

INTRODUÇÃO À PROGRAMAÇÃO

Aula 4

Arquivos de script e o Editor/Debugador

As operações no MATLAB podem ser feitas de 2 formas:

1. Modo interativo – comandos inseridos diretamente na janela de comandos
2. Arquivo script - rodando um programa MATLAB armazenado em um arquivo script.

Arquivos de script e o Editor/Debugador

Arquivos script

- São editados e gravados em arquivos M (arquivos com extensão .m)
- O editor do MATLAB pode ser utilizado para escrever e salvar os arquivos M
- Os arquivos M podem também serem escritos em outros editores (Ex: Notepad++)

Arquivos de script e o Editor/Debugador

Criando e utilizando um arquivos script

- O símbolo % é utilizado para criar um comentário no MATLAB. Comentários são utilizados para documentar os arquivos.
- O símbolo de comentário pode ser inserido em qualquer lugar da linha de comando.
- O MATLAB desconsidera tudo que estiver à direita do símbolo %.

```
>> % Isto é um comentário
```

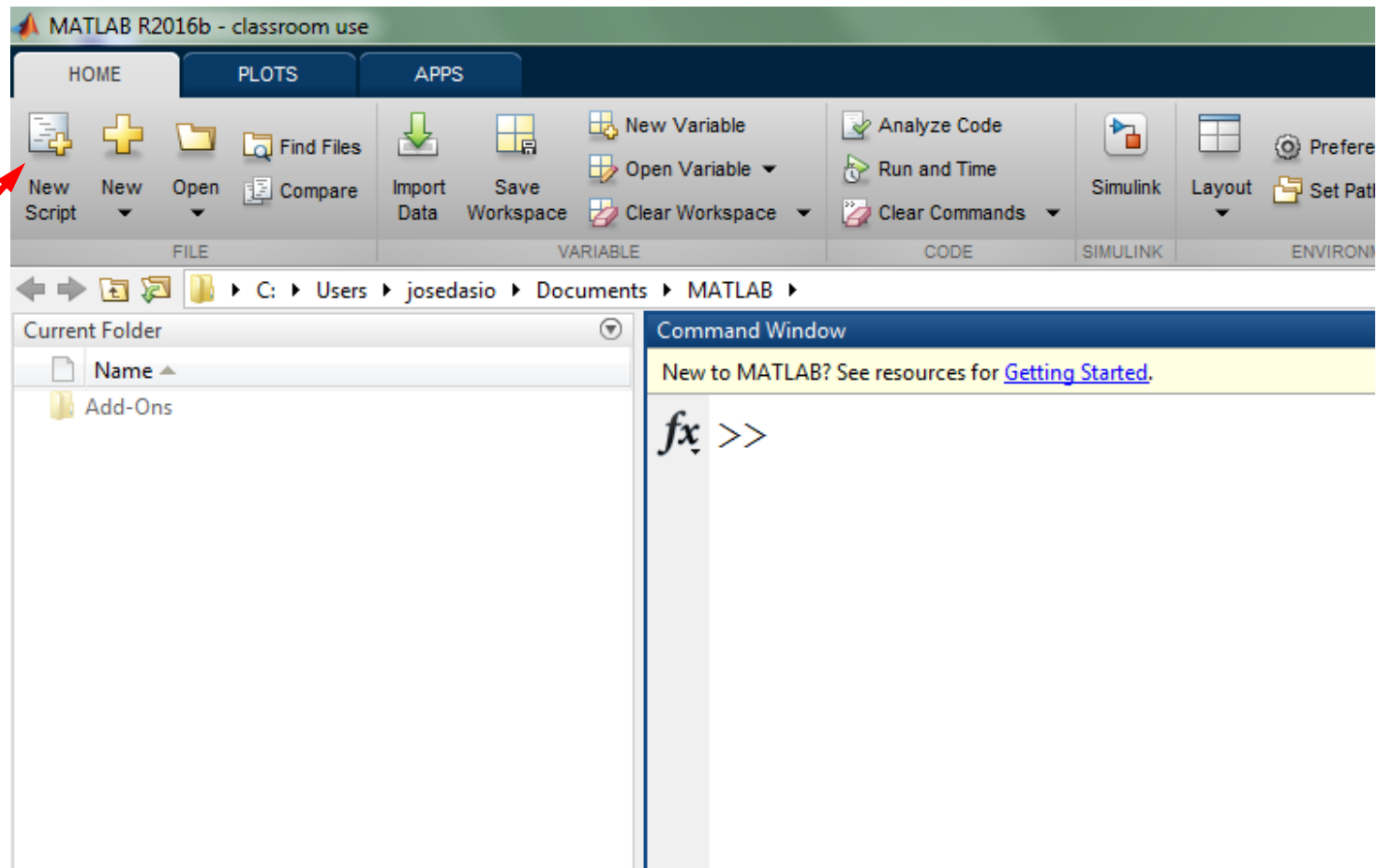
```
>> x = 2 + 3 % Isto também é
```

```
x =
```

```
5
```

