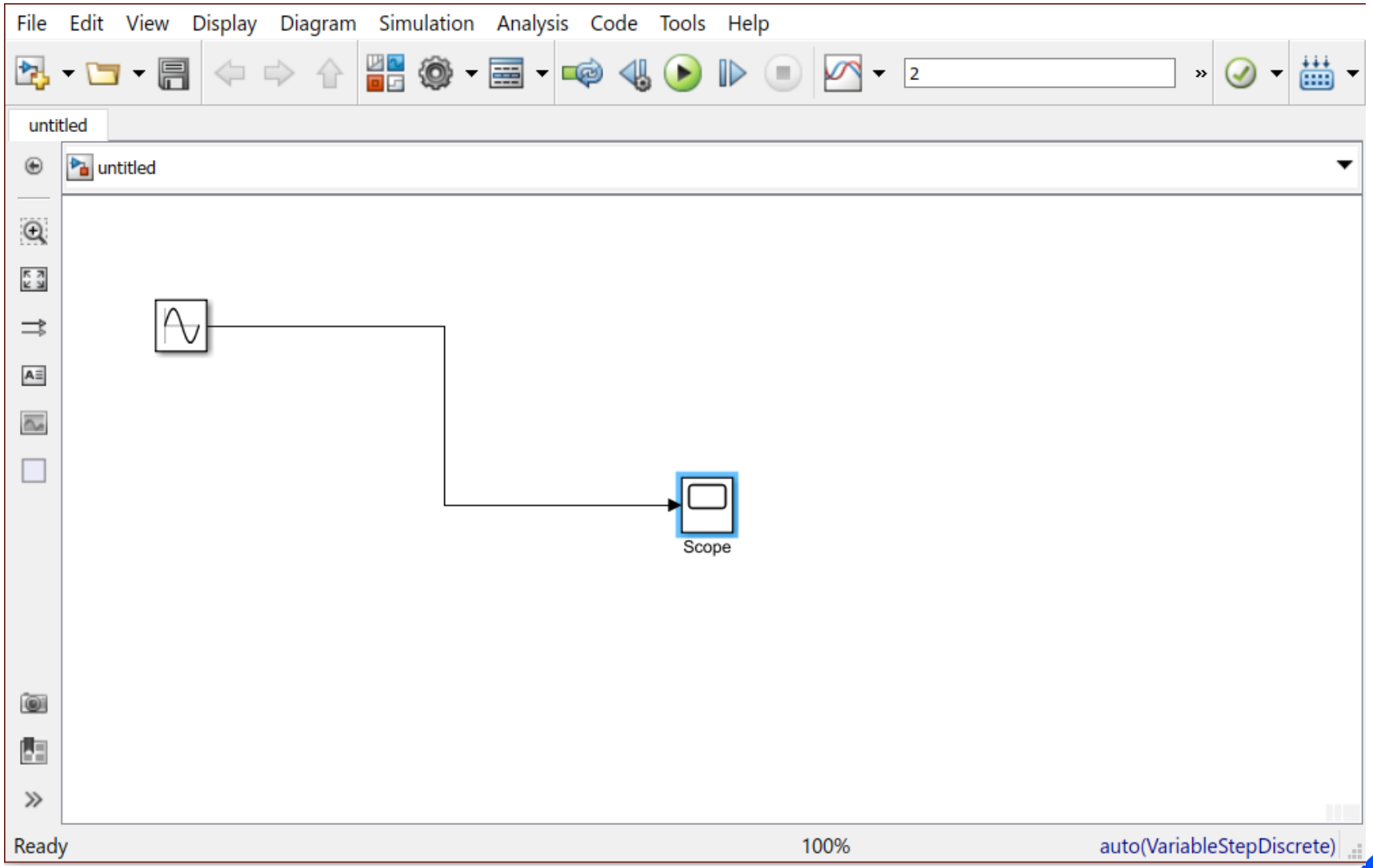
MINISTÉRIO DA EDUCAÇÃO

UFMG

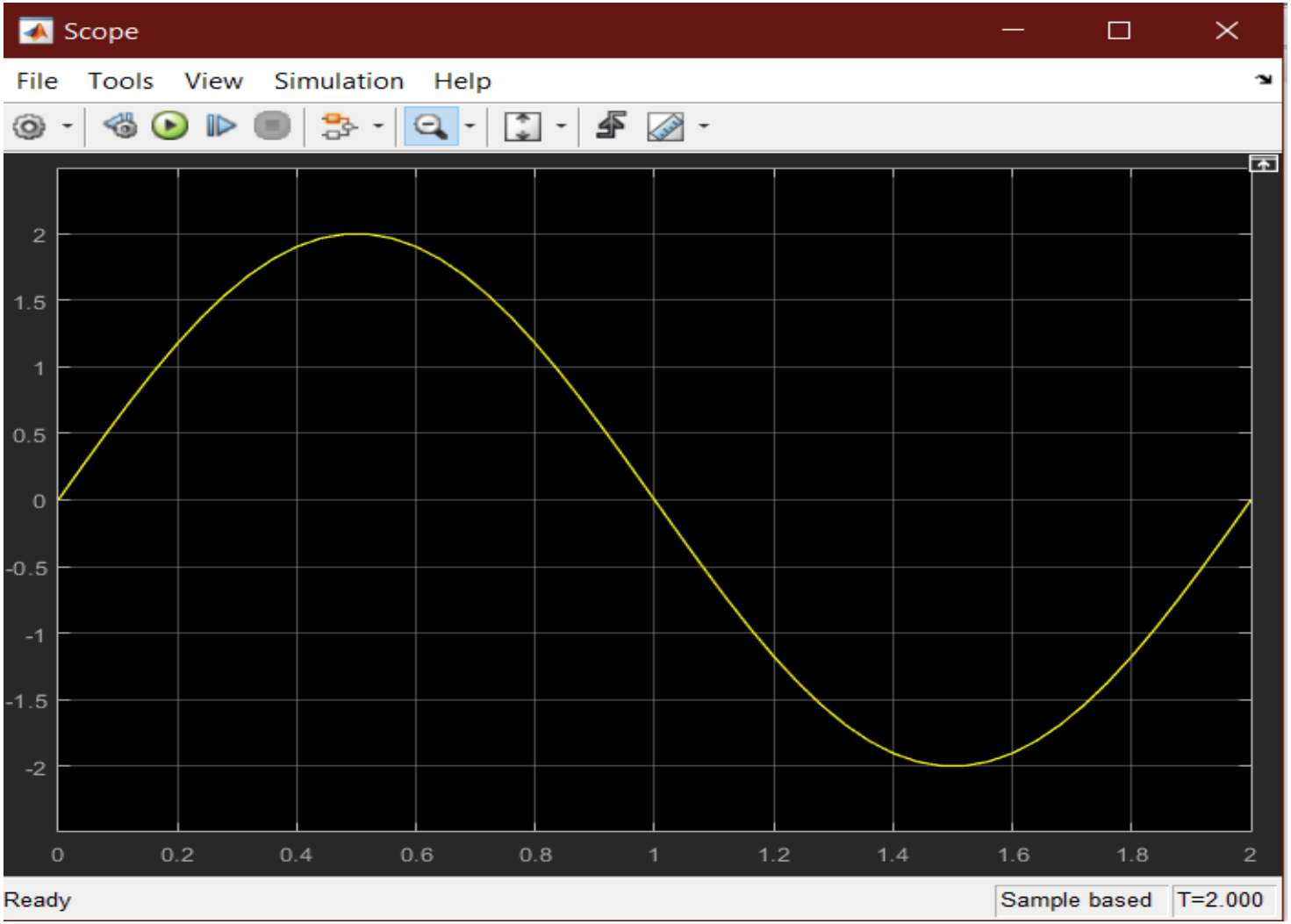
Exemplo


untitled * - Simulink
File Edit View Display Diagram
untitled
untitled
Sine Wave
Clique
duplo
Ready
VariableStepAuto
Block Parameters: Sine Wave
Sine Wave
Output a sine wave:
O(t) = Amp*Sin(Freq*t+Phase) + Bias
Sine type determines the computational technique used. The parameters in the two types are related through:
Samples per period = 2*pi / (Frequency * Sample time)
Number of offset samples = Phase * Samples per period / (2*pi)
Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.
Parameters
Sine type: Time based
Time (t): Use simulation time
Amplitude:
1
Bias:
0
Frequency (rad/sec):
1
Phase (rad):
0
OK
Cancel
Help
Apply

