College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 1 (About MATLAB)

Zeyad T. Shareef

Objectives

After studying this lecture, you should be able to:

- Understand what MATLAB is and why it is widely used in engineering and science
- Understand the advantages and limitations of the student edition of MATLAB
- Formulate problems by using a structured problem-solving approach

Section 1.1
What is MATLAB?

- MATLAB is one of a number of commercially available, sophisticated mathematical computation tools
- Others include
  - Maple
  - Mathematica
  - MathCad

MATLAB excels at:

- Numerical calculations
  - Especially involving matrices
- Graphics
- MATLAB stands for
  Matrix **Lab**oratory

Why MATLAB

- Easy to use
- Versatile
- Built in programming language
- Not a general purpose language like C++ or Java

MATLAB was originally written in Fortran, then later rewritten in C

Section 1.2
Student Edition of MATLAB

- MATLAB comes in both a student and professional edition
- Student editions are available for
  - Windows Operating Systems
  - Mac OS
  - Linux
- The student edition typically lags the professional edition by one release

The command prompt is the biggest difference you’ll notice

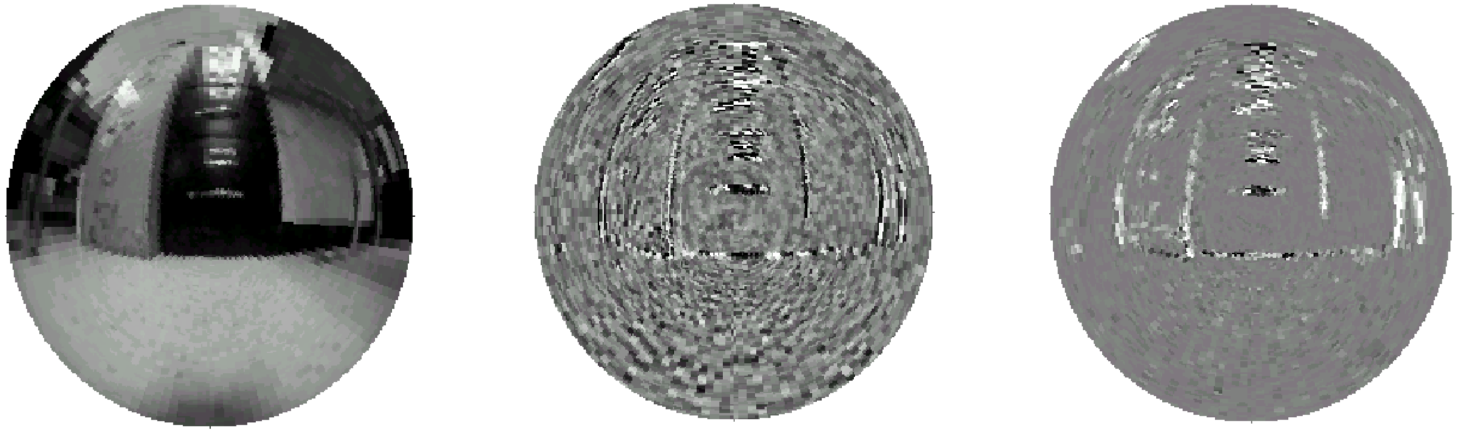
>> is the command prompt for the professional version

EDU>> is the command prompt for the student version

Section 1.3
How is MATLAB used in Industry?

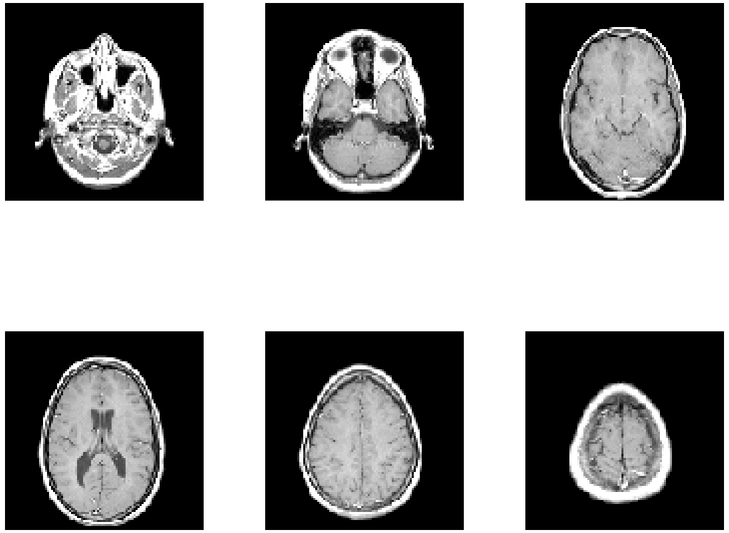
- Widespread, especially in the signal processing field
- Tool of choice in Academia for most engineering fields
- Some examples….

Electrical Engineering



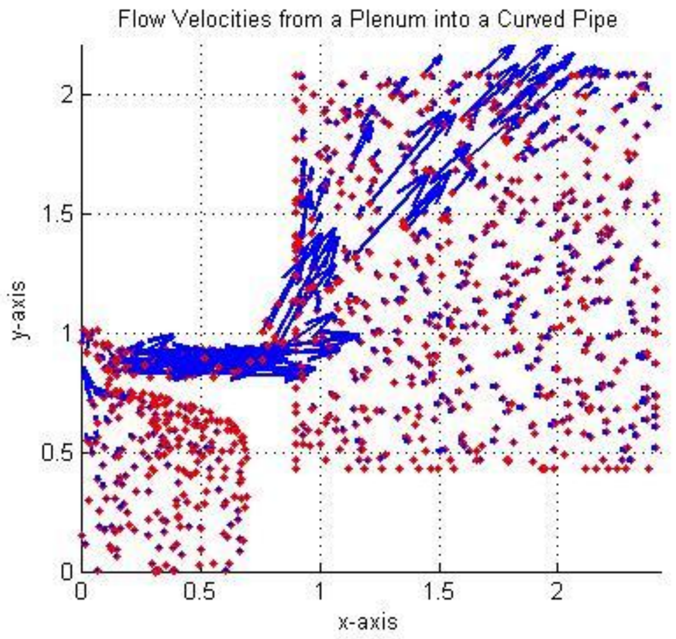
These images simulate the visual system used in a housefly brain to detect collisions. The techniques developed are being used in autonomous robot systems that depend upon vision for navigation. The data was processed using MATLAB

Biomedical Engineering



These images were created from MRI scan data using MATLAB. The actual data set is included with the standard MATLAB installation, allowing you experiment with manipulating the data yourself.

Fluid Dynamics



Results from a finite element analysis code were post processed using MATLAB to create this image.

Section 1.4
Problem Solving in Engineering and Science

1. State the Problem
2. Describe the input and output
3. Develop an algorithm
4. Solve the problem
5. Test the solution

State the Problem

- If you don’t have a clear understanding of the problem, it’s unlikely that you’ll be able to solve it
- Drawing a picture often helps you understand the system better

Describe the Input and Output

- Be careful to include units
- Identify constants
- Label your sketch
- Group information into tables

Develop an Algorithm

- Identify any equations relating the knowns and unknowns
- Work through a simplified version of the problem by hand or with a calculator
- Developing a flow chart is often useful for complicated problems

Solve the problem

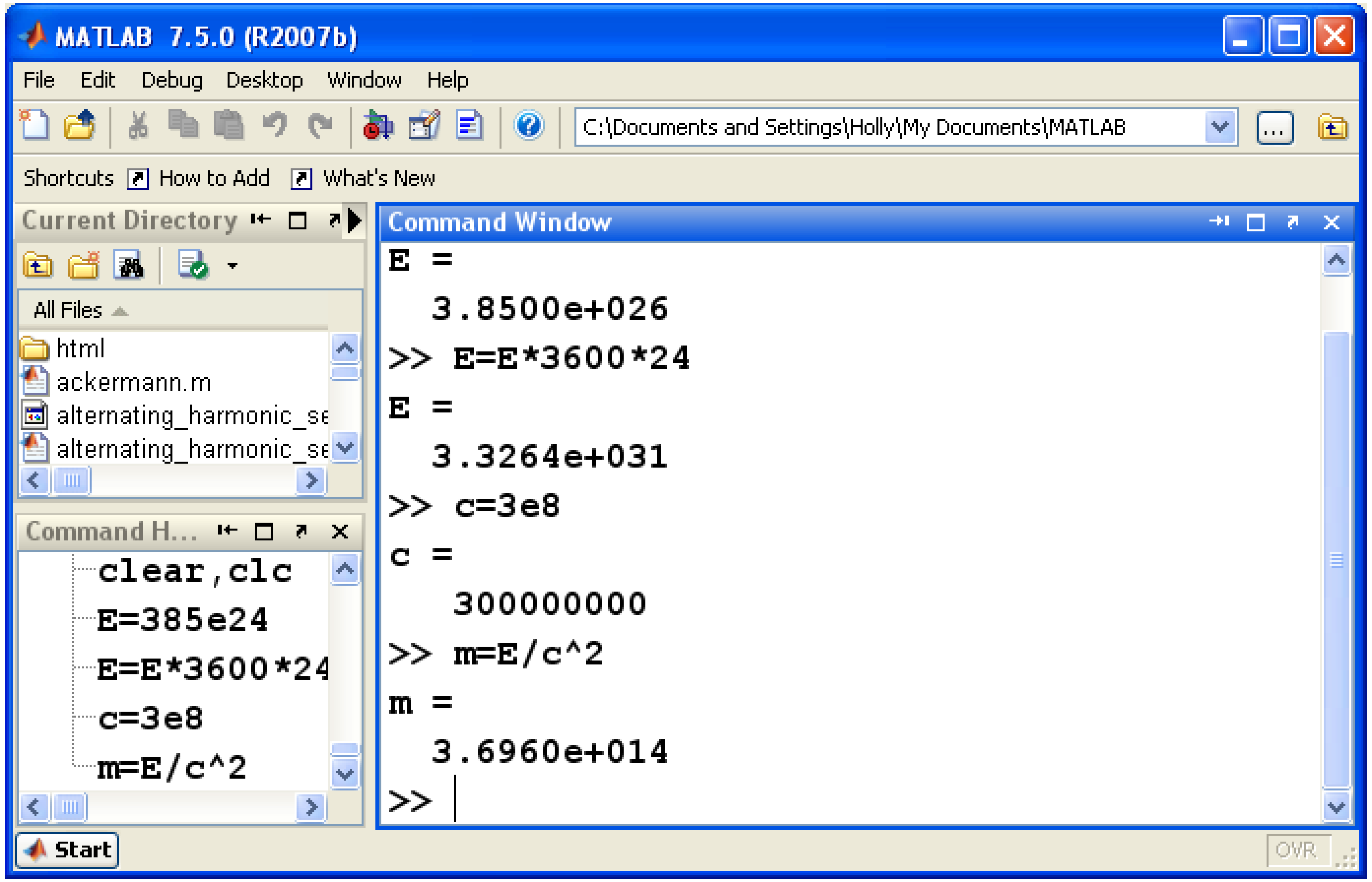
- Create a MATLAB solution
- Be generous with comments, so that others can follow your work

Test the Solution

- Compare to the hand solution
- Do your answers make sense physically?
- Is your answer really what was asked for?
- Graphs are often useful ways to check your calculations for reasonableness

Develop a MATLAB Solution to Solve the Problem

- We’ll start learning the details of how to use MATLAB in the next chapter.
- However, you can see from the following demonstration just how easy it is to use the command window



C:\Documents and Settings\Holly\My Documents\MATLAB

```
clear,clc
E=385e24
E=E*3600*24
c=3e8
m=E/c^2
```

```
E =
  3.8500e+026
>> E=E*3600*24
E =
  3.3264e+031
>> c=3e8
c =
   300000000
>> m=E/c^2
m =
  3.6960e+014
>>
```

Summary

- MATLAB is widely used
- MATLAB is easy to use
- A systematic problem solving strategy makes it more likely you’ve found the right answer

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 2 (MATLAB Environment)

Zeyad T. Shareef

Objectives

After studying this lecture, you should be able to

- Start the MATLAB program and solve simple problems in the command window
- Understand MATLAB's use of matrices
- Identify and use the various MATLAB windows
- Define and use simple matrices
- Name and use variables
- Understand the order of operations in MATLAB
- Understand the difference between scalar, array and matrix calculations in MATLAB

Objectives - continued

After studying this lecture, you should be able to

- Express numbers in either floating-point or scientific notation
- Adjust the format used to display numbers in the command window
- Save the value of variables used in a MATLAB session
- Save a series of commands in an M-file

In this lecture we’ll…

- Get started with MATLAB
- Explore the MATLAB windows
- Solve some problems using MATLAB
- Learn how to save our work

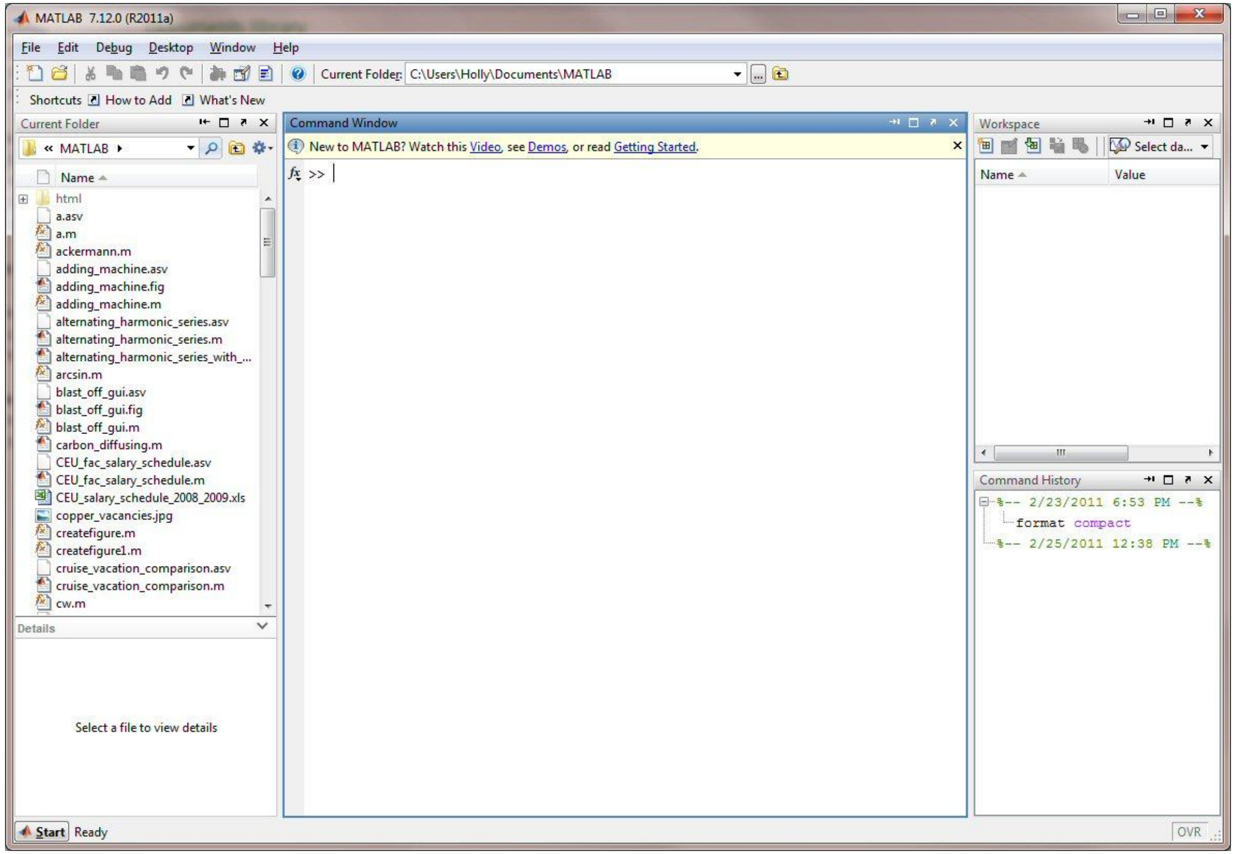
Section 2.1
Getting Started

- In Windows or Apple operating systems click on the desktop icon
- In Unix type

```
MATLAB
```

At the shell prompt

MATLAB opens to a default window configuration



Current Folder: C:\Users\Holly\Documents\MATLAB

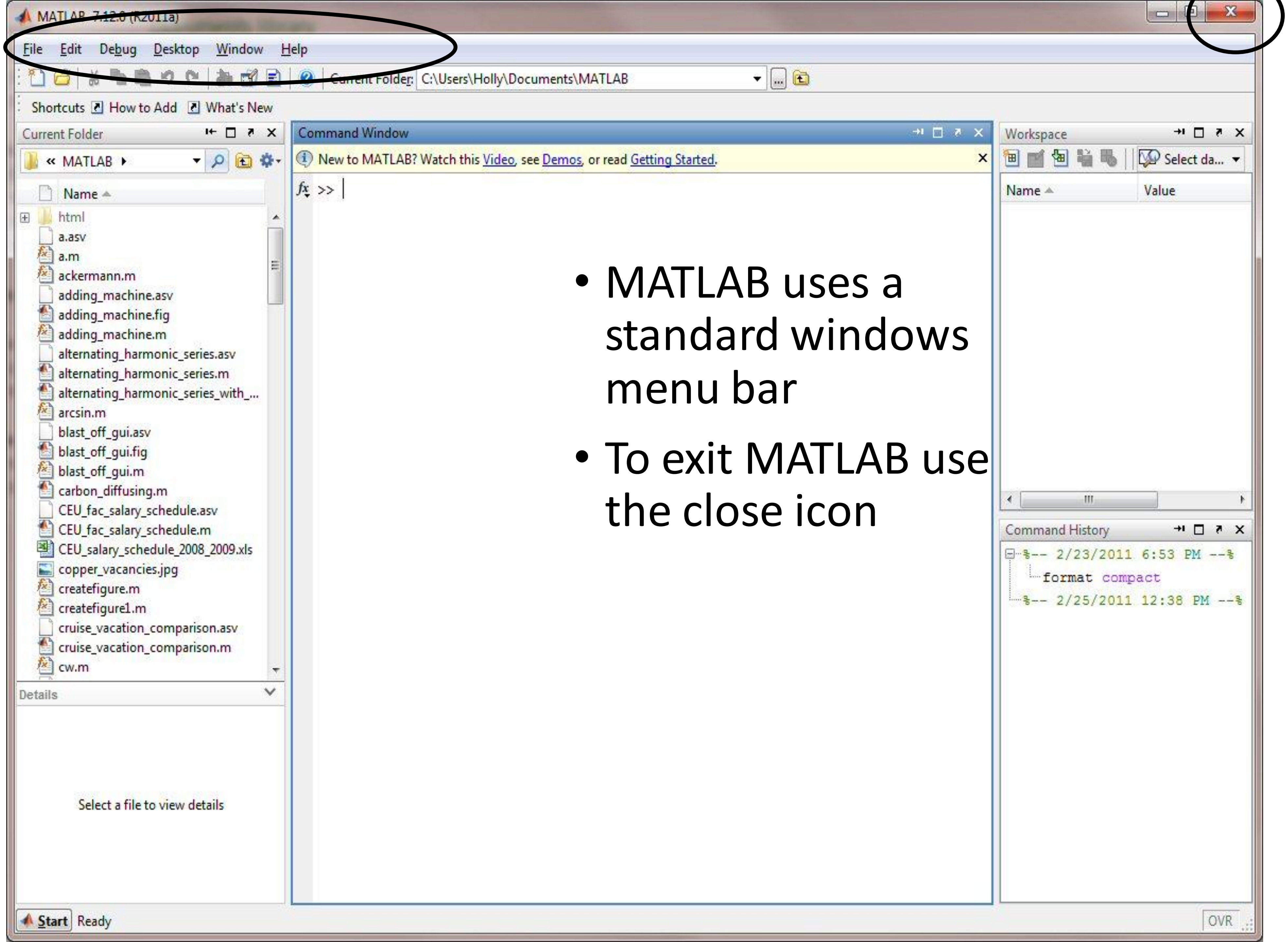
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with_...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
cruise_vacation_comparison.m
cw.m

Select a file to view details

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
fx >>
```

```
%-- 2/23/2011 6:53 PM --%
  format compact
%-- 2/25/2011 12:38 PM --%
```



Current Folder: C:\Users\Holly\Documents\MATLAB

« MATLAB ▸

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
fx >>
```

Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with_...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
cruise_vacation_comparison.m
cw.m

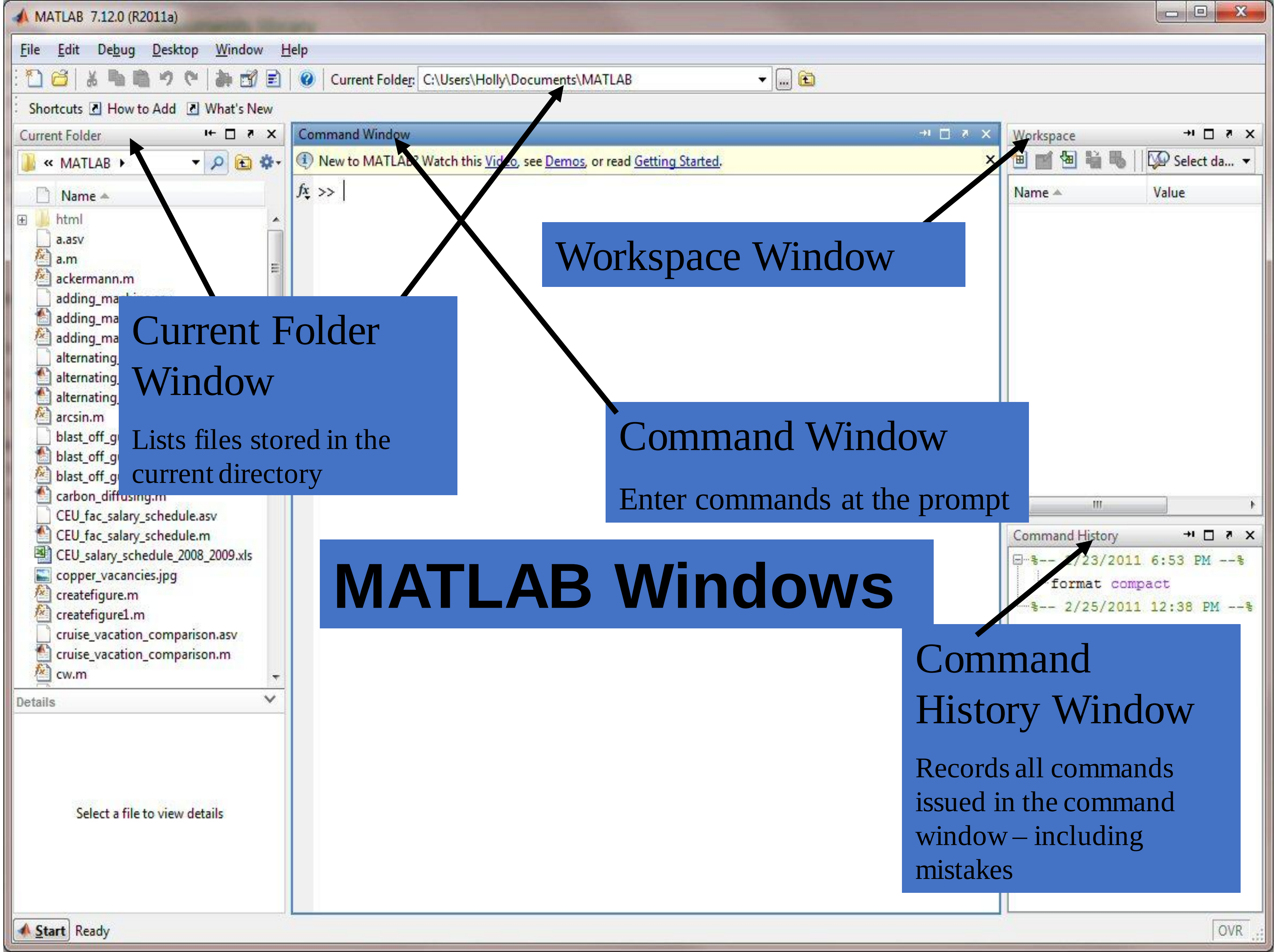
Select a file to view details

```
%-- 2/23/2011 6:53 PM --%
  format compact
%-- 2/25/2011 12:38 PM --%
```

- MATLAB uses a standard windows menu bar
- To exit MATLAB use the close icon

Section 2.2
MATLAB Windows

- MATLAB uses several different windows to display data, commands and results.
- They are not necessarily all open at once



Current Folder: C:\Users\Holly\Documents\MATLAB

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
fx >>
```

```
%-- 2/23/2011 6:53 PM --%
  format compact
%-- 2/25/2011 12:38 PM --%
```

Select a file to view details

Ready

MATLAB Windows

Current Folder Window

Lists files stored in the current directory

Workspace Window

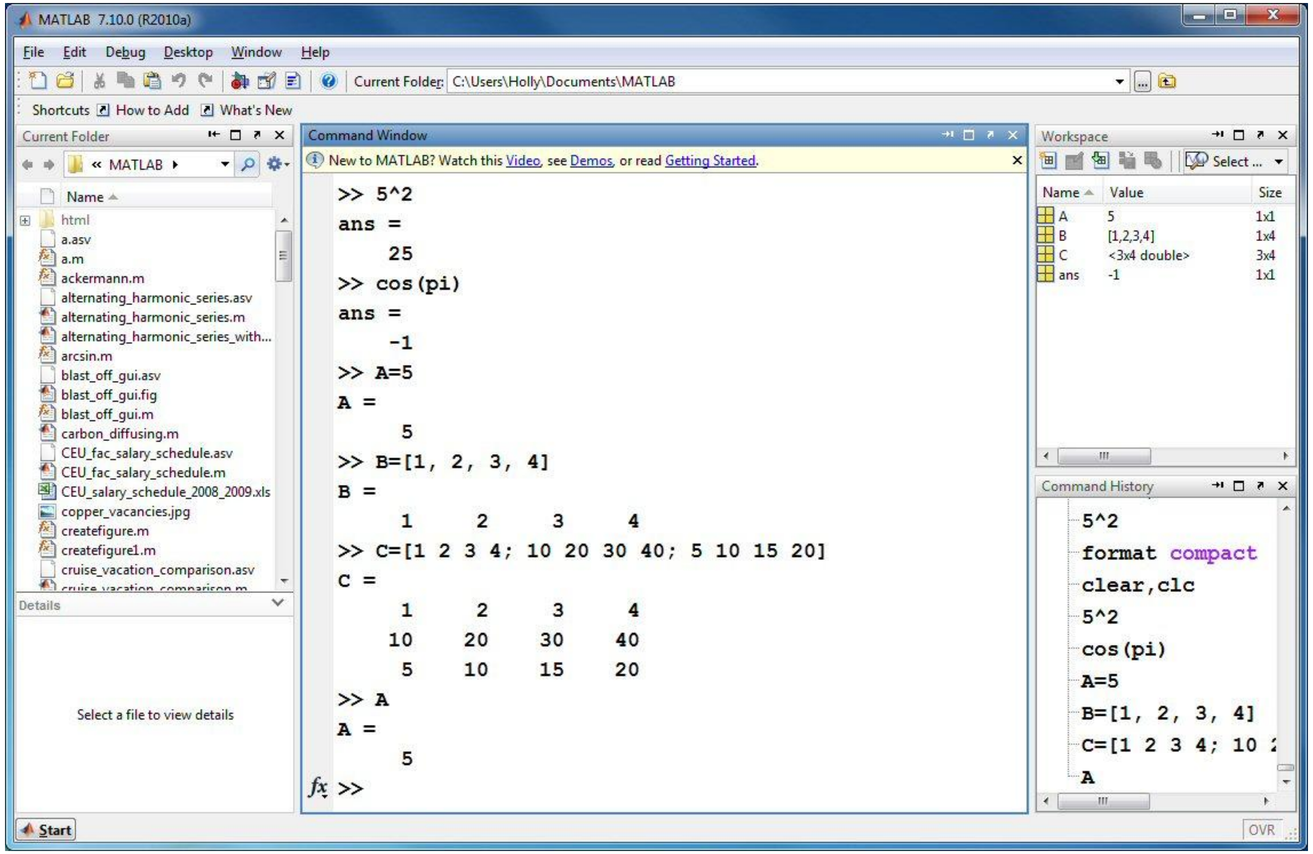
Command Window

Enter commands at the prompt

Command History Window

Records all commands issued in the command window – including mistakes

Let’s look at the windows one at a time



html
a.asv
a.m
ackermann.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
cruise_vacation_comparison.m

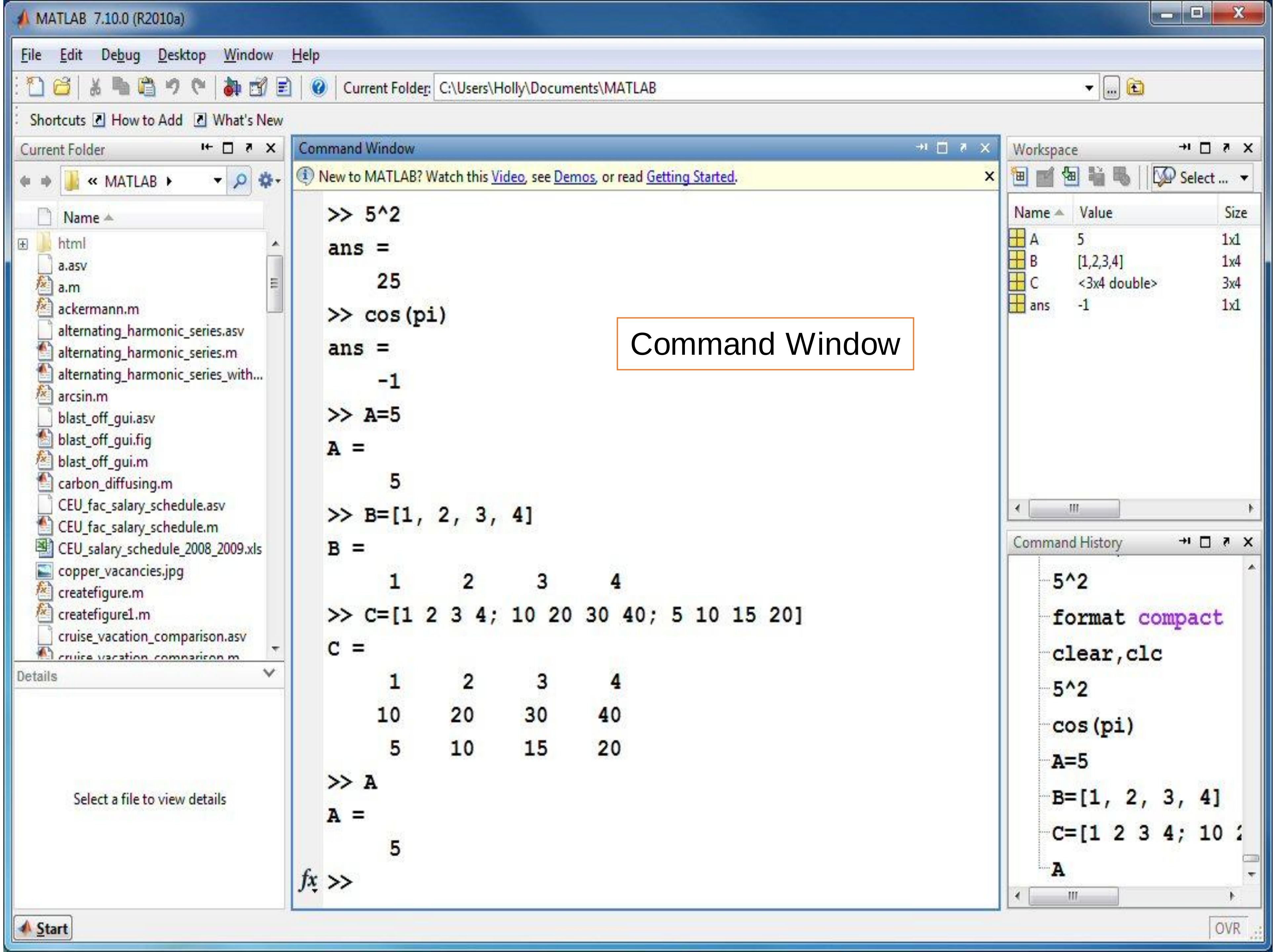
Select a file to view details

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> 5^2
ans =
    25
>> cos(pi)
ans =
    -1
>> A=5
A =
     5
>> B=[1, 2, 3, 4]
B =
     1     2     3     4
>> C=[1 2 3 4; 10 20 30 40; 5 10 15 20]
C =
     1     2     3     4
    10    20    30    40
     5    10    15    20
>> A
A =
     5
>>
```

Name	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```


MATLAB 7.10.0 (R2010a)
File
Edit
Debug
Desktop
Window
Help
Current Folder: C:\Users\Holly\Documents\MATLAB
Shortcuts
How to Add
What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
cruise_vacation_comparison.m
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Select ...
Command History
Start
OVR

```
>> 5^2
ans =
    25
>> cos(pi)
ans =
    -1
>> A=5
A =
     5
>> B=[1, 2, 3, 4]
B =
     1     2     3     4
>> C=[1 2 3 4; 10 20 30 40; 5 10 15 20]
C =
     1     2     3     4
    10    20    30    40
     5    10    15    20
>> A
A =
     5
fx >>
```

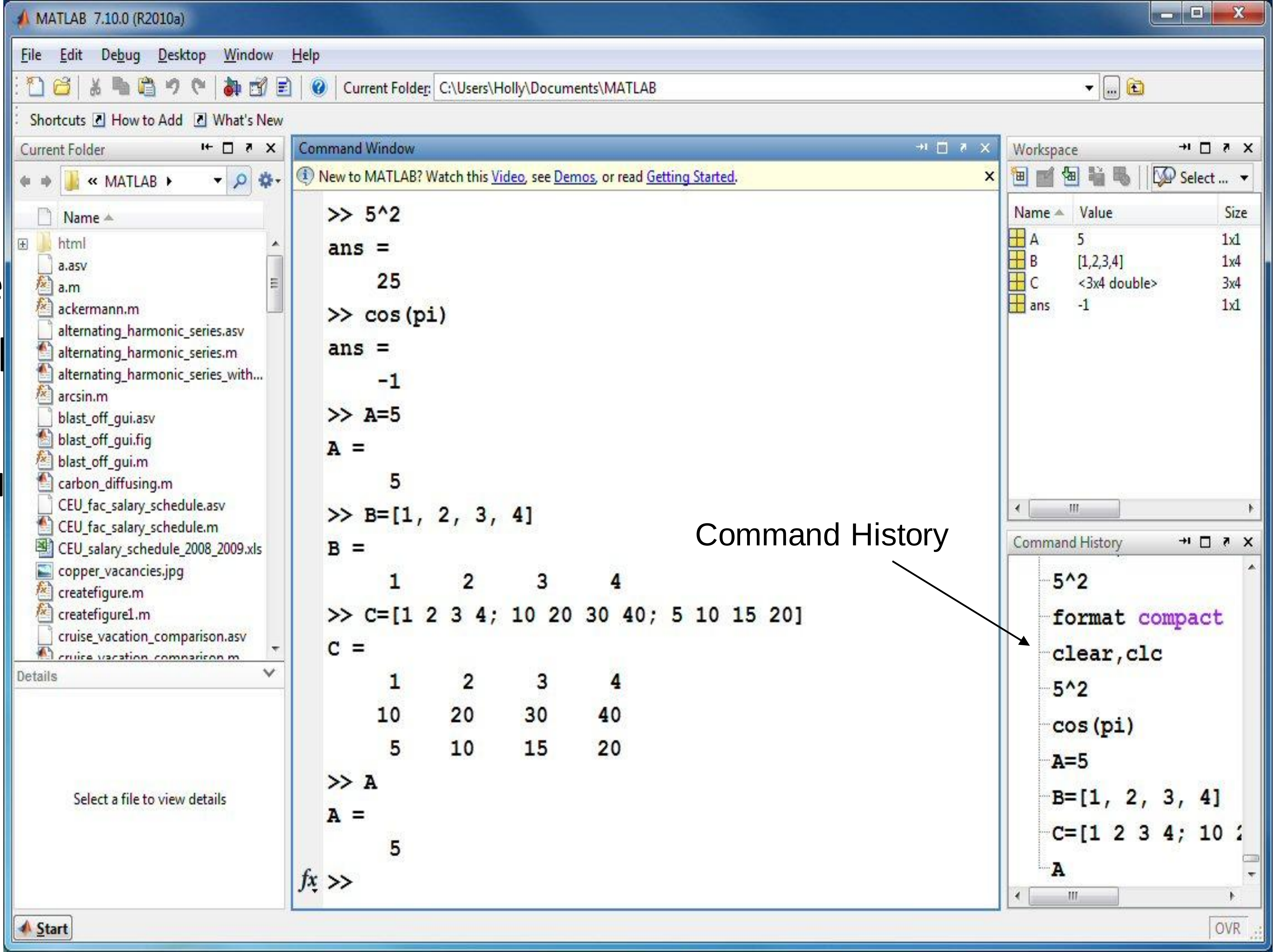
Name	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```

Command Window

Co

- Re
- Wl
  co
- Bu


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Current Folder: C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Select ...
Command History
Start
OVR

```
>> 5^2
ans =
    25
>> cos(pi)
ans =
    -1
>> A=5
A =
     5
>> B=[1, 2, 3, 4]
B =
     1     2     3     4
>> C=[1 2 3 4; 10 20 30 40; 5 10 15 20]
C =
     1     2     3     4
    10    20    30    40
     5    10    15    20
>> A
A =
     5
fx >>
```

Name	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

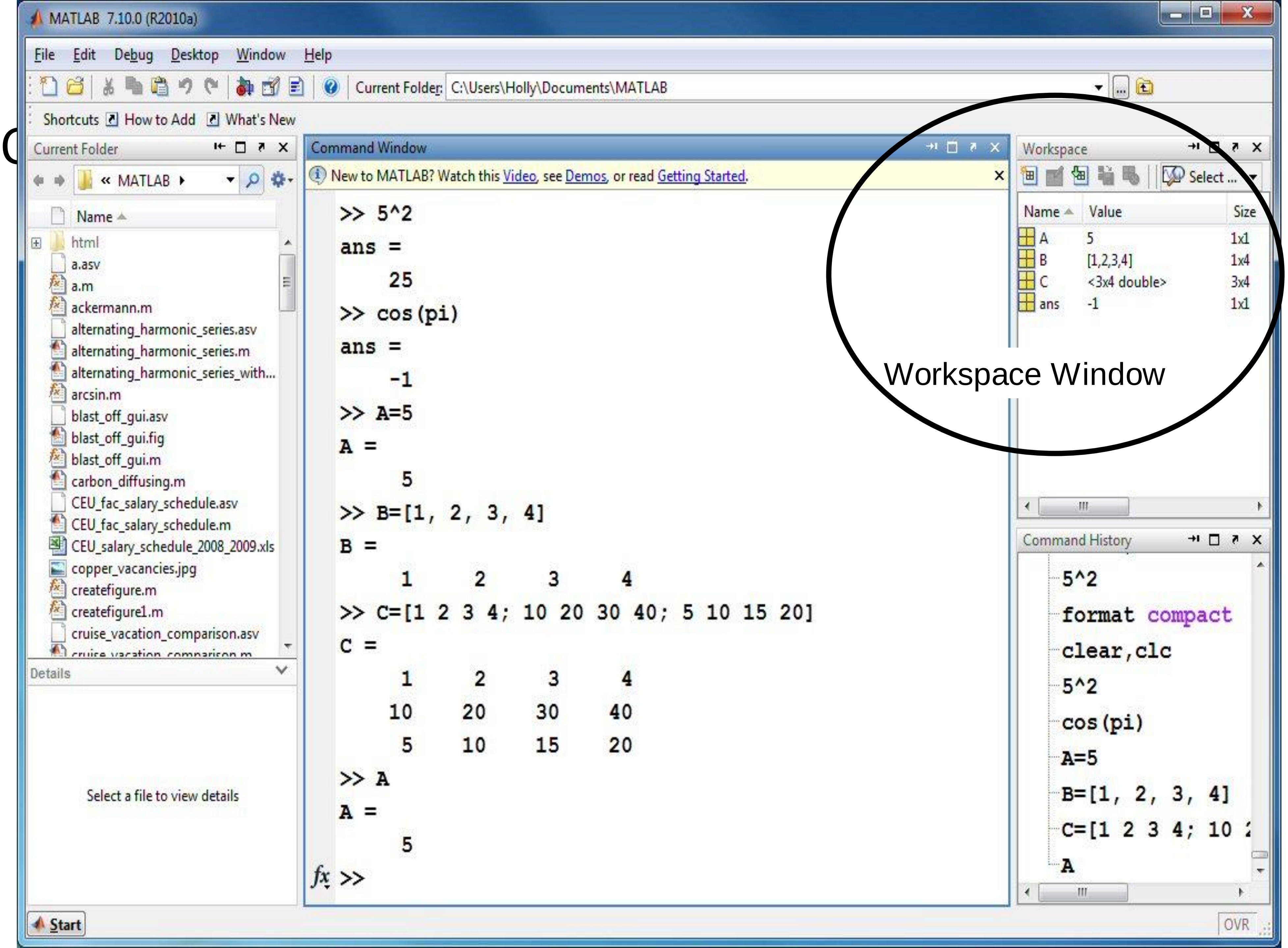
Command History

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```

Command History

- You can transfer commands from the command history to the command window
    - Double click on a command
        - It executes immediately
    - Click and drag into the command window
        - You can edit the command before executing

Wo



MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Current Folder: C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Select ...
Command History
Start
OVR

Current Folder:

- html
- a.asv
- a.m
- ackermann.m
- alternating_harmonic_series.asv
- alternating_harmonic_series.m
- alternating_harmonic_series_with...
- arcsin.m
- blast_off_gui.asv
- blast_off_gui.fig
- blast_off_gui.m
- carbon_diffusing.m
- CEU_fac_salary_schedule.asv
- CEU_fac_salary_schedule.m
- CEU_salary_schedule_2008_2009.xls
- copper_vacancies.jpg
- createfigure.m
- createfigure1.m
- cruise_vacation_comparison.asv
- cruise_vacation_comparison.m

Command Window:

```
>> 5^2
ans =
    25
>> cos(pi)
ans =
    -1
>> A=5
A =
     5
>> B=[1, 2, 3, 4]
B =
     1     2     3     4
>> C=[1 2 3 4; 10 20 30 40; 5 10 15 20]
C =
     1     2     3     4
    10    20    30    40
     5    10    15    20
>> A
A =
     5
fx >>
```

Workspace:

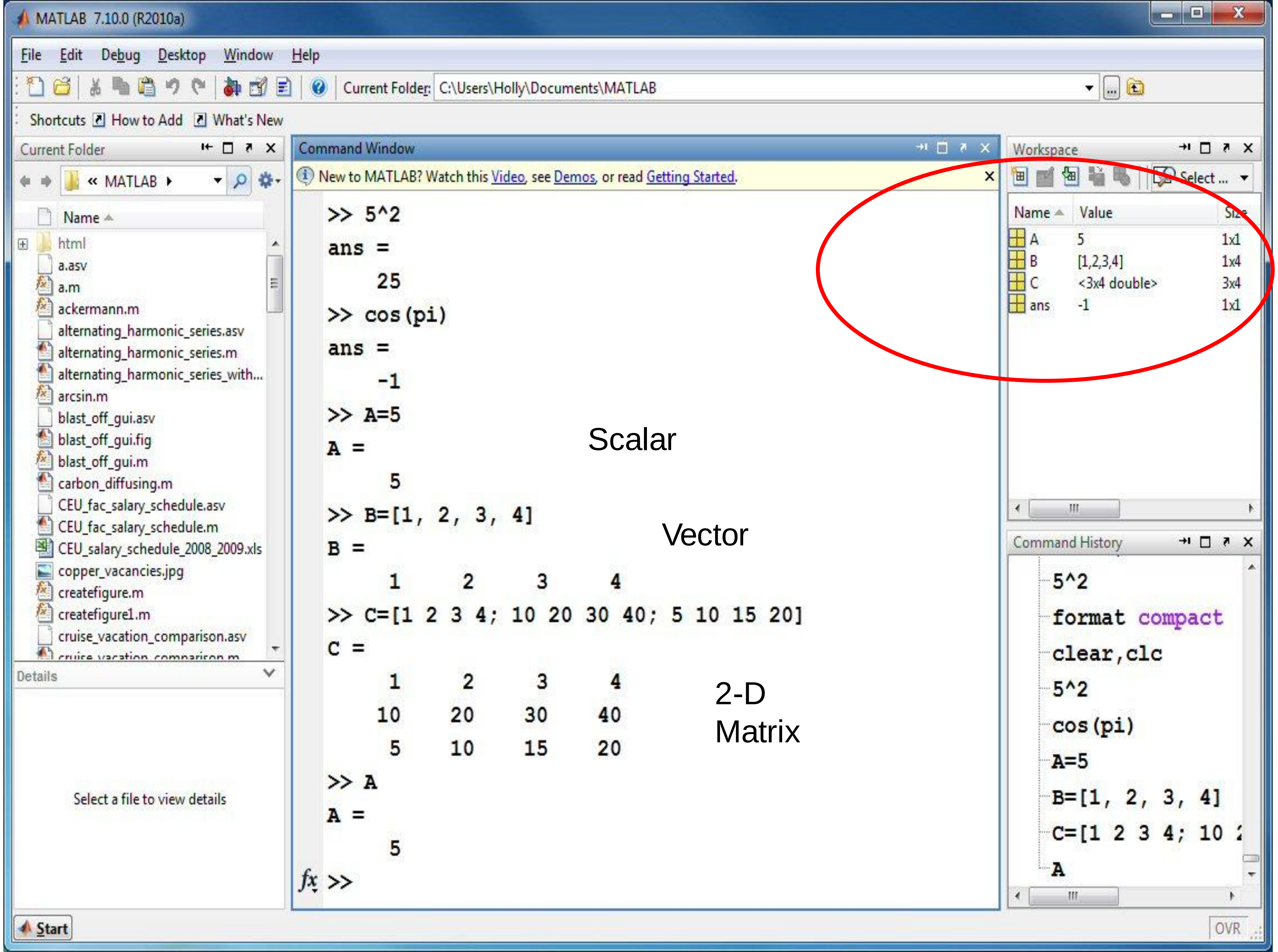
Name	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

Workspace Window

Command History:

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```

When you define variables in the command window, they are listed in the workspace window



html
a.asv
a.m
ackermann.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
cruise_vacation_comparison.m

```
>> 5^2
ans =
    25
>> cos(pi)
ans =
    -1
>> A=5
A =
     5
>> B=[1, 2, 3, 4]
B =
     1     2     3     4
>> C=[1 2 3 4; 10 20 30 40; 5 10 15 20]
C =
     1     2     3     4
    10    20    30    40
     5    10    15    20
>> A
A =
     5
fx >>
```

Scalar

Vector

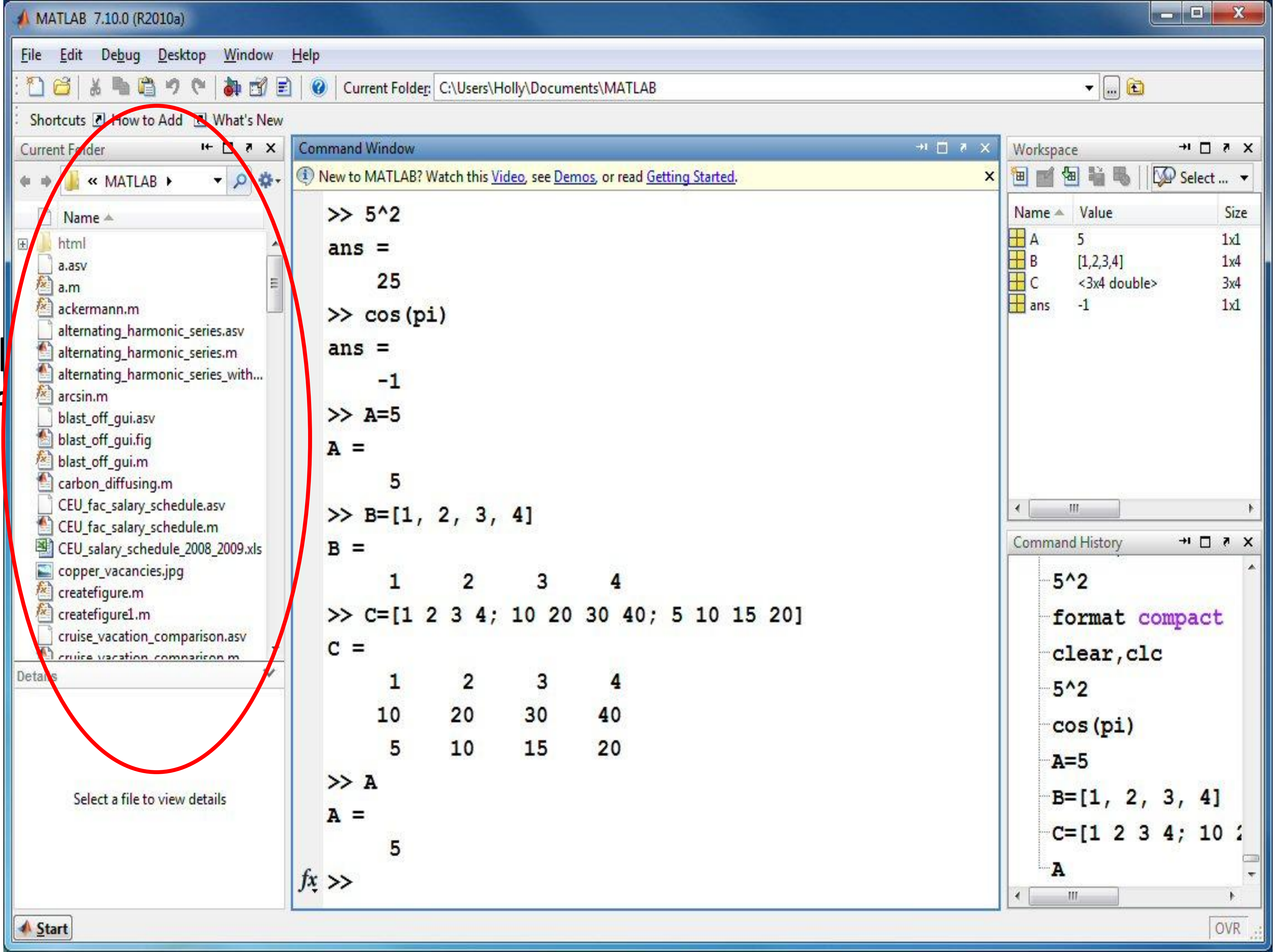
2-D Matrix

Name	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```

Cu

- Th
- Wl
  inf



New to MATLAB? Watch this Video, see Demos, or read Getting Started.

Name ▲

- html
- a.asv
- a.m
- ackermann.m
- alternating_harmonic_series.asv
- alternating_harmonic_series.m
- alternating_harmonic_series_with...
- arcsin.m
- blast_off_gui.asv
- blast_off_gui.fig
- blast_off_gui.m
- carbon_diffusing.m
- CEU_fac_salary_schedule.asv
- CEU_fac_salary_schedule.m
- CEU_salary_schedule_2008_2009.xls
- copper_vacancies.jpg
- createfigure.m
- createfigure1.m
- cruise_vacation_comparison.asv
- cruise_vacation_comparison.m

Select a file to view details

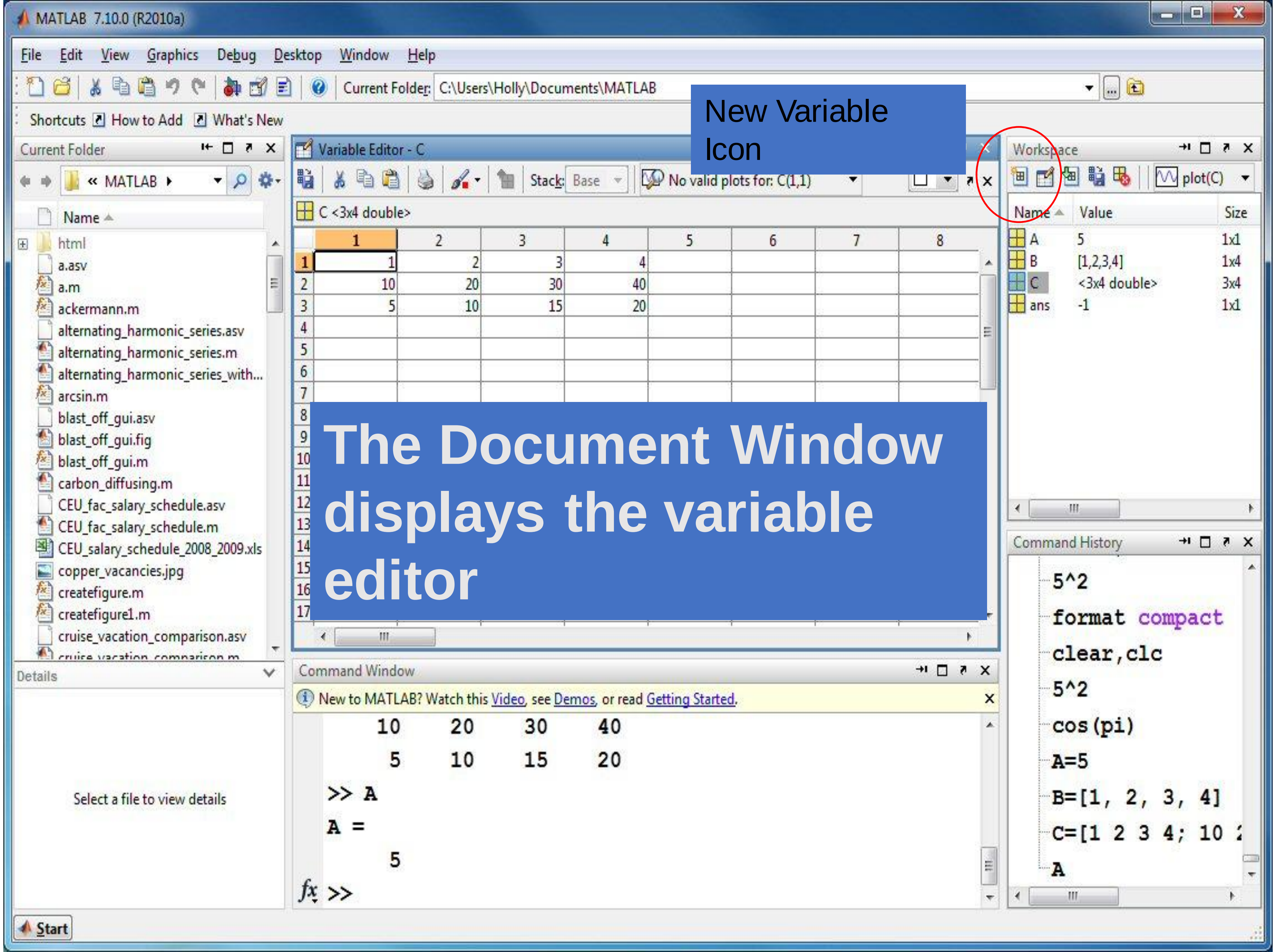
```
>> 5^2
ans =
    25
>> cos(pi)
ans =
    -1
>> A=5
A =
     5
>> B=[1, 2, 3, 4]
B =
     1     2     3     4
>> C=[1 2 3 4; 10 20 30 40; 5 10 15 20]
C =
     1     2     3     4
    10    20    30    40
     5    10    15    20
>> A
A =
     5
fx >>
```

Name ▲	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```

Document Window

- If you double click on any variable in the workspace window MATLAB launches a **document** window containing the **array editor**
- You can edit variables in the array editor


MATLAB 7.10.0 (R2010a)
File Edit View Graphics Debug Desktop Window Help
Current Folder: C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
alternating_harmonic_series.asv
alternating_harmonic_series.m
alternating_harmonic_series_with...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008_2009.xls
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparison.asv
cruise_vacation_comparison.m
Details
Select a file to view details
Variable Editor - C
Stack: Base
No valid plots for: C(1,1)
Workspace
plot(C)
Command Window
Command History
Start

New Variable Icon

C <3x4 double>

	1	2	3	4	5	6	7	8
1	1	2	3	4				
2	10	20	30	40				
3	5	10	15	20				

The Document Window displays the variable editor

Name ▲	Value	Size
A	5	1x1
B	[1,2,3,4]	1x4
C	<3x4 double>	3x4
ans	-1	1x1

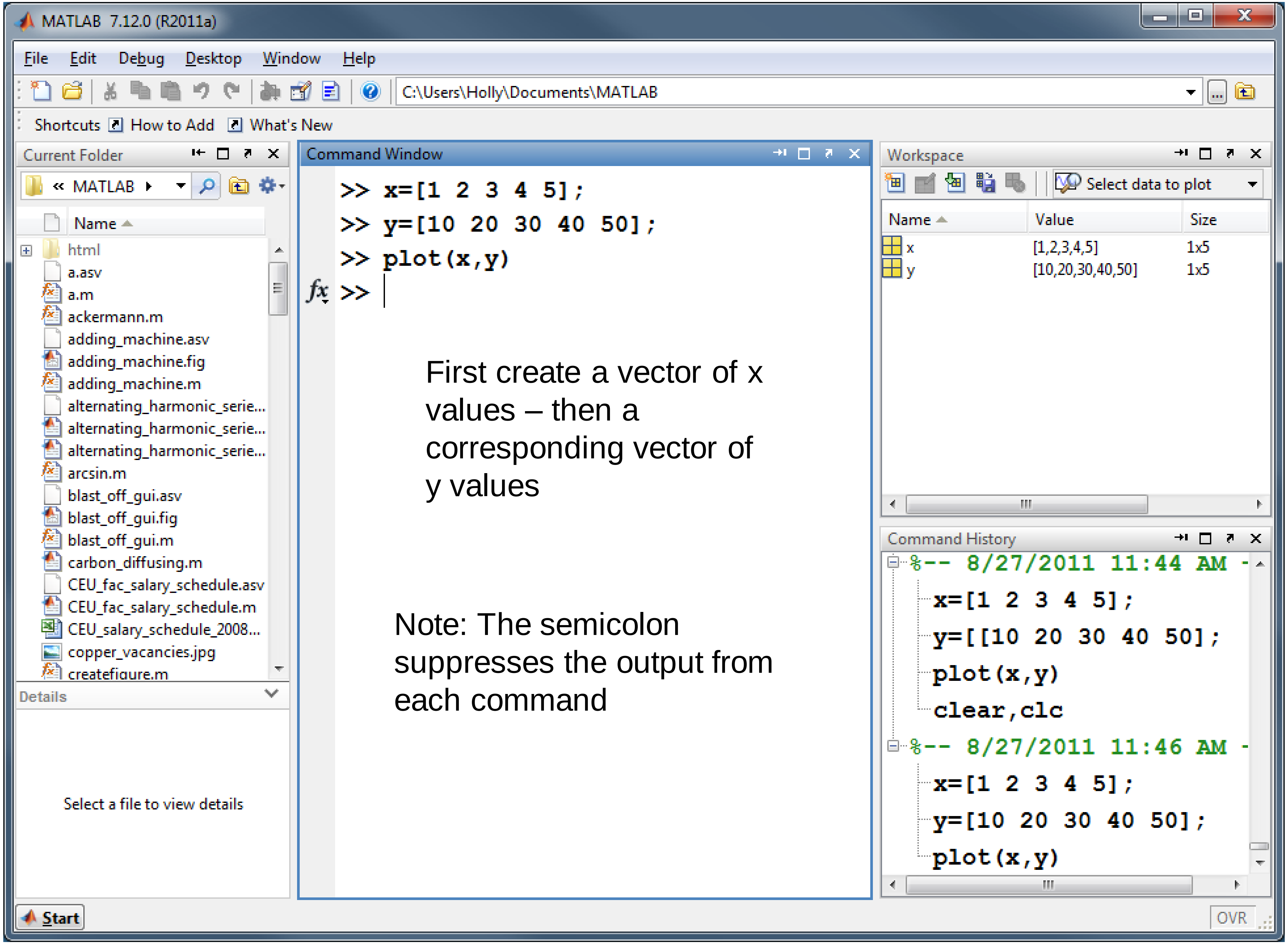
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
    10    20    30    40
     5    10    15    20
>> A
A =
     5
fx >>
```

```
5^2
format compact
clear,clc
5^2
cos(pi)
A=5
B=[1, 2, 3, 4]
C=[1 2 3 4; 10 2
A
```

Figure Window

- When Figures are created a new window opens
- It’s extremely easy to create graphs in MATLAB



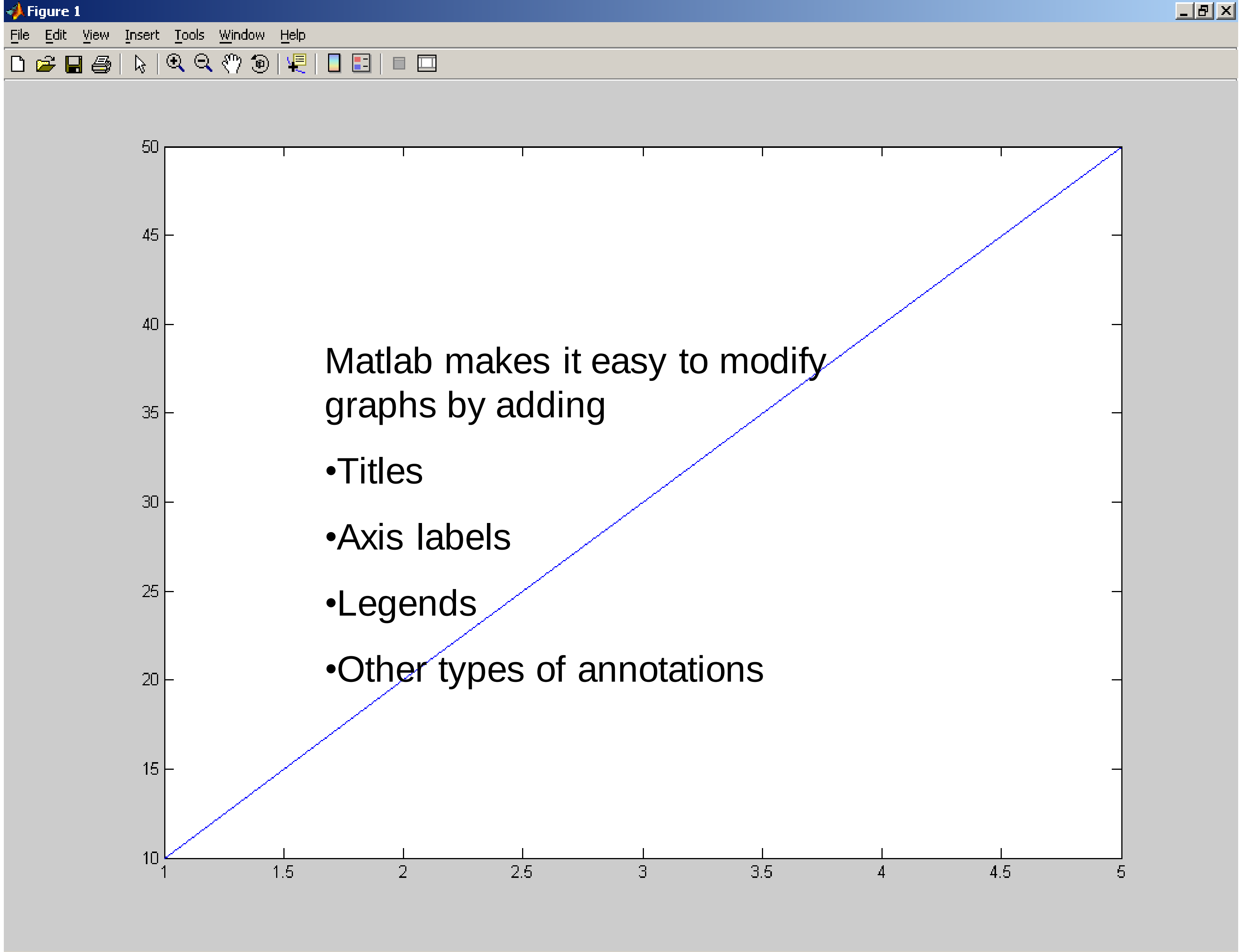
```
>> x=[1 2 3 4 5];
>> y=[10 20 30 40 50];
>> plot(x,y)
fx >> 
```

Name	Value	Size
x	[1,2,3,4,5]	1x5
y	[10,20,30,40,50]	1x5

```
%-- 8/27/2011 11:44 AM -
  x=[1 2 3 4 5];
  y=[[10 20 30 40 50];
  plot(x,y)
  clear,clc
%-- 8/27/2011 11:46 AM -
  x=[1 2 3 4 5];
  y=[10 20 30 40 50];
  plot(x,y)
```

First create a vector of x values – then a corresponding vector of y values

Note: The semicolon suppresses the output from each command

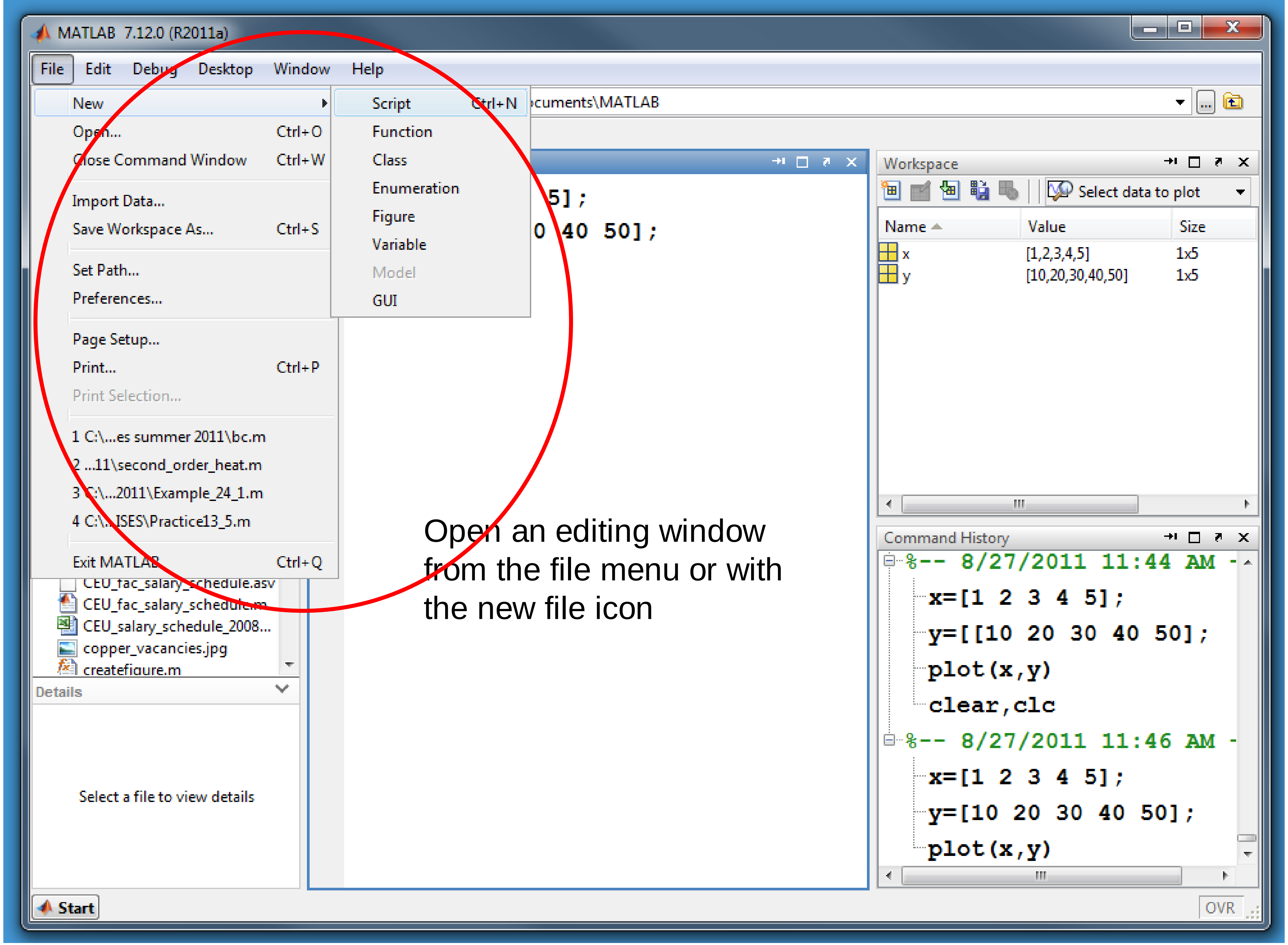


Matlab makes it easy to modify graphs by adding

- Titles
- Axis labels
- Legends
- Other types of annotations

Editing Window

- This window allows you to type and save a series of commands without executing them
- There are several ways to open an editing window
  - From the file menu
  - With the new file icon

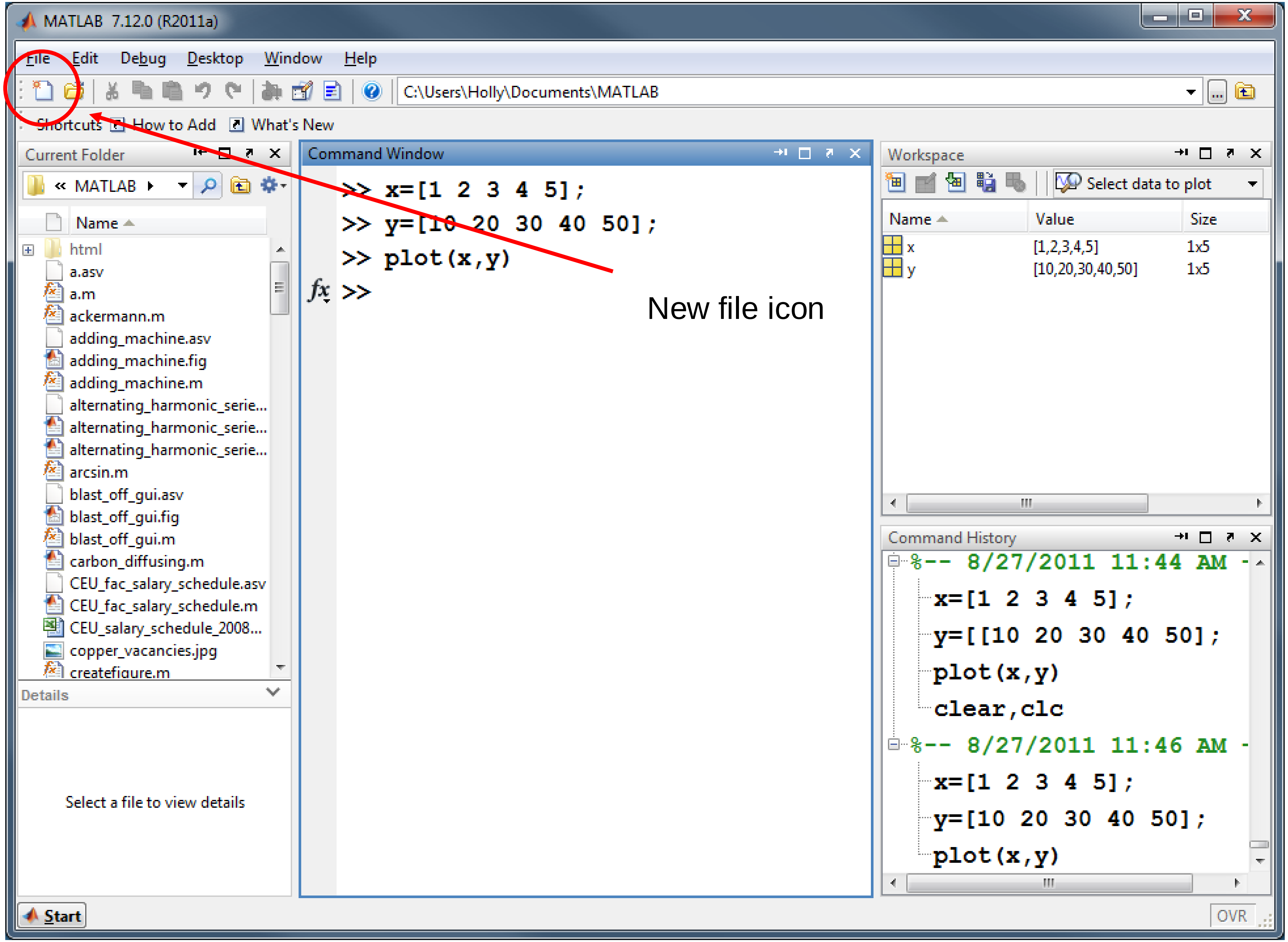


```
5];
0 40 50];
```

Name	Value	Size
x	[1,2,3,4,5]	1x5
y	[10,20,30,40,50]	1x5

```
%-- 8/27/2011 11:44 AM --
x=[1 2 3 4 5];
y=[[10 20 30 40 50];
plot(x,y)
clear,clc
%-- 8/27/2011 11:46 AM --
x=[1 2 3 4 5];
y=[10 20 30 40 50];
plot(x,y)
```

Open an editing window from the file menu or with the new file icon

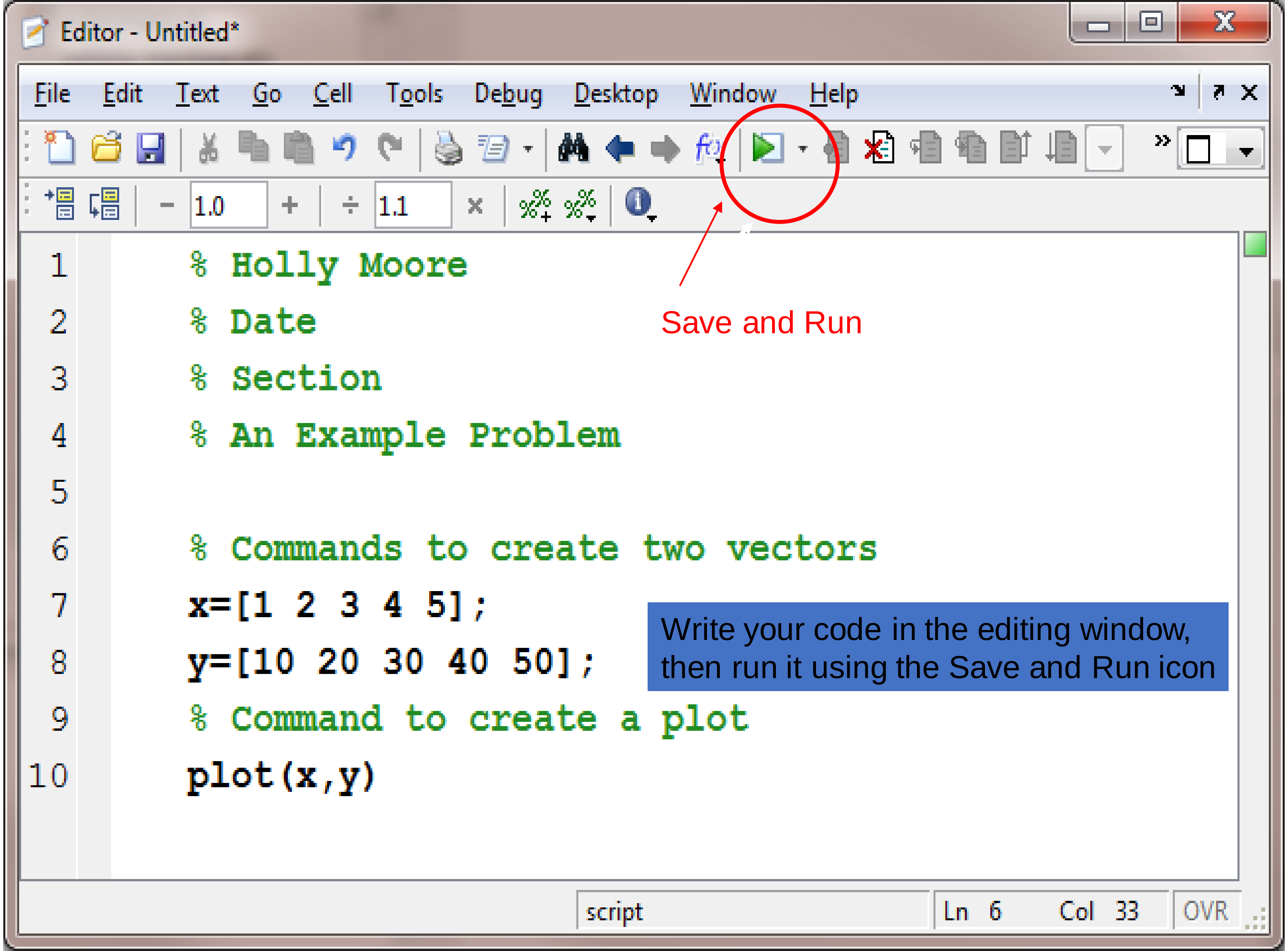


New file icon

```
>> x=[1 2 3 4 5];
>> y=[10 20 30 40 50];
>> plot(x,y)
fx >>
```

Name	Value	Size
x	[1,2,3,4,5]	1x5
y	[10,20,30,40,50]	1x5

```
%-- 8/27/2011 11:44 AM -
   x=[1 2 3 4 5];
   y=[[10 20 30 40 50];
   plot(x,y)
   clear,clc
%-- 8/27/2011 11:46 AM -
   x=[1 2 3 4 5];
   y=[10 20 30 40 50];
   plot(x,y)
```



```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x=[1 2 3 4 5];
y=[10 20 30 40 50];
% Command to create a plot
plot(x,y)
```

Save and Run

Write your code in the editing window, then run it using the Save and Run icon

Section 2.3
Solving Problems with MATLAB

- We've already solved some simple problems
- We need to understand how MATLAB works to solve more complicated problems

Variables

- MATLAB allows you to assign a value to a variable
- A=3
- Should be read as A is assigned a value of 3
- Use the variables in subsequent calculations

Naming Variables

- All names must start with a letter
- They may contain letters, numbers and the underscore (_)
- Names are case sensitive
- There are certain keywords you can’t use

Use the iskeyword function for a list of keywords

```
 iskeyword

ans =
    'break'                      'global'
    'case'                       'if'
    'catch'                      'otherwise'
    'classdef'                   'parfor'
    'continue'                   'persistent'
    'else'                       'return‘
    'elseif'                     ‘spmd’
    'end‘                        'switch'
    'for‘                        'try'
   'function'                    'while'
```

Keywords are not acceptable variable names

You can reassign function names

- MATLAB will let you use built-in function names as variables – but it’s a really bad idea
- sin = 3 changes sin from a function to a variable name
- clear sin resets sin back to a function

Practice Exercise 2.2
Which of these names are allowed in MATLAB?

- test
- Test
- if X
- my-book X
- my_book
- Thisisoneverylongnamebutisitstillallowed? X
- X 1stgroup
- group_one
- zzaAbc
- z34wAwy?12# X X
- sin
- log

}

bad idea

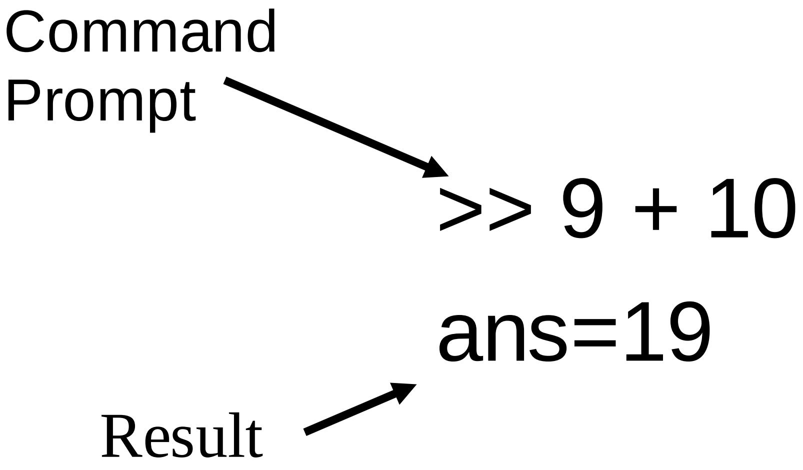
2.3.2 Matrices in MATLAB

The basic data type

- Group of numbers arranged into rows and columns
- Single Value (Scalar)
  - Matrix with one row and one column
- Vector (One dimensional matrix)
  - One row or one column
- Matrix (Two dimensional)

Scalar Calculations

- You can use MATLAB like you'd use a calculator



Command Prompt

```
>> 9 + 10
ans=19
```

Result

Assignment Operator

- To define a variable **a** we might type

  a=1+2
  which should be read as:
  “a is assigned a value of 1+2 “

How is the assignment operator different from an equality?

- In algebra the equation
  x=3+5
  means that both sides are the same
- In computers when we say
  x=3+5
  we are telling the machine to store the value on the right hand side of the equation in a memory location, and to name that location x

Is that really different?

- Yes!!!
- In algebra this is not a true statement
  x=x+1
- In computers (assignment statements) it means replace the value in the memory location named x, with a new value equal to x+1

Order of Operation

- Same as you've learned in math class
- Same as your calculator
  - Parentheses first
  - Exponentiation
  - Multiplication / division
  - Addition / subtraction

Order of Operation

5*(3+6)   = 45

5*3+6   = 21

White space does not matter!!!

5*3 + 6   = 21

Adding a space around + and – signs makes the expression more readable

Parentheses

- Use only ( )
- { } and [ ] mean something different
- MATLAB does not assume operators

5 * (3+4) not 5(3+4)

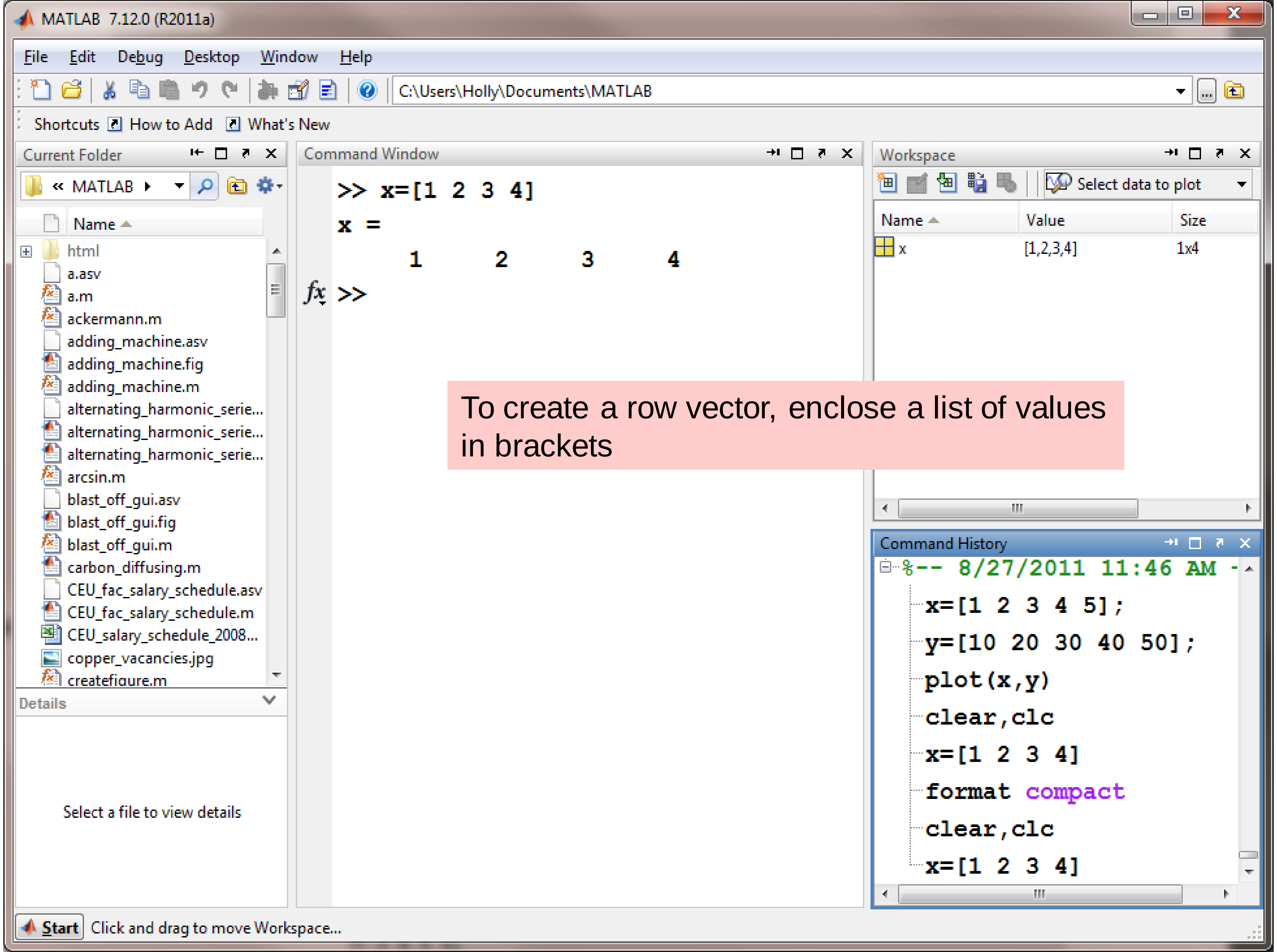
Compute from left to right

5*6/6*5   = 25

5*6/(6*5)   = 1

Array Operations

- Using MATLAB as a glorified calculator is OK, but its real strength is in matrix manipulations

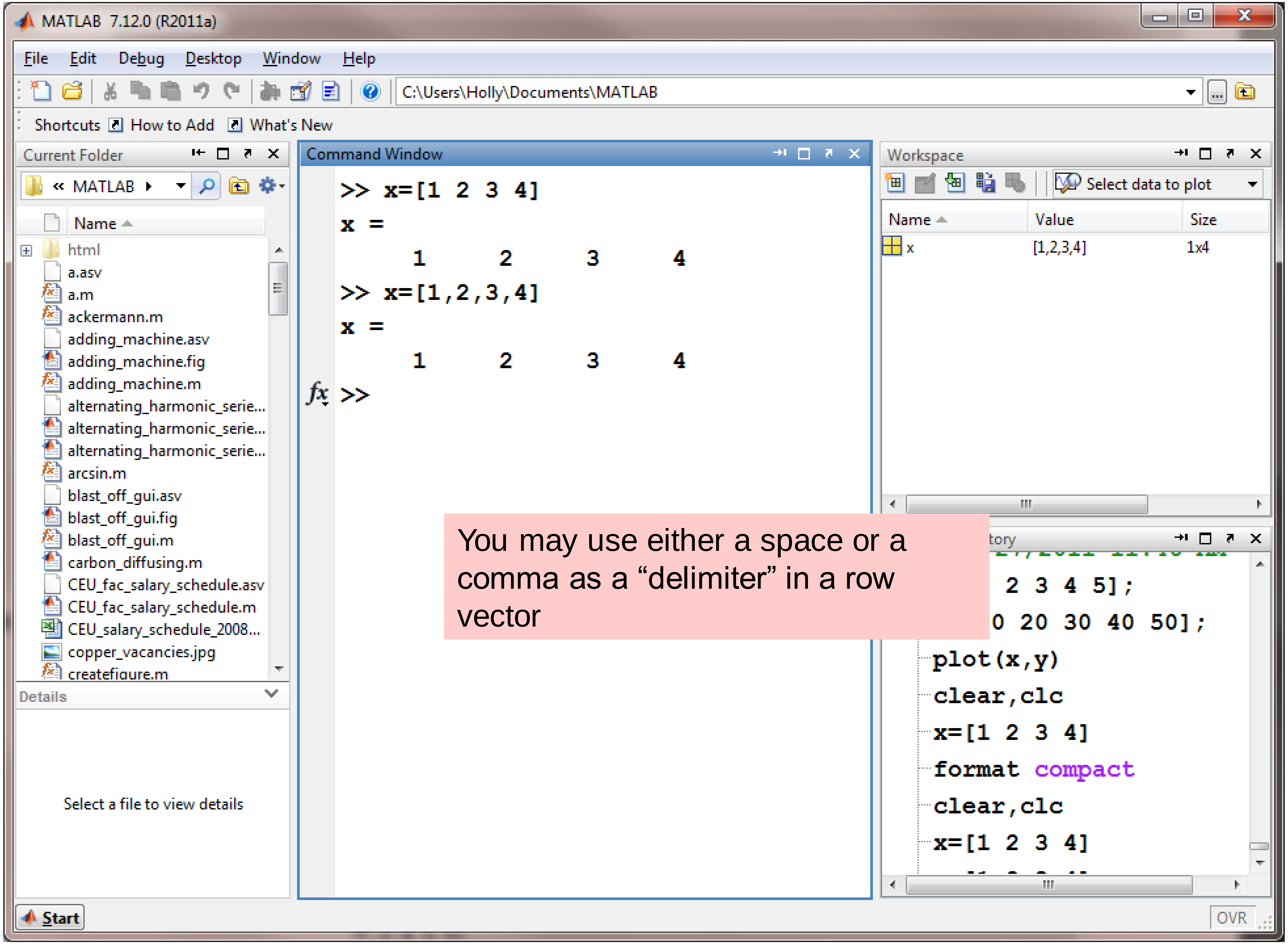

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
MATLAB
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
Details
Select a file to view details
Command Window
Workspace
Select data to plot
Command History
Start
Click and drag to move Workspace...

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

Name	Value	Size
x	[1,2,3,4]	1x4