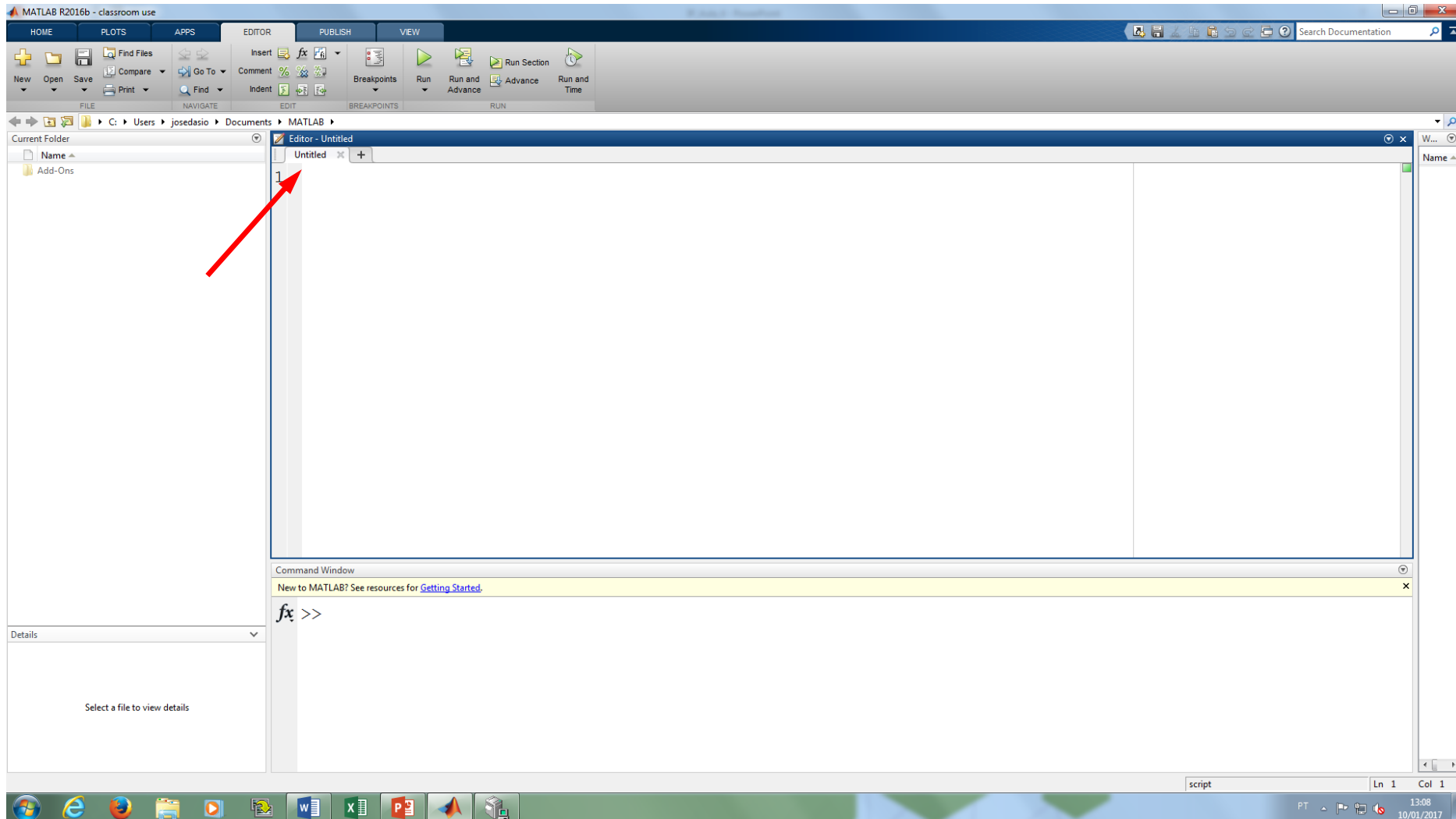
Arquivos de script e o Editor/Debugador

Criando e utilizando um arquivos script



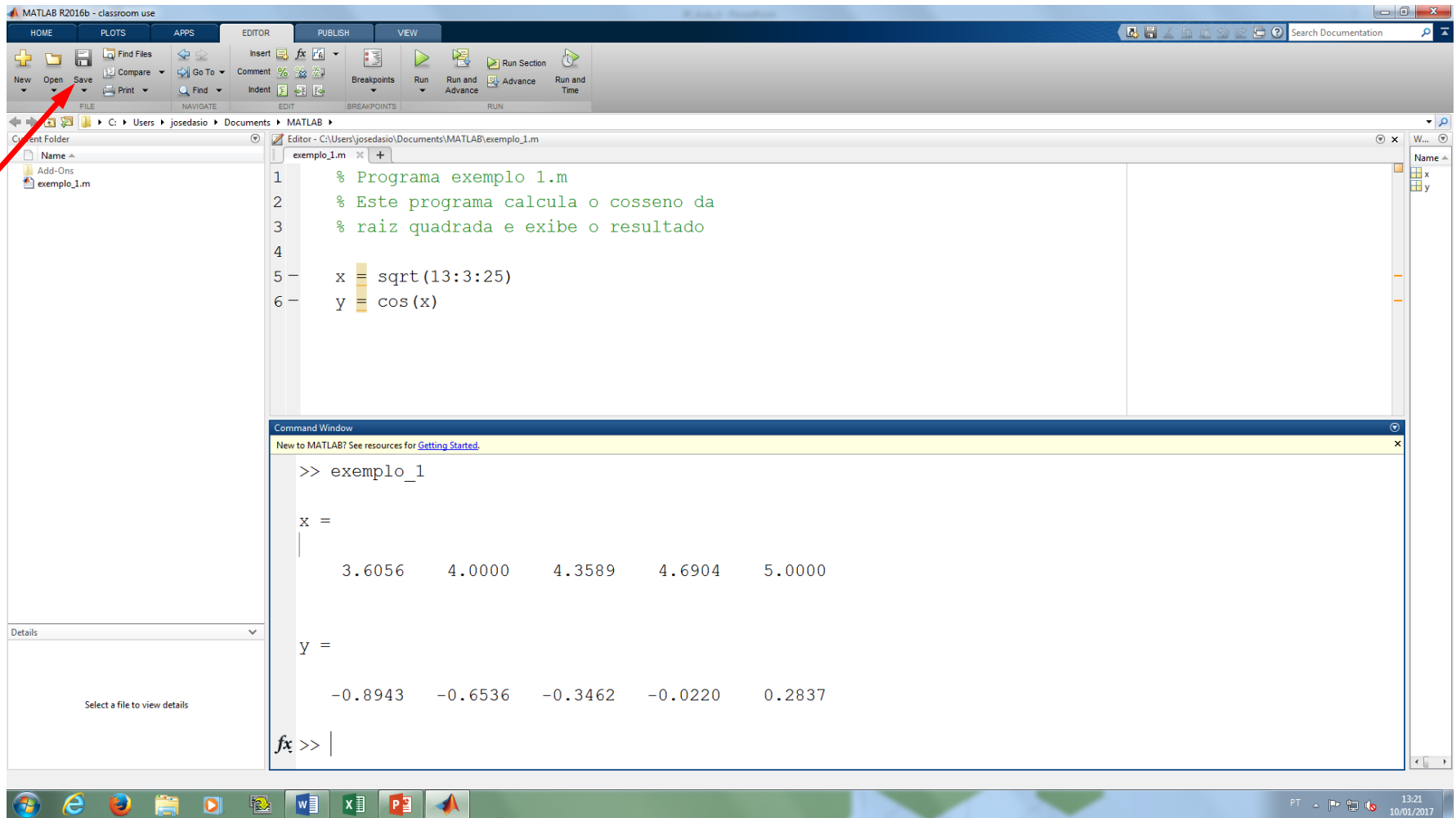
Arquivos de script e o Editor/Debugador

Criando e utilizando um arquivos script



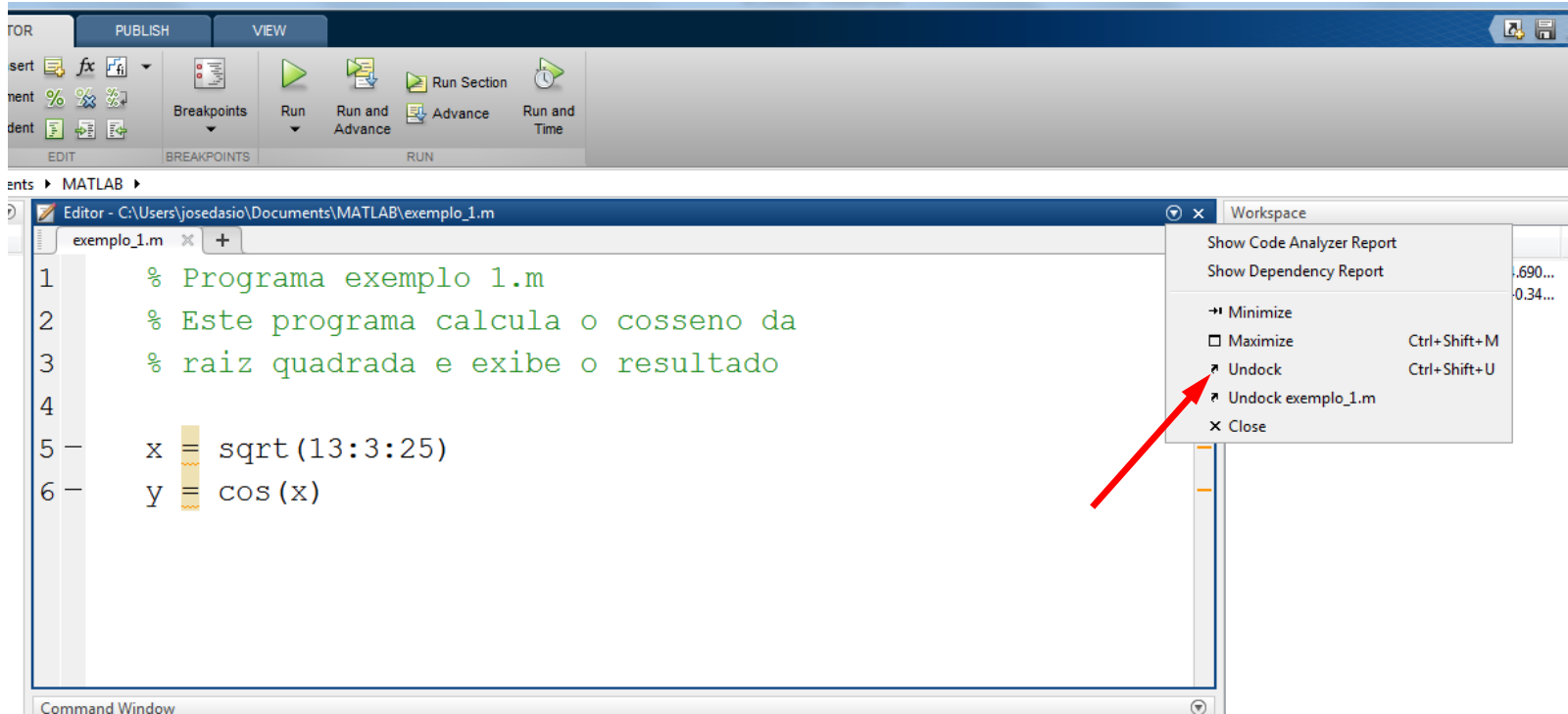
Arquivos de script e o Editor/Debugador

Criando e utilizando um arquivos script



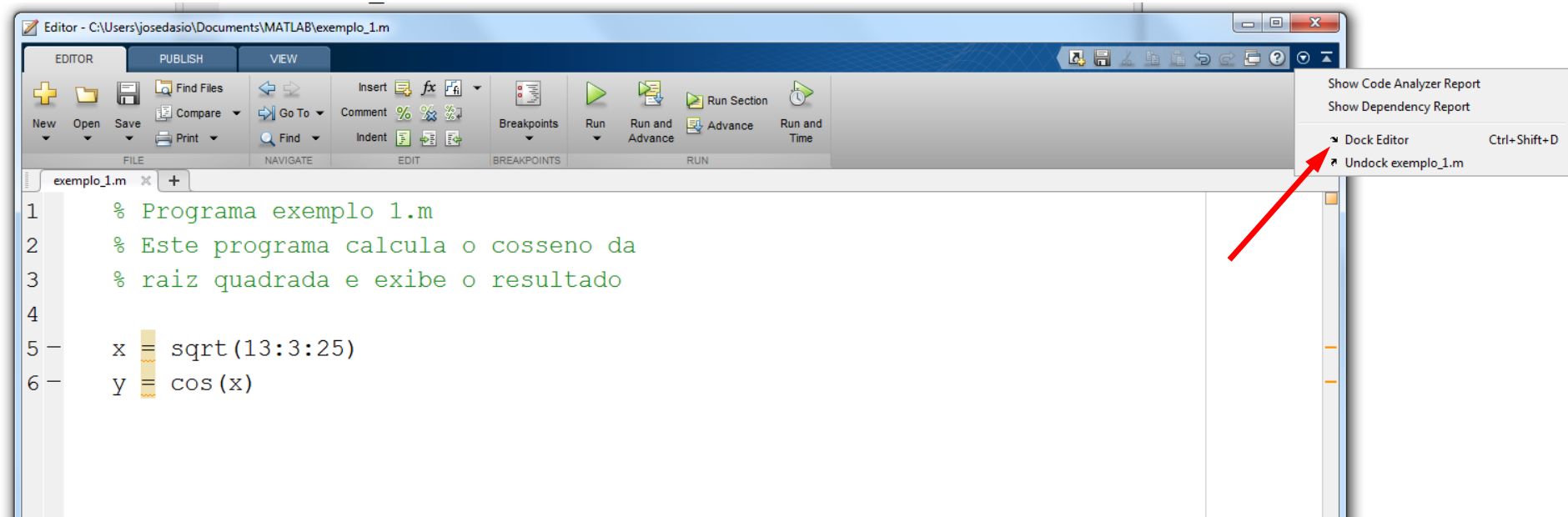
Arquivos de script e o Editor/Debugador

Criando e utilizando um arquivos script



Arquivos de script e o Editor/Debugador

Criando e utilizando um arquivos script



Arquivos de script e o Editor/Debugador

Nunca salvar script com o nome de funções ou comandos já existentes no MATLAB.

O comando `exist` verifica se uma determinada variável já existe, se o retorno da função for 0 a variável não existe, se o retorno for 1 a variável existe.

```
>> exist('ifpe')
```

```
ans =
```

```
0
```

```
>>
```

Arquivos de script e o Editor/Debugador

O comando `exist` também pode ser usado para verificar se um arquivo já existe, retorna 0 se não existir, ou 2 se o arquivo existir.

```
>> exist('exemplo_1')
```

```
ans =
```

```
2
```

```
>> exist('exemplo_1', 'file')
```

```
ans =
```

```
2
```

```
>>
```

Arquivos de script e o Editor/Debugador

Debugando arquivos de script