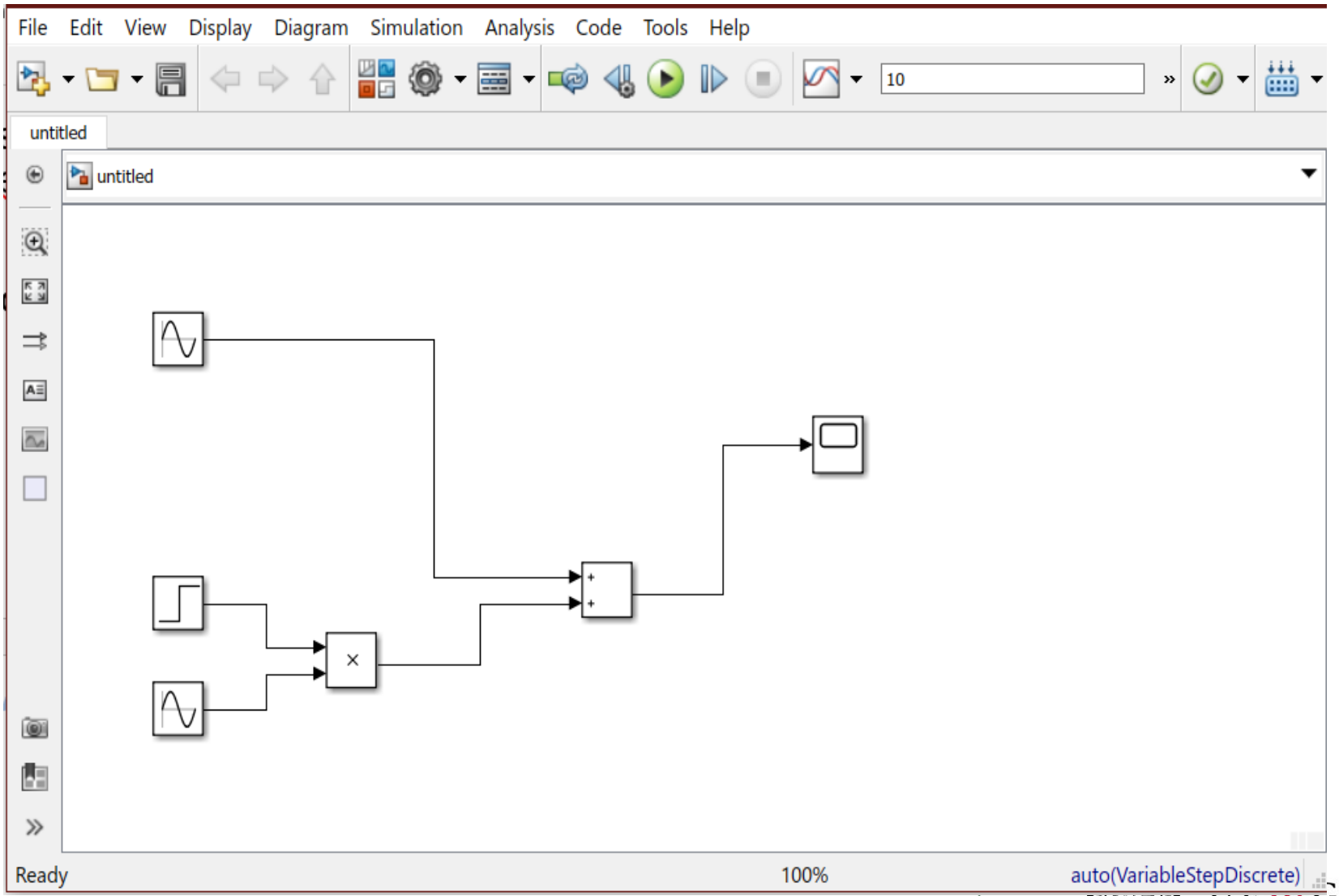
Exemplo


File Edit View Display Diagram Simulation Analysis Code Tools Help
2
untitled
untitled
Scope
Ready
100%
auto(VariableStepDiscrete)

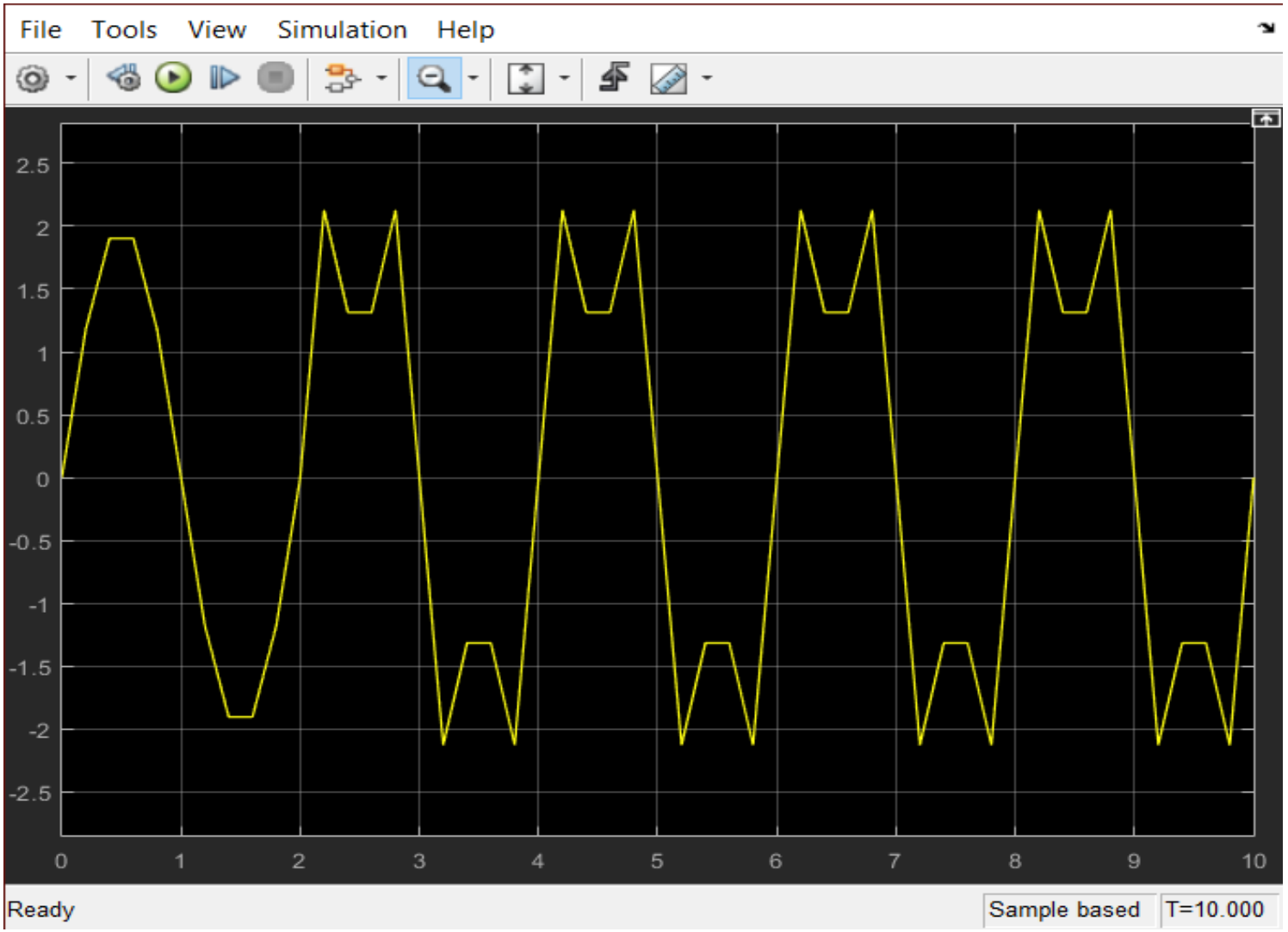
Exemplo


Scope
File Tools View Simulation Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Ready
Sample based
T=2.000