```
%-- 8/27/2011 11:46 AM -
x=[1 2 3 4 5];
y=[10 20 30 40 50];
plot(x,y)
clear,clc
x=[1 2 3 4]
format compact
clear,clc
x=[1 2 3 4]
```

To create a row vector, enclose a list of values in brackets

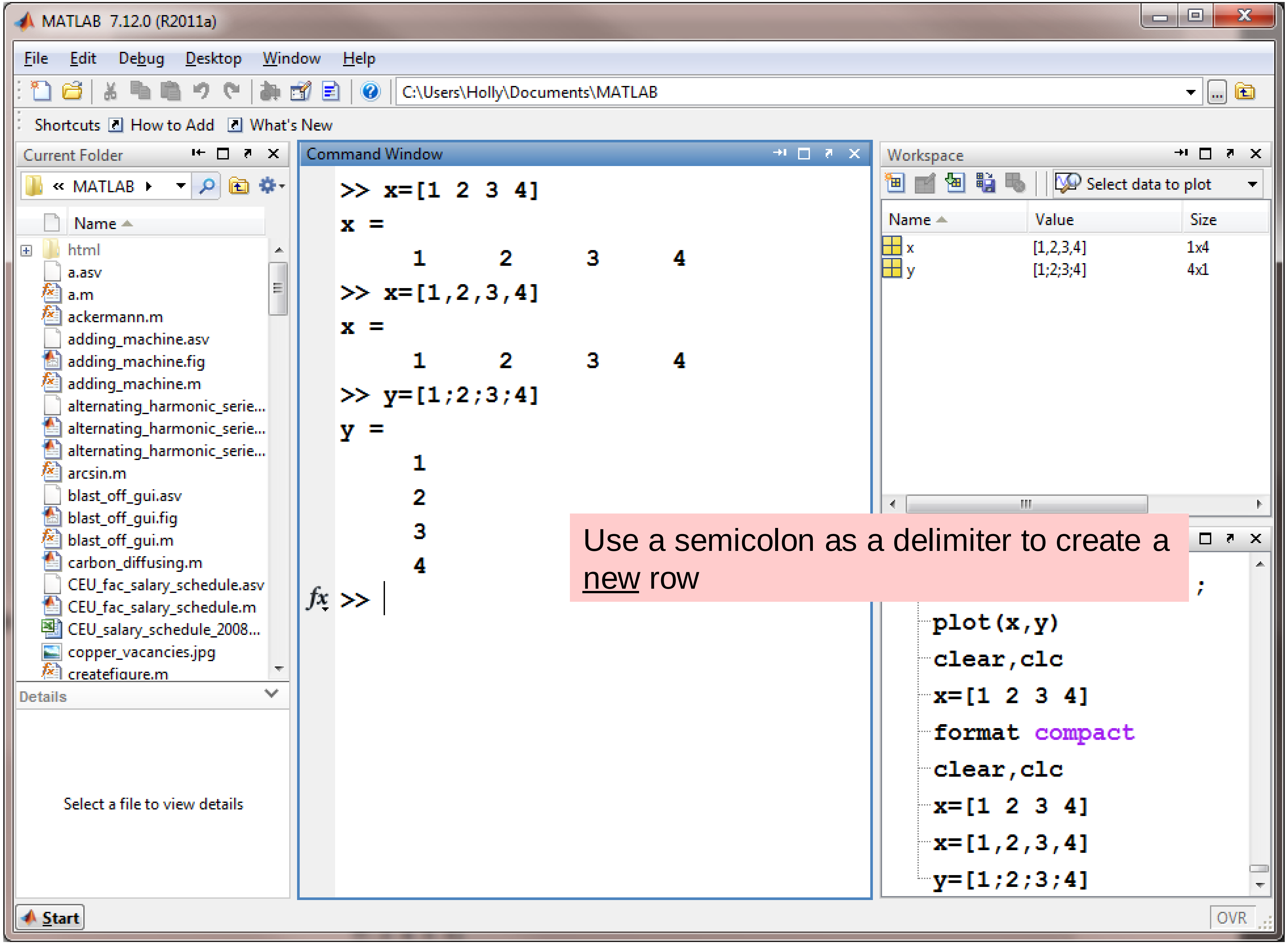


```
>> x=[1 2 3 4]
x =
     1     2     3     4
>> x=[1,2,3,4]
x =
     1     2     3     4
fx >>
```

Name	Value	Size
x	[1,2,3,4]	1x4

```
 2 3 4 5];
0 20 30 40 50];
plot(x,y)
clear,clc
x=[1 2 3 4]
format compact
clear,clc
x=[1 2 3 4]
```

You may use either a space or a comma as a "delimiter" in a row vector

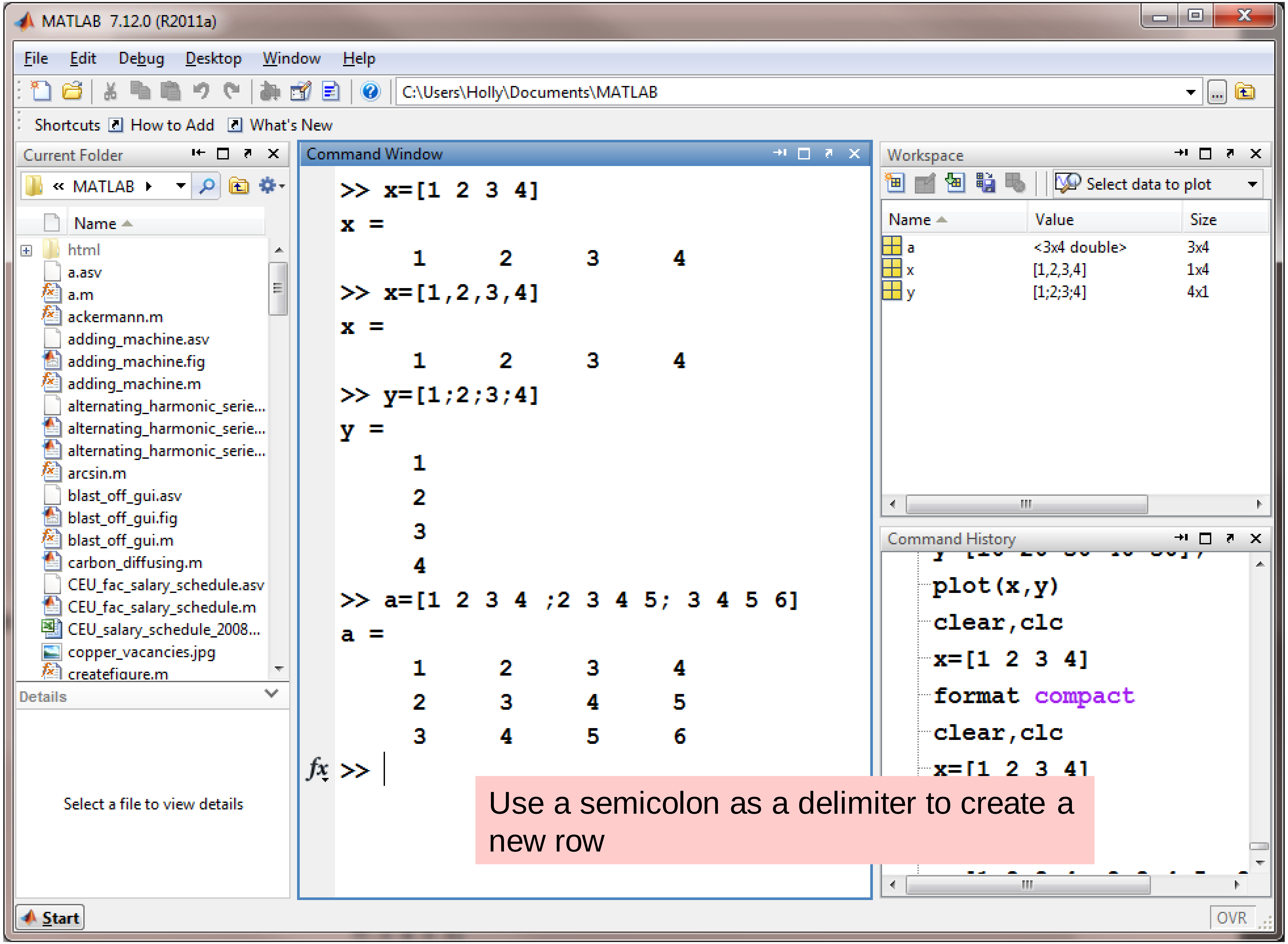

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
Details
Select a file to view details
Command Window
Workspace
Select data to plot
Start
OVR

```
>> x=[1 2 3 4]
x =
     1     2     3     4
>> x=[1,2,3,4]
x =
     1     2     3     4
>> y=[1;2;3;4]
y =
     1
     2
     3
     4
>> 
```

Name	Value	Size
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
;
plot(x,y)
clear,clc
x=[1 2 3 4]
format compact
clear,clc
x=[1 2 3 4]
x=[1,2,3,4]
y=[1;2;3;4]
```

Use a semicolon as a delimiter to create a new row


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
Details
Select a file to view details
Command Window
Workspace
Select data to plot
Command History
Start
OVR

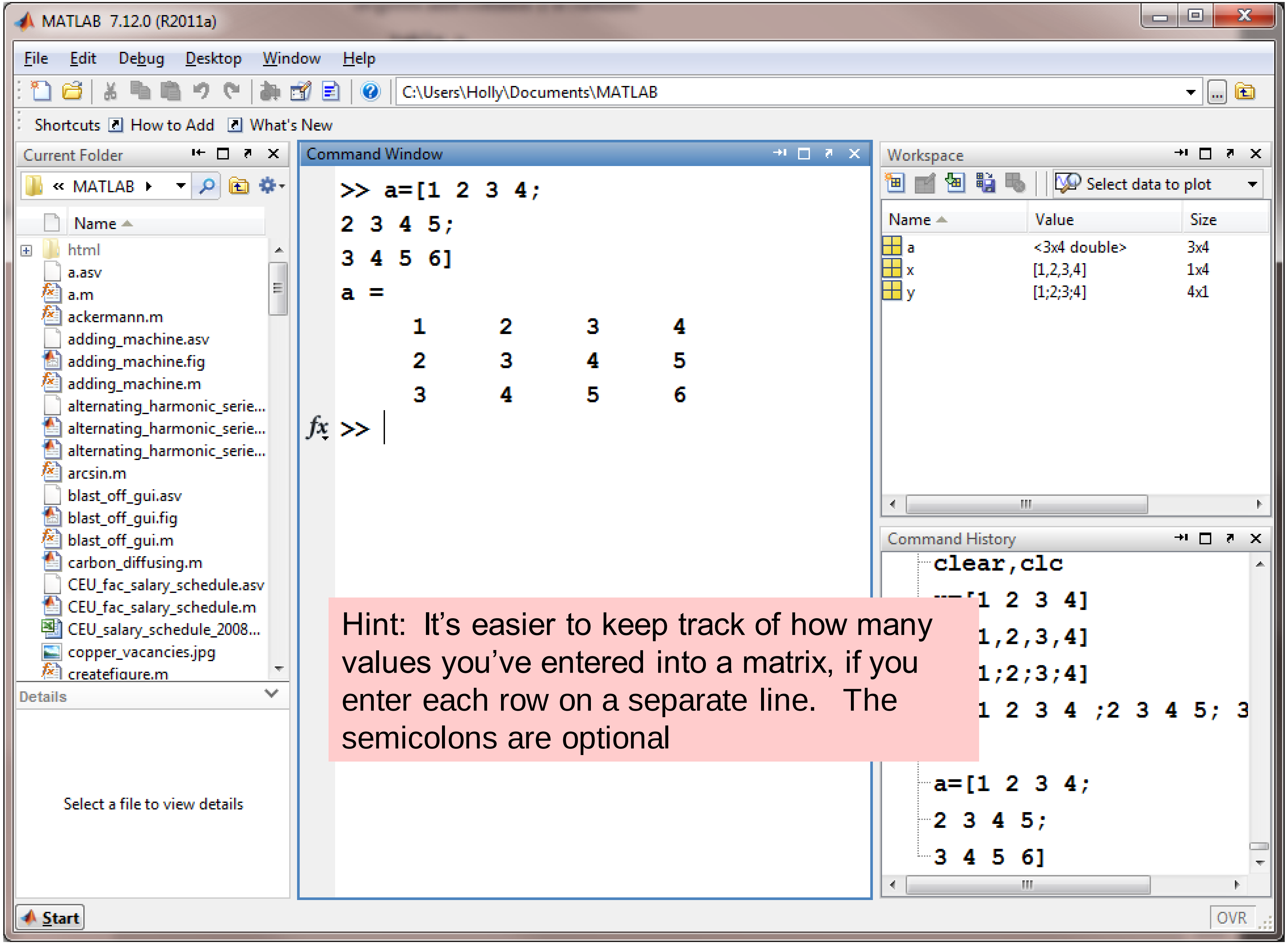
```
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
```

```
>> x=[1 2 3 4]
x =
     1     2     3     4
>> x=[1,2,3,4]
x =
     1     2     3     4
>> y=[1;2;3;4]
y =
     1
     2
     3
     4
>> a=[1 2 3 4 ;2 3 4 5; 3 4 5 6]
a =
     1     2     3     4
     2     3     4     5
     3     4     5     6
>>
```

Name	Value	Size
a	<3x4 double>	3x4
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
plot(x,y)
clear,clc
x=[1 2 3 4]
format compact
clear,clc
x=[1 2 3 4]
```

Use a semicolon as a delimiter to create a new row



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

Name ▲	Value	Size
a	<3x4 double>	3x4
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
clear,clc
x=[1 2 3 4]
1,2,3,4]
1;2;3;4]
1 2 3 4 ;2 3 4 5; 3
a=[1 2 3 4;
2 3 4 5;
3 4 5 6]
```

Hint: It's easier to keep track of how many values you've entered into a matrix, if you enter each row on a separate line. The semicolons are optional

Shortcuts

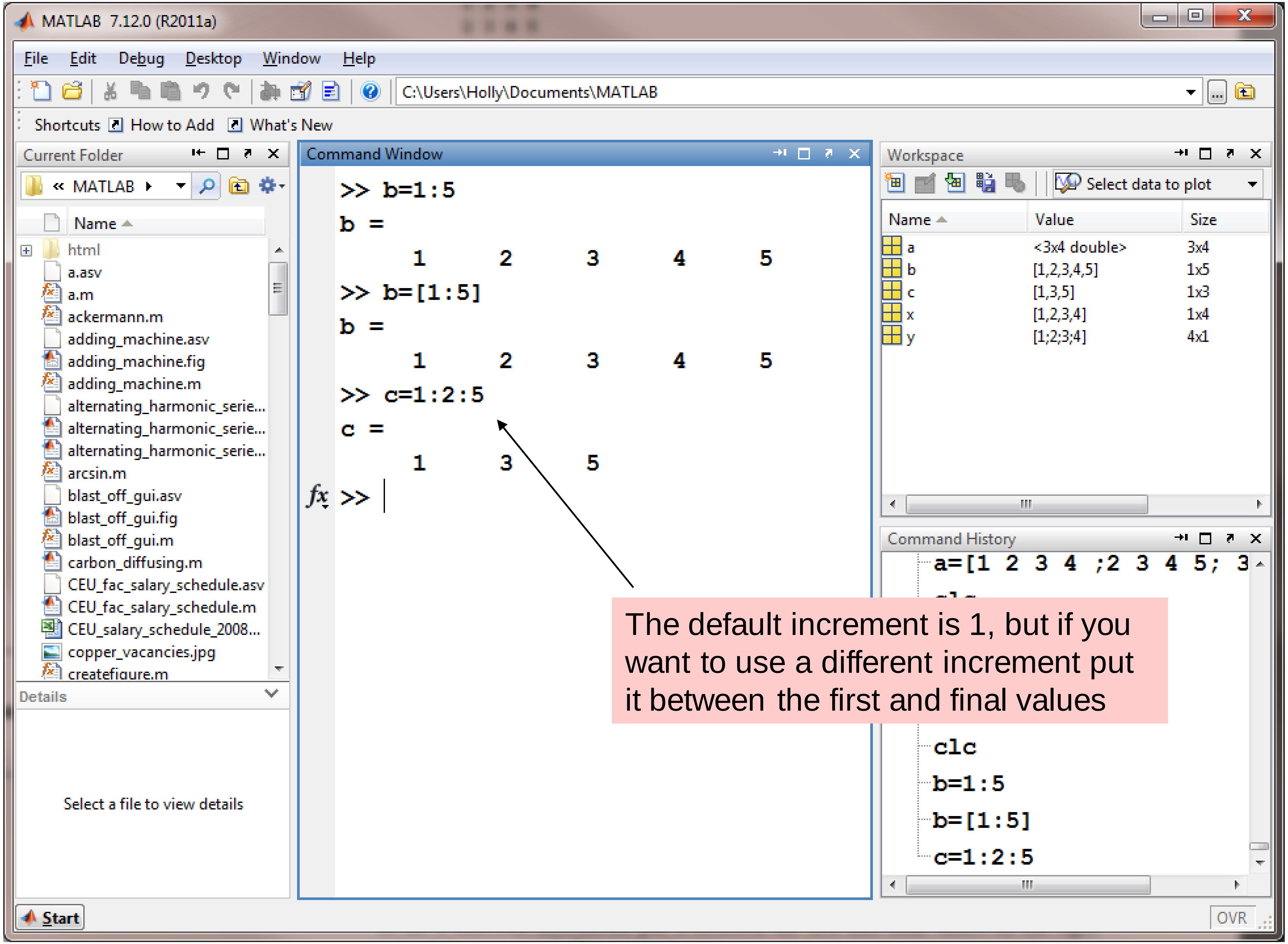
- While a complicated matrix might have to be entered by hand, evenly spaced matrices can be entered much more readily. The command

  b= 1:5

  or the command

  b = [1:5]

  both return a row matrix



C:\Users\Holly\Documents\MATLAB

```
>> b=1:5
b =
     1     2     3     4     5
>> b=[1:5]
b =
     1     2     3     4     5
>> c=1:2:5
c =
     1     3     5
fx >> 
```

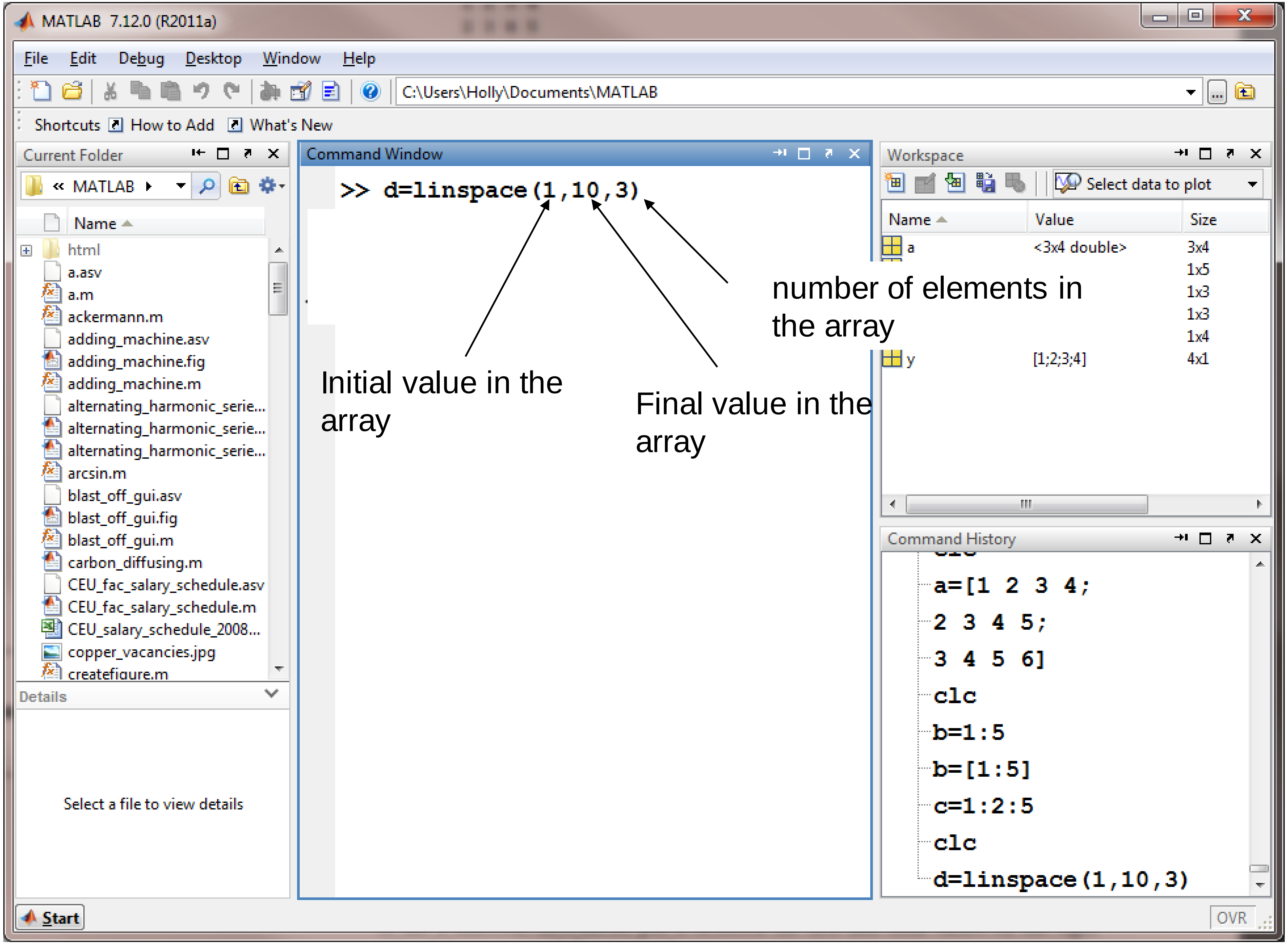
Name	Value	Size
a	<3x4 double>	3x4
b	[1,2,3,4,5]	1x5
c	[1,3,5]	1x3
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
a=[1 2 3 4 ;2 3 4 5; 3
clc
clc
b=1:5
b=[1:5]
c=1:2:5
```

The default increment is 1, but if you want to use a different increment put it between the first and final values

To calculate spacing between elements use…

- linspace
- logspace



```
>> d=linspace(1,10,3)
```

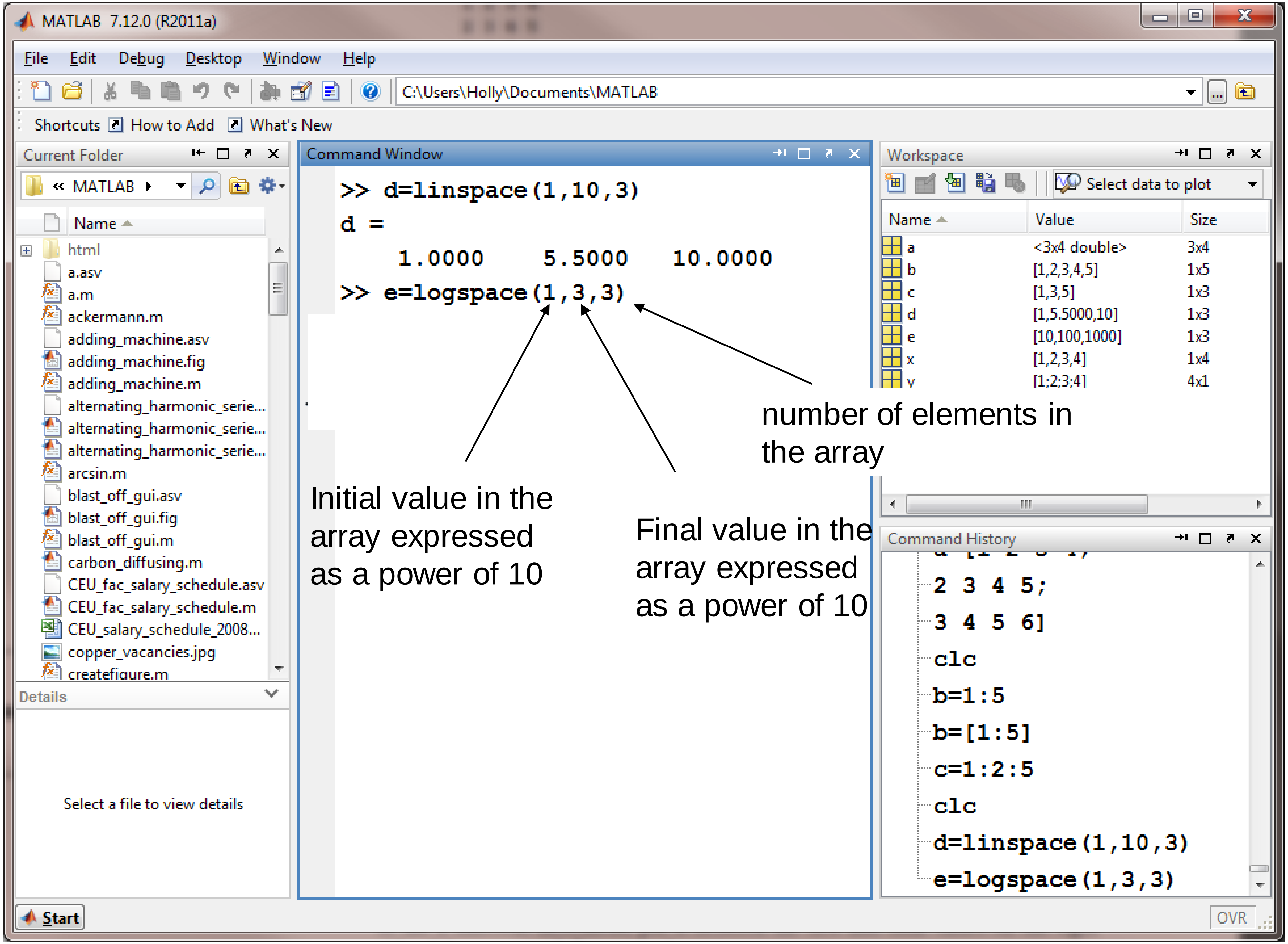
Initial value in the array

Final value in the array

number of elements in the array

Name	Value	Size
a	<3x4 double>	3x4
		1x5
		1x3
		1x3
		1x4
y	[1;2;3;4]	4x1

```
clc
a=[1 2 3 4;
2 3 4 5;
3 4 5 6]
clc
b=1:5
b=[1:5]
c=1:2:5
clc
d=linspace(1,10,3)
```



```
>> d=linspace(1,10,3)
d =
     1.0000     5.5000    10.0000
>> e=logspace(1,3,3)
```

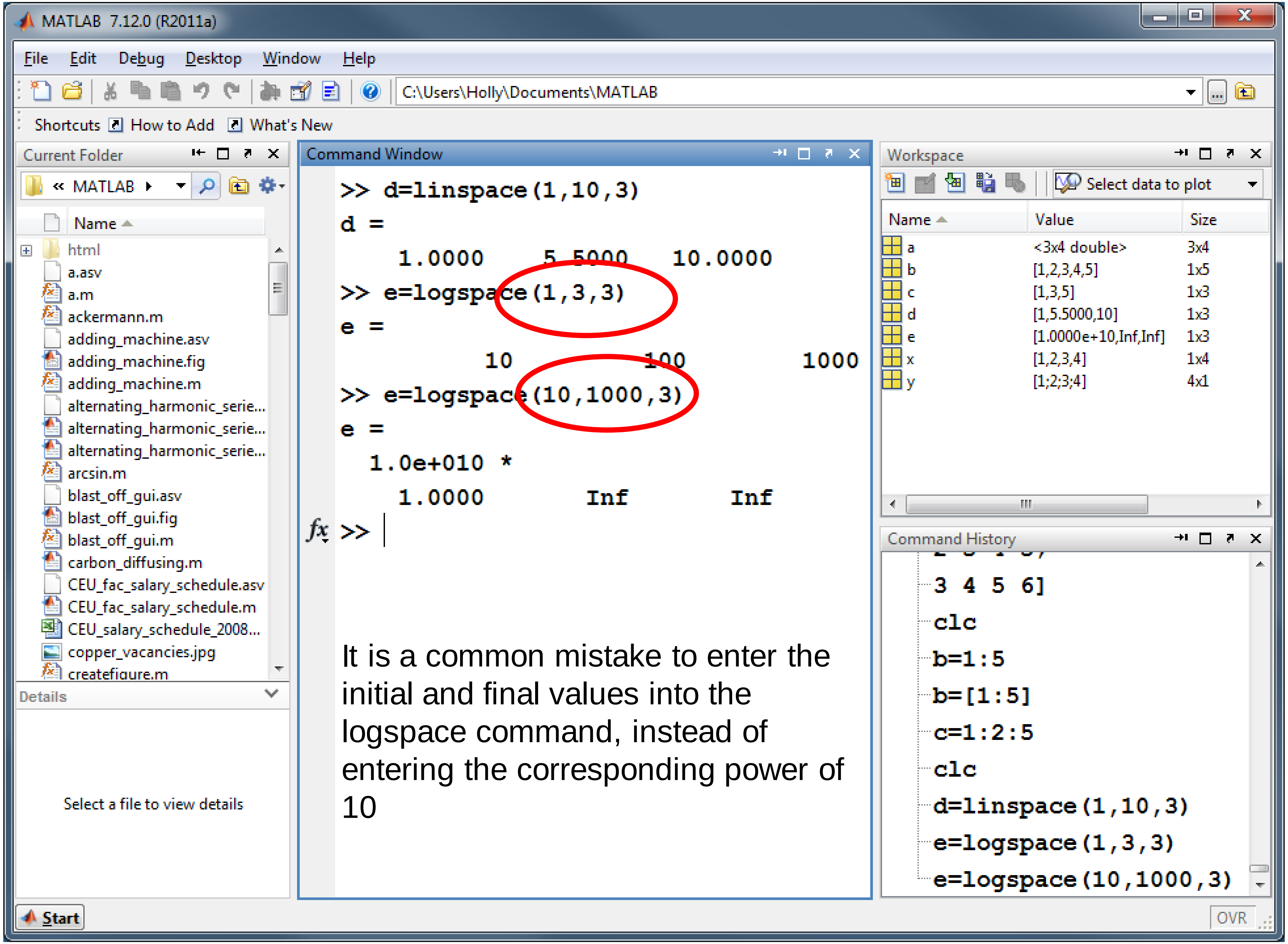
Name	Value	Size
a	<3x4 double>	3x4
b	[1,2,3,4,5]	1x5
c	[1,3,5]	1x3
d	[1,5.5000,10]	1x3
e	[10,100,1000]	1x3
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
a=[1 2 3 4;
2 3 4 5;
3 4 5 6]
clc
b=1:5
b=[1:5]
c=1:2:5
clc
d=linspace(1,10,3)
e=logspace(1,3,3)
```

Initial value in the array expressed as a power of 10

Final value in the array expressed as a power of 10

number of elements in the array



```
>> d=linspace(1,10,3)
d =
    1.0000    5.5000   10.0000
>> e=logspace(1,3,3)
e =
          10         100        1000
>> e=logspace(10,1000,3)
e =
  1.0e+010 *
    1.0000       Inf       Inf
>> 
```

Name	Value	Size
a	<3x4 double>	3x4
b	[1,2,3,4,5]	1x5
c	[1,3,5]	1x3
d	[1,5.5000,10]	1x3
e	[1.0000e+10,Inf,Inf]	1x3
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
3 4 5 6]
clc
b=1:5
b=[1:5]
c=1:2:5
clc
d=linspace(1,10,3)
e=logspace(1,3,3)
e=logspace(10,1000,3)
```

It is a common mistake to enter the initial and final values into the logspace command, instead of entering the corresponding power of 10

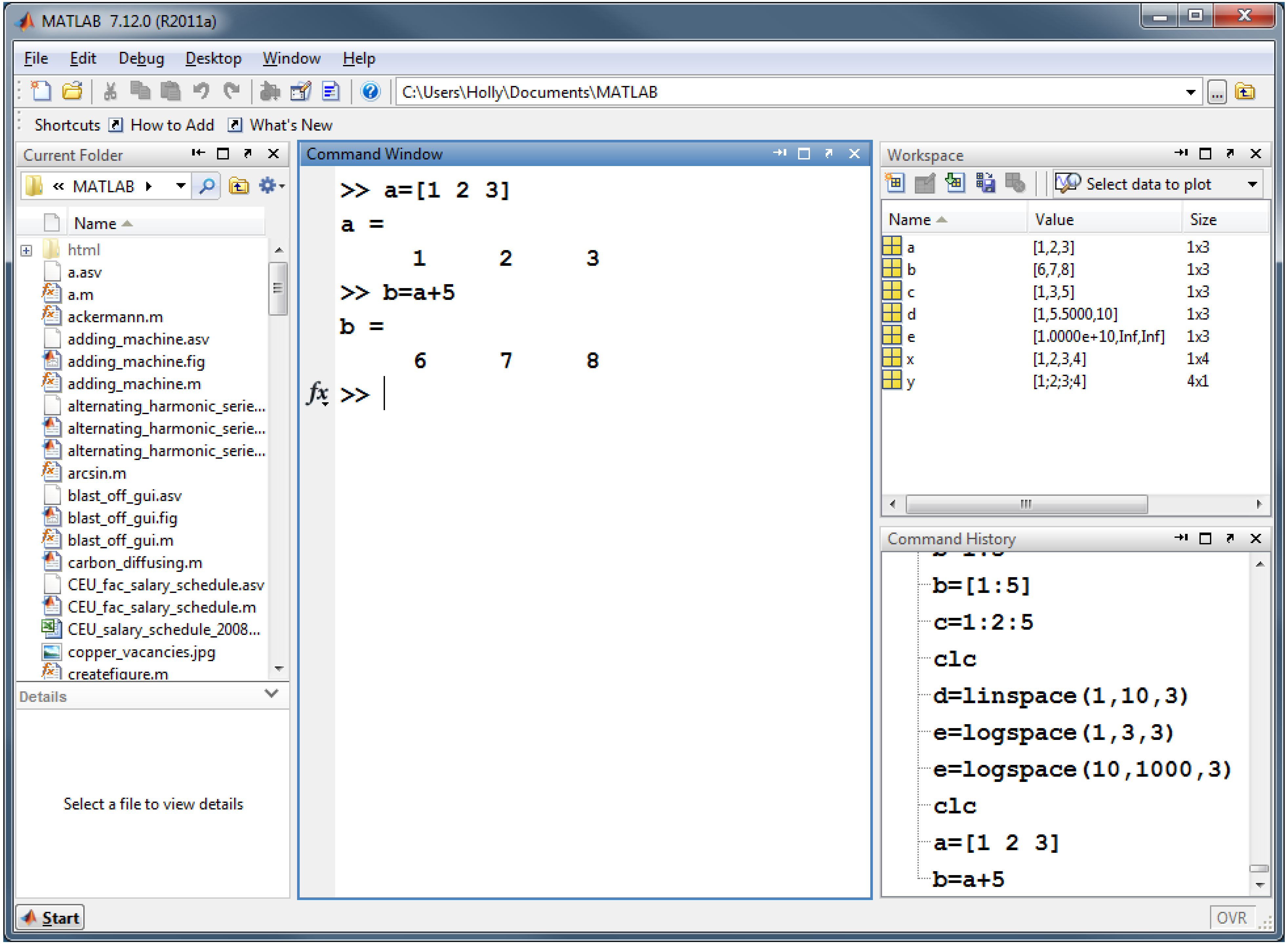
Hint

- You can include mathematical operations inside a matrix definition statement.
- For example

```
a = [0: pi/10: pi]
```

Mixed calculations between scalars and arrays

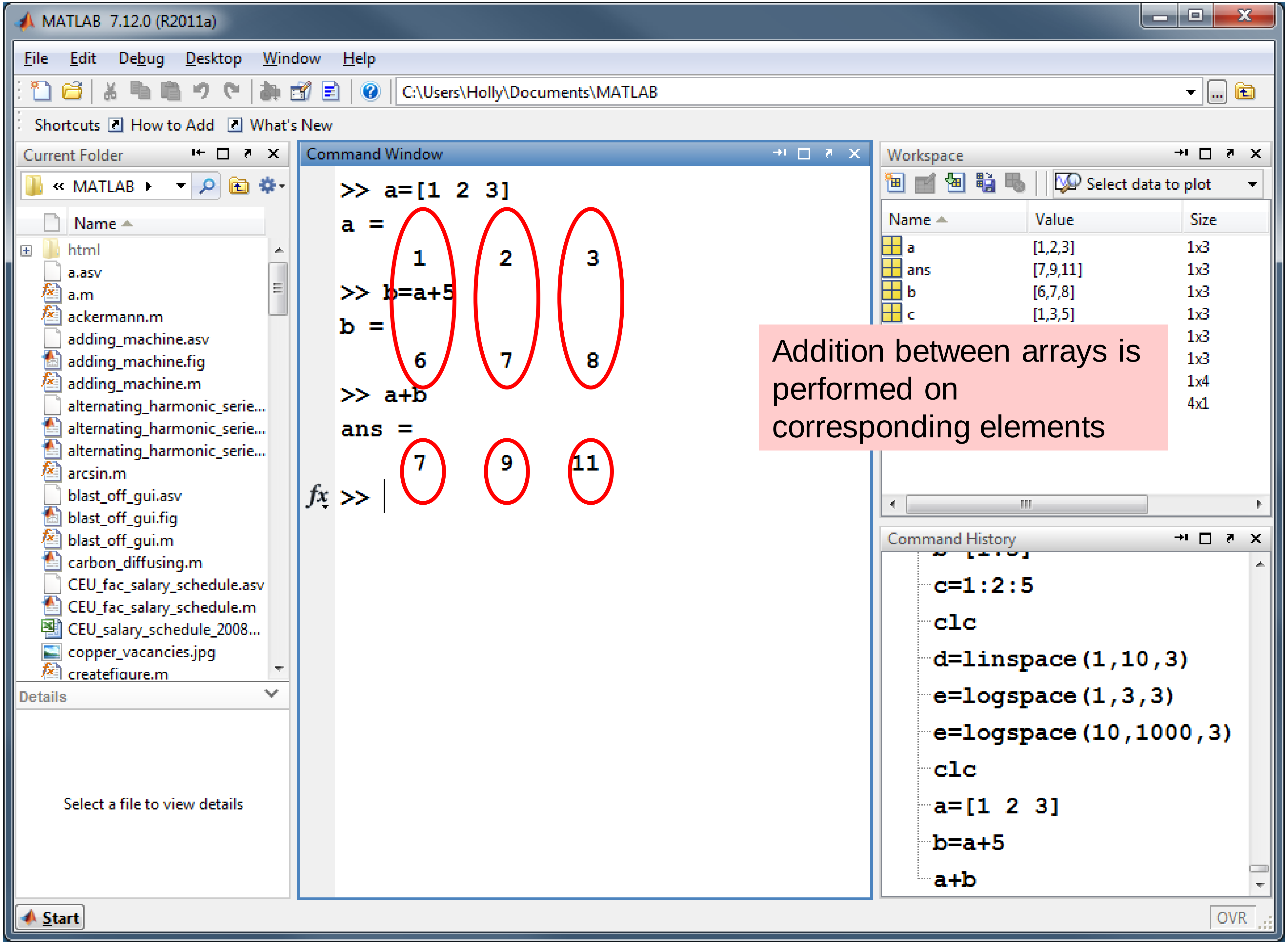
- Matrices can be used in many calculations with scalars
- There is no confusion when we perform addition and subtraction
- Multiplication and division are a little different
- In matrix mathematics the multiplication operator (*) has a very specific meaning