Debugar um programa é o processo de encontrar e remover os erros ou “bugs”

1. Erros de Sintaxe – Omissão de parênteses, vírgulas, digitações incorretas.

O MATLAB detecta os erros e exibe uma mensagem descrevendo o erro e sua localização.

2. Erros em tempo de execução.

São erros devido a um procedimento matemático incorreto. Ex: divisão por zero.

Arquivos de script e o Editor/Debugador

Debugando arquivos de script

Debugar um programa é o processo de encontrar e remover os erros ou “bugs”

1. Erros de Sintaxe – Omissão de parênteses, vírgulas, digitações incorretas.

O MATLAB detecta os erros e exibe uma mensagem descrevendo o erro e sua localização.

2. Erros em tempo de execução.

São erros devido a um procedimento matemático incorreto. Ex: divisão por zero.

Arquivos de script e o Editor/Debugador

Encontrando os erros

1. Sempre realizar um teste com problemas simples, cuja resposta pode ser checada por cálculos a mão.
2. Exiba alguns cálculos intermediários removendo o sinal de ponto e vírgula no final das sentenças.
3. Utilizar as funcionalidades do Editor/Debugador, que serão introduzidas mais adiante.

Arquivos de script e o Editor/Debugador

Estilo de programação

1. Seção de comentários

1. Nome do programa e algumas palavras-chave na primeira linha.
2. Data de criação do programa e nome do criador na segunda linha.