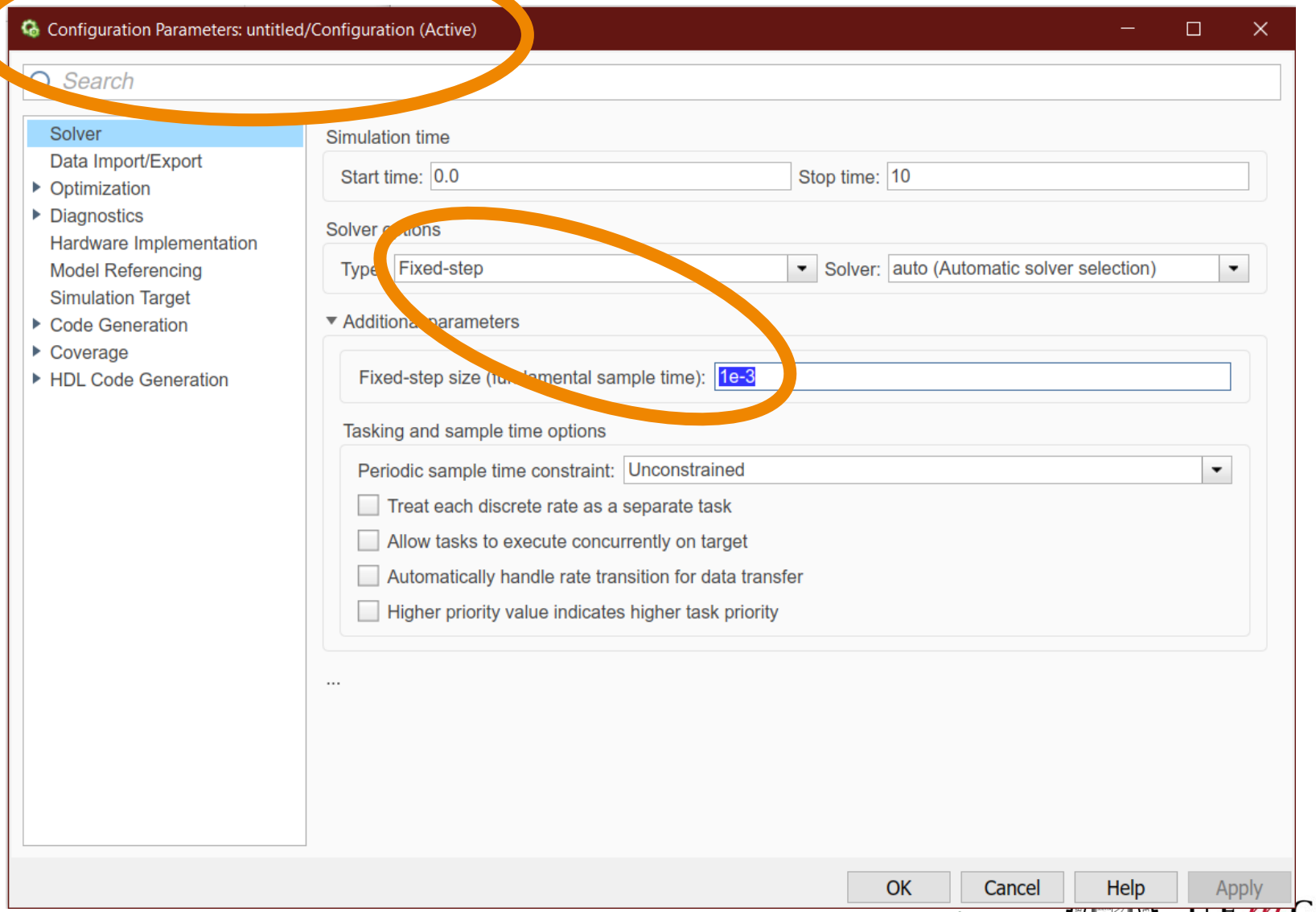
Exemplo


File Edit View Display Diagram Simulation Analysis Code Tools Help
10
untitled
untitled
Ready
100%
auto(VariableStepDiscrete)