```
>> a=[1 2 3]
a =
     1     2     3
>> b=a+5
b =
     6     7     8
>>
```

Name	Value	Size
a	[1,2,3]	1x3
b	[6,7,8]	1x3
c	[1,3,5]	1x3
d	[1,5.5000,10]	1x3
e	[1.0000e+10,Inf,Inf]	1x3
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

```
b=[1:5]
c=1:2:5
clc
d=linspace(1,10,3)
e=logspace(1,3,3)
e=logspace(10,1000,3)
clc
a=[1 2 3]
b=a+5
```

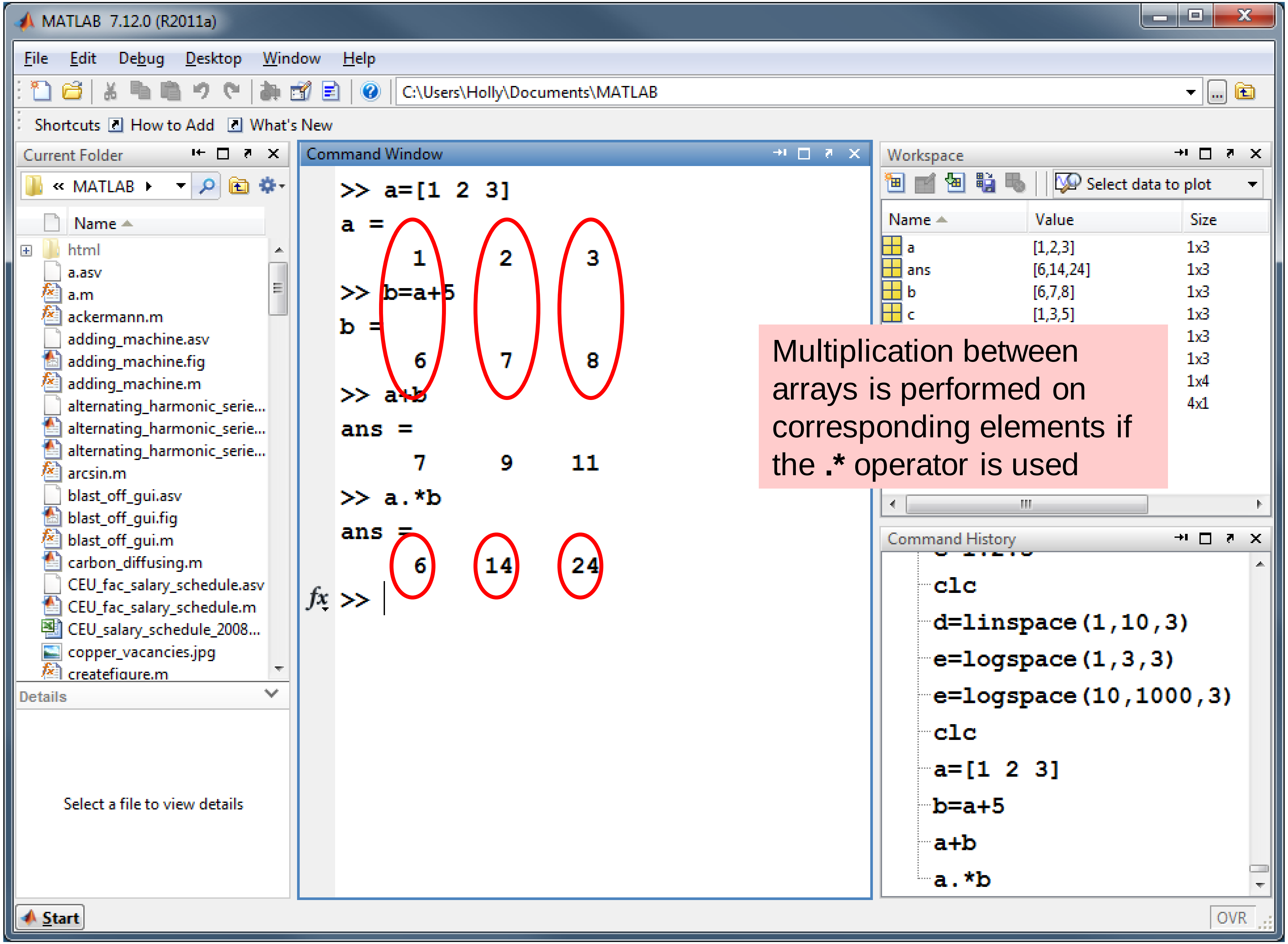


```
>> a=[1 2 3]
a =
     1     2     3

>> b=a+5
b =
     6     7     8

>> a+b
ans =
     7     9    11

fx >> 
```

Name	Value	Size
a	[1,2,3]	1x3
ans	[7,9,11]	1x3
b	[6,7,8]	1x3
c	[1,3,5]	1x3
		1x3
		1x3
		1x4
		4x1

```
c=1:2:5
clc
d=linspace(1,10,3)
e=logspace(1,3,3)
e=logspace(10,1000,3)
clc
a=[1 2 3]
b=a+5
a+b
```

Addition between arrays is performed on corresponding elements

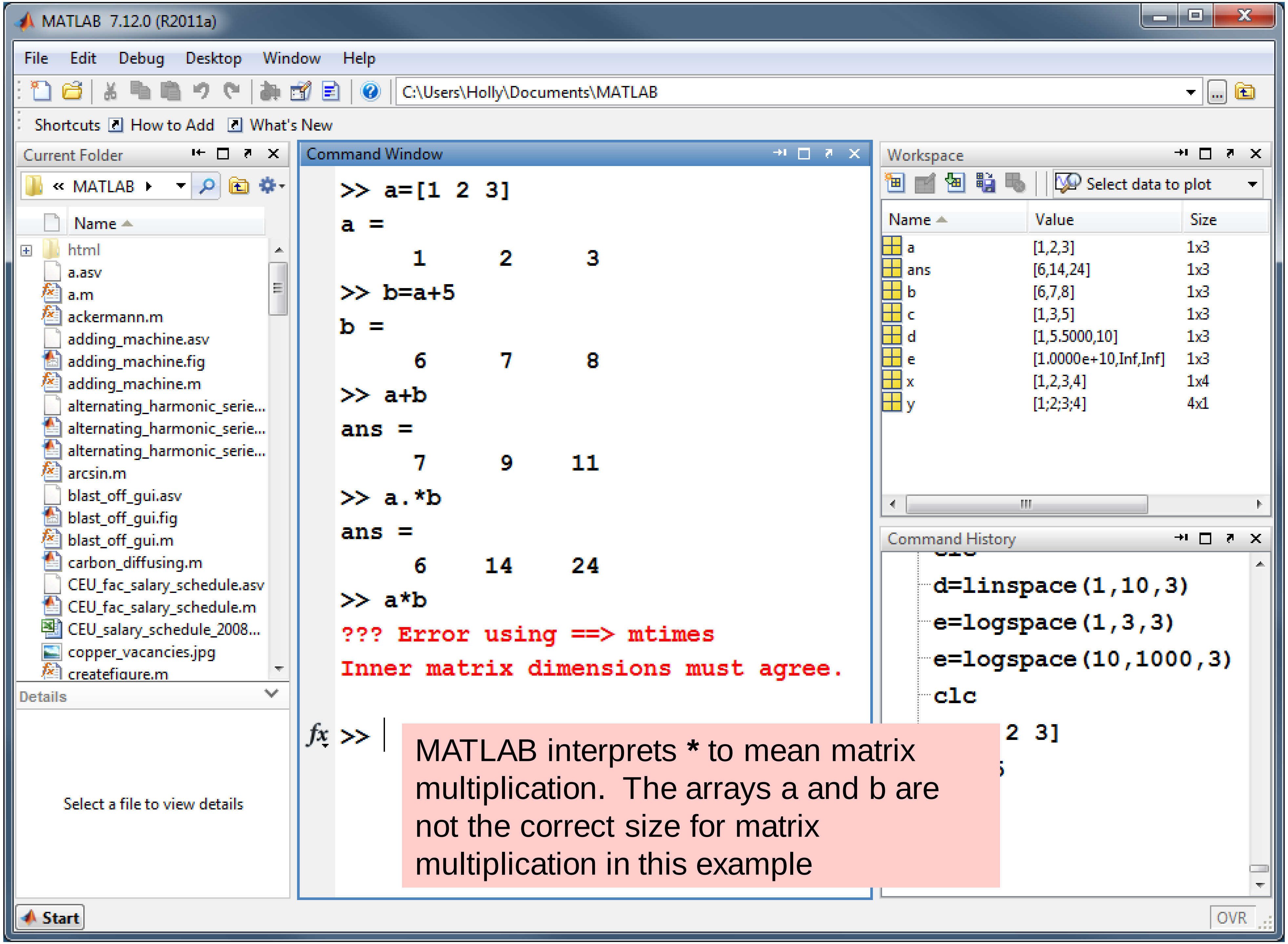


```
>> a=[1 2 3]
a =

     1     2     3

>> b=a+5
b =

     6     7     8

>> a+b
ans =

     7     9    11

>> a.*b
ans =

     6    14    24

fx >> 
```

Name ▲	Value	Size
a	[1,2,3]	1x3
ans	[6,14,24]	1x3
b	[6,7,8]	1x3
c	[1,3,5]	1x3
		1x3
		1x3
		1x4
		4x1

```
clc
d=linspace(1,10,3)
e=logspace(1,3,3)
e=logspace(10,1000,3)
clc
a=[1 2 3]
b=a+5
a+b
a.*b
```

Multiplication between arrays is performed on corresponding elements if the .* operator is used


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
Details
Select a file to view details
Command Window
Workspace
Select data to plot
Command History
Start
OVR

Current Folder:

- html
- a.asv
- a.m
- ackermann.m
- adding_machine.asv
- adding_machine.fig
- adding_machine.m
- alternating_harmonic_serie...
- alternating_harmonic_serie...
- alternating_harmonic_serie...
- arcsin.m
- blast_off_gui.asv
- blast_off_gui.fig
- blast_off_gui.m
- carbon_diffusing.m
- CEU_fac_salary_schedule.asv
- CEU_fac_salary_schedule.m
- CEU_salary_schedule_2008...
- copper_vacancies.jpg
- createfigure.m

Command Window:

```
>> a=[1 2 3]
a =
     1     2     3
>> b=a+5
b =
     6     7     8
>> a+b
ans =
     7     9    11
>> a.*b
ans =
     6    14    24
>> a*b
??? Error using ==> mtimes
Inner matrix dimensions must agree.

fx >>
```

Workspace:

Name	Value	Size
a	[1,2,3]	1x3
ans	[6,14,24]	1x3
b	[6,7,8]	1x3
c	[1,3,5]	1x3
d	[1,5.5000,10]	1x3
e	[1.0000e+10,Inf,Inf]	1x3
x	[1,2,3,4]	1x4
y	[1;2;3;4]	4x1

Command History:

```
clc
d=linspace(1,10,3)
e=logspace(1,3,3)
e=logspace(10,1000,3)
clc
        2 3]
5
```

MATLAB interprets * to mean matrix multiplication. The arrays a and b are not the correct size for matrix multiplication in this example

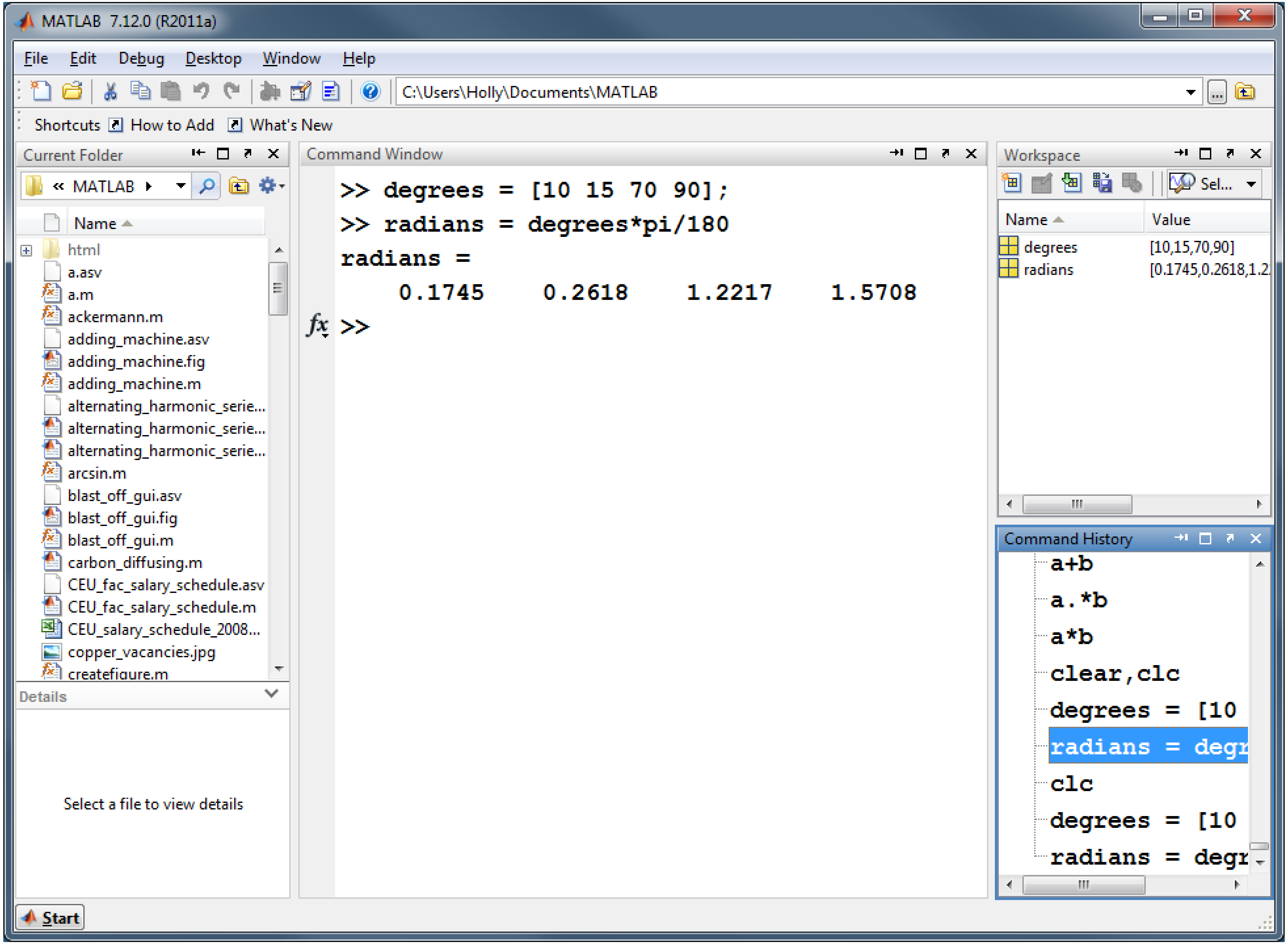
Array Operations

- Array multiplication .*
- Array division ./
- Array exponentiation .^

In each case the size of the arrays must match

- The matrix capability of MATLAB makes it easy to do repetitive calculations

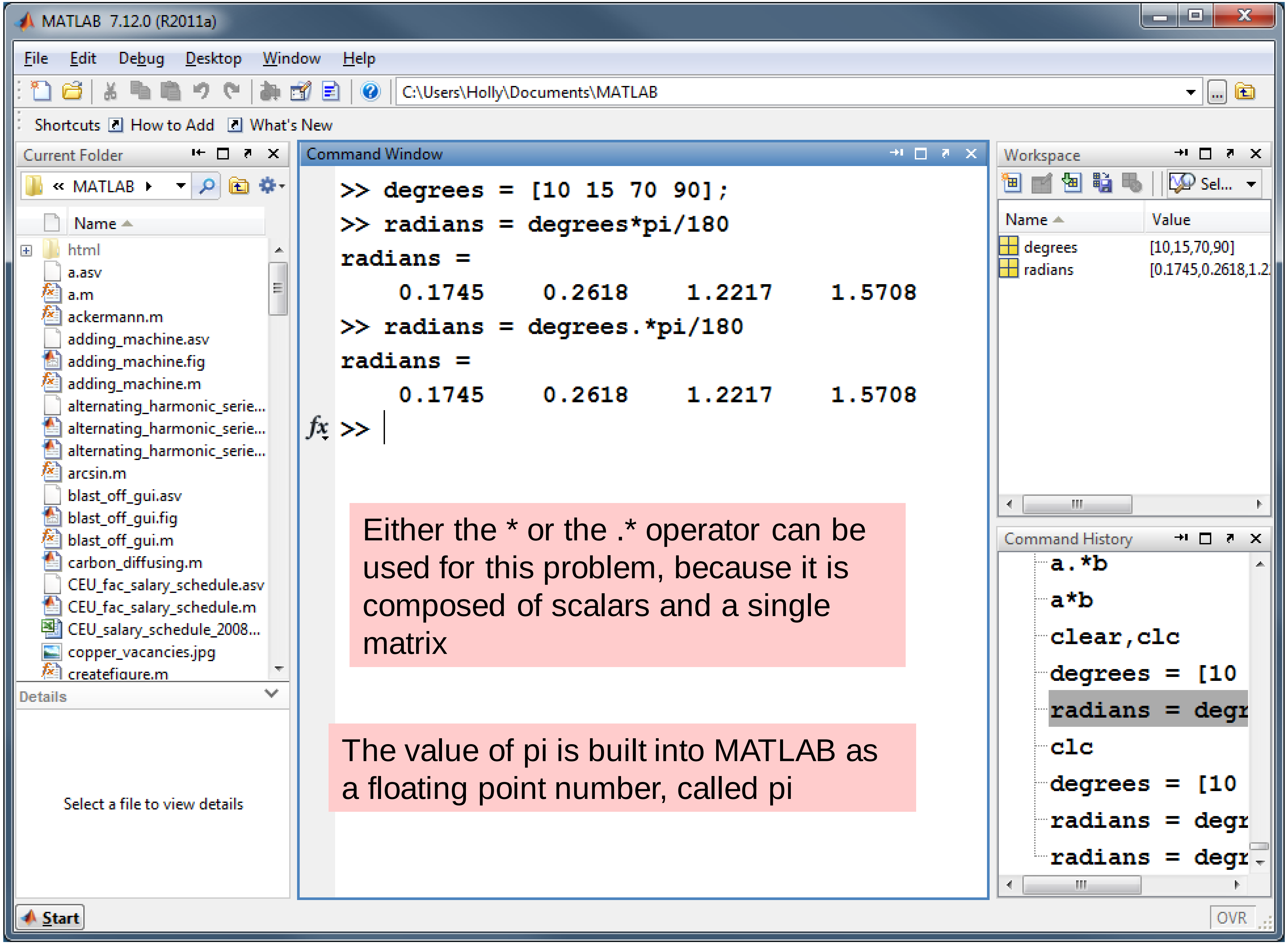
- For example, assume you have a list of angles in degrees that you would like to convert to radians.
  - First put the values into a matrix.
  - Perform the calculation



```
>> degrees = [10 15 70 90];
>> radians = degrees*pi/180
radians =
    0.1745    0.2618    1.2217    1.5708
>>
```

Name	Value
degrees	[10,15,70,90]
radians	[0.1745,0.2618,1.2

```
a+b
a.*b
a*b
clear,clc
degrees = [10
radians = degr
clc
degrees = [10
radians = degr
```



```
>> degrees = [10 15 70 90];
>> radians = degrees*pi/180
radians =
    0.1745    0.2618    1.2217    1.5708
>> radians = degrees.*pi/180
radians =
    0.1745    0.2618    1.2217    1.5708
fx >> 
```

Name ▲	Value
degrees	[10,15,70,90]
radians	[0.1745,0.2618,1.2

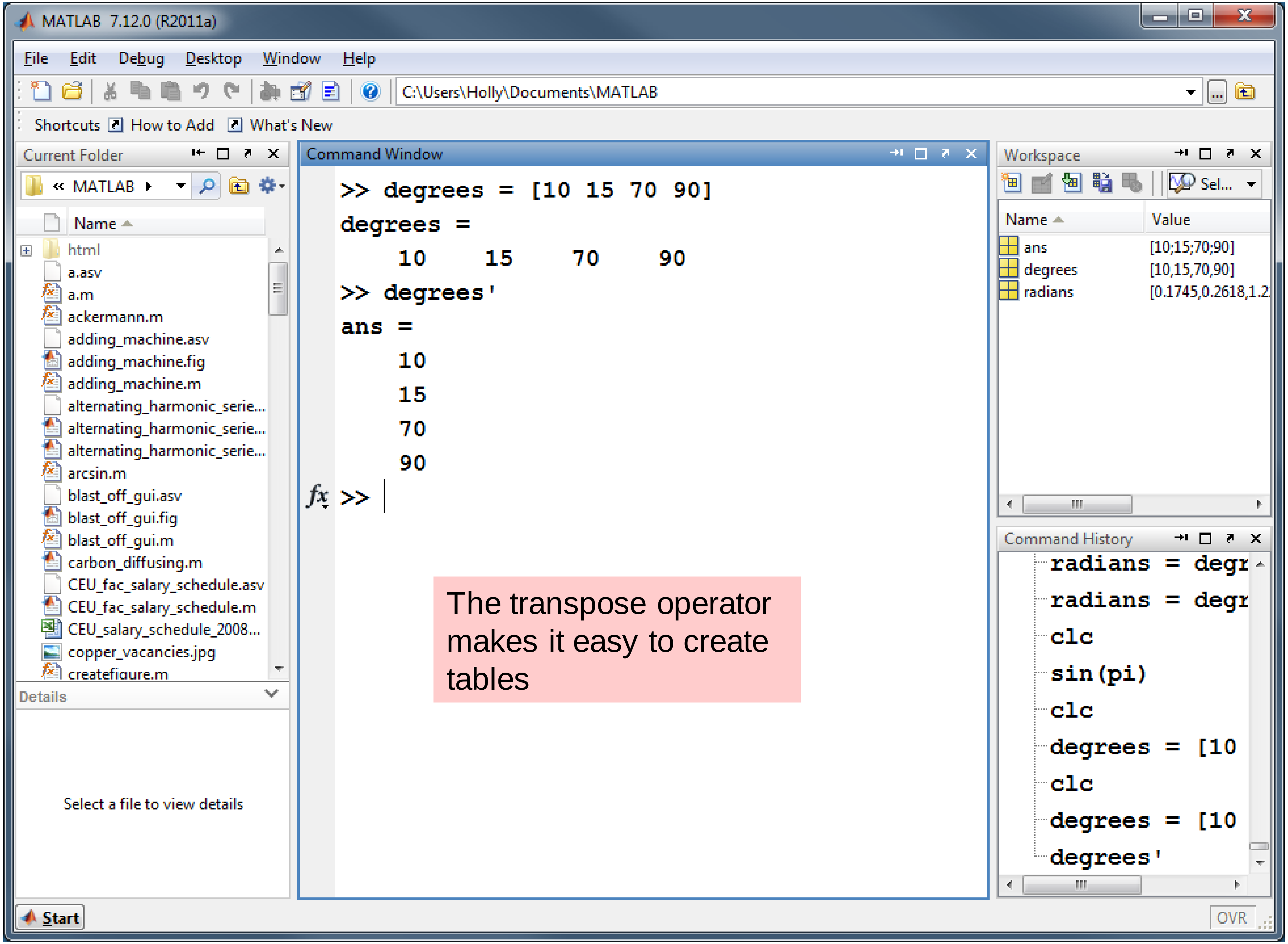
```
a.*b
a*b
clear,clc
degrees = [10
radians = degr
clc
degrees = [10
radians = degr
radians = degr
```

Either the * or the .* operator can be used for this problem, because it is composed of scalars and a single matrix

The value of pi is built into MATLAB as a floating point number, called pi

Transpose

- The transpose operator changes rows to columns or vice versa.

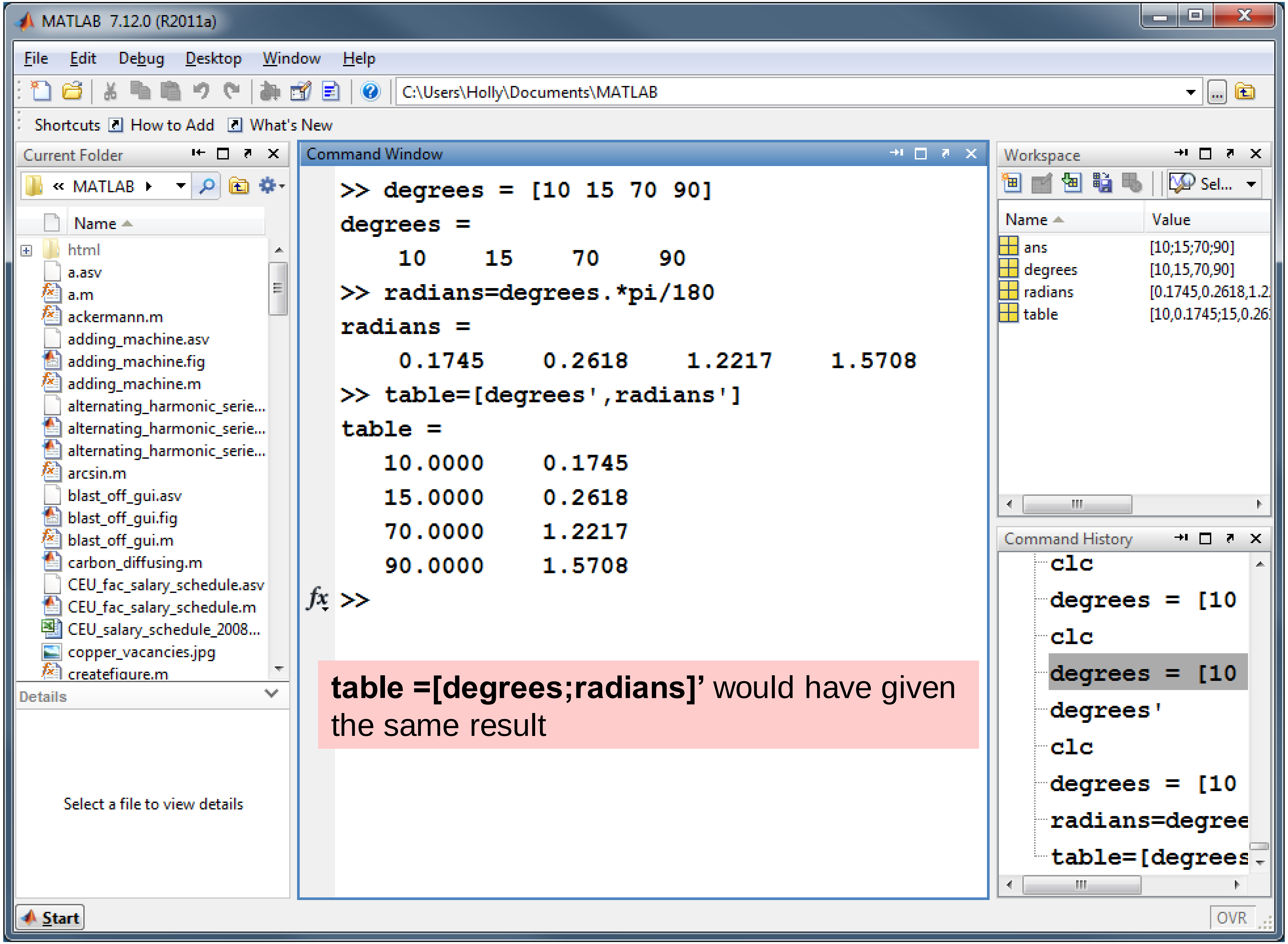


```
>> degrees = [10 15 70 90]
degrees =
    10    15    70    90
>> degrees'
ans =
    10
    15
    70
    90
>> 
```

Name	Value
ans	[10;15;70;90]
degrees	[10,15,70,90]
radians	[0.1745,0.2618,1.2

```
radians = degr
radians = degr
clc
sin(pi)
clc
degrees = [10
clc
degrees = [10
degrees'
```

The transpose operator makes it easy to create tables

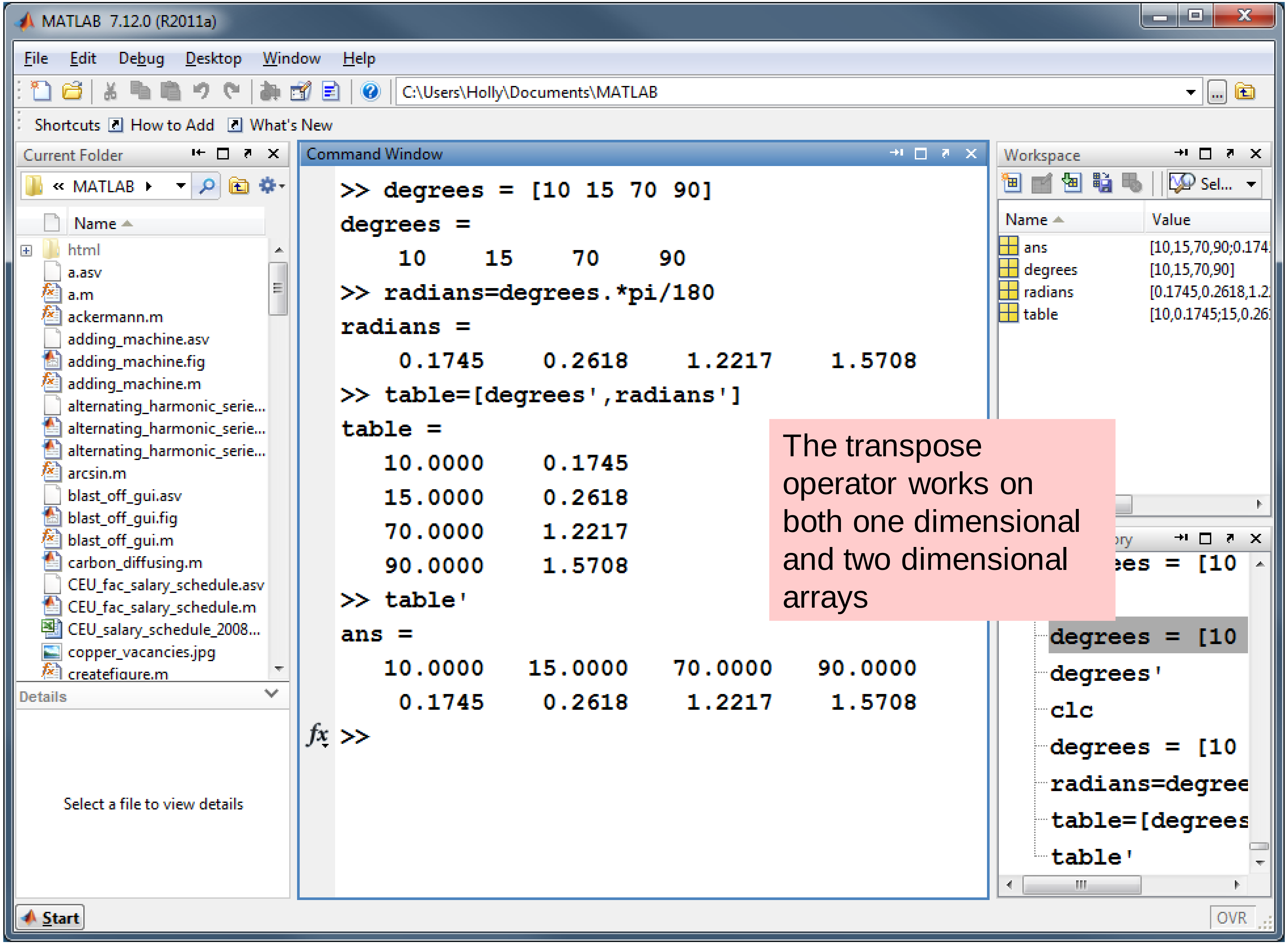


```
>> degrees = [10 15 70 90]
degrees =
    10    15    70    90
>> radians=degrees.*pi/180
radians =
    0.1745    0.2618    1.2217    1.5708
>> table=[degrees',radians']
table =
   10.0000    0.1745
   15.0000    0.2618
   70.0000    1.2217
   90.0000    1.5708
fx >>
```

Name ▲	Value
ans	[10;15;70;90]
degrees	[10,15,70,90]
radians	[0.1745,0.2618,1.2
table	[10,0.1745;15,0.26

```
clc
degrees = [10
clc
degrees = [10
degrees'
clc
degrees = [10
radians=degree
table=[degrees
```

table =[degrees;radians]' would have given the same result


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name ▲
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
Details
Select a file to view details
Command Window
Workspace
Sel...
Start
OVR

```
>> degrees = [10 15 70 90]
degrees =
    10    15    70    90
>> radians=degrees.*pi/180
radians =
    0.1745    0.2618    1.2217    1.5708
>> table=[degrees',radians']
table =
   10.0000    0.1745
   15.0000    0.2618
   70.0000    1.2217
   90.0000    1.5708
>> table'
ans =
   10.0000   15.0000   70.0000   90.0000
    0.1745    0.2618    1.2217    1.5708
>>
```

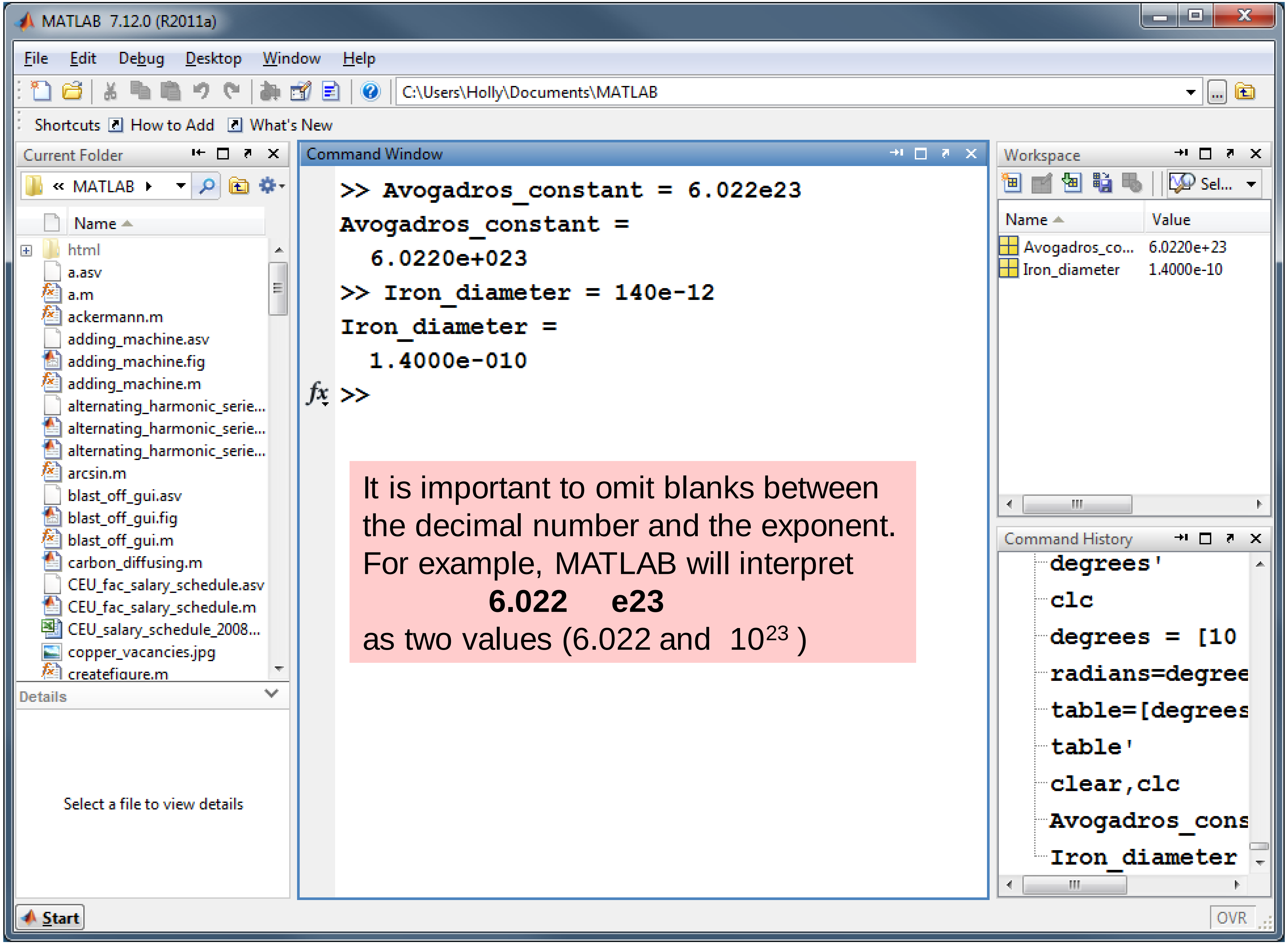
Name ▲	Value
ans	[10,15,70,90;0.174
degrees	[10,15,70,90]
radians	[0.1745,0.2618,1.2
table	[10,0.1745;15,0.26

```
ees = [10
degrees = [10
degrees'
clc
degrees = [10
radians=degree
table=[degrees
table'
```

The transpose operator works on both one dimensional and two dimensional arrays

Number Display

- Scientific Notation
  - Although you can enter any number in decimal notation, it isn’t always the best way to represent very large or very small numbers
  - In MATLAB, values in scientific notation are designated with an e between the decimal number and exponent. (Your calculator probably uses similar notation.)