3. As definições dos nomes para as variáveis de entrada e saída, criar uma subseção de entrada e outra subseção de saída. Pode ser criada também subseção para variáveis utilizadas nos cálculos.

Arquivos de script e o Editor/Debugador

Estilo de programação

1. Seção de Entrada

Nesta seção insira os dados de entrada e/ou funções de entrada.

2. Seção de cálculos

Insira os cálculos nesta seção. Inclua comentários quando apropriado.

3. Seção de saída

Insira as funções de saída. Inclua comentários quando apropriado.

Arquivos de script e o Editor/Debugador

Controlando Entradas e Saídas

disp(A) – Exibe o conteúdo, mas não o nome do Arranjo A.

```
>> A = 2
```

```
A =
```

```
2
```

```
>> disp(A)
```

```
2
```

Arquivos de script e o Editor/Debugador

Controlando Entradas e Saídas

disp('texto') → Exibe o texto entre aspas simples.

```
>> disp('A velocidade prevista é:')
```

A velocidade prevista é:

x = input('texto') → Exibe o texto entre aspas simples na tela, espera pela entrada do usuário a partir do teclado, e armazena o valor em x.

```
>> x = input('entre com o valor de x :')
```

entre com o valor de x :10

x =

10

Arquivos de script e o Editor/Debugador

Controlando Entradas e Saídas

`x = input('texto', 's')` → Exibe o texto entre aspas simples na tela, espera pela entrada.