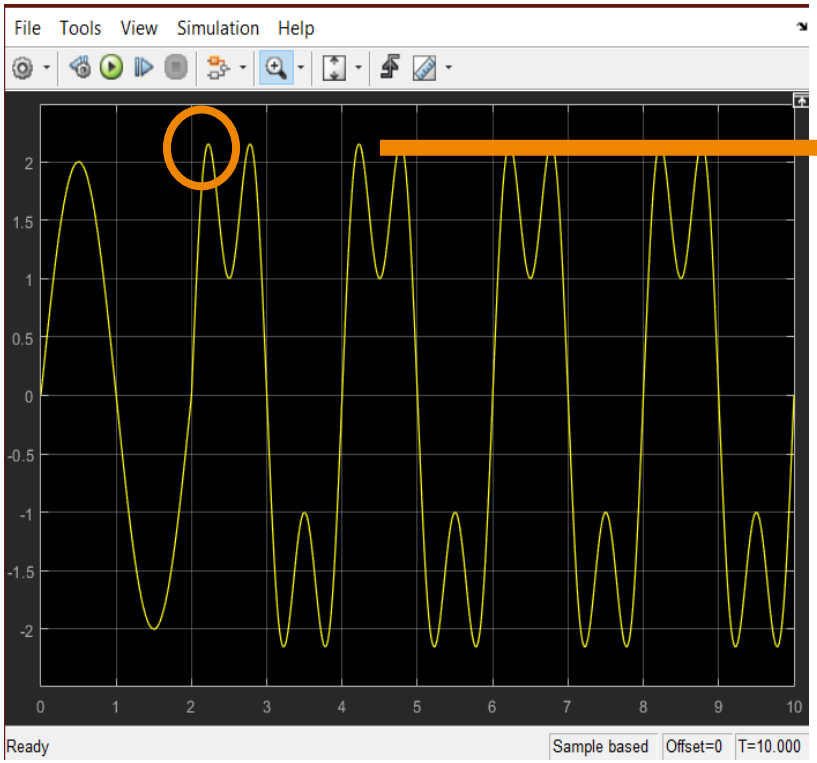
Exemplo

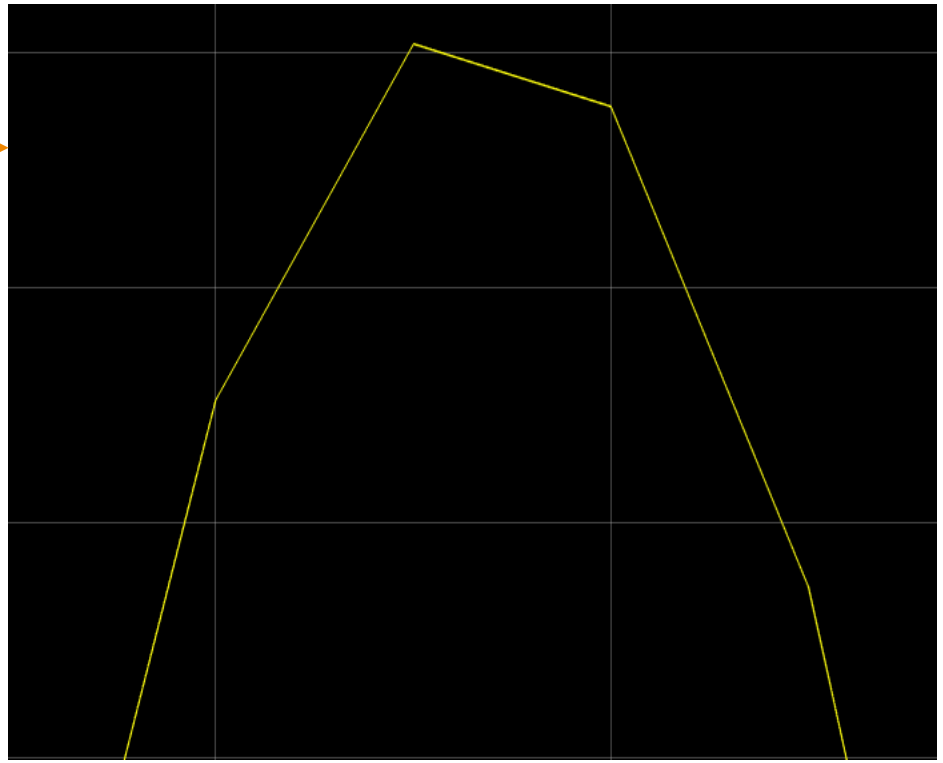

File Tools View Simulation Help
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000