```
>> Avogadros_constant = 6.022e23
Avogadros_constant =
  6.0220e+023
>> Iron_diameter = 140e-12
Iron_diameter =
  1.4000e-010
>>
```

Name	Value
Avogadros_co...	6.0220e+23
Iron_diameter	1.4000e-10

```
degrees'
clc
degrees = [10
radians=degree
table=[degrees
table'
clear,clc
Avogadros_cons
Iron_diameter
```

It is important to omit blanks between the decimal number and the exponent. For example, MATLAB will interpret

6.022   e23

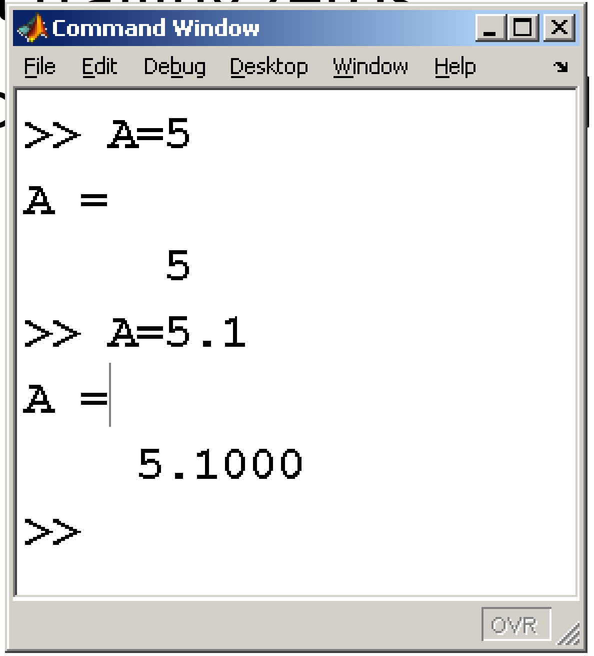
as two values (6.022 and 10^{23})

Display Format

- Multiple display formats are available
- No matter what display format you choose, MATLAB uses double precision floating point numbers in its calculations
- MATLAB handles both integers and decimal numbers as floating point numbers

Default

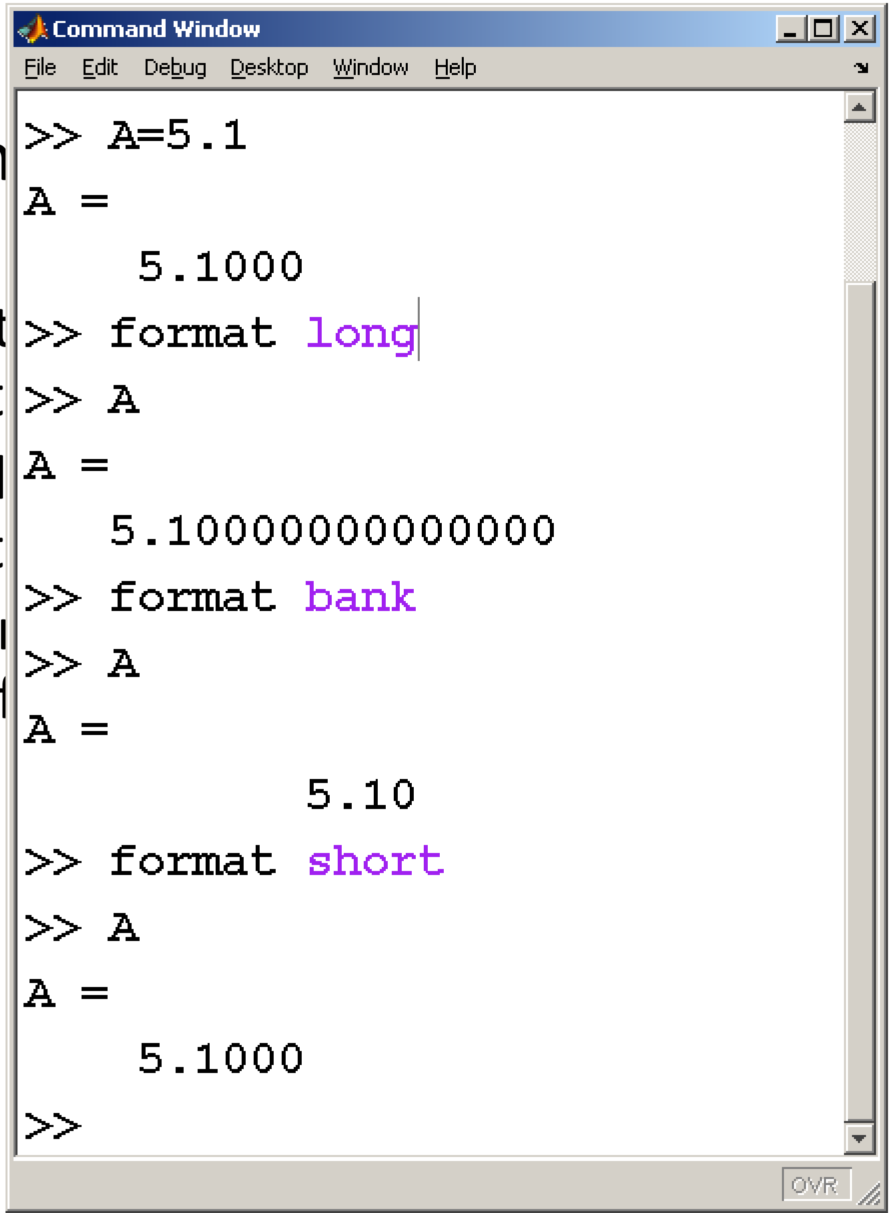
- The default format is called short
- If an integer is entered it is displayed without trailing zeros
- If a floating point number is entered four dec layed



```
>> A=5
A =
     5
>> A=5.1
A =
    5.1000
>>
```

Other formats

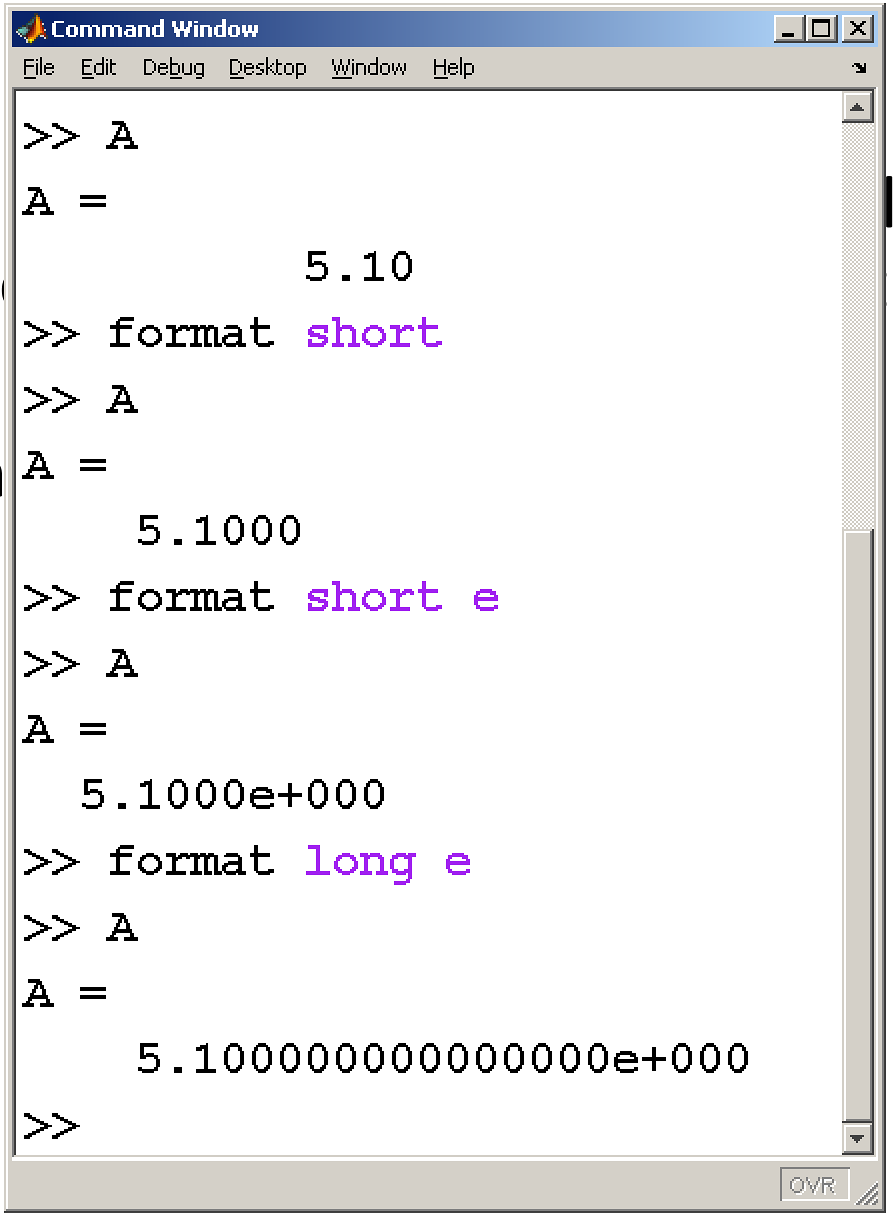
- Changing the form displays
  - format long result the decimal point
  - format bank resul the decimal point
  - format short retu 4 decimal digits af



```
>> A=5.1
A =
     5.1000
>> format long
>> A
A =
   5.10000000000000
>> format bank
>> A
A =
           5.10
>> format short
>> A
A =
     5.1000
>>
```

Really Big and Really Small

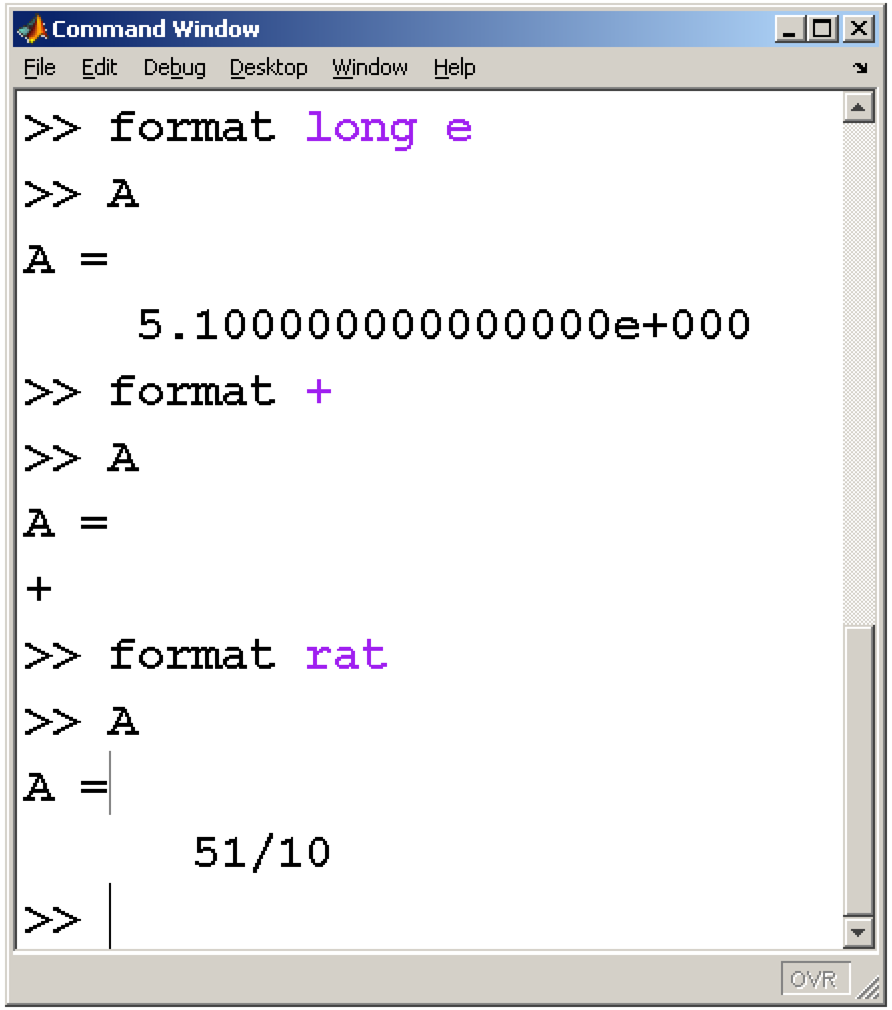
- When numbers become too large or lay using the default format, it automati ific notation
- You can force scientific notation with
  - format short e
  - format long e



```
>> A

A =

          5.10

>> format short

>> A

A =

    5.1000

>> format short e

>> A

A =

  5.1000e+000

>> format long e

>> A

A =

    5.100000000000000e+000

>>
```

Two other formats

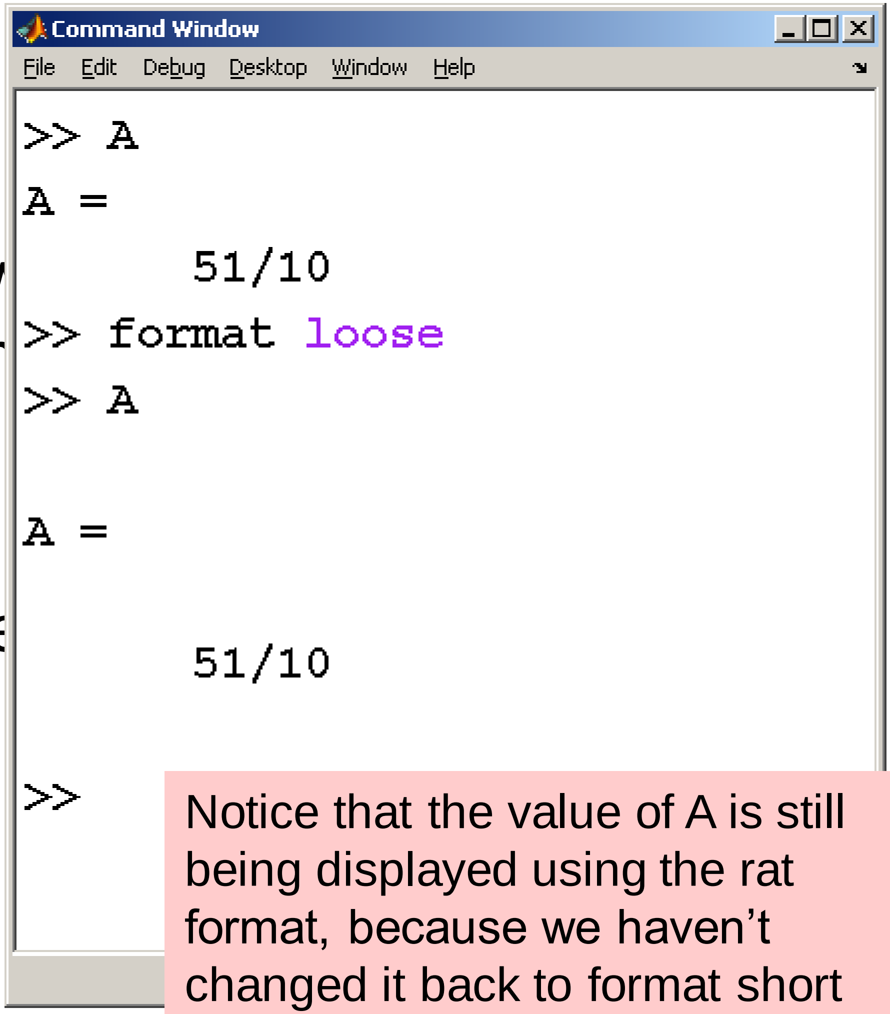
- format +
- format rat



```
>> format long e
>> A
A =
     5.100000000000000e+000
>> format +
>> A
A =
+
>> format rat
>> A
A =
       51/10
>> 
```

Spacing in the comman

- The **format** command also allov information is spaced in the cor
  - format compact
  - format loose – (default)
- Most of the examples in this pre **ct**

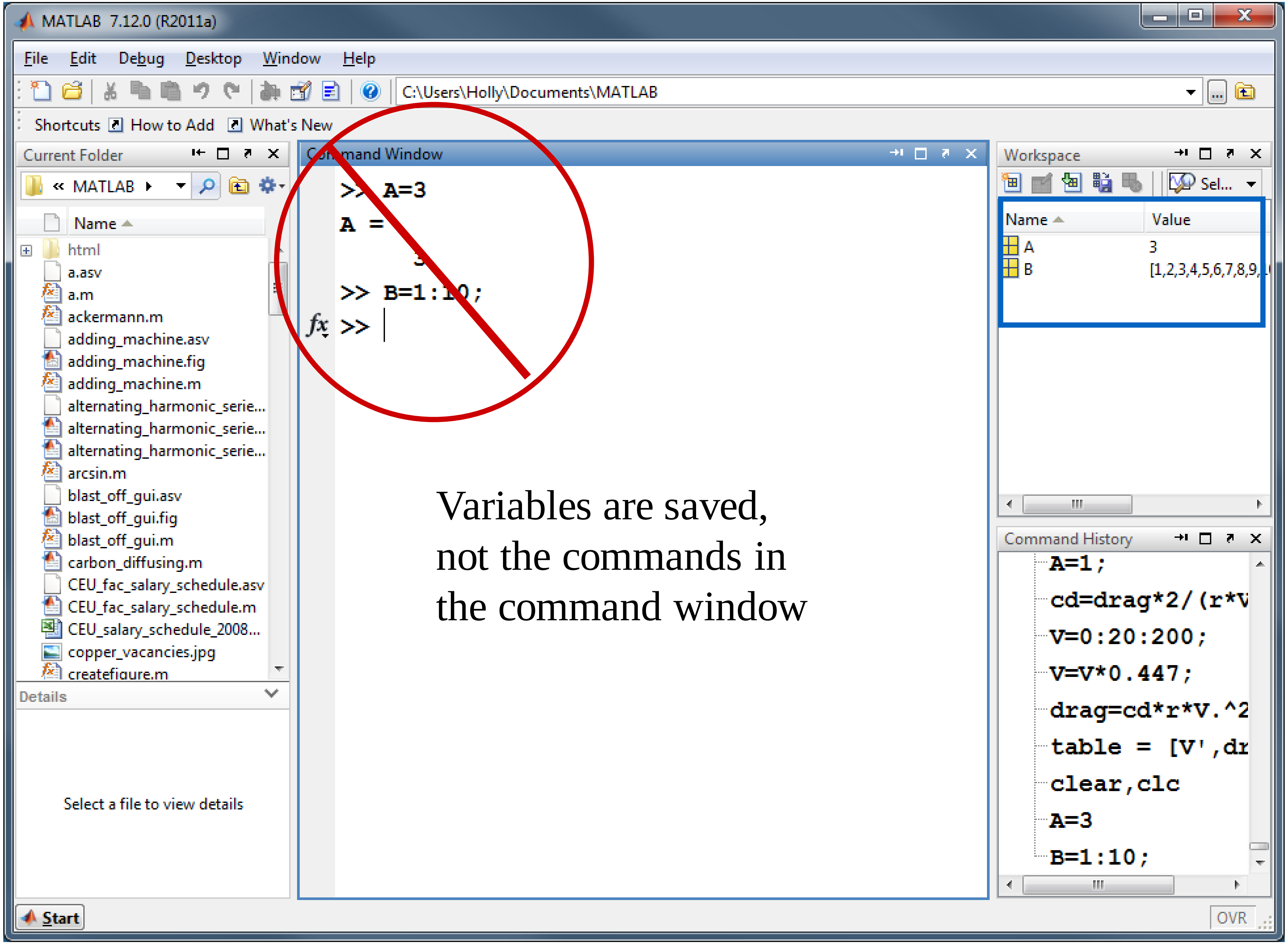


```
>> A
A =
      51/10
>> format loose
>> A


A =


      51/10


>>
```

Notice that the value of A is still being displayed using the rat format, because we haven’t changed it back to format short

Section 2.4
Saving Your Work

- If you save a MATLAB session performed in the command window, all that is saved are the values of the variables you have named


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
MATLAB
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> A=3

A =

     3

>> B=1:10;
>> 
```

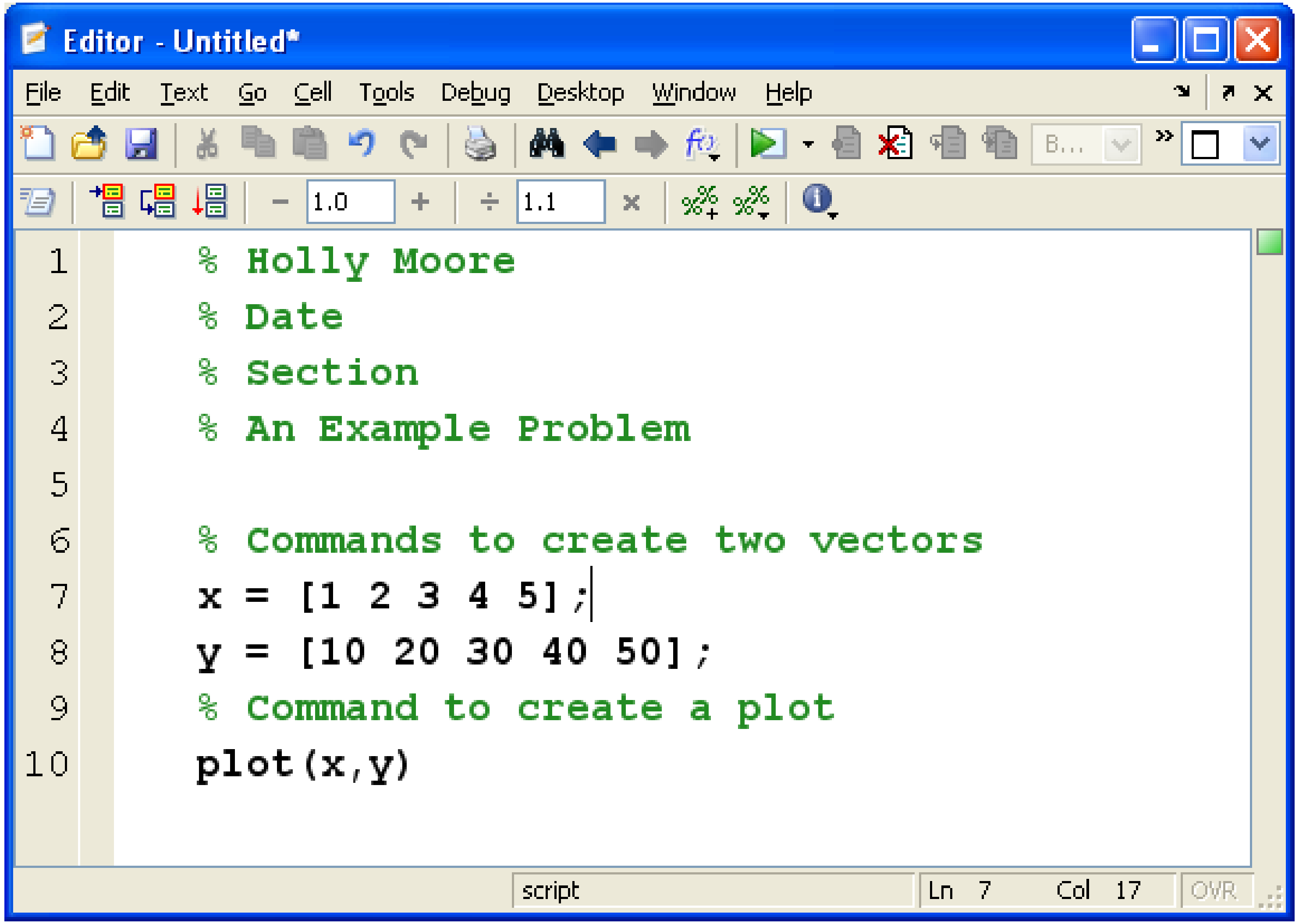
Name	Value
A	3
B	[1,2,3,4,5,6,7,8,9,1

```
A=1;
cd=drag*2/(r*V
V=0:20:200;
V=V*0.447;
drag=cd*r*V.^2
table = [V',dr
clear,clc
A=3
B=1:10;
```

Variables are saved, not the commands in the command window

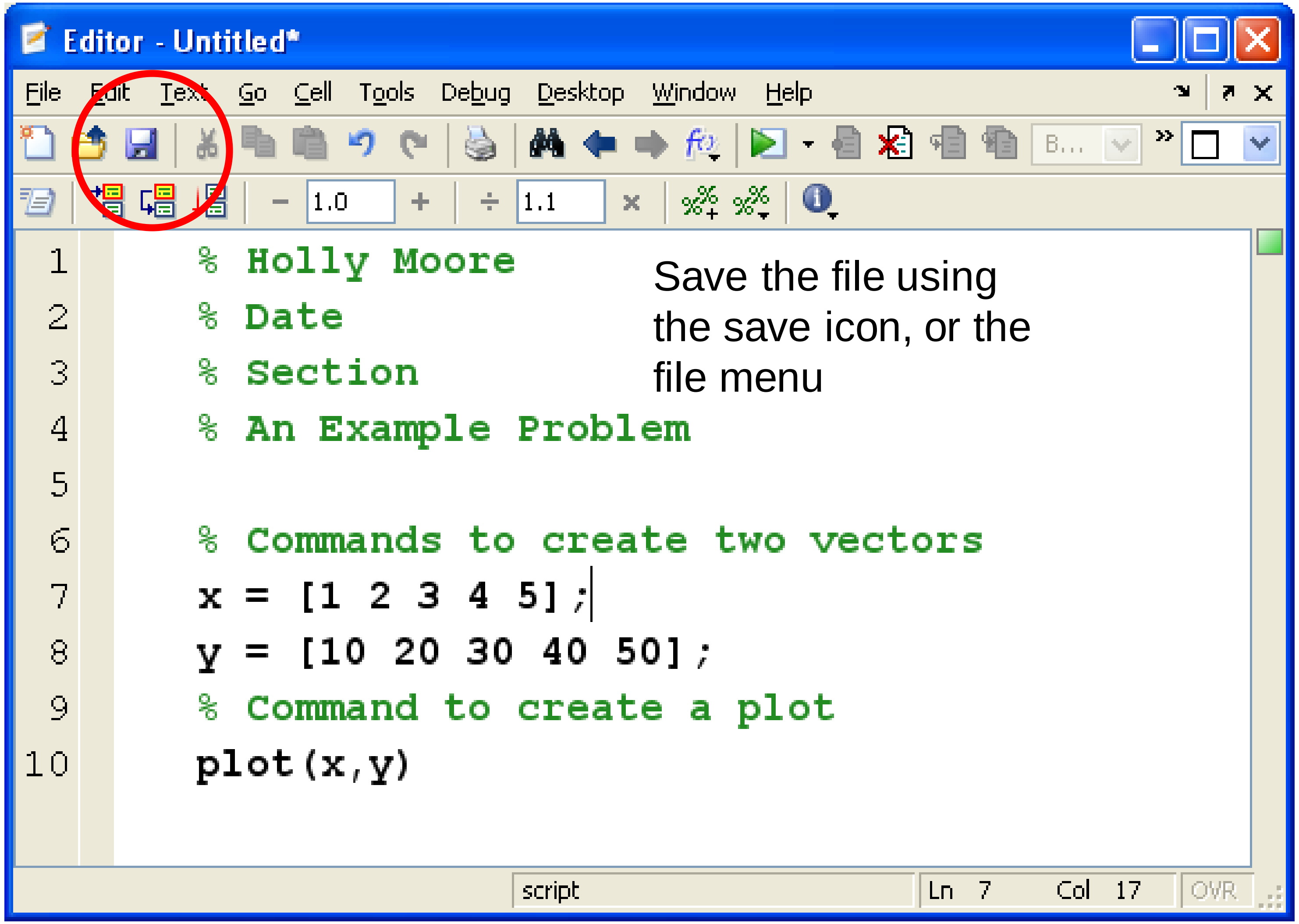
Script M-files

- If you want to save your work, (the commands you entered) you need to create an M-file
- File->New->M-file
- Type your commands in the edit window that opens



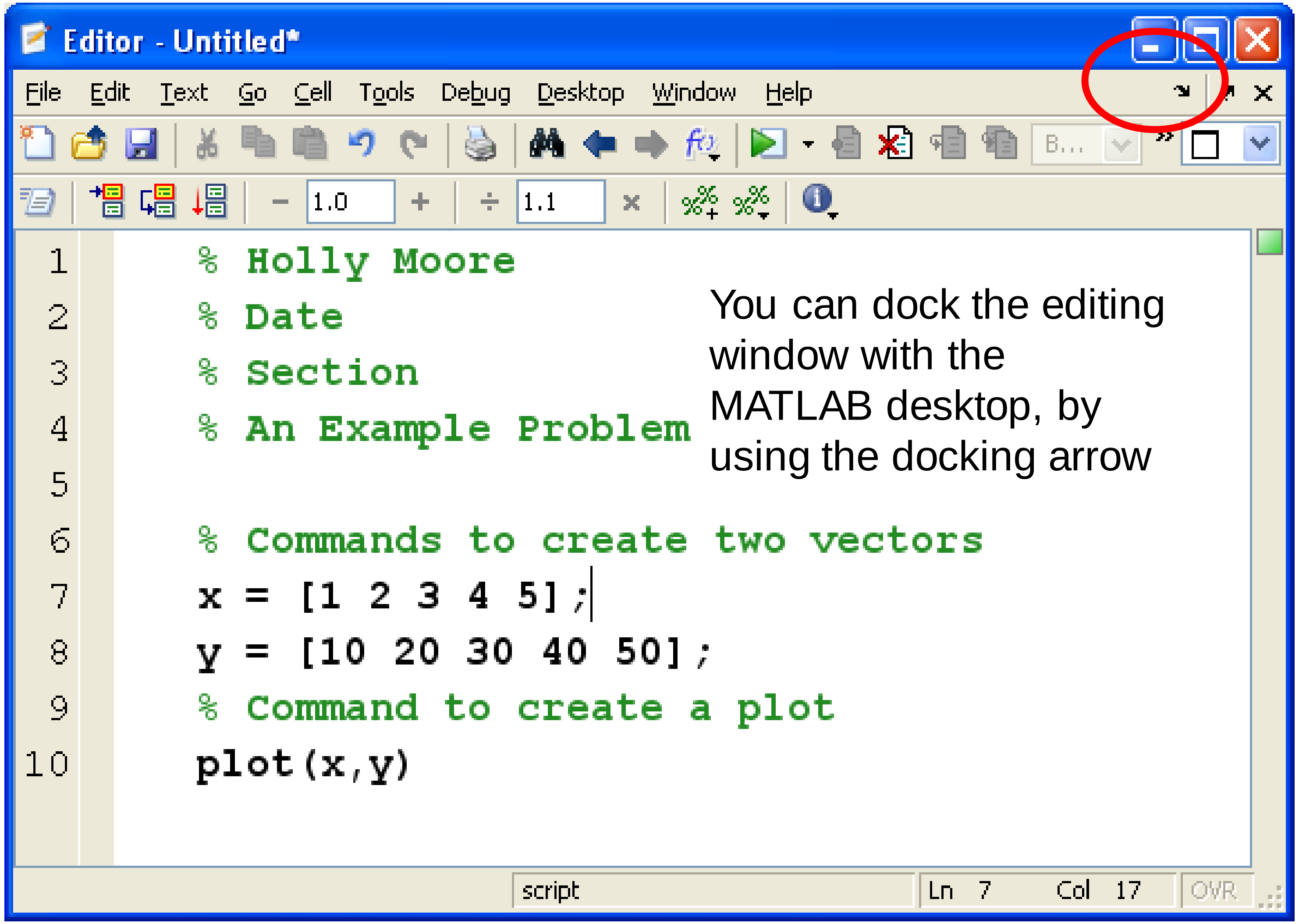
```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10 20 30 40 50];
% Command to create a plot
plot(x,y)
```

- The file can be saved into the current folder/directory
- It runs in the command window



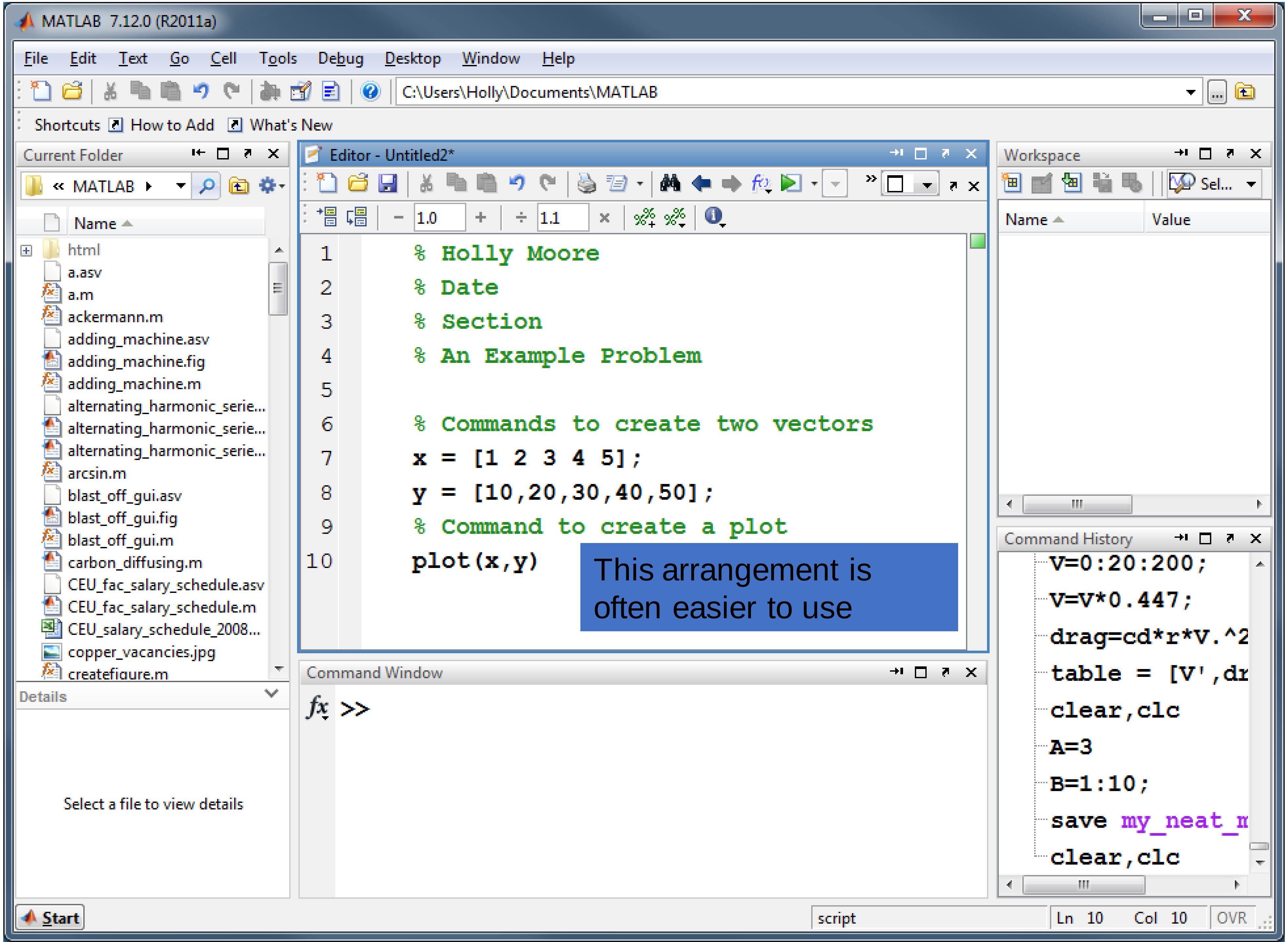
```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10 20 30 40 50];
% Command to create a plot
plot(x,y)
```

Save the file using the save icon, or the file menu



```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10 20 30 40 50];
% Command to create a plot
plot(x,y)
```

You can dock the editing window with the MATLAB desktop, by using the docking arrow

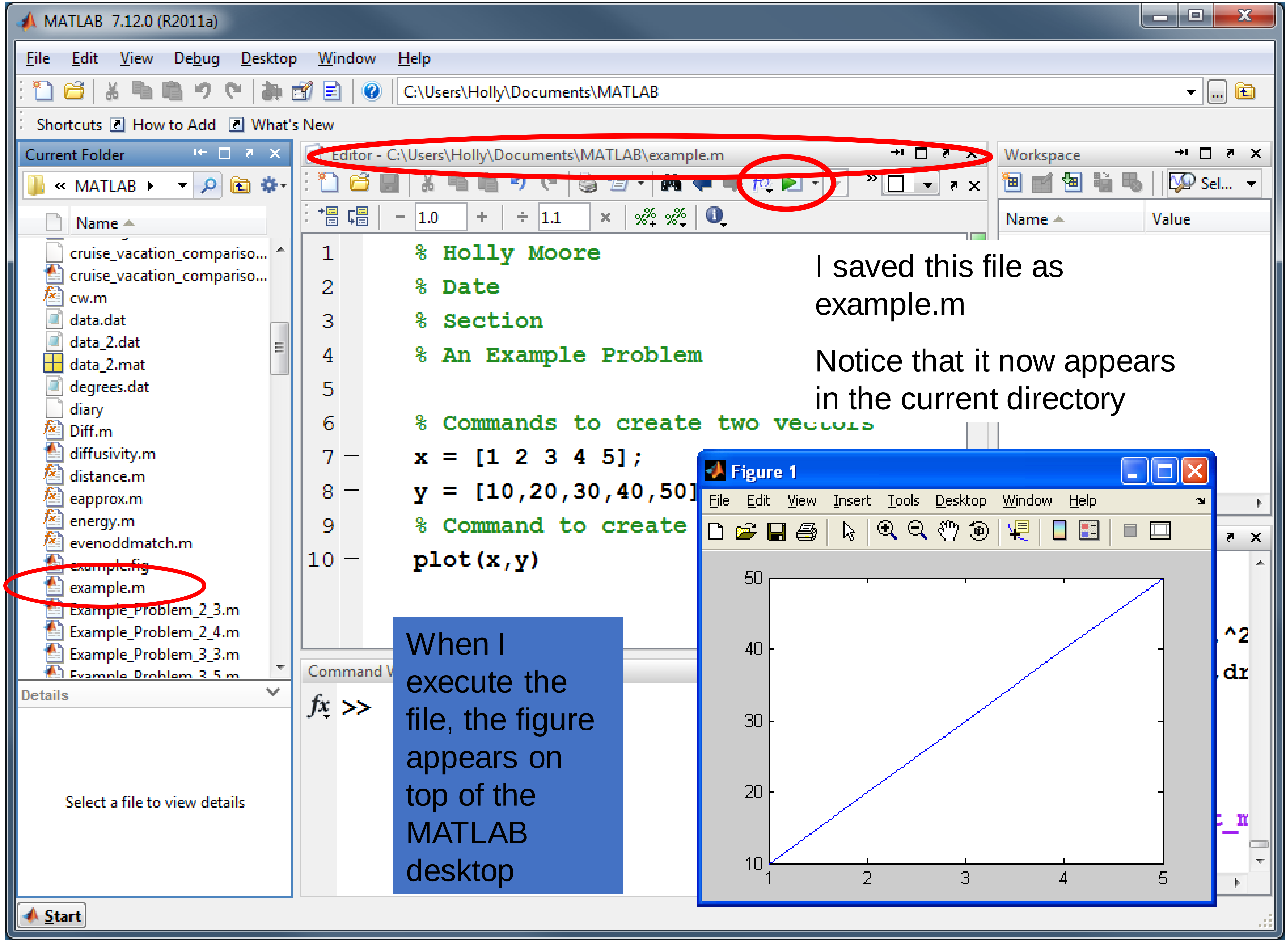


```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10,20,30,40,50];
% Command to create a plot
plot(x,y)
```

```
fx >>
```

```
V=0:20:200;
V=V*0.447;
drag=cd*r*V.^2
table = [V',dr
clear,clc
A=3
B=1:10;
save my_neat_m
clear,clc
```

This arrangement is often easier to use

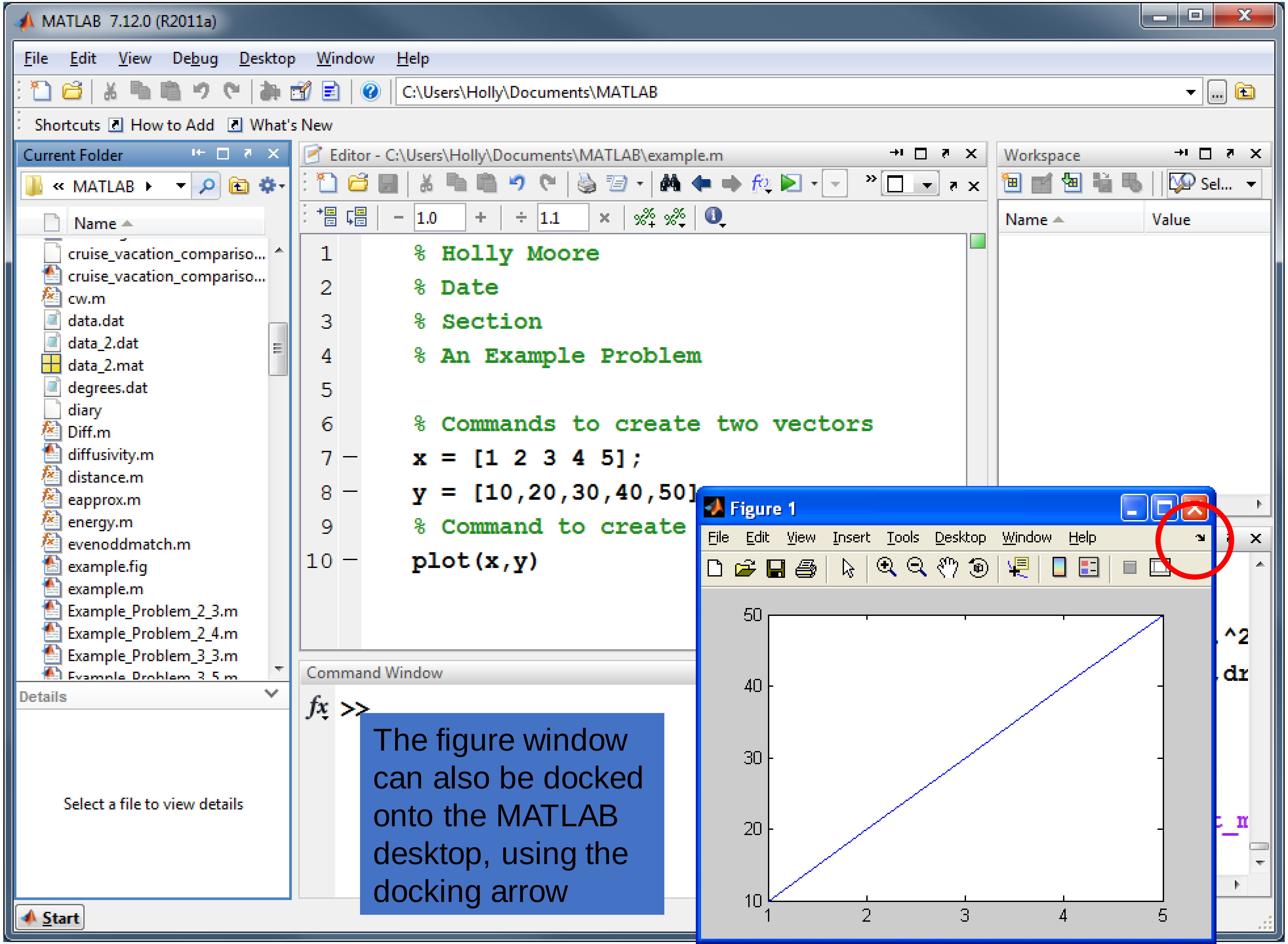


```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10,20,30,40,50]
% Command to create
plot(x,y)
```

I saved this file as example.m

Notice that it now appears in the current directory

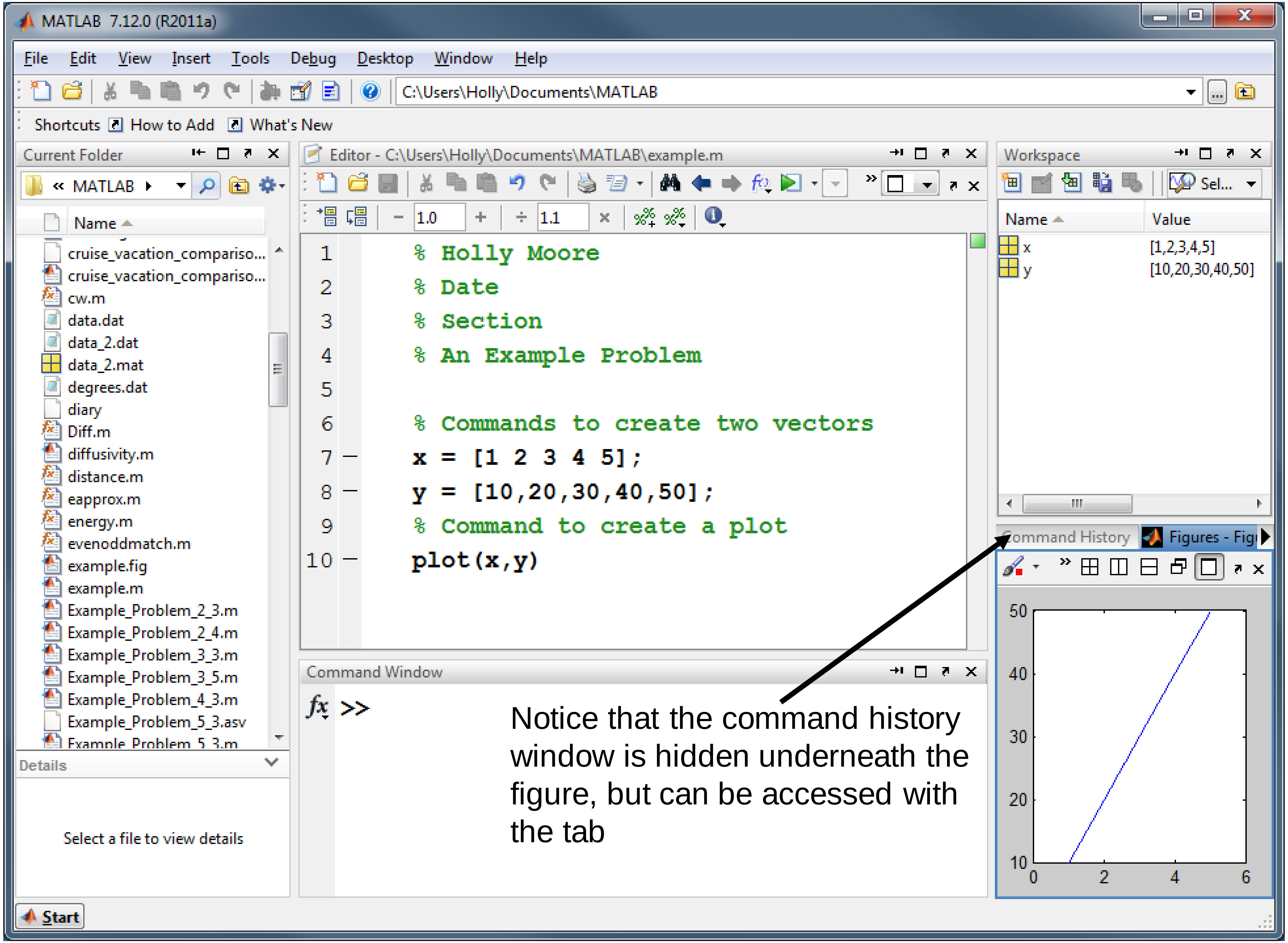
When I execute the file, the figure appears on top of the MATLAB desktop



```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10,20,30,40,50]
% Command to create
plot(x,y)
```

```
>>
```