do usuário a partir do teclado, e armazena a entrada como uma string em `x`.

```
>> Calendario = input('Entre com o dia da semana: ', 's')
```

```
Entre com o dia da semana: Quarta-feira
```

```
Calendario =
```

```
Quarta-feira
```

Arquivos de script e o Editor/Debugador

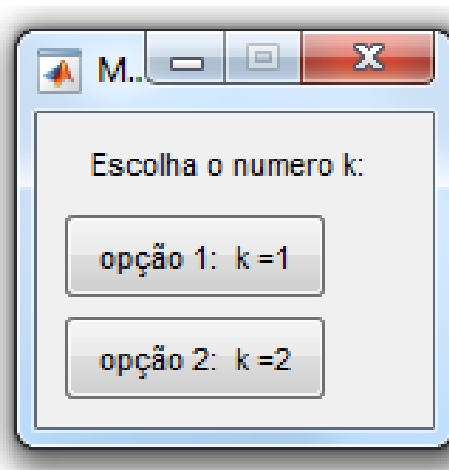
Controlando Entradas e Saídas

`k = menu('title', 'option1', 'option2',...)` → Exibe um menu cujo título é a variável string 'title' e cujas opções são 'option1', 'option2', e assim por diante.

```
>> k = menu('Escolha o numero k: ','opção 1: k =1','opção 2: k =2')
```

k =

1



Arquivos de script e o Editor/Debugador

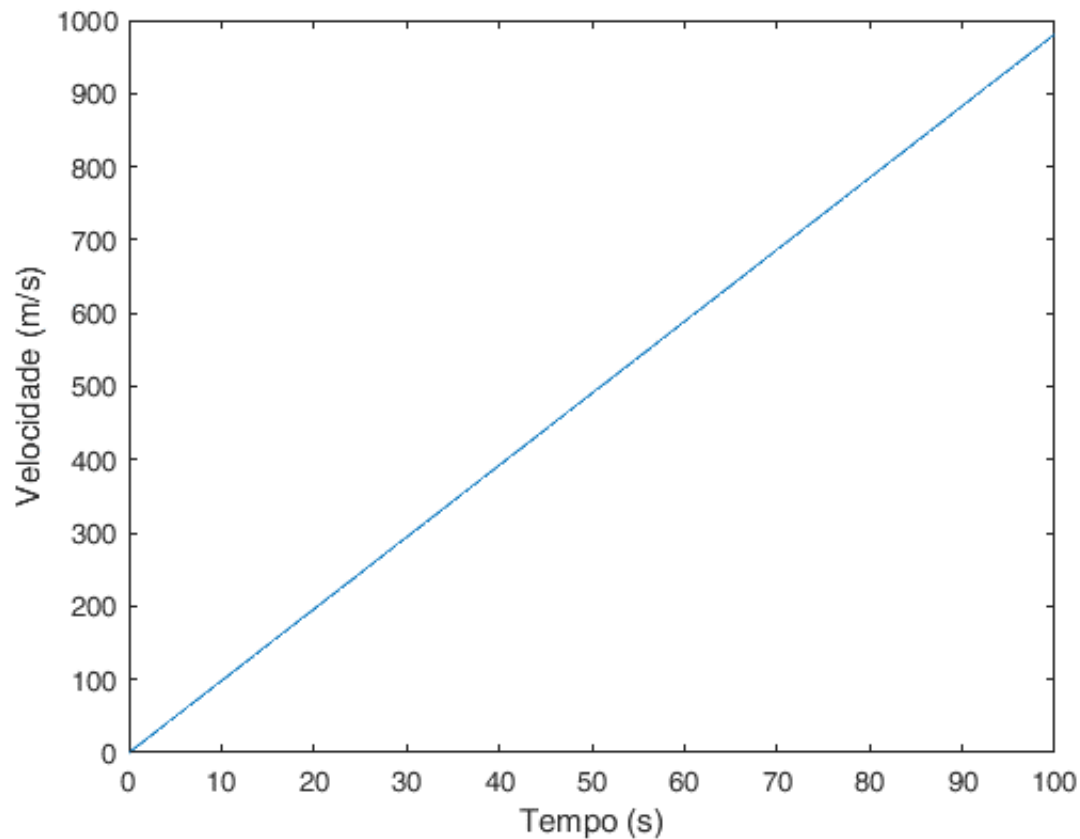
Exemplo de um arquivo de script

```
1      % Programa Velocidade_de_Queda.m: plota a velocidade de queda de um objeto.
2      % Criado em 01/03/2009, por W. Palm III
3      %
4      % Variável de entrada:
5      % tfinal = tempo final (em segundos)
6      %
7      % Variáveis de saída:
8      % t = arranjo de instantes de tempo em que a velocidade é calculada
9      % (Segundos)
10     % v = arranjo de velocidades (metros/segundo)
11     % Valor de parâmetro:
12 -   g = 9.81; % Aceleração em unidades do SI (m/(s^2))
13     %
14     % Seção de entrada:
15 -   tfinal = input ('Entre com o tempo final em segundos: ');
16     %
17     % Seção de cálculo:
18 -   dt = tfinal/500;
19 -   t = 0:dt:tfinal; % Cria um arranjo com 501 valores de tempo.
20 -   v = g*t; % Movimento uniformemente variado
21     %
22     % Seção de saída:
23 -   plot(t,v);
24 -   xlabel('Tempo (s)');
25 -   ylabel('Velocidade (m/s)');
```