Exemplo

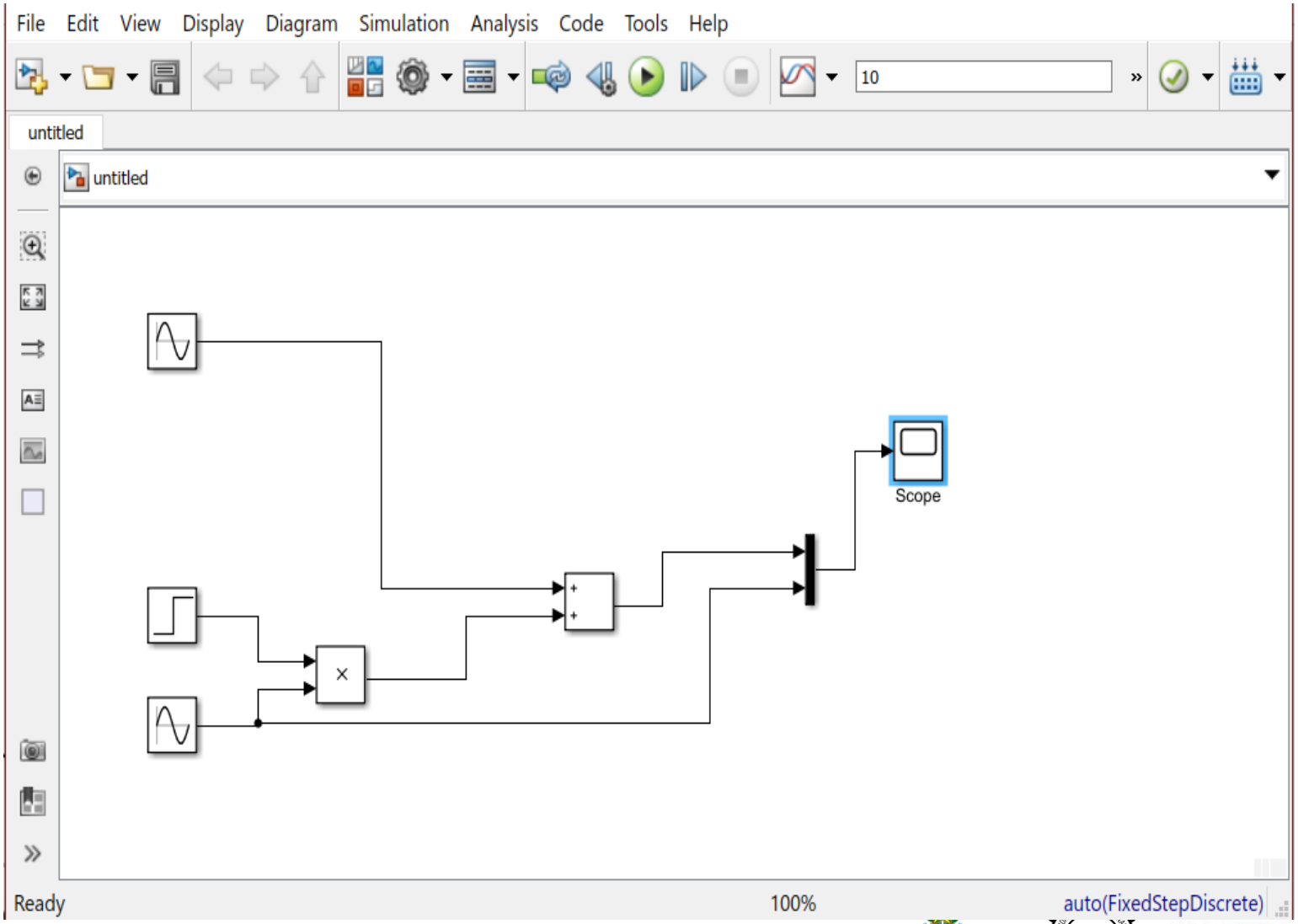

Configuration Parameters: untitled/Configuration (Active)
Search
Solver
Data Import/Export
Optimization
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Coverage
HDL Code Generation
Simulation time
Start time: 0.0
Stop time: 10
Type: Fixed-step
Solver: auto (Automatic solver selection)
Fixed-step size (fundamental sample time): 1e-3
Tasking and sample time options
Periodic sample time constraint: Unconstrained
Treat each discrete rate as a separate task
Allow tasks to execute concurrently on target
Automatically handle rate transition for data transfer
Higher priority value indicates higher task priority
OK
Cancel
Help
Apply