The figure window can also be docked onto the MATLAB desktop, using the docking arrow


MATLAB 7.12.0 (R2011a)
File Edit View Insert Tools Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
MATLAB
Name
Editor - C:\Users\Holly\Documents\MATLAB\example.m
1.0
1.1
Workspace
Sel...
Command Window
Command History
Figures - Fig
Details
Select a file to view details
Start
50
40
30
20
10
0
2
4
6

cruise_vacation_compariso...
cruise_vacation_compariso...
cw.m
data.dat
data_2.dat
data_2.mat
degrees.dat
diary
Diff.m
diffusivity.m
distance.m
eapprox.m
energy.m
evenoddmatch.m
example.fig
example.m
Example_Problem_2_3.m
Example_Problem_2_4.m
Example_Problem_3_3.m
Example_Problem_3_5.m
Example_Problem_4_3.m
Example_Problem_5_3.asv
Example_Problem_5_3.m

```
% Holly Moore
% Date
% Section
% An Example Problem

% Commands to create two vectors
x = [1 2 3 4 5];
y = [10,20,30,40,50];
% Command to create a plot
plot(x,y)
```

Name	Value
x	[1,2,3,4,5]
y	[10,20,30,40,50]

```
fx >>
```

Notice that the command history window is hidden underneath the figure, but can be accessed with the tab

Comments

- Be sure to comment your code
  - Add your name
  - Date
  - Section #
  - Assignment #
  - Descriptions of what you are doing and why

The % sign identifies comments

You need one on each line

Summary

- Introduced the MATLAB Windows
- Basic matrix definition
- Save and retrieve MATLAB data
- Create and use script M-files

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 3
(Built-in MATLAB Functions)

Zeyad T. Shareef

Objectives

After studying this lecture, you should be able to:

- Use a variety of common mathematical functions
- Understand and use trigonometric functions in MATLAB
- Compute and use statistical and data analysis functions
- Generate uniform and Gaussian random-number matrices
- Understand the computational limits of MATLAB
- Recognize and be able to use the special values and functions built into MATLAB

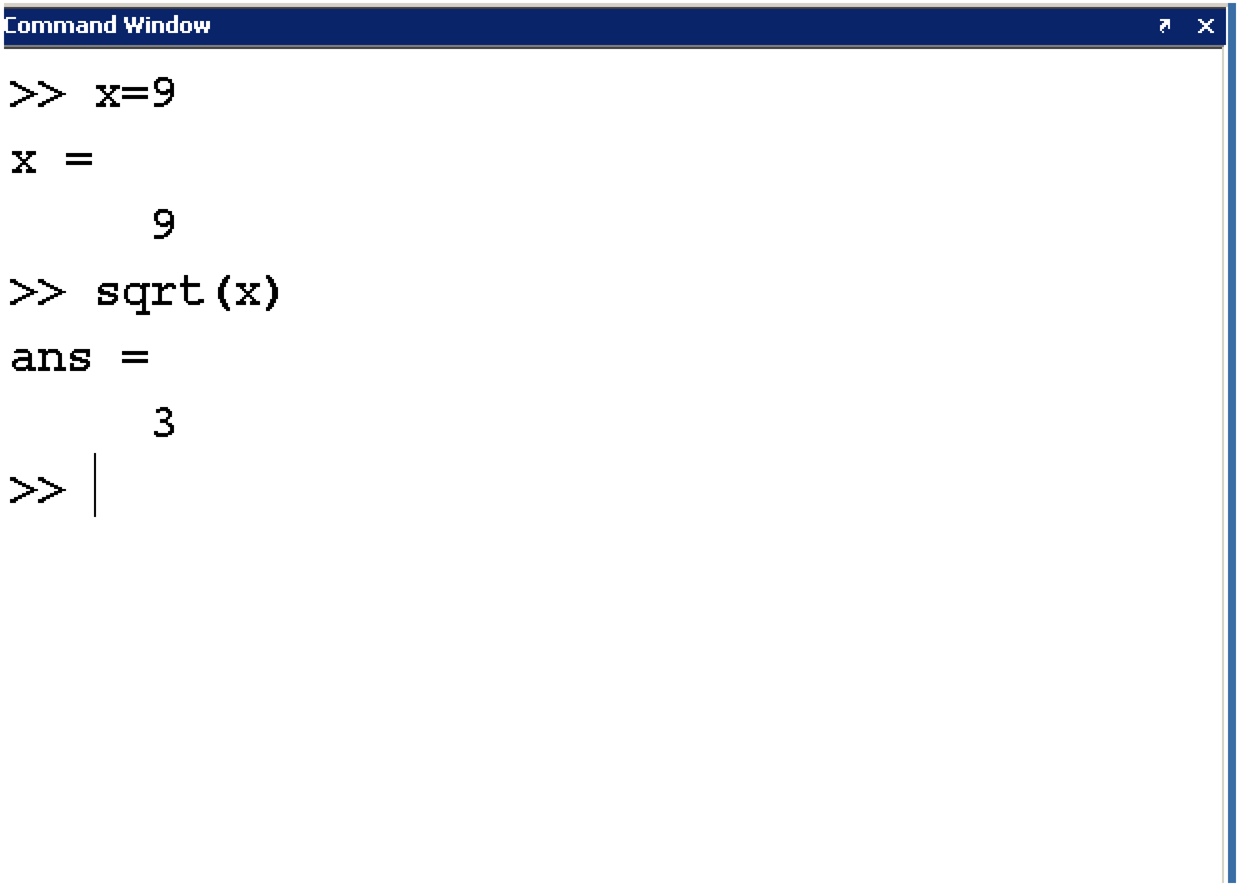
3.1 Using Built-in Functions

MATLAB uses function names consistent with most major programming languages

For example

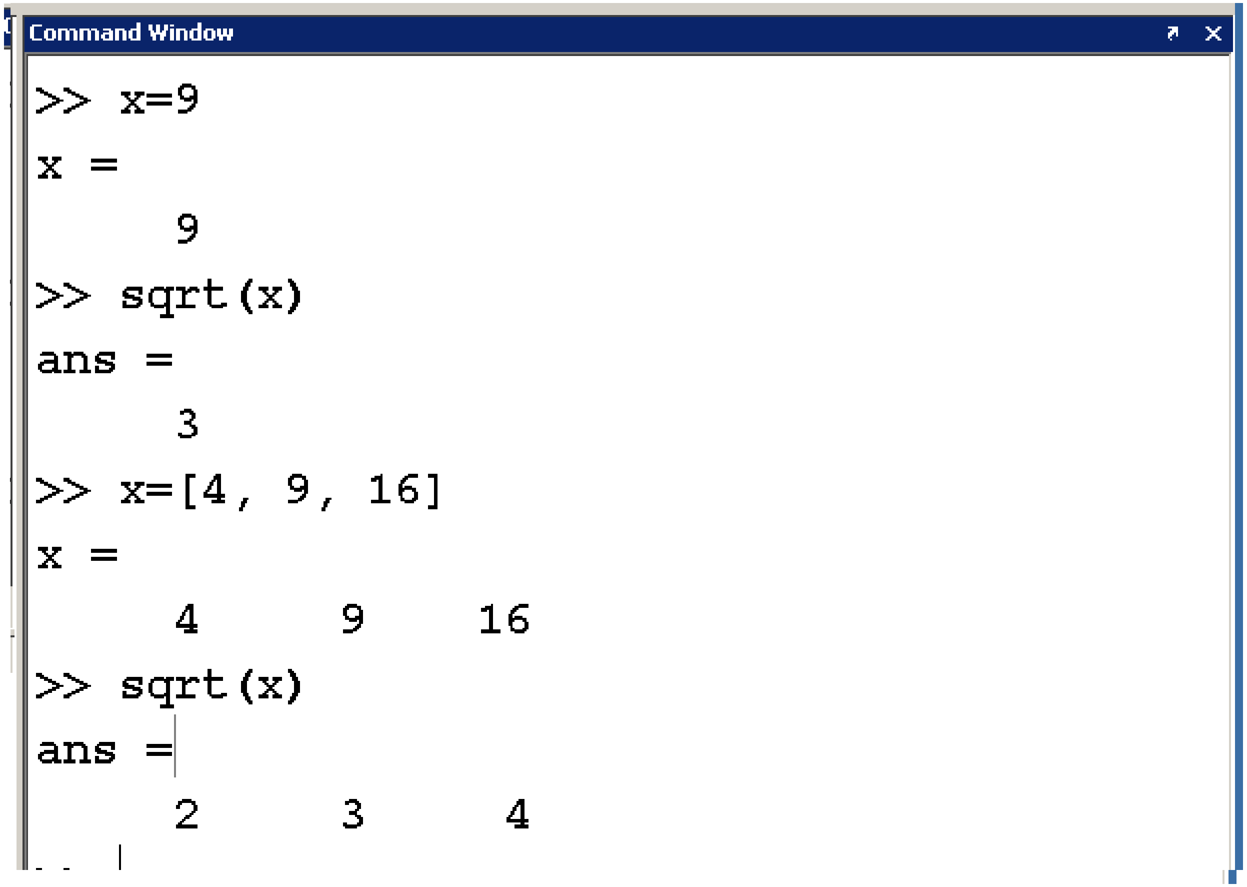
- sqrt
- sin
- cos
- log

Function Input can be either scalars or matrices



```
>> x=9

x =

     9

>> sqrt(x)

ans =

     3

>> 
```

Function Input can be either scalars or matrices



```
>> x=9
x =
     9
>> sqrt(x)
ans =
     3
>> x=[4, 9, 16]
x =
     4     9    16
>> sqrt(x)
ans =
     2     3     4
```

Using Predefined Functions

- Functions consist of
  - Name
  - Input argument(s)
  - Output

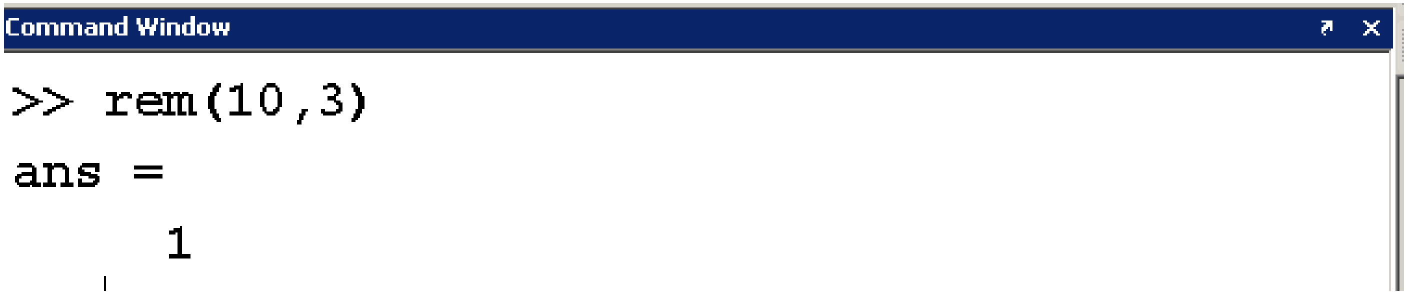
In MATLAB

sqrt (x)= result

```
sqrt(4)
 ans = 2
```

Some functions require multiple inputs

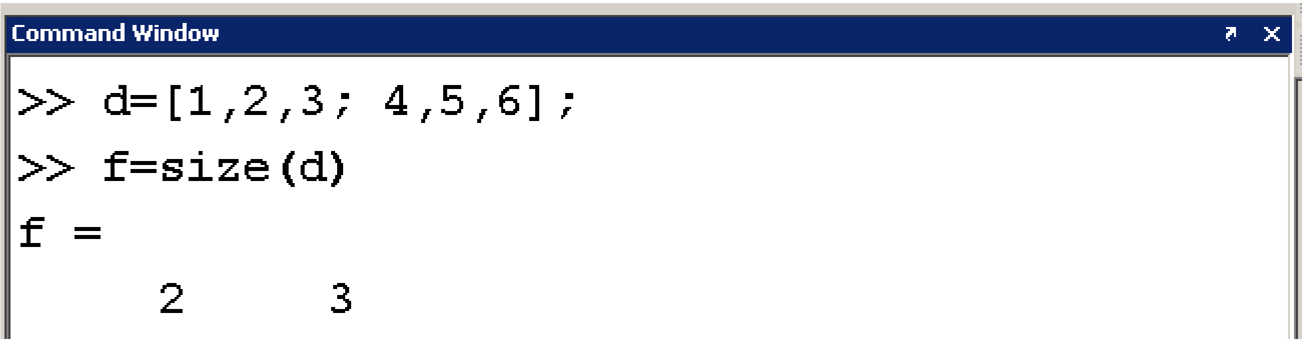
- Remainder function returns the remainder in a division problem
- For example, the remainder of 10/3, is 1



```
>> rem(10,3)
ans =
     1
```

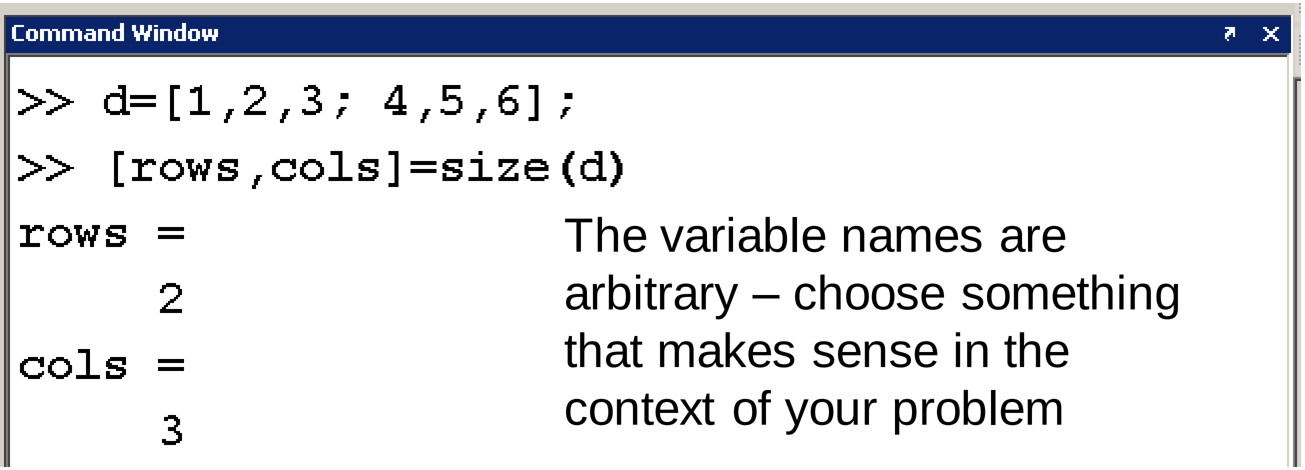
Some functions return multiple results

- size function determines the number of rows and columns



```
>> d=[1,2,3; 4,5,6];
>> f=size(d)
f =
     2     3
```

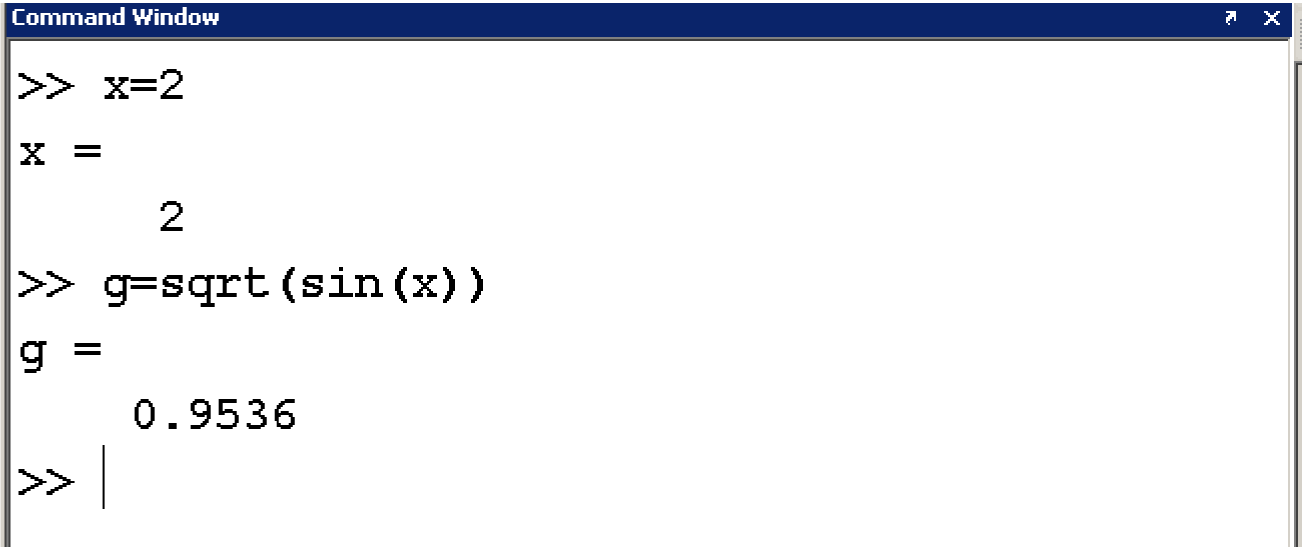
You can assign names to the output



```
>> d=[1,2,3; 4,5,6];
>> [rows,cols]=size(d)
rows =
     2
cols =
     3
```

The variable names are arbitrary – choose something that makes sense in the context of your problem

Nesting Functions

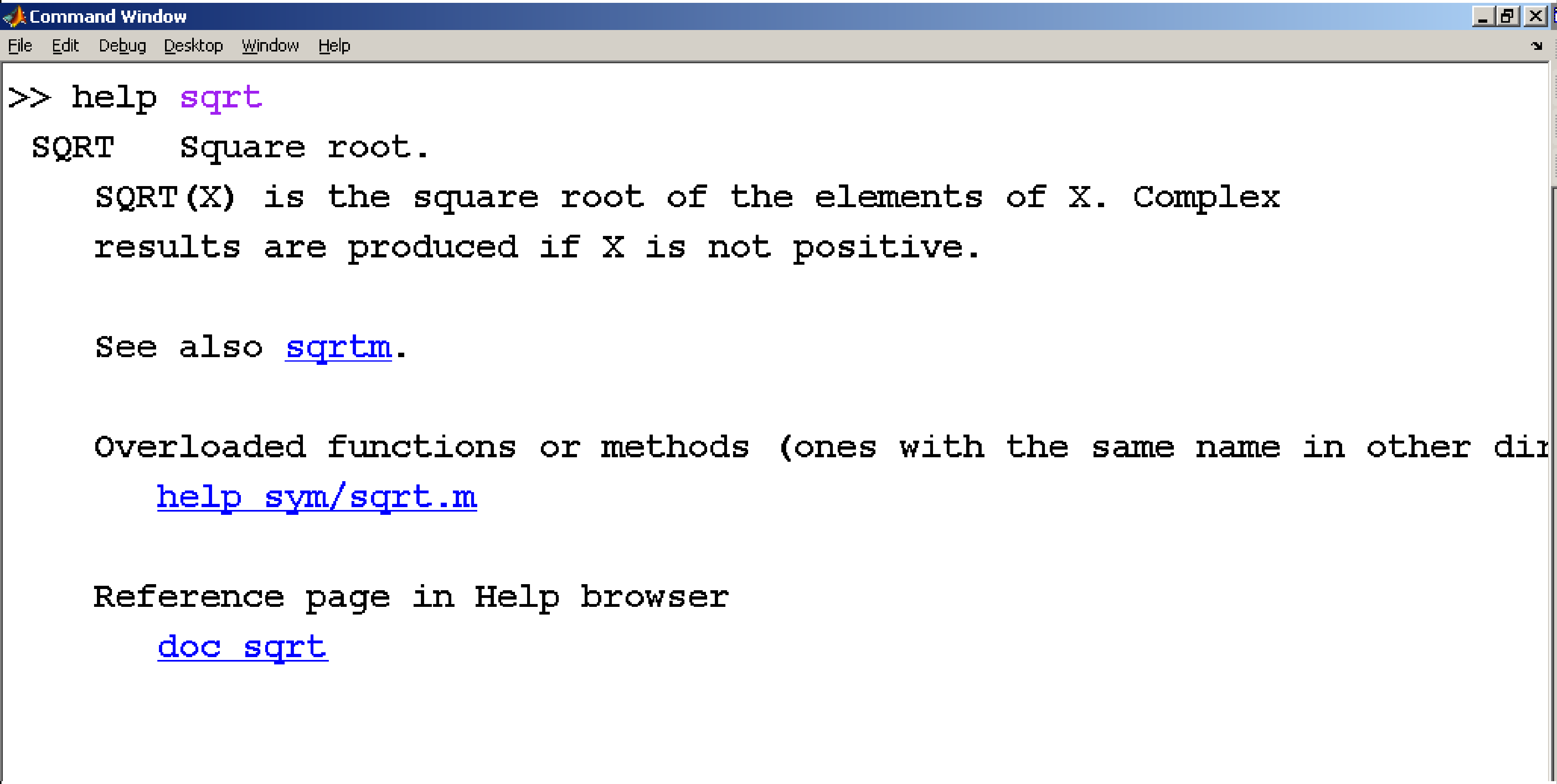


```
>> x=2

x =

     2

>> g=sqrt(sin(x))

g =

    0.9536

>>
```

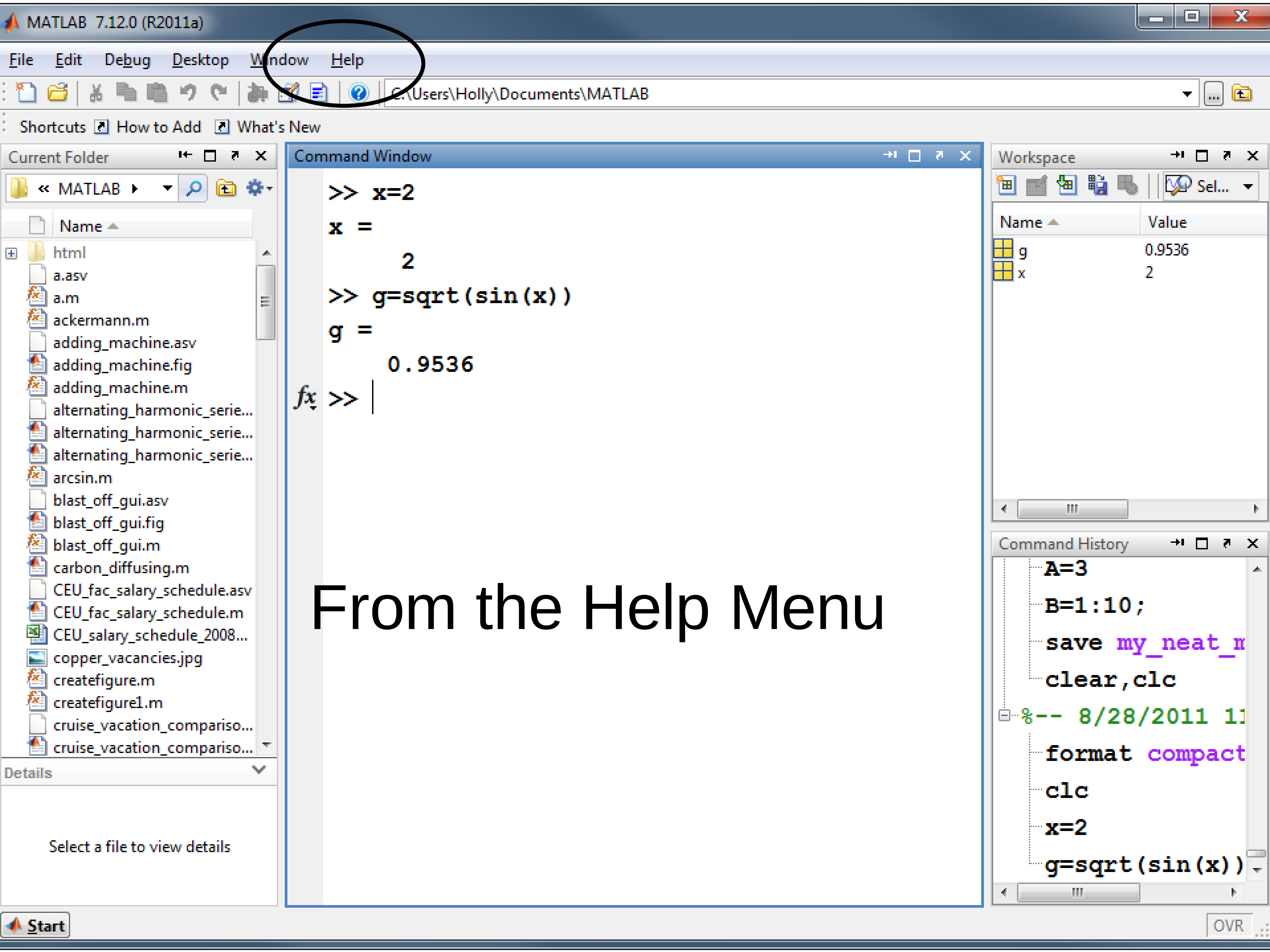
3.2 Using the Help Feature

- There are functions for almost anything you want to do
- Use the help feature to find out what they are and how to use them
  - From the command window
  - From the help selection on the menu bar

From the Command Window



```
>> help sqrt
 SQRT   Square root.
    SQRT(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.

    See also sqrtm.

    Overloaded functions or methods (ones with the same name in other dir
       help sym/sqrt.m

    Reference page in Help browser
       doc sqrt
```

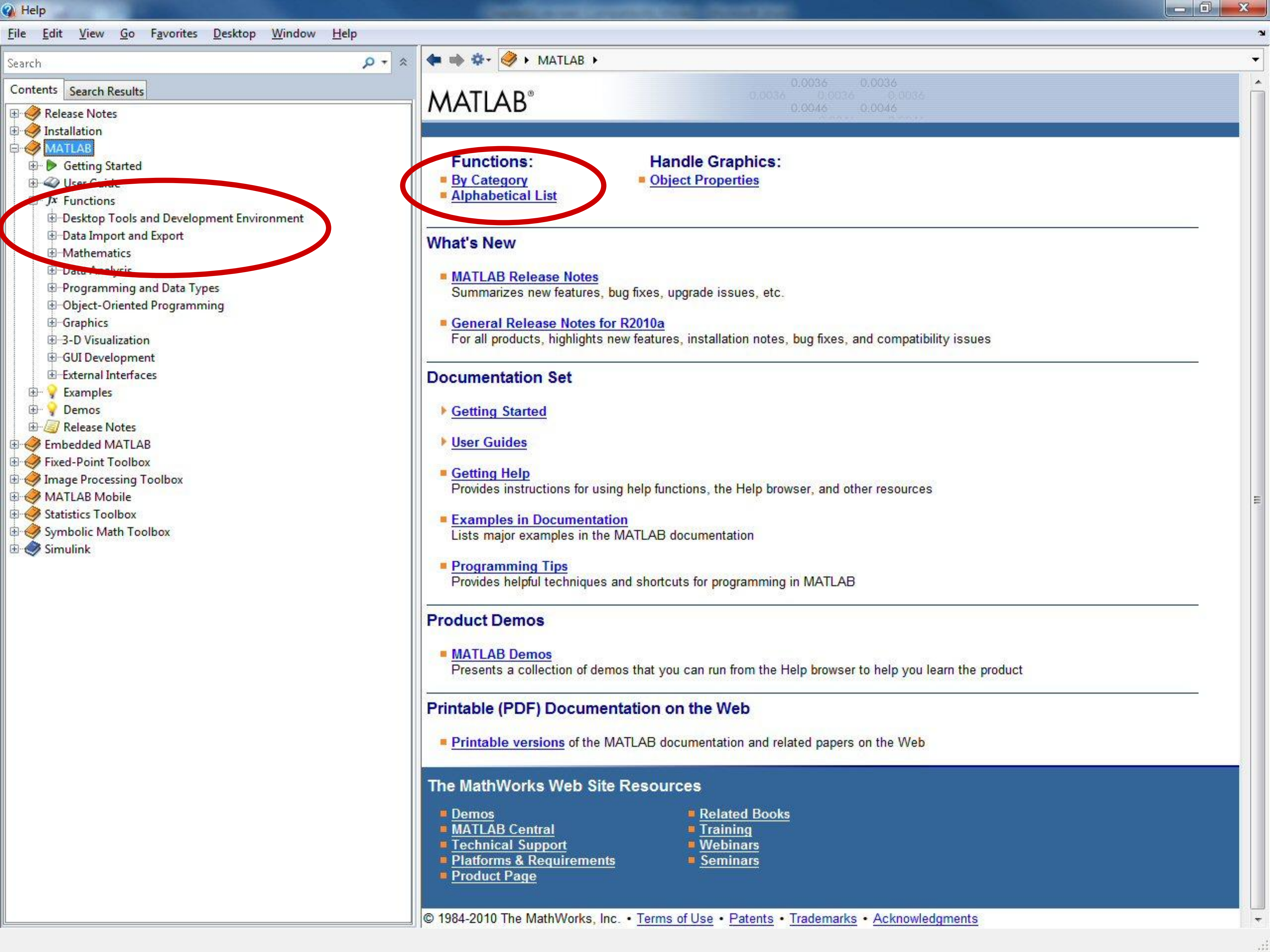


```
>> x=2
x =
     2
>> g=sqrt(sin(x))
g =
    0.9536
fx >>
```

Name	Value
g	0.9536
x	2

```
A=3
B=1:10;
save my_neat_m
clear,clc
%-- 8/28/2011 1
format compact
clc
x=2
g=sqrt(sin(x))
```

From the Help Menu



- Release Notes
- Installation
- MATLAB
  - Getting Started
  - User Guide
  - Functions
    - Desktop Tools and Development Environment
    - Data Import and Export
    - Mathematics
    - Data Analysis
    - Programming and Data Types
    - Object-Oriented Programming
    - Graphics
    - 3-D Visualization
    - GUI Development
    - External Interfaces
  - Examples
  - Demos
  - Release Notes
- Embedded MATLAB
- Fixed-Point Toolbox
- Image Processing Toolbox
- MATLAB Mobile
- Statistics Toolbox
- Symbolic Math Toolbox
- Simulink

MATLAB®

Functions:
- By Category
- Alphabetical List

Handle Graphics:
- Object Properties

What's New

- MATLAB Release Notes
  Summarizes new features, bug fixes, upgrade issues, etc.
- General Release Notes for R2010a
  For all products, highlights new features, installation notes, bug fixes, and compatibility issues

Documentation Set

- Getting Started
- User Guides
- Getting Help
  Provides instructions for using help functions, the Help browser, and other resources
- Examples in Documentation
  Lists major examples in the MATLAB documentation
- Programming Tips
  Provides helpful techniques and shortcuts for programming in MATLAB

Product Demos

- MATLAB Demos
  Presents a collection of demos that you can run from the Help browser to help you learn the product

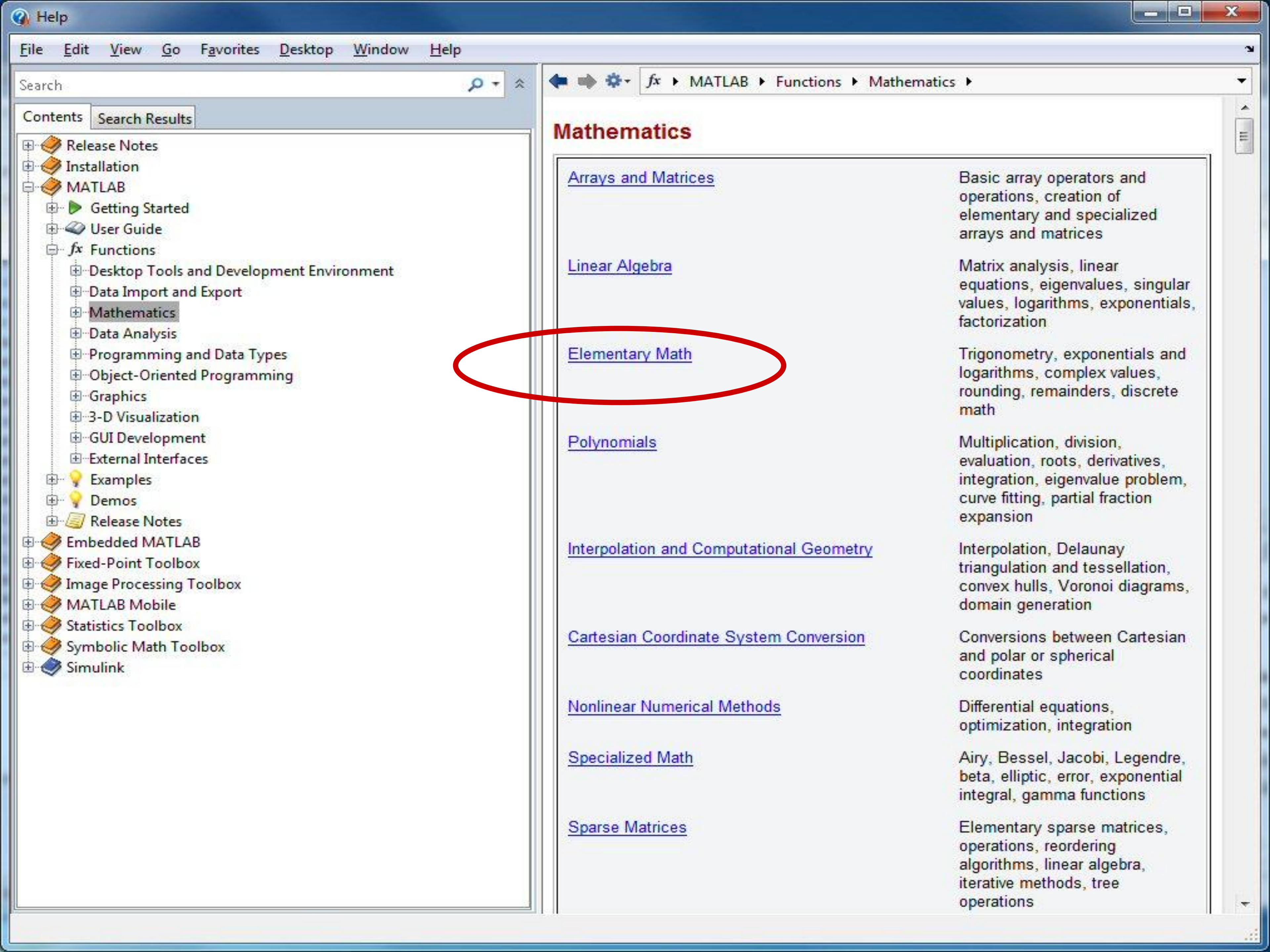
Printable (PDF) Documentation on the Web

- Printable versions of the MATLAB documentation and related papers on the Web

The MathWorks Web Site Resources

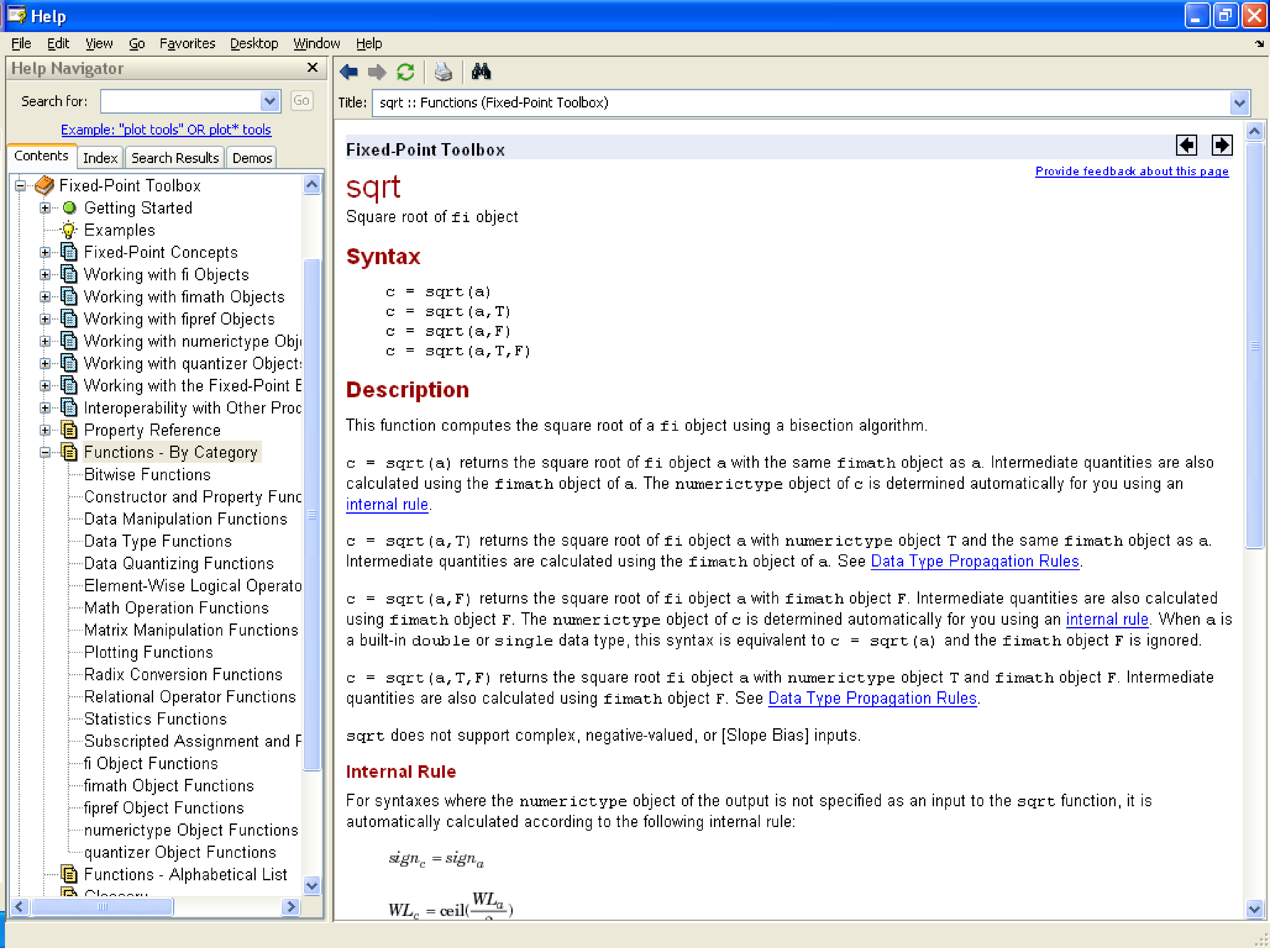
- Demos
- MATLAB Central
- Technical Support
- Platforms & Requirements
- Product Page
- Related Books
- Training
- Webinars
- Seminars

© 1984-2010 The MathWorks, Inc. • Terms of Use • Patents • Trademarks • Acknowledgments



Mathematics

Arrays and Matrices	Basic array operators and operations, creation of elementary and specialized arrays and matrices
Linear Algebra	Matrix analysis, linear equations, eigenvalues, singular values, logarithms, exponentials, factorization
Elementary Math	Trigonometry, exponentials and logarithms, complex values, rounding, remainders, discrete math
Polynomials	Multiplication, division, evaluation, roots, derivatives, integration, eigenvalue problem, curve fitting, partial fraction expansion
Interpolation and Computational Geometry	Interpolation, Delaunay triangulation and tessellation, convex hulls, Voronoi diagrams, domain generation
Cartesian Coordinate System Conversion	Conversions between Cartesian and polar or spherical coordinates
Nonlinear Numerical Methods	Differential equations, optimization, integration
Specialized Math	Airy, Bessel, Jacobi, Legendre, beta, elliptic, error, exponential integral, gamma functions
Sparse Matrices	Elementary sparse matrices, operations, reordering algorithms, linear algebra, iterative methods, tree operations


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search for:
Go
Example: "plot tools" OR plot* tools
Contents
Index
Search Results
Demos
Fixed-Point Toolbox
Getting Started
Examples
Fixed-Point Concepts
Working with fi Objects
Working with fimath Objects
Working with fipref Objects
Working with numerictype Obj
Working with quantizer Object
Working with the Fixed-Point E
Interoperability with Other Proc
Property Reference
Functions - By Category
Bitwise Functions
Constructor and Property Func
Data Manipulation Functions
Data Type Functions
Data Quantizing Functions
Element-Wise Logical Operato
Math Operation Functions
Matrix Manipulation Functions
Plotting Functions
Radix Conversion Functions
Relational Operator Functions
Statistics Functions
Subscripted Assignment and F
fi Object Functions
fimath Object Functions
fipref Object Functions
numerictype Object Functions
quantizer Object Functions
Functions - Alphabetical List
Glossary
Title: sqrt :: Functions (Fixed-Point Toolbox)
Fixed-Point Toolbox
Provide feedback about this page

sqrt

Square root of `fi` object

Syntax

```
c = sqrt(a)
c = sqrt(a,T)
c = sqrt(a,F)
c = sqrt(a,T,F)
```

Description

This function computes the square root of a `fi` object using a bisection algorithm.

`c = sqrt(a)` returns the square root of `fi` object `a` with the same `fimath` object as `a`. Intermediate quantities are also calculated using the `fimath` object of `a`. The `numerictype` object of `c` is determined automatically for you using an internal rule.

`c = sqrt(a,T)` returns the square root of `fi` object `a` with `numerictype` object `T` and the same `fimath` object as `a`. Intermediate quantities are calculated using the `fimath` object of `a`. See Data Type Propagation Rules.

`c = sqrt(a,F)` returns the square root of `fi` object `a` with `fimath` object `F`. Intermediate quantities are also calculated using `fimath` object `F`. The `numerictype` object of `c` is determined automatically for you using an internal rule. When `a` is a built-in `double` or `single` data type, this syntax is equivalent to `c = sqrt(a)` and the `fimath` object `F` is ignored.

`c = sqrt(a,T,F)` returns the square root `fi` object `a` with `numerictype` object `T` and `fimath` object `F`. Intermediate quantities are also calculated using `fimath` object `F`. See Data Type Propagation Rules.

`sqrt` does not support complex, negative-valued, or [Slope Bias] inputs.