Arquivos de script e o Editor/Debugador

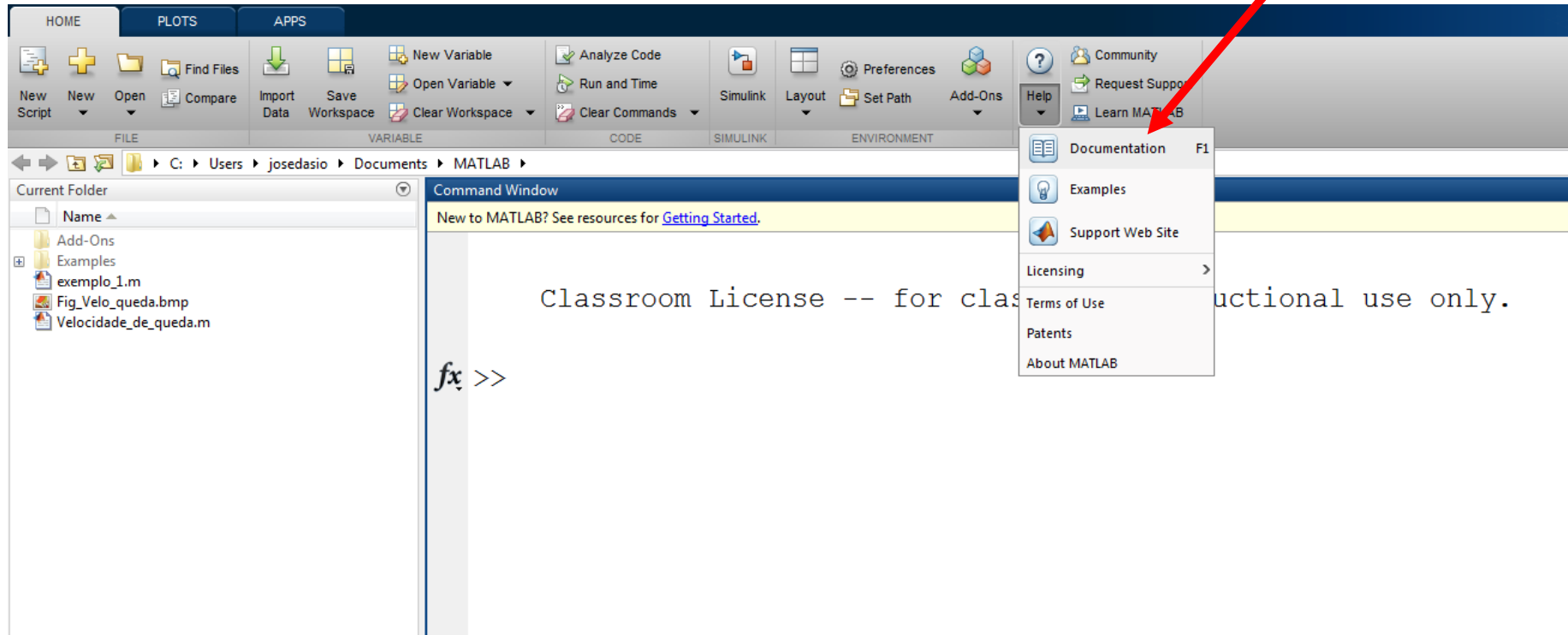
Exemplo de um arquivo de script

```
tfinal = 100;
```