Exemplo

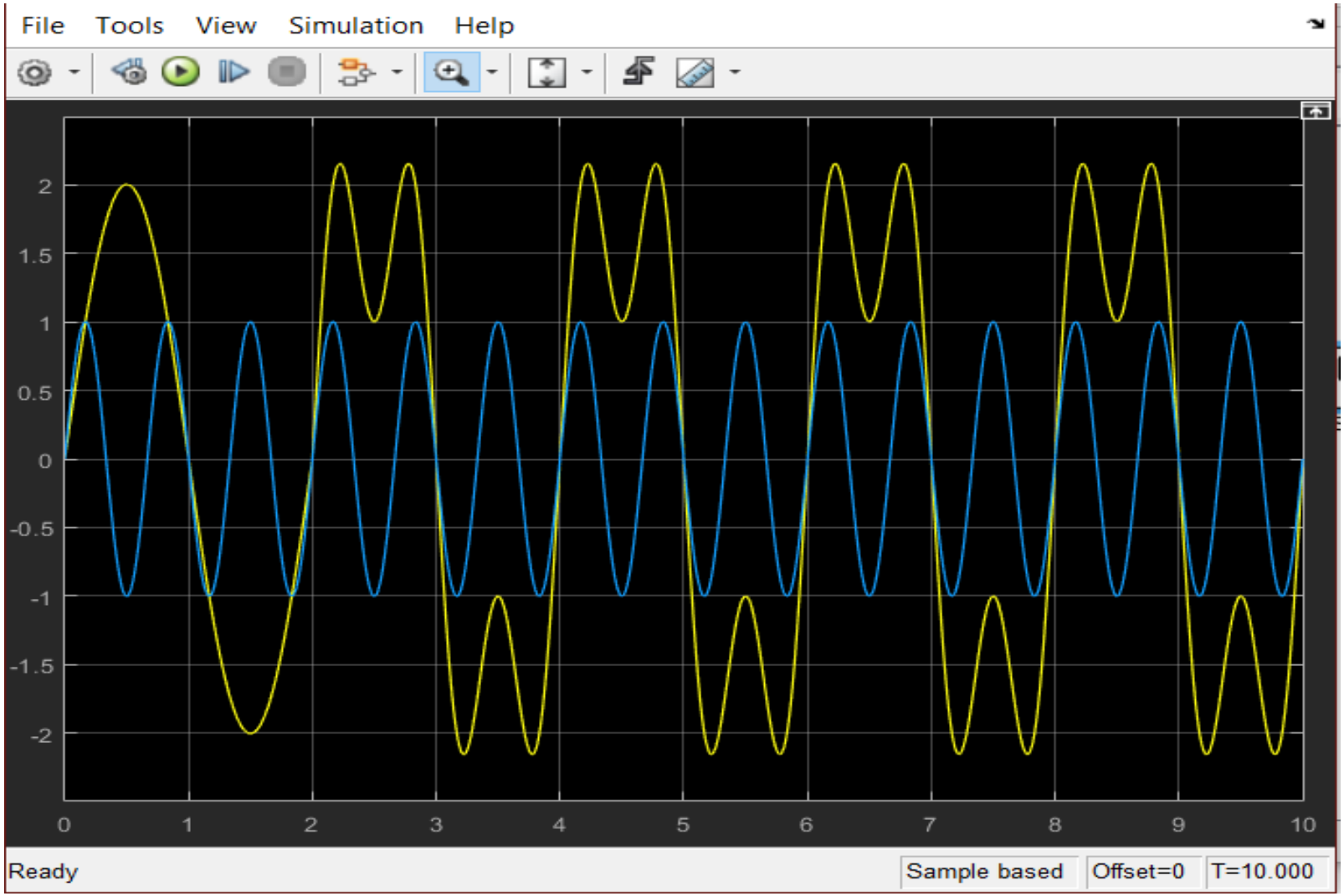

File Tools View Simulation Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
Offset=0
T=10.000



Exemplo


File Edit View Display Diagram Simulation Analysis Code Tools Help
10
untitled
untitled
Scope
Ready
100%
auto(FixedStepDiscrete)

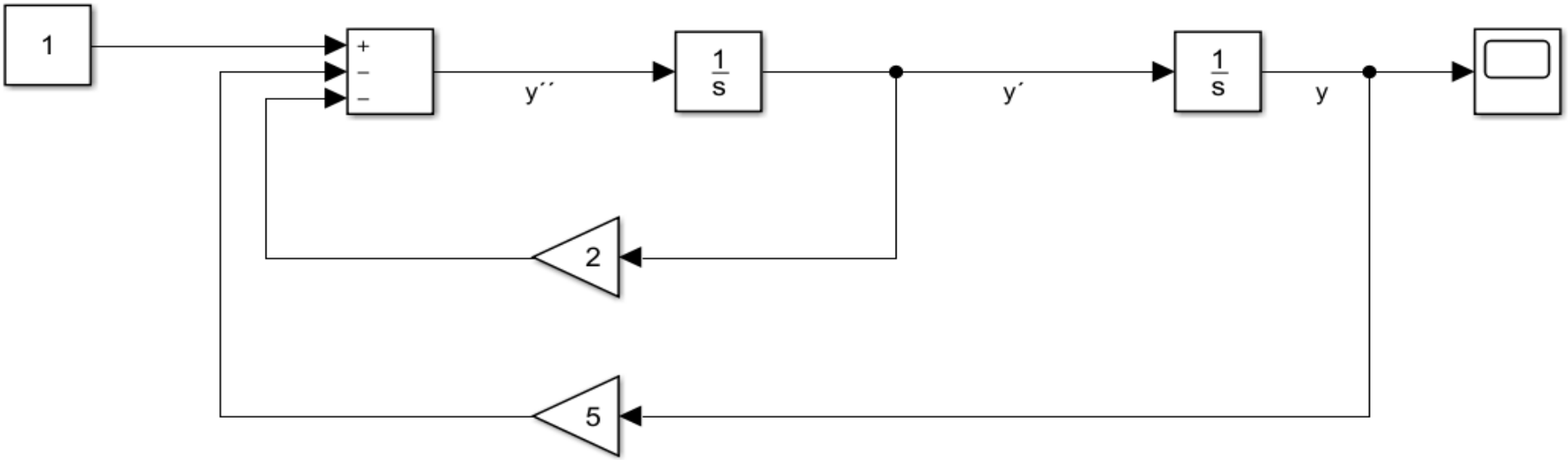
Exemplo

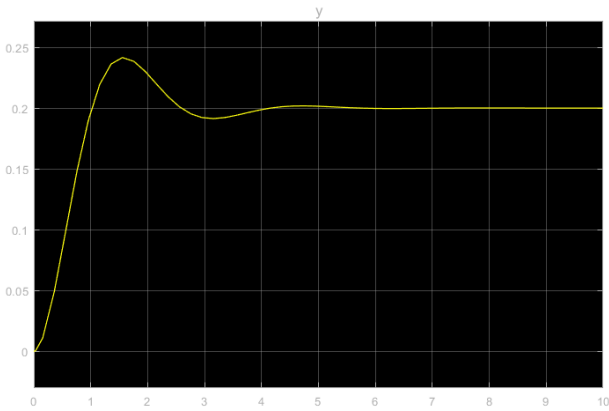

File Tools View Simulation Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
Offset=0
T=10.000