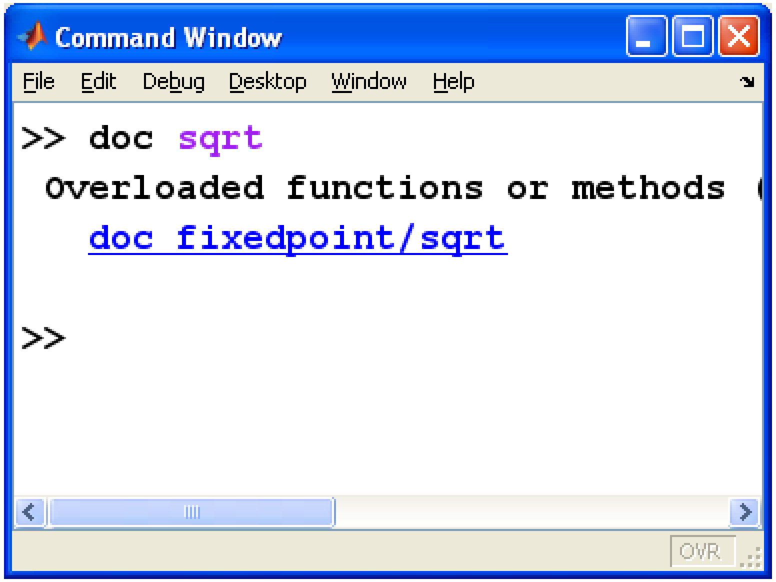
Internal Rule

For syntaxes where the `numerictype` object of the output is not specified as an input to the `sqrt` function, it is automatically calculated according to the following internal rule:

$$sign_c = sign_a$$

$$WL_c = \mathrm{ceil}(\frac{WL_a}{2})$$

The windowed help function can also be accessed using the doc command



```
>> doc sqrt
 Overloaded functions or methods (
   doc fixedpoint/sqrt

>>
```

3.3 Elementary Math Functions

3.3.1 Common Computations

As in most computer languages, log(x) is the syntax for the natural log – there is no ln function defined in MATLAB

• log(x)	natural log
• log10(x)	log base 10

3.3.2 Rounding Functions

- round(x)

  rounds towards nearest decimal or integer

- fix(x)

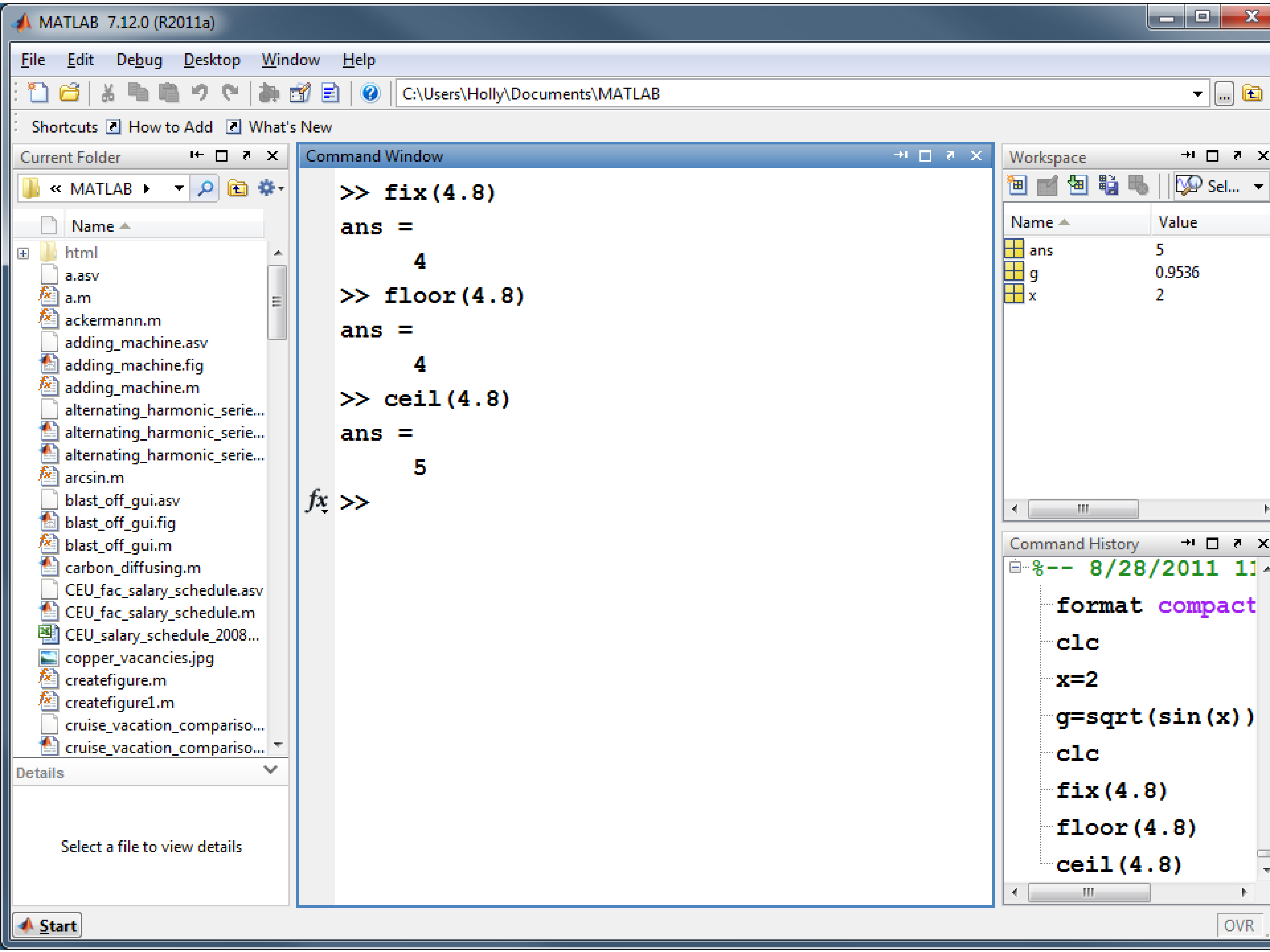
  rounds towards zero

- floor(x)

  rounds towards minus infinity

- ceil(x)

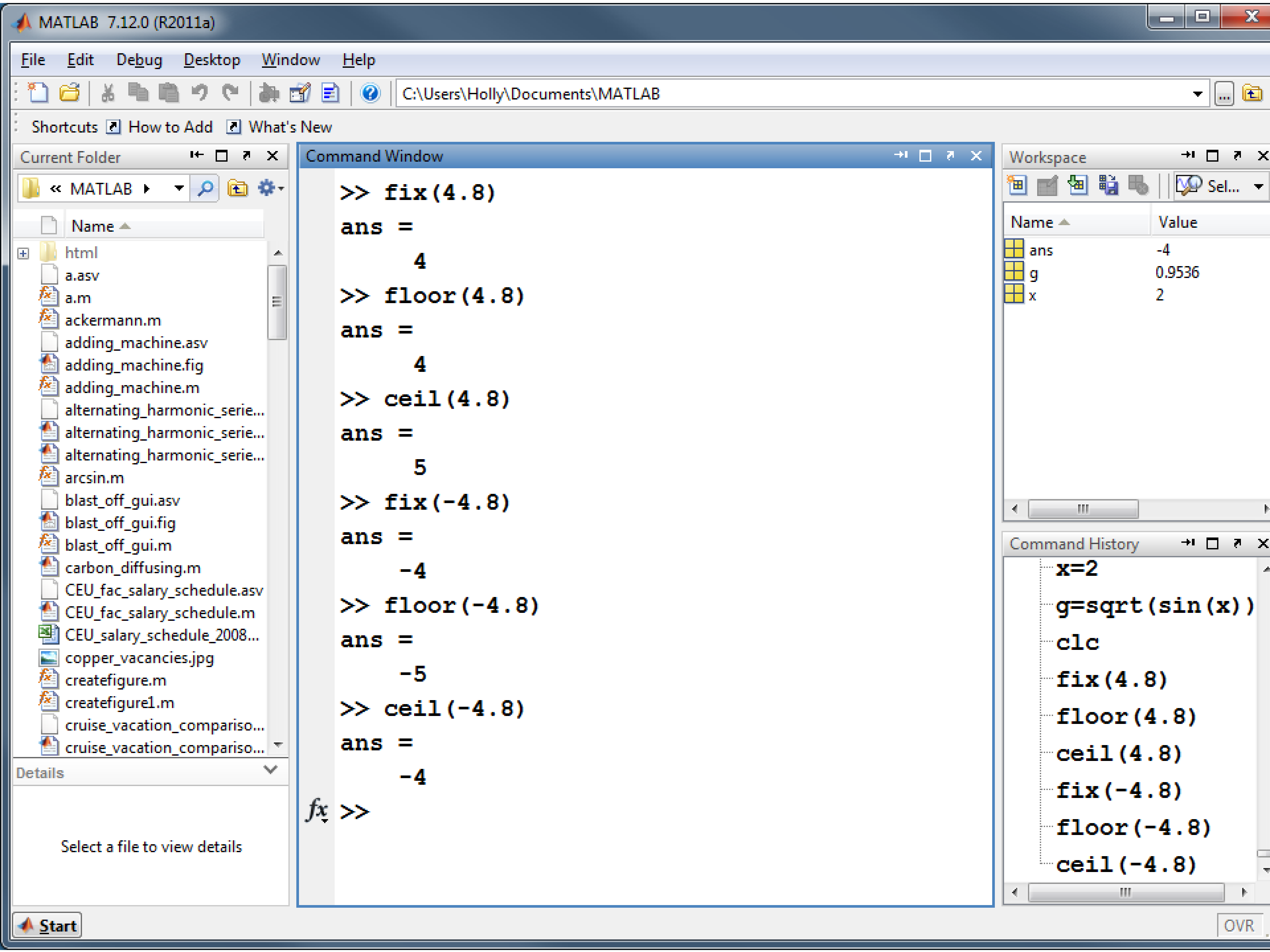
  rounds towards plus infinity



```
>> fix(4.8)
ans =
     4
>> floor(4.8)
ans =
     4
>> ceil(4.8)
ans =
     5
>>
```

Name	Value
ans	5
g	0.9536
x	2

```
%-- 8/28/2011 11
format compact
clc
x=2
g=sqrt(sin(x))
clc
fix(4.8)
floor(4.8)
ceil(4.8)
```



C:\Users\Holly\Documents\MATLAB

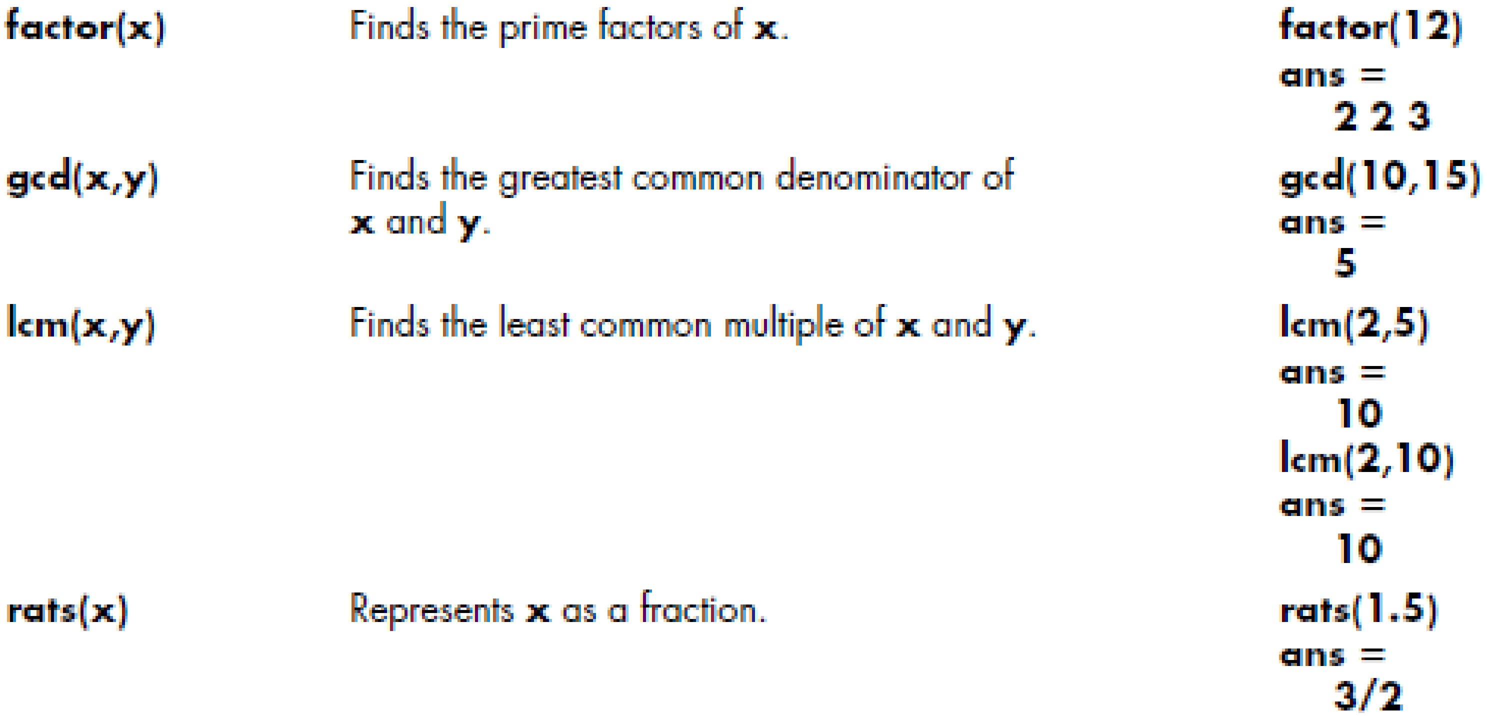
- html
- a.asv
- a.m
- ackermann.m
- adding_machine.asv
- adding_machine.fig
- adding_machine.m
- alternating_harmonic_serie...
- alternating_harmonic_serie...
- alternating_harmonic_serie...
- arcsin.m
- blast_off_gui.asv
- blast_off_gui.fig
- blast_off_gui.m
- carbon_diffusing.m
- CEU_fac_salary_schedule.asv
- CEU_fac_salary_schedule.m
- CEU_salary_schedule_2008...
- copper_vacancies.jpg
- createfigure.m
- createfigure1.m
- cruise_vacation_compariso...
- cruise_vacation_compariso...

```
>> fix(4.8)
ans =
     4
>> floor(4.8)
ans =
     4
>> ceil(4.8)
ans =
     5
>> fix(-4.8)
ans =
    -4
>> floor(-4.8)
ans =
    -5
>> ceil(-4.8)
ans =
    -4
fx >>
```

Name	Value
ans	-4
g	0.9536
x	2

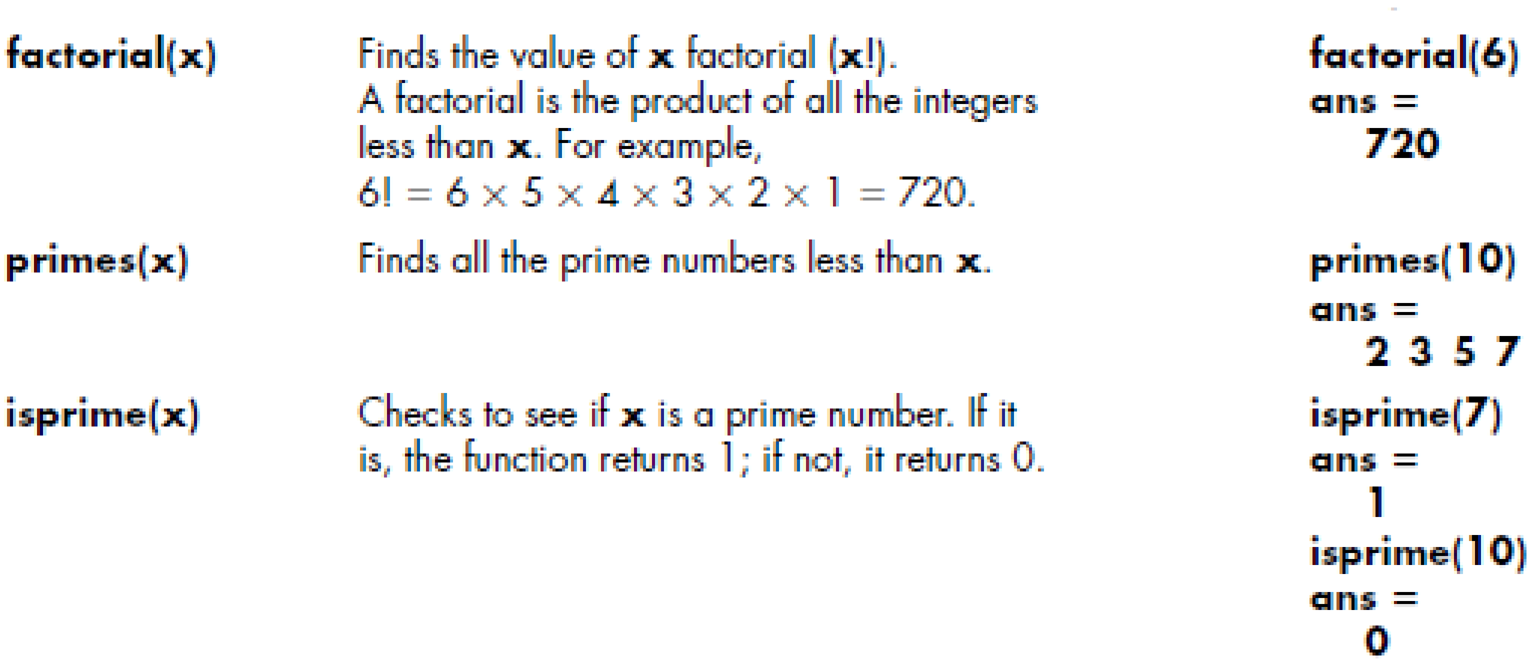
```
x=2
g=sqrt(sin(x))
clc
fix(4.8)
floor(4.8)
ceil(4.8)
fix(-4.8)
floor(-4.8)
ceil(-4.8)
```

3.3.3 Discrete Mathematics



factor(x)	Finds the prime factors of **x**.	**factor(12)** **ans =** **2 2 3**
gcd(x,y)	Finds the greatest common denominator of **x** and **y**.	**gcd(10,15)** **ans =** **5**
lcm(x,y)	Finds the least common multiple of **x** and **y**.	**lcm(2,5)** **ans =** **10** **lcm(2,10)** **ans =** **10**
rats(x)	Represents **x** as a fraction.	**rats(1.5)** **ans =** **3/2**

3.3.3 Discrete Mathematics



factorial(x)	Finds the value of **x** factorial (**x**!). A factorial is the product of all the integers less than **x**. For example, 6! = 6 × 5 × 4 × 3 × 2 × 1 = 720.	**factorial(6)** **ans =** **720**
primes(x)	Finds all the prime numbers less than **x**.	**primes(10)** **ans =** **2 3 5 7**
isprime(x)	Checks to see if **x** is a prime number. If it is, the function returns 1; if not, it returns 0.	**isprime(7)** **ans =** **1** **isprime(10)** **ans =** **0**

3.4 Trigonometric Functions

• sin(x)	sine
• cos(x)	cosine
• tan(x)	tangent
• asin(x)	inverse sine
• sinh(x)	hyperbolic sine
• asinh(x)	inverse hyperbolic sine
• sind(x)	sine with degree input
• asind(x)	inverse sin with degree output

3.5 Data Analysis Functions

- max(x)
- min(x)
- mean(x)
- median(x)
- sum(x)
- prod(x)
- sort(x)
- sortrows(x)
- size(x)
- length(x)
- numel(x)
- std(x)
- var(x)

mean(x)	Computes the mean value (or average value) of a **vector x**. For example if $\mathbf{x} = [1 \quad 5 \quad 3]$, the mean value is 3.	**x=[1, 5, 3];**
		mean(x)
		ans =
		3.0000

	Returns a row vector containing the mean value from each column of a **matrix x**.
	For example, if $\mathbf{x} = \begin{bmatrix} 1 & 5 & 3 \\ 2 & 4 & 6 \end{bmatrix}$ then the mean value of column 1 is 1.5, the mean value of column 2 is 4.5, and the mean value of column 3 is 4.5.
		x=[1, 5, 3; 2, 4, 6];
		mean(x)
		ans =
		1.5 4.5 4.5

median(x)	Finds the median of the elements of a **vector x**. For example, if $\mathbf{x} = [1 \quad 5 \quad 3]$, the median value is 3.	**x=[1, 5, 3];**
		median(x)
		ans =
		3

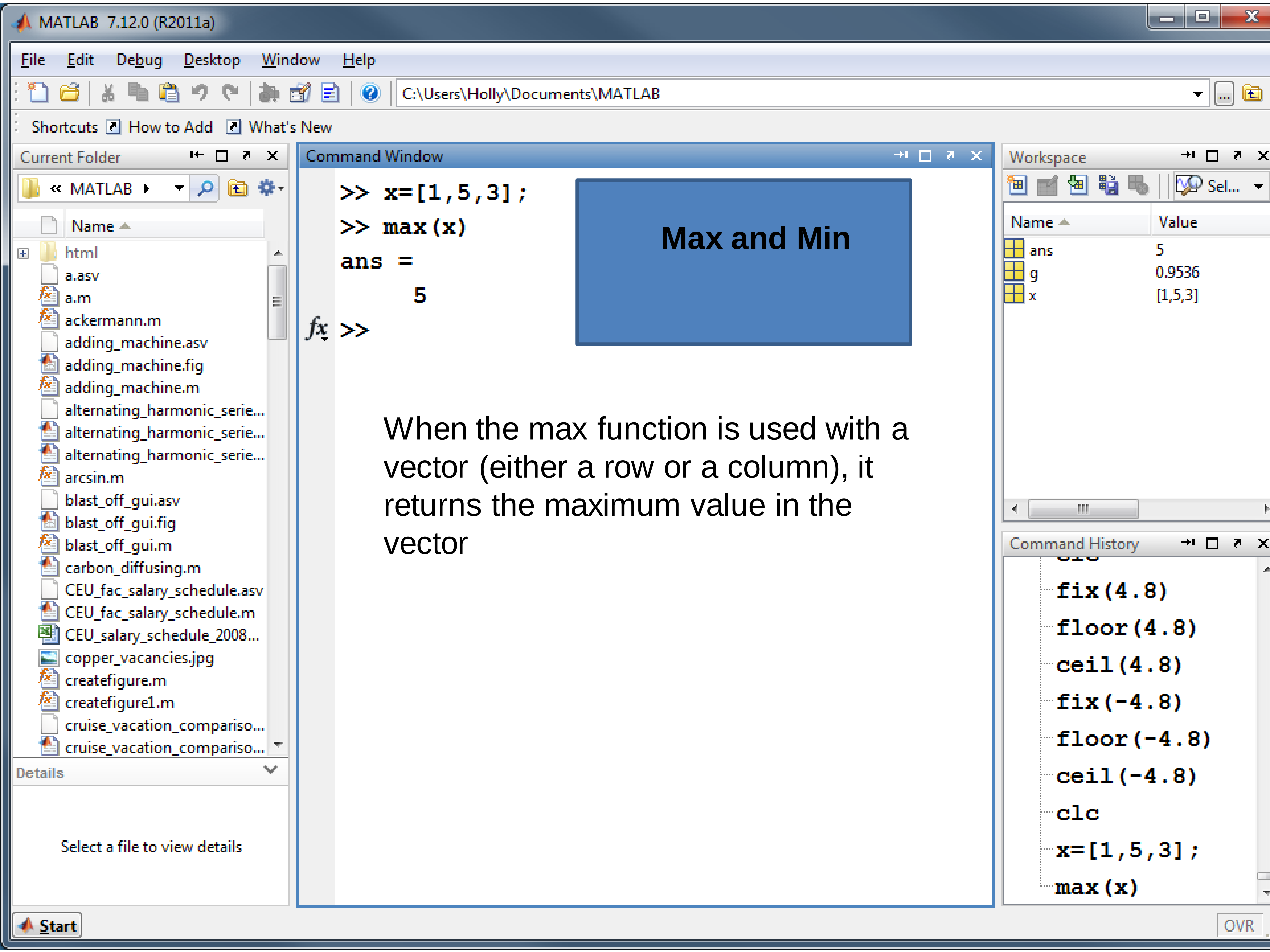
	Returns a row vector containing the median value from each column of a **matrix x**.
	For example, if $\mathbf{x} = \begin{bmatrix} 1 & 5 & 3 \\ 2 & 4 & 6 \\ 3 & 8 & 4 \end{bmatrix}$,
	then the median value from column 1 is 2, the median value from column 2 is 5, and the median value from column 3 is 4.
		x=[1, 5, 3;
		2, 4, 6;
		3, 8, 4];
		median(x)
		ans =
		2 5 4

sum(x)	Sums the elements in **vector x**. For example, if **x** = $[1 \quad 5 \quad 3]$, the sum is 9.	**x=[1, 5, 3];**
		sum(x)
		ans =
		9

	Computes a row vector containing the sum of the elements in each column of a **matrix x**. For example, if **x** = $\begin{bmatrix} 1 & 5 & 3 \\ 2 & 4 & 6 \end{bmatrix}$ then the sum of column 1 is 3, the sum of column 2 is 9, and the sum of column 3 is 9.	**x=[1, 5, 3; 2, 4, 6];**
		sum(x)
		ans =
		3 9 9

prod(x)	Computes the product of the elements of a **vector x**. For example, if **x** = $[1 \quad 5 \quad 3]$ the product is 15.	**x=[1, 5, 3];**
		prod(x)
		ans =
		15

	Computes a row vector containing the product of the elements in each column of a **matrix x**. For example, if $x = \begin{bmatrix} 1 & 5 & 3 \\ 2 & 4 & 6 \end{bmatrix}$, then the product of column 1 is 2, the product of column 2 is 20, and the product of column 3 is 18.	**x=[1, 5, 3; 2, 4, 6];**
		prod(x)
		ans =
		2 20 18



Max and Min

```
>> x=[1,5,3];
>> max(x)
ans =
     5
>>
```

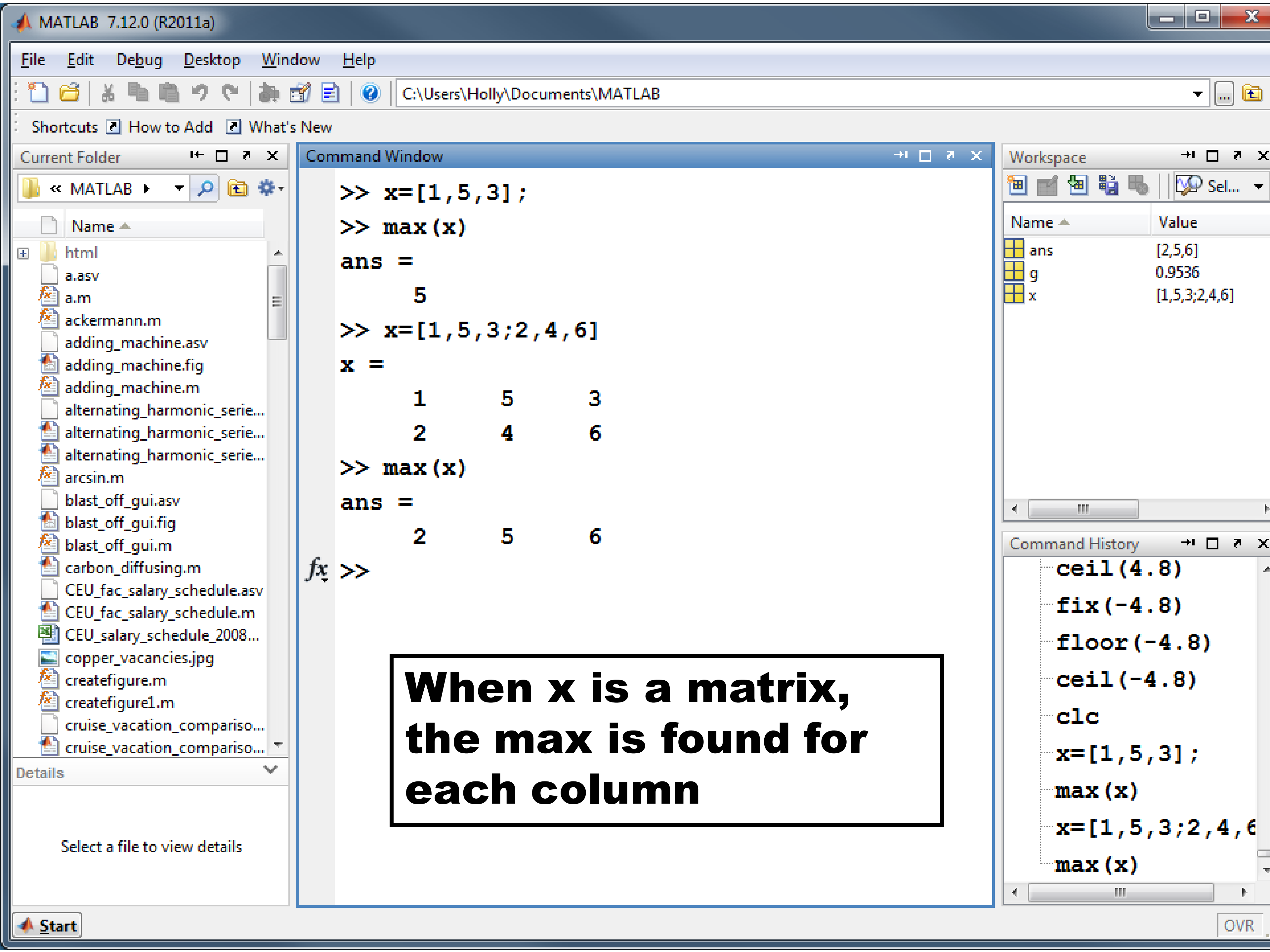
When the max function is used with a vector (either a row or a column), it returns the maximum value in the vector

html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...

Select a file to view details

Name ▲	Value
ans	5
g	0.9536
x	[1,5,3]

```
fix(4.8)
floor(4.8)
ceil(4.8)
fix(-4.8)
floor(-4.8)
ceil(-4.8)
clc
x=[1,5,3];
max(x)
```

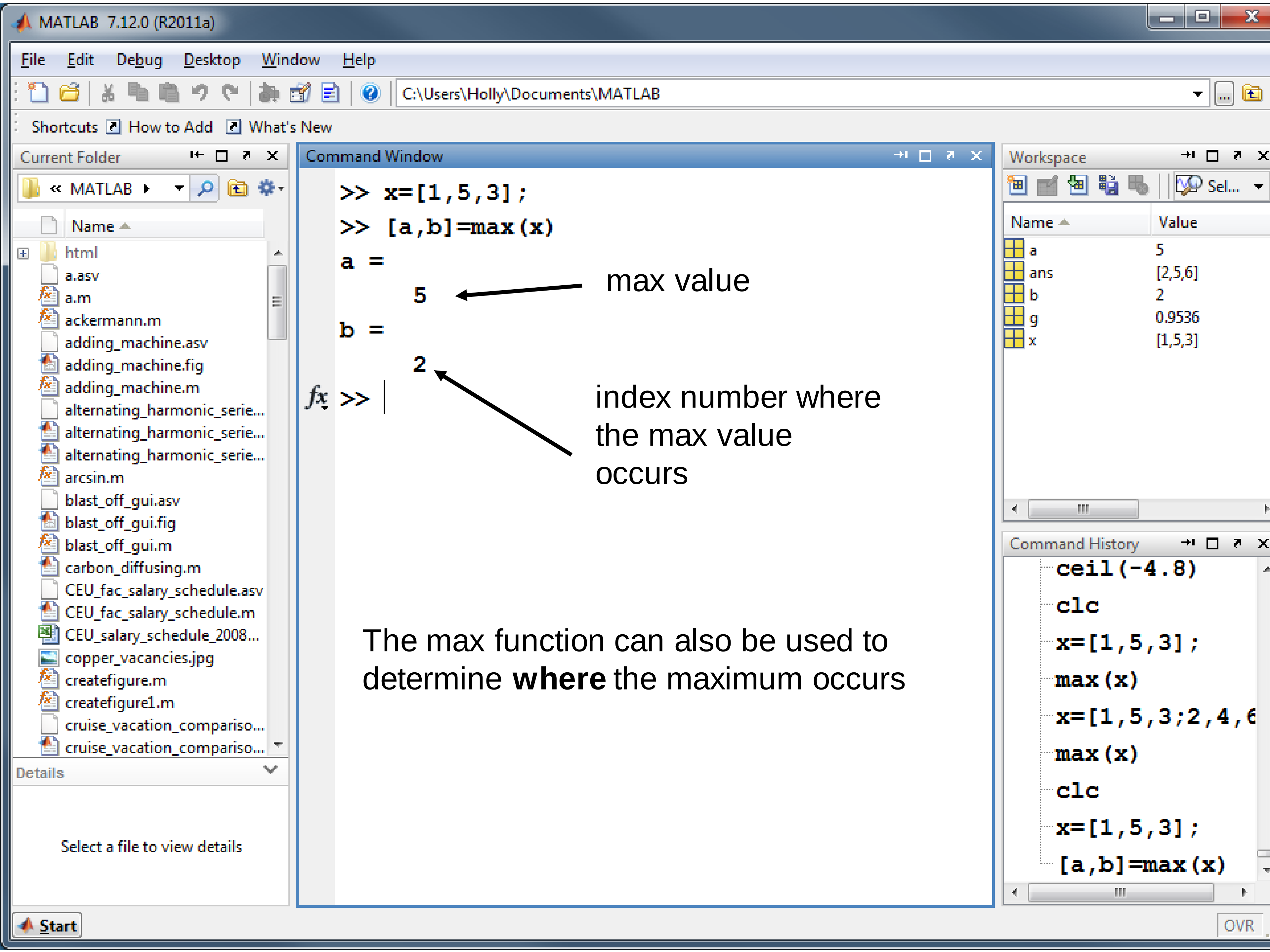


```
>> x=[1,5,3];
>> max(x)
ans =
     5
>> x=[1,5,3;2,4,6]
x =
     1     5     3
     2     4     6
>> max(x)
ans =
     2     5     6
fx >>
```

Name	Value
ans	[2,5,6]
g	0.9536
x	[1,5,3;2,4,6]

```
ceil(4.8)
fix(-4.8)
floor(-4.8)
ceil(-4.8)
clc
x=[1,5,3];
max(x)
x=[1,5,3;2,4,6
max(x)
```

When x is a matrix, the max is found for each column


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=[1,5,3];
>> [a,b]=max(x)
a =
        5
b =
        2
fx >>
```

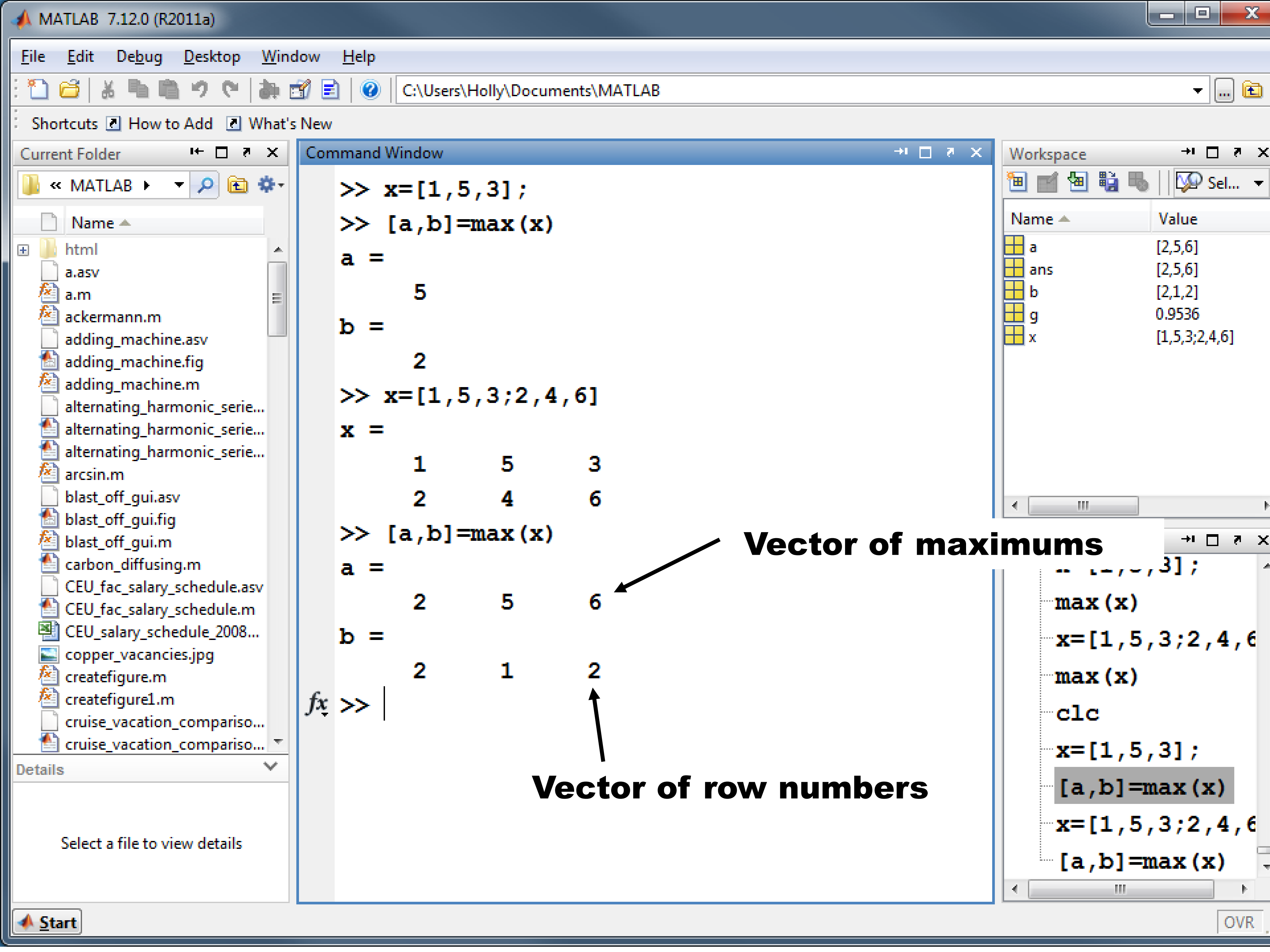
max value

index number where the max value occurs

The max function can also be used to determine **where** the maximum occurs

Name	Value
a	5
ans	[2,5,6]
b	2
g	0.9536
x	[1,5,3]

```
ceil(-4.8)
clc
x=[1,5,3];
max(x)
x=[1,5,3;2,4,6
max(x)
clc
x=[1,5,3];
[a,b]=max(x)
```



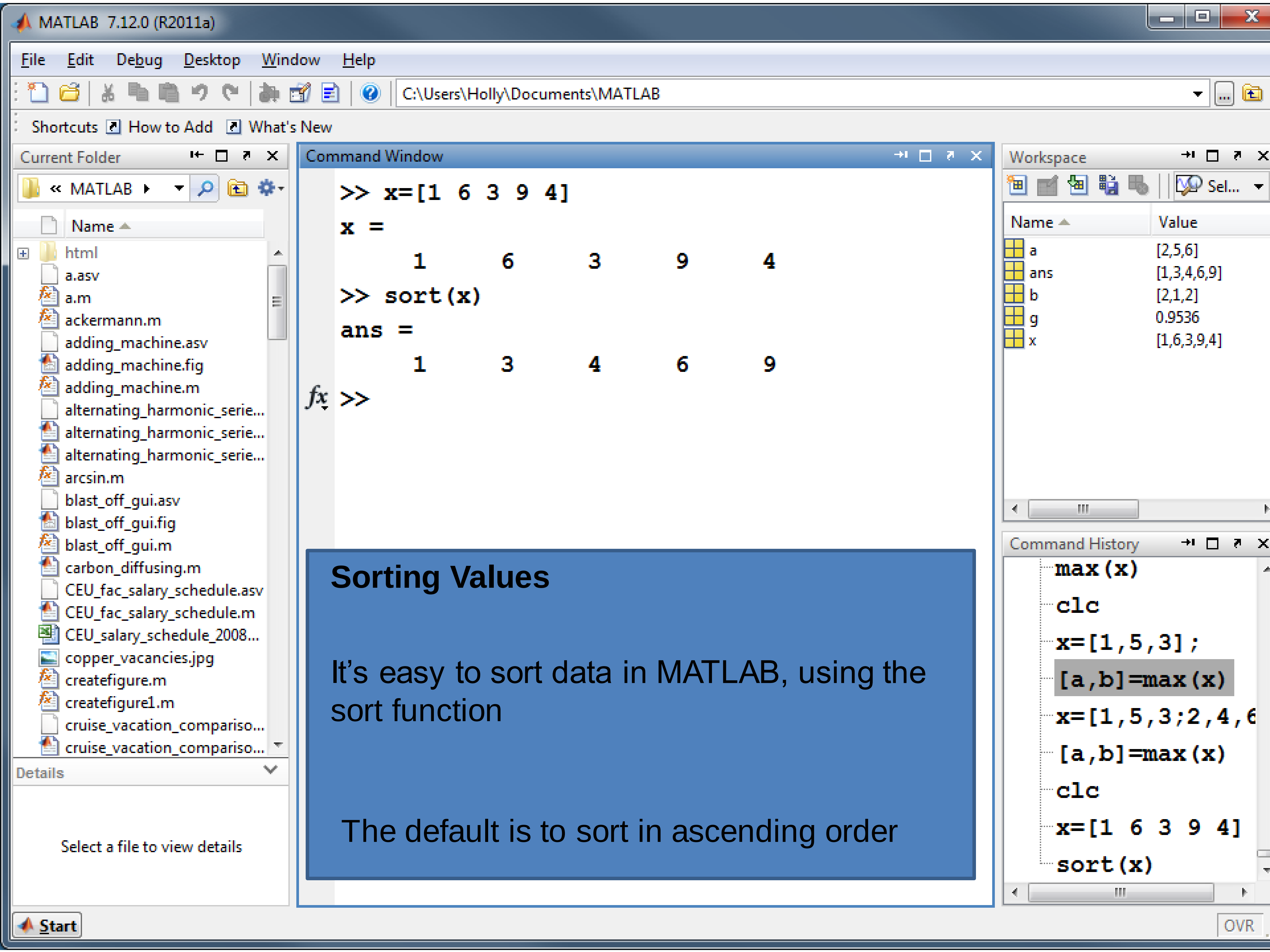
```
>> x=[1,5,3];
>> [a,b]=max(x)
a =

     5

b =

     2

>> x=[1,5,3;2,4,6]
x =

     1     5     3
     2     4     6

>> [a,b]=max(x)
a =

     2     5     6

b =

     2     1     2

fx >> 
```

Name	Value
a	[2,5,6]
ans	[2,5,6]
b	[2,1,2]
g	0.9536
x	[1,5,3;2,4,6]

```
x=[1,5,3];
max(x)
x=[1,5,3;2,4,6
max(x)
clc
x=[1,5,3];
[a,b]=max(x)
x=[1,5,3;2,4,6
[a,b]=max(x)
```

Vector of maximums

Vector of row numbers



MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
MATLAB
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=[1 6 3 9 4]
x =
     1     6     3     9     4
>> sort(x)
ans =
     1     3     4     6     9
fx >>
```