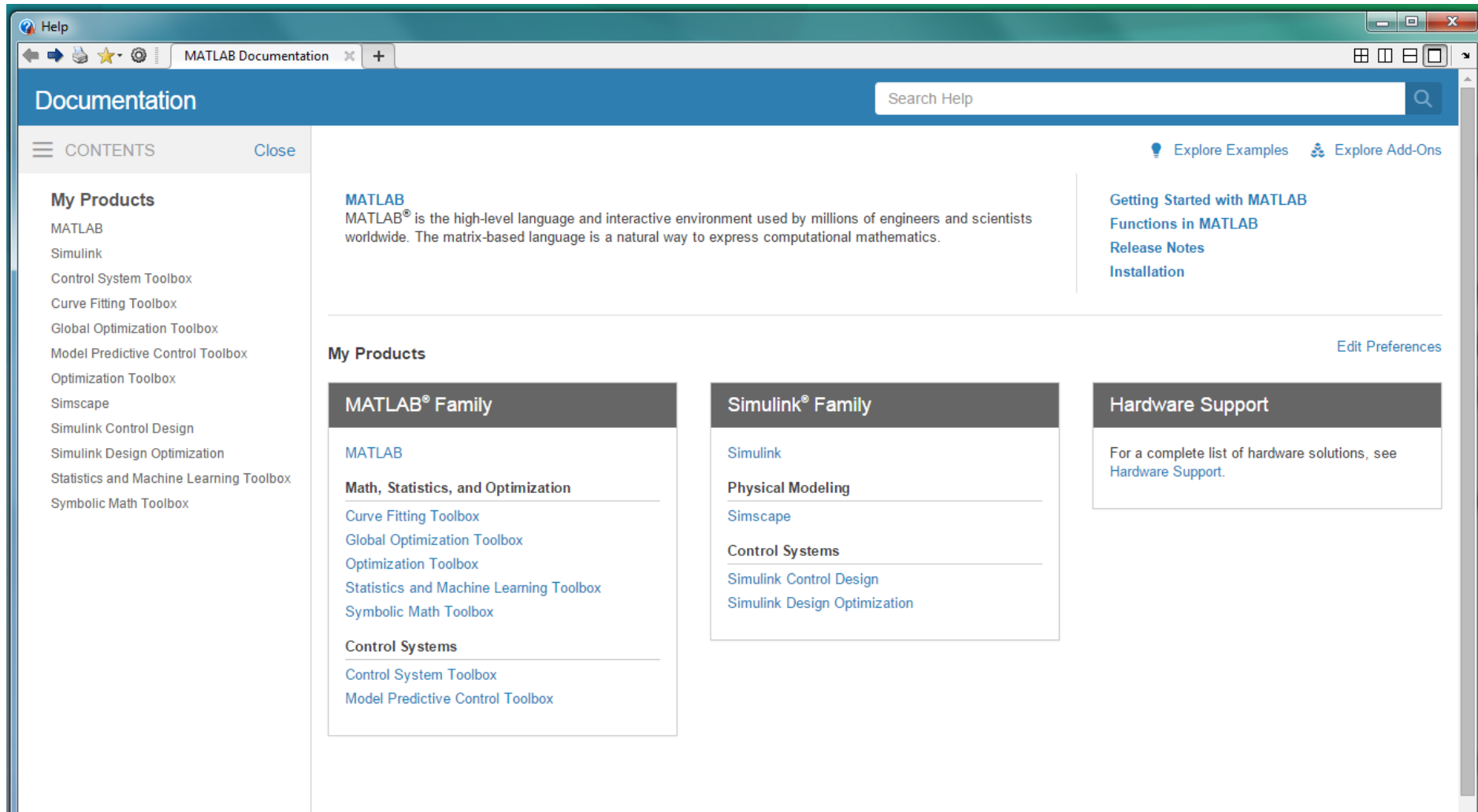
Sistema de ajuda do MATLAB

O Navegador de Funções



Sistema de ajuda do MATLAB

O Navegador de Funções



Sistema de ajuda do MATLAB

O Navegador de Funções

The screenshot displays the MATLAB R2016b interface. The top ribbon includes tabs for HOME, PLOTS, and APPS. The HOME tab is active, showing various toolbars for file operations (New Script, New, Open, Compare), workspace management (Import Data, Save Workspace, Clear Workspace), code execution (Analyze Code, Run and Time, Clear Commands), Simulink, Layout, Preferences, Set Path, Add-Ons, Help, Community, Request Support, and Learn MATLAB. The Command Window shows a message: "New to MATLAB? See resources for [Getting Started](#)." Below this, the Command Window prompt is `fx >>`. A red arrow points from the title "Sistema de ajuda do MATLAB" to the Command Window. The Function Navigator is open, displaying a list of functions. The "plot" function is selected, and its details are shown in the right pane. The details include the function's description: "2-D line plot. This MATLAB function creates a 2-D line plot of the data in Y versus the corresponding values in X." and a list of function signatures: `plot(X,Y)`, `plot(X,Y,LineSpec)`, `plot(X1,Y1,...,Xn,Yn)`, `plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)`, `plot(Y)`, `plot(Y,LineSpec)`, `plot(__,Name,Value)`, `plot(ax,___)`, and `h = plot(___)`. A "More Help..." link is also present.

MATLAB R2016b - classroom use

HOME PLOTS APPS

New Script New Open Compare Import Data Save Workspace Clear Workspace Analyze Code Run and Time Clear Commands Simulink Layout Preferences Set Path Add-Ons Help Community Request Support Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder

Name

- Add-Ons
- Examples
- exemplo_1.m
- Fig_Velo_queda.bmp
- Velocidade_de_queda.m

Command Window

New to MATLAB? See resources for [Getting Started](#).

`fx >>`

plot

Categories

- Plot Customization (control)
- 2-D and 3-D Plots
- DOE Plots (stats)
- Date and Time Arithmetic and Plotting

Functions

Function	Description
<code>plot</code>	2-D line plot
<code>plot(simulink)</code>	Draw graph connecting series of p..
<code>plot(curvefit)</code>	Plot cfit or sfit object
<code>plot(mpc)</code>	Plot responses generated by MPC ..
<code>plot(stats)</code>	Plot Bayesian optimization results
<code>plot3</code>	3-D line plot
<code>plotyy</code>	(Not recommended) Create graph .
<code>plotedit</code>	Interactively edit and annotate plots

All installed products

plot [More Help...](#)

2-D line plot

This MATLAB function creates a 2-D line plot of the data in Y versus the corresponding values in X.

`plot(X,Y)`
`plot(X,Y,LineSpec)`
`plot(X1,Y1,...,Xn,Yn)`
`plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)`
`plot(Y)`
`plot(Y,LineSpec)`
`plot(__,Name,Value)`
`plot(ax,___)`
`h = plot(___)`

Sistema de ajuda do MATLAB

Funções de Ajuda

A função help

> help plot

```
Command Window
New to MATLAB? See resources for Getting Started.

>> help plot
plot    Linear plot.

plot(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
then the vector is plotted versus the rows or columns of the matrix,
whichever line up. If X is a scalar and Y is a vector, disconnected
line objects are created and plotted as discrete points vertically at
X.

plot(Y) plots the columns of Y versus their index.
If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)).
In all other uses of plot, the imaginary part is ignored.

Various line types, plot symbols and colors may be obtained with
plot(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:
```