Exemplo 2

$$y'' + 2\,y' + 5y = 1$$

$$y'(0) = y(0) = 0$$

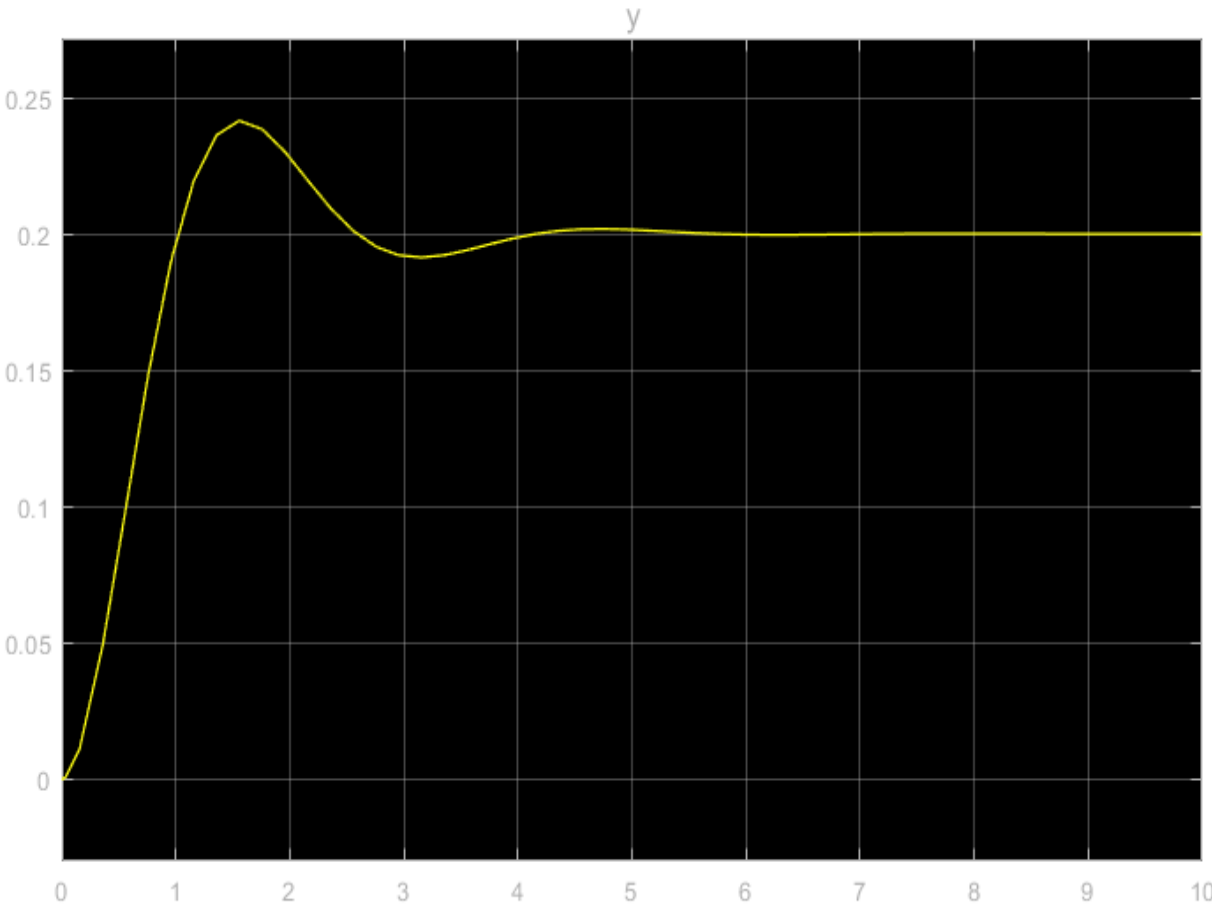




Exemplo 2

$$y'' + 2\,y' + 5y = 1$$

$$y'(0) = y(0) = 0$$

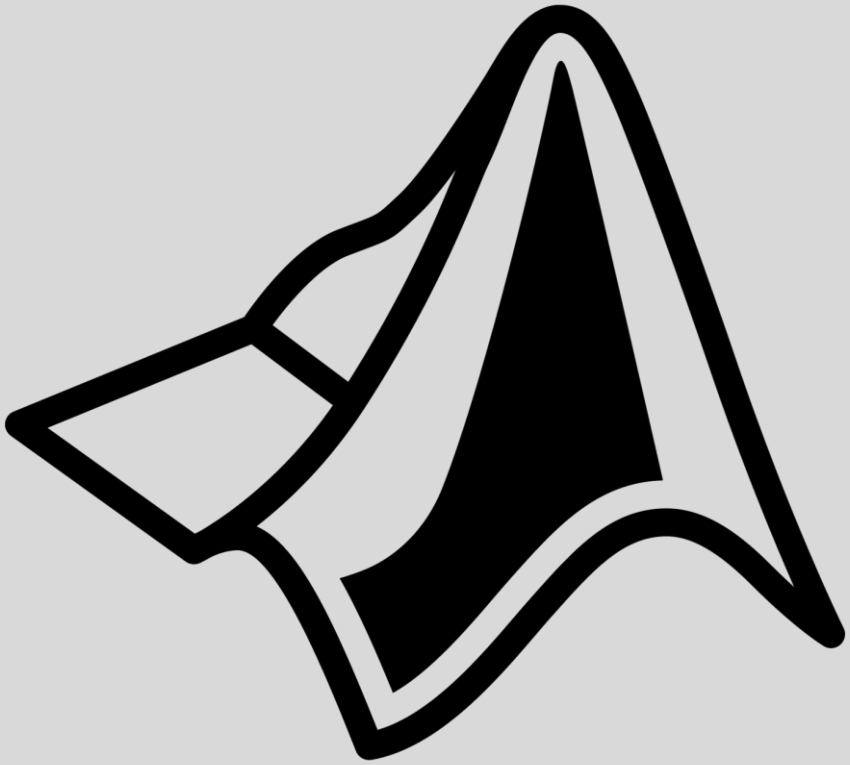


PETEE UFMG

UFMG

MINICURSO MATLAB 1.0

Slide-aula minicurso 1.0



PROGRAMA DE EDUCAÇÃO TUTORIAL – ENGENHARIA ELÉTRICA –
UNIVERSIDADE FEDERAL DE MINAS GERAIS