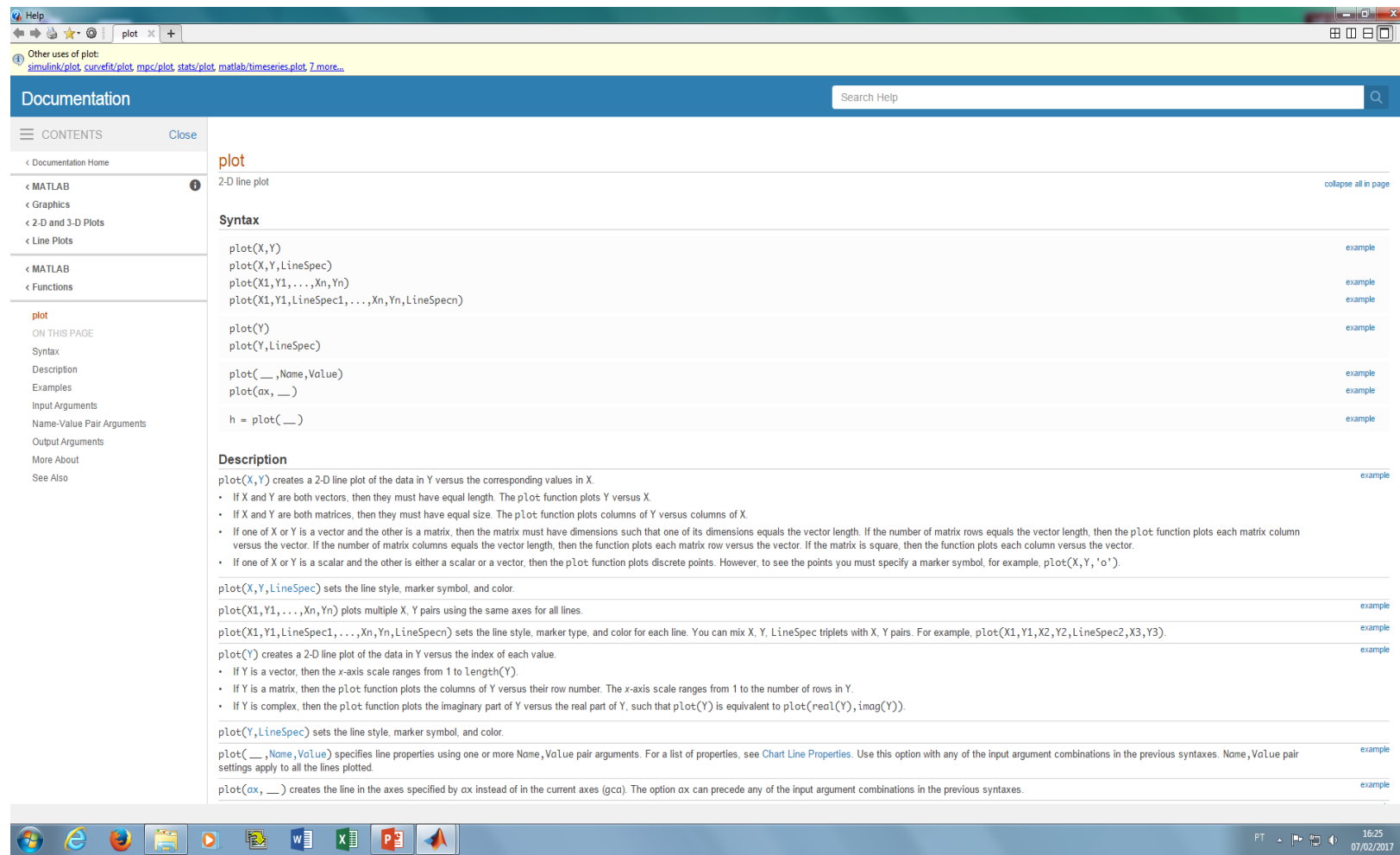
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		

Sistema de ajuda do MATLAB

Funções de Ajuda

A função doc

> doc plot



The screenshot shows the MATLAB Help browser window. The address bar displays 'plot'. The page title is 'Documentation'. The left sidebar shows the 'CONTENTS' menu with 'plot' selected. The main content area displays the 'plot' function documentation, including its syntax, description, and examples.

plot
2-D line plot

Syntax

- `plot(X,Y)` [example](#)
- `plot(X,Y,LineSpec)` [example](#)
- `plot(X1,Y1,...,Xn,Yn)` [example](#)
- `plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)` [example](#)
- `plot(Y)` [example](#)
- `plot(Y,LineSpec)` [example](#)
- `plot(__,Name,Value)` [example](#)
- `plot(ax, __)` [example](#)
- `h = plot(__)` [example](#)

Description

`plot(X,Y)` creates a 2-D line plot of the data in Y versus the corresponding values in X.

- If X and Y are both vectors, then they must have equal length. The `plot` function plots Y versus X.
- If X and Y are both matrices, then they must have equal size. The `plot` function plots columns of Y versus columns of X.
- If one of X or Y is a vector and the other is a matrix, then the matrix must have dimensions such that one of its dimensions equals the vector length. If the number of matrix rows equals the vector length, then the `plot` function plots each matrix column versus the vector. If the number of matrix columns equals the vector length, then the function plots each matrix row versus the vector. If the matrix is square, then the function plots each column versus the vector.
- If one of X or Y is a scalar and the other is either a scalar or a vector, then the `plot` function plots discrete points. However, to see the points you must specify a marker symbol, for example, `plot(X,Y,'o')`.

`plot(X,Y,LineSpec)` sets the line style, marker symbol, and color.

`plot(X1,Y1,...,Xn,Yn)` plots multiple X, Y pairs using the same axes for all lines.

`plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)` sets the line style, marker type, and color for each line. You can mix X, Y, LineSpec triplets with X, Y pairs. For example, `plot(X1,Y1,X2,Y2,LineSpec2,X3,Y3)`.

`plot(Y)` creates a 2-D line plot of the data in Y versus the index of each value.

- If Y is a vector, then the x-axis scale ranges from 1 to `length(Y)`.
- If Y is a matrix, then the `plot` function plots the columns of Y versus their row number. The x-axis scale ranges from 1 to the number of rows in Y.
- If Y is complex, then the `plot` function plots the imaginary part of Y versus the real part of Y, such that `plot(Y)` is equivalent to `plot(real(Y),imag(Y))`.

`plot(Y,LineSpec)` sets the line style, marker symbol, and color.

`plot(__,Name,Value)` specifies line properties using one or more Name,Value pair arguments. For a list of properties, see [Chart Line Properties](#). Use this option with any of the input argument combinations in the previous syntaxes. Name,Value pair settings apply to all the lines plotted.

`plot(ax, __)` creates the line in the axes specified by ax instead of in the current axes (gca). The option ax can precede any of the input argument combinations in the previous syntaxes.

Sistema de ajuda do MATLAB

Funções de Ajuda

A função doc

> doc