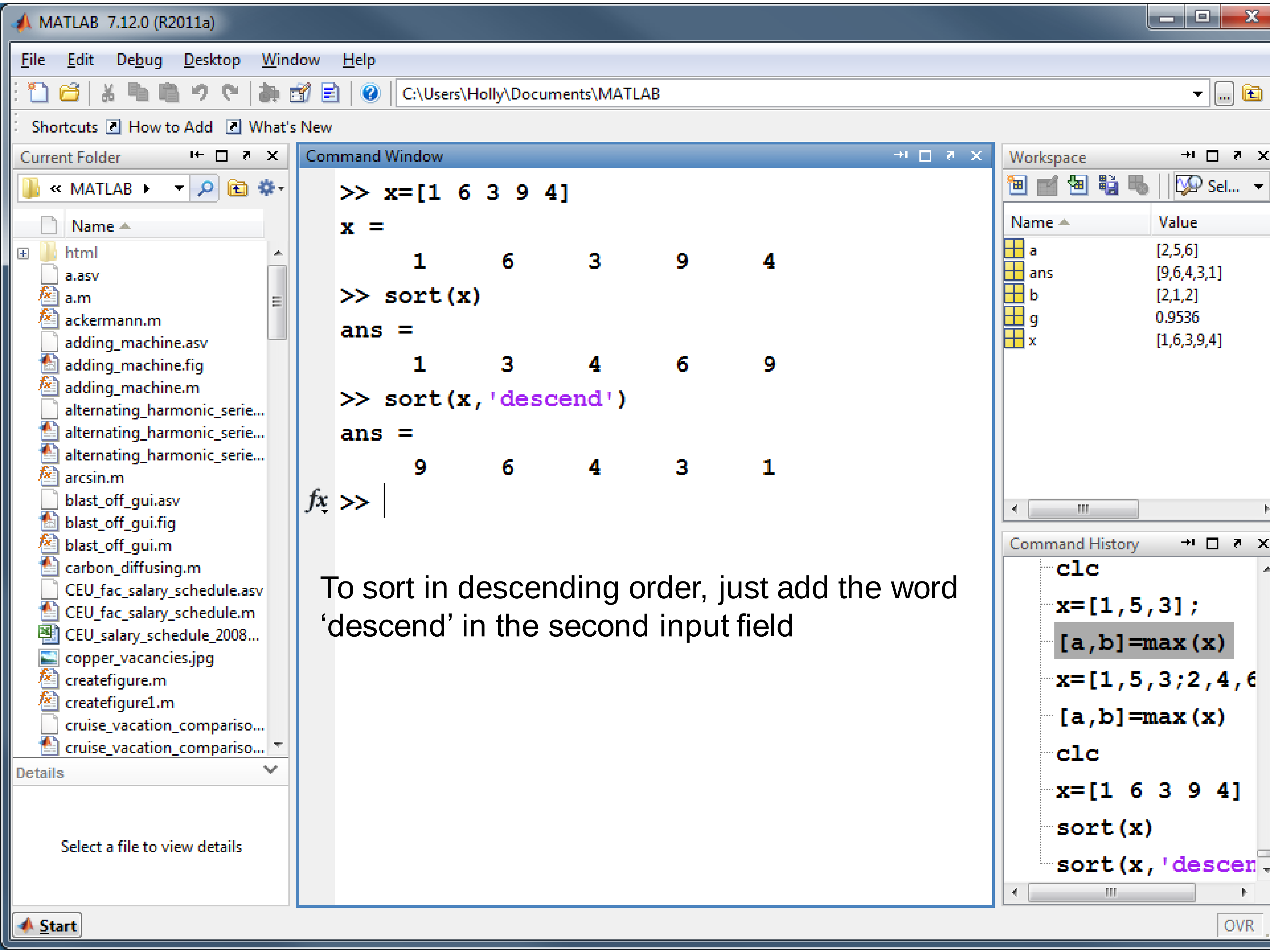
Name	Value
a	[2,5,6]
ans	[1,3,4,6,9]
b	[2,1,2]
g	0.9536
x	[1,6,3,9,4]

```
max(x)
clc
x=[1,5,3];
[a,b]=max(x)
x=[1,5,3;2,4,6
[a,b]=max(x)
clc
x=[1 6 3 9 4]
sort(x)
```

Sorting Values

It's easy to sort data in MATLAB, using the sort function

The default is to sort in ascending order

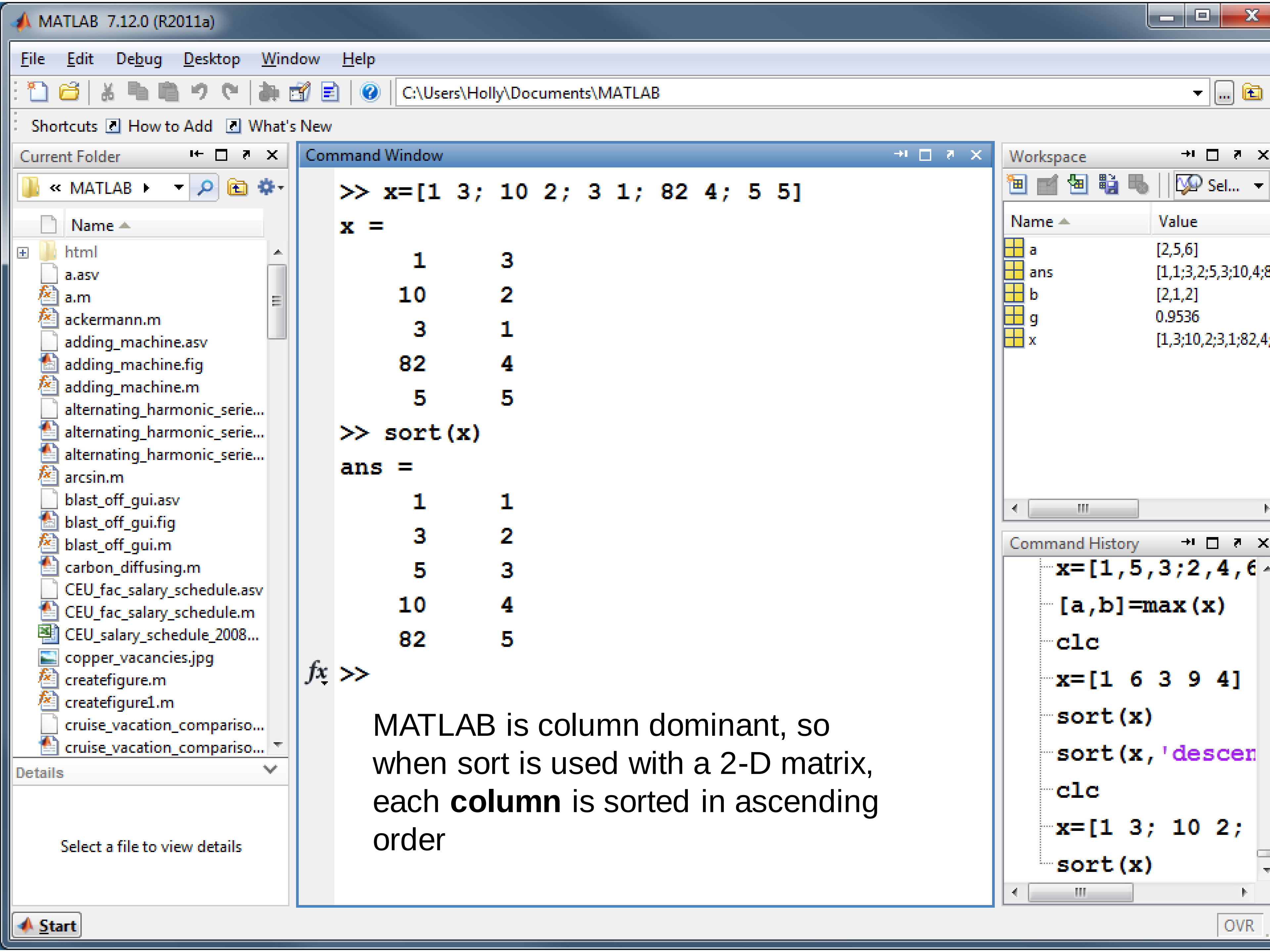

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=[1 6 3 9 4]
x =
     1     6     3     9     4
>> sort(x)
ans =
     1     3     4     6     9
>> sort(x,'descend')
ans =
     9     6     4     3     1
>> 
```

Name	Value
a	[2,5,6]
ans	[9,6,4,3,1]
b	[2,1,2]
g	0.9536
x	[1,6,3,9,4]

```
clc
x=[1,5,3];
[a,b]=max(x)
x=[1,5,3;2,4,6
[a,b]=max(x)
clc
x=[1 6 3 9 4]
sort(x)
sort(x,'descen
```

To sort in descending order, just add the word 'descend' in the second input field

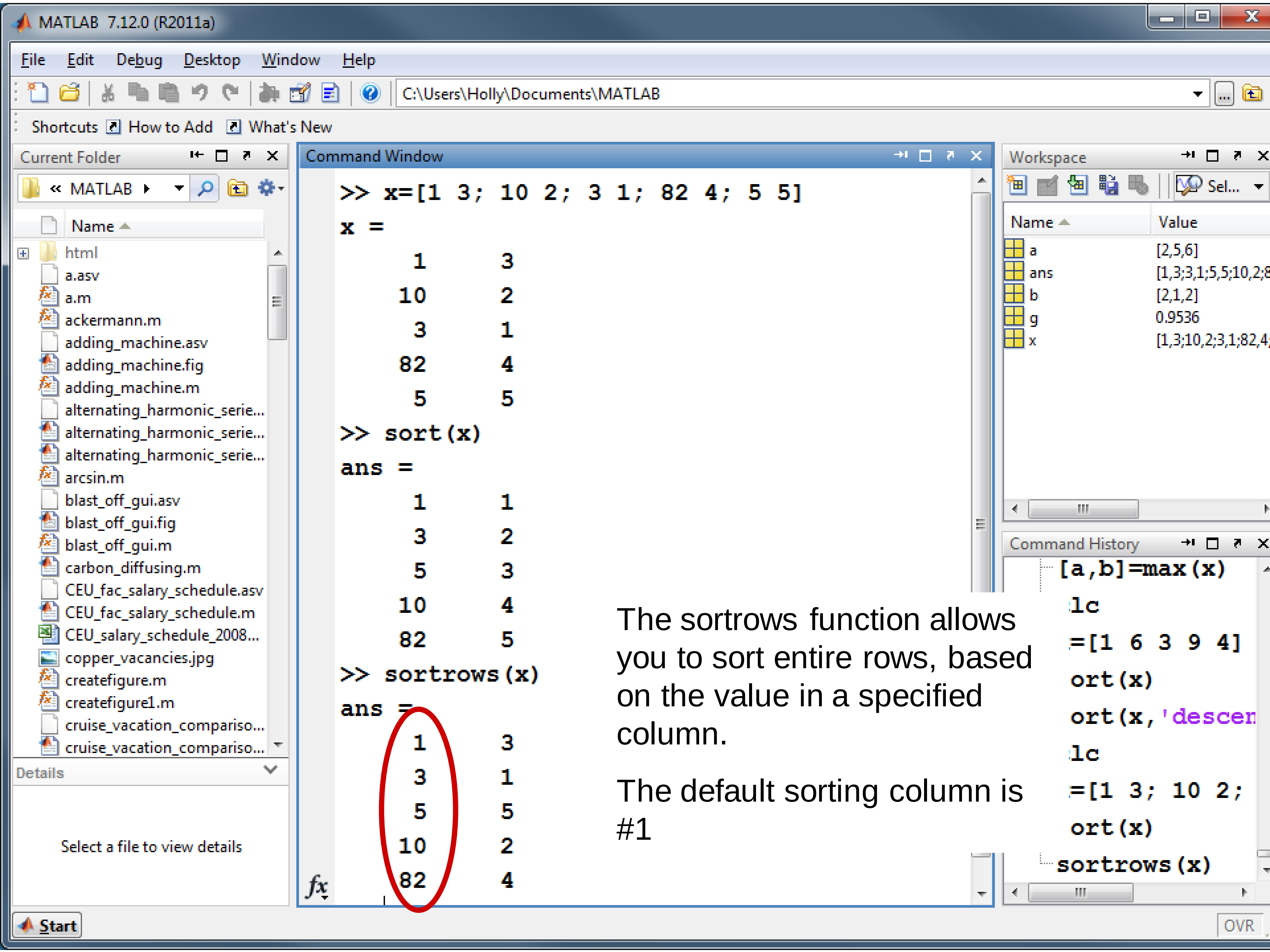

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=[1 3; 10 2; 3 1; 82 4; 5 5]
x =
     1     3
    10     2
     3     1
    82     4
     5     5
>> sort(x)
ans =
     1     1
     3     2
     5     3
    10     4
    82     5
fx >>
```

MATLAB is column dominant, so when sort is used with a 2-D matrix, each **column** is sorted in ascending order

Name	Value
a	[2,5,6]
ans	[1,1;3,2;5,3;10,4;8
b	[2,1,2]
g	0.9536
x	[1,3;10,2;3,1;82,4,

```
x=[1,5,3;2,4,6
[a,b]=max(x)
clc
x=[1 6 3 9 4]
sort(x)
sort(x,'descen
clc
x=[1 3; 10 2;
sort(x)
```



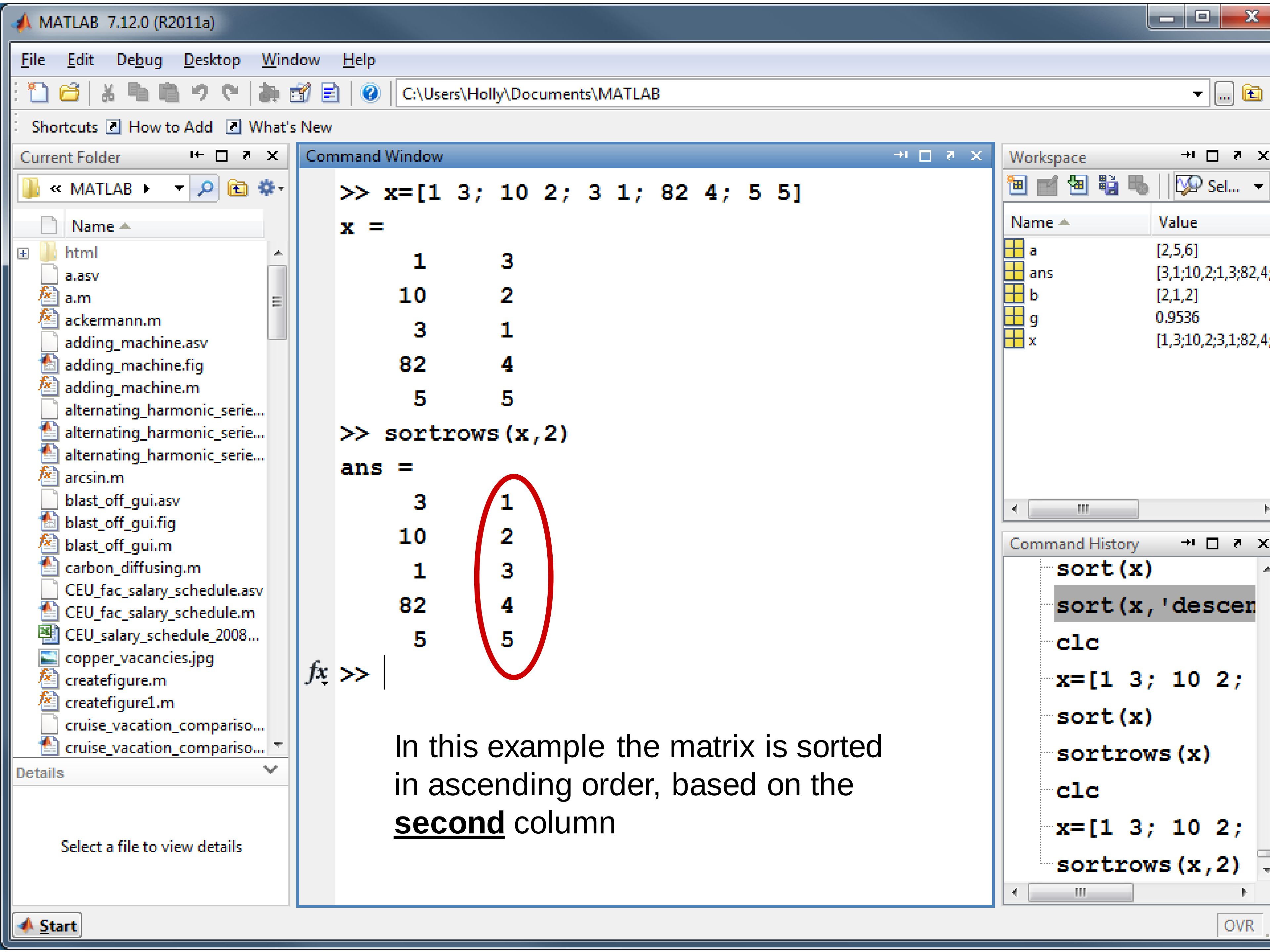
```
>> x=[1 3; 10 2; 3 1; 82 4; 5 5]
x =
     1     3
    10     2
     3     1
    82     4
     5     5
>> sort(x)
ans =
     1     1
     3     2
     5     3
    10     4
    82     5
>> sortrows(x)
ans =
     1     3
     3     1
     5     5
    10     2
    82     4
```

Name ▲	Value
a	[2,5,6]
ans	[1,3;3,1;5,5;10,2;8
b	[2,1,2]
g	0.9536
x	[1,3;10,2;3,1;82,4,

```
[a,b]=max(x)
lc
=[1 6 3 9 4]
ort(x)
ort(x,'descen
lc
=[1 3; 10 2;
ort(x)
sortrows(x)
```

The sortrows function allows you to sort entire rows, based on the value in a specified column.

The default sorting column is #1

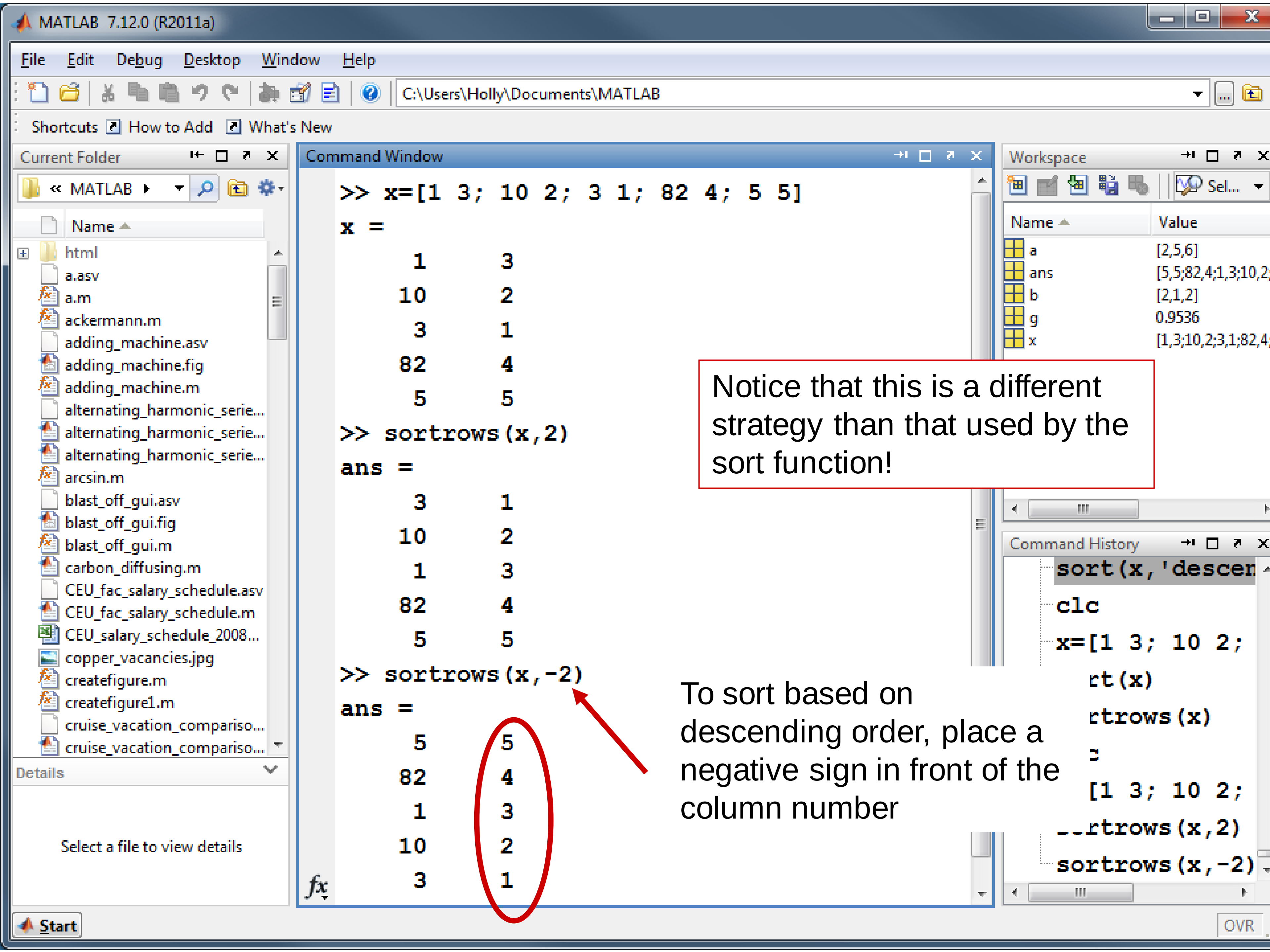

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=[1 3; 10 2; 3 1; 82 4; 5 5]
x =
     1     3
    10     2
     3     1
    82     4
     5     5
>> sortrows(x,2)
ans =
     3     1
    10     2
     1     3
    82     4
     5     5
>>
```

In this example the matrix is sorted in ascending order, based on the **second** column

Name	Value
a	[2,5,6]
ans	[3,1;10,2;1,3;82,4;
b	[2,1,2]
g	0.9536
x	[1,3;10,2;3,1;82,4;

```
sort(x)
sort(x,'descen
clc
x=[1 3; 10 2;
sort(x)
sortrows(x)
clc
x=[1 3; 10 2;
sortrows(x,2)
```


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=[1 3; 10 2; 3 1; 82 4; 5 5]
x =
     1     3
    10     2
     3     1
    82     4
     5     5
>> sortrows(x,2)
ans =
     3     1
    10     2
     1     3
    82     4
     5     5
>> sortrows(x,-2)
ans =
     5     5
    82     4
     1     3
    10     2
     3     1
```

Name	Value
a	[2,5,6]
ans	[5,5;82,4;1,3;10,2,
b	[2,1,2]
g	0.9536
x	[1,3;10,2;3,1;82,4,

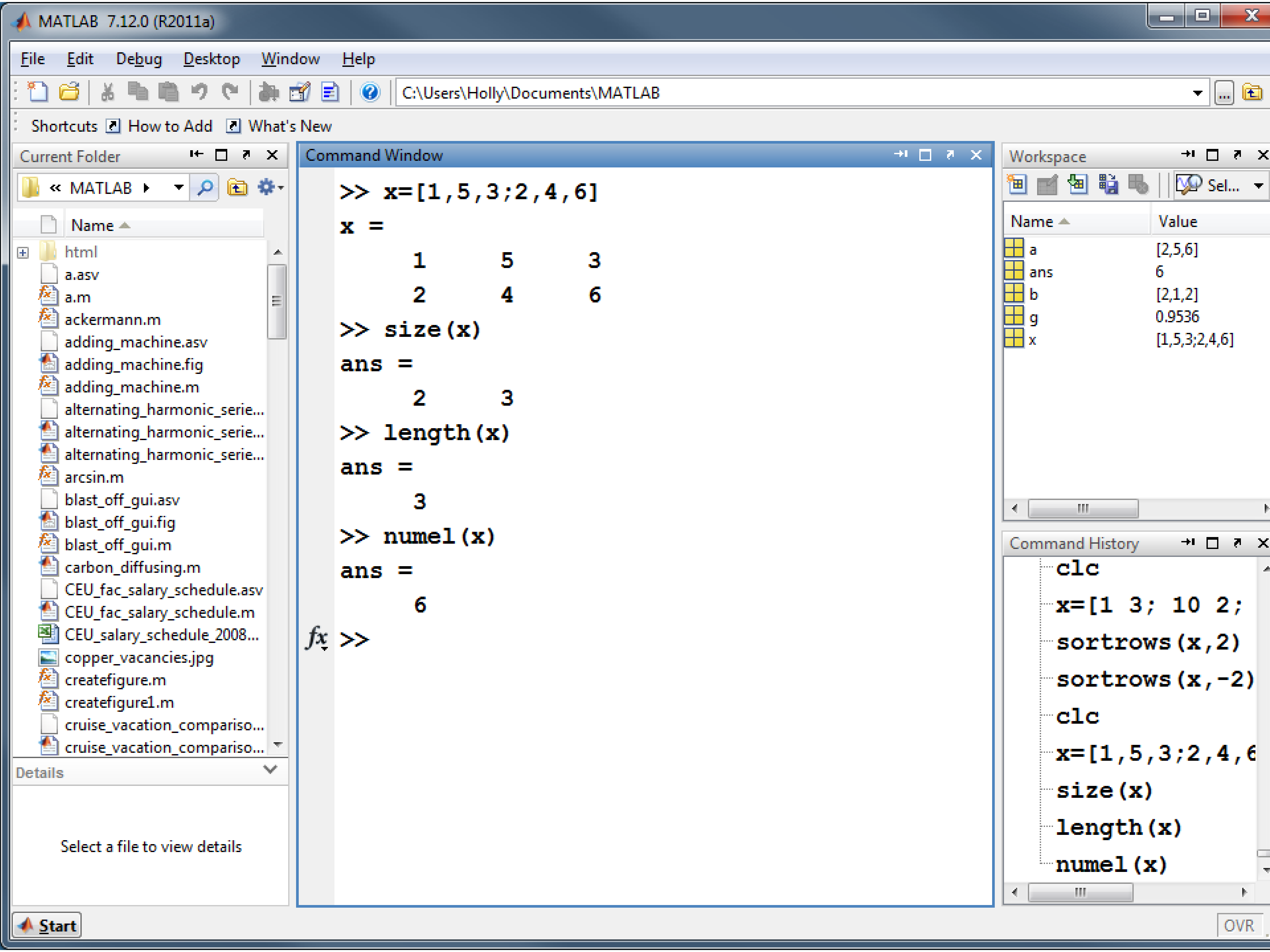
```
sort(x,'descen
clc
x=[1 3; 10 2;
rt(x)
rtrows(x)
c
[1 3; 10 2;
ortrows(x,2)
sortrows(x,-2)
```

Notice that this is a different strategy than that used by the sort function!

To sort based on descending order, place a negative sign in front of the column number

Determining Matrix Size

- size(x) number of rows and columns
- length(x) biggest dimension
- numel(x) total number of elements



```
>> x=[1,5,3;2,4,6]
x =
     1     5     3
     2     4     6
>> size(x)
ans =
     2     3
>> length(x)
ans =
     3
>> numel(x)
ans =
     6
>>
```

Name	Value
a	[2,5,6]
ans	6
b	[2,1,2]
g	0.9536
x	[1,5,3;2,4,6]

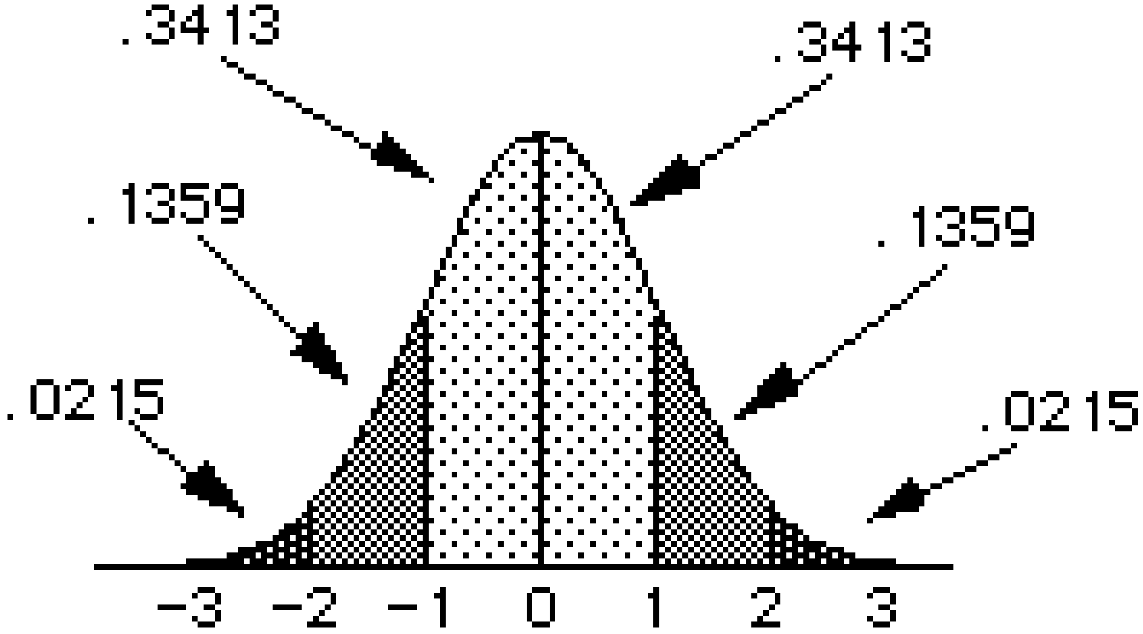
```
clc
x=[1 3; 10 2;
sortrows(x,2)
sortrows(x,-2)
clc
x=[1,5,3;2,4,6
size(x)
length(x)
numel(x)
```

Variance and Standard Deviation

- std(x) σ
- var(x) σ^2

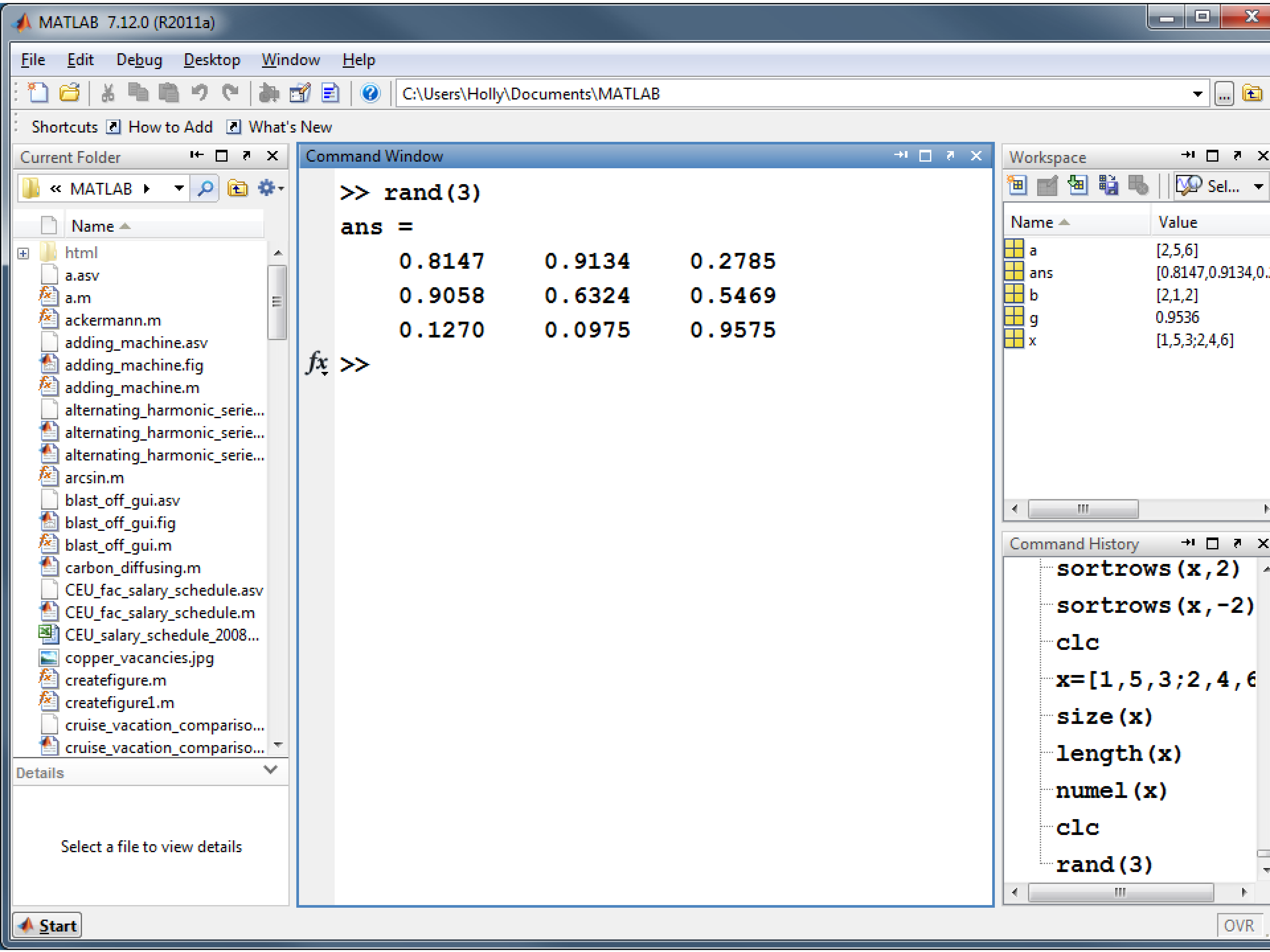
$$\sigma^2 = \frac{\sum_{k=1}^{N}\left(x_k - \mu\right)^2}{N-1}$$

Standard Deviation



3.6 Random Numbers

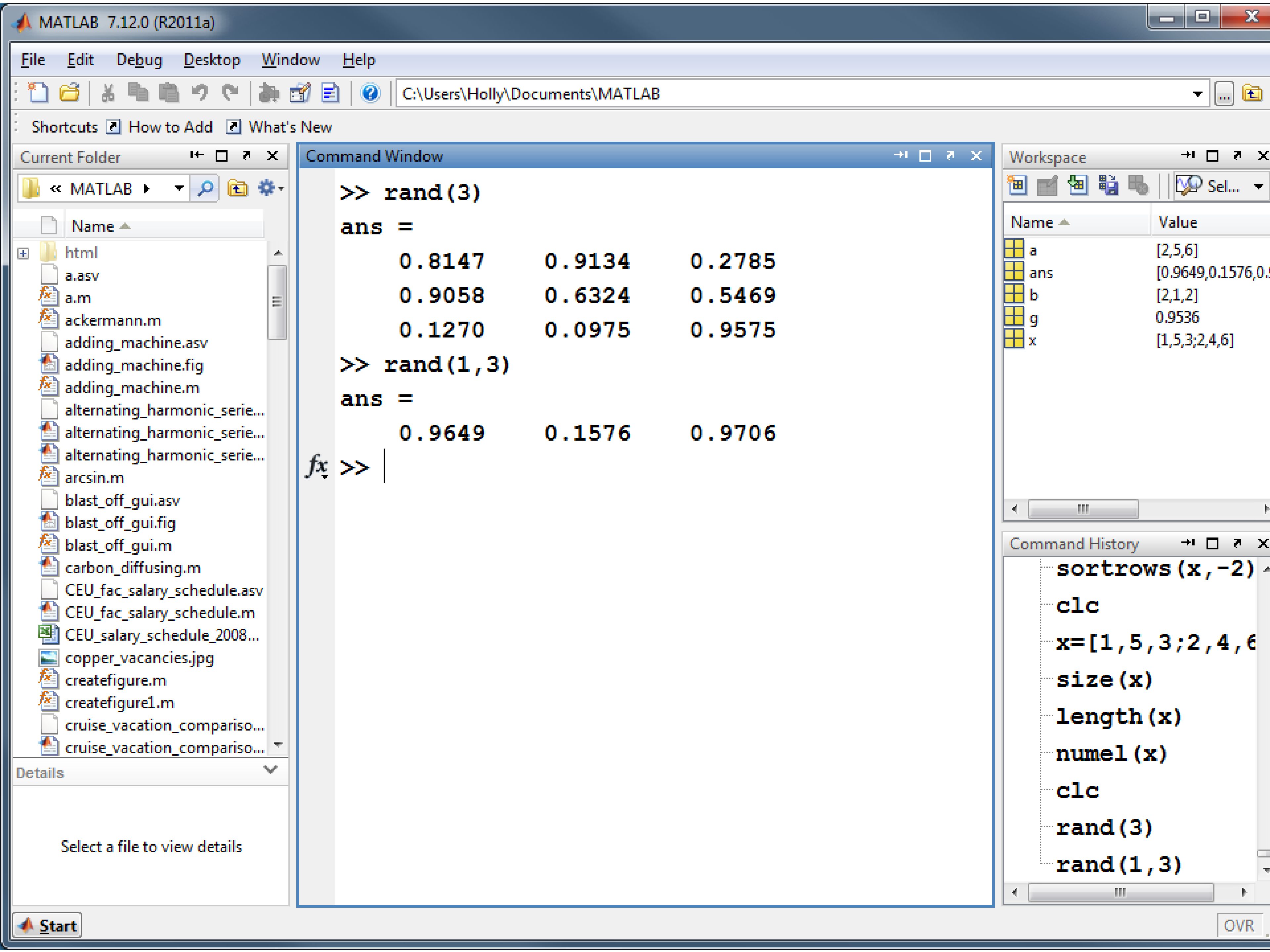
- rand(x)
  - Returns an x by x matrix of random numbers between 0 and 1
- rand(n,m)
  - Returns an n by m matrix of random numbers
- These random numbers are evenly distributed



```
>> rand(3)
ans =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
>>
```

Name	Value
a	[2,5,6]
ans	[0.8147,0.9134,0.
b	[2,1,2]
g	0.9536
x	[1,5,3;2,4,6]

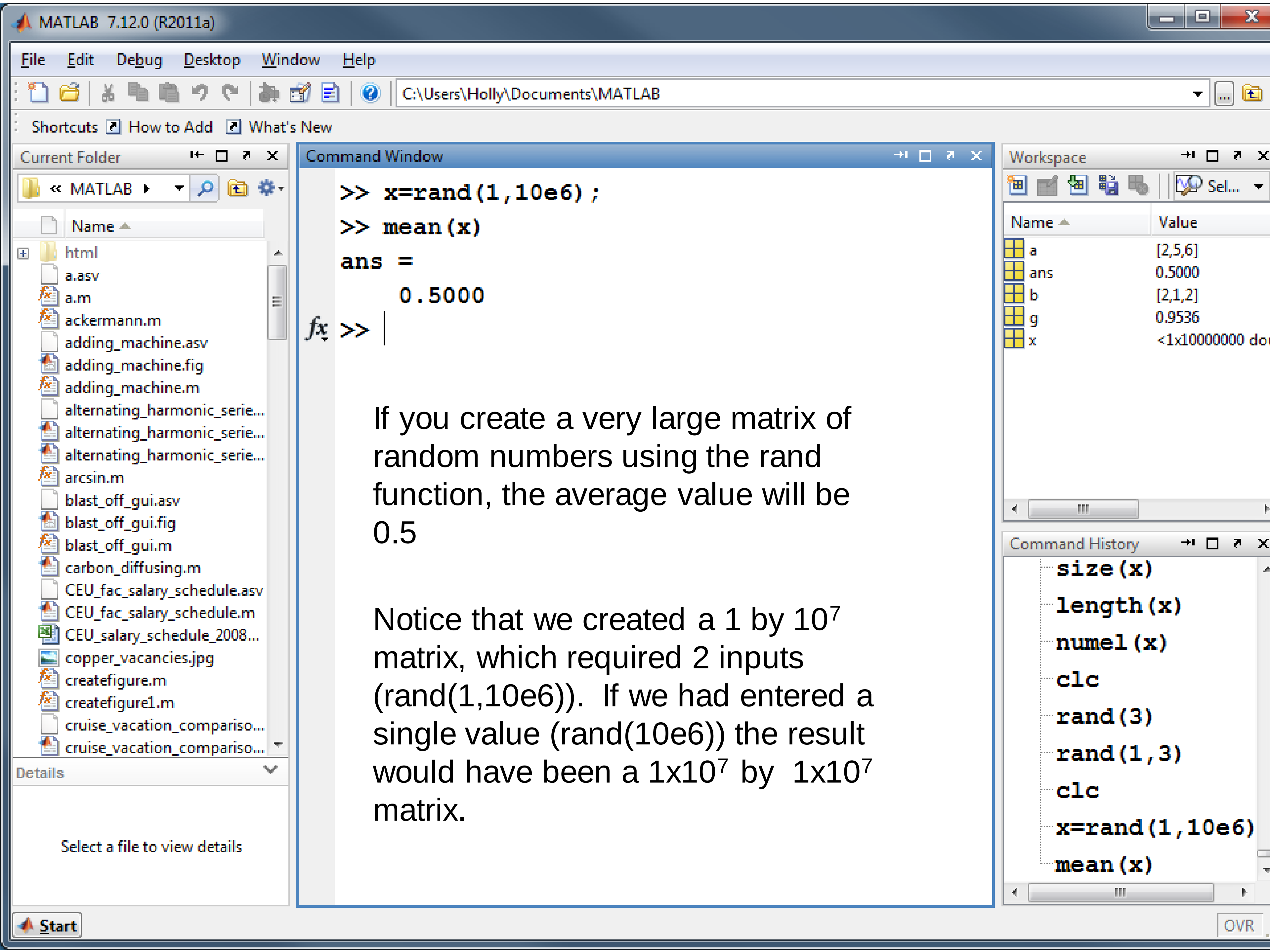
```
sortrows(x,2)
sortrows(x,-2)
clc
x=[1,5,3;2,4,6
size(x)
length(x)
numel(x)
clc
rand(3)
```



```
>> rand(3)
ans =
     0.8147    0.9134    0.2785
     0.9058    0.6324    0.5469
     0.1270    0.0975    0.9575
>> rand(1,3)
ans =
     0.9649    0.1576    0.9706
fx >> 
```

Name	Value
a	[2,5,6]
ans	[0.9649,0.1576,0.9
b	[2,1,2]
g	0.9536
x	[1,5,3;2,4,6]

```
sortrows(x,-2)
clc
x=[1,5,3;2,4,6
size(x)
length(x)
numel(x)
clc
rand(3)
rand(1,3)
```


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name ▲
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=rand(1,10e6);
>> mean(x)

ans =

    0.5000

fx >> 
```

If you create a very large matrix of random numbers using the rand function, the average value will be 0.5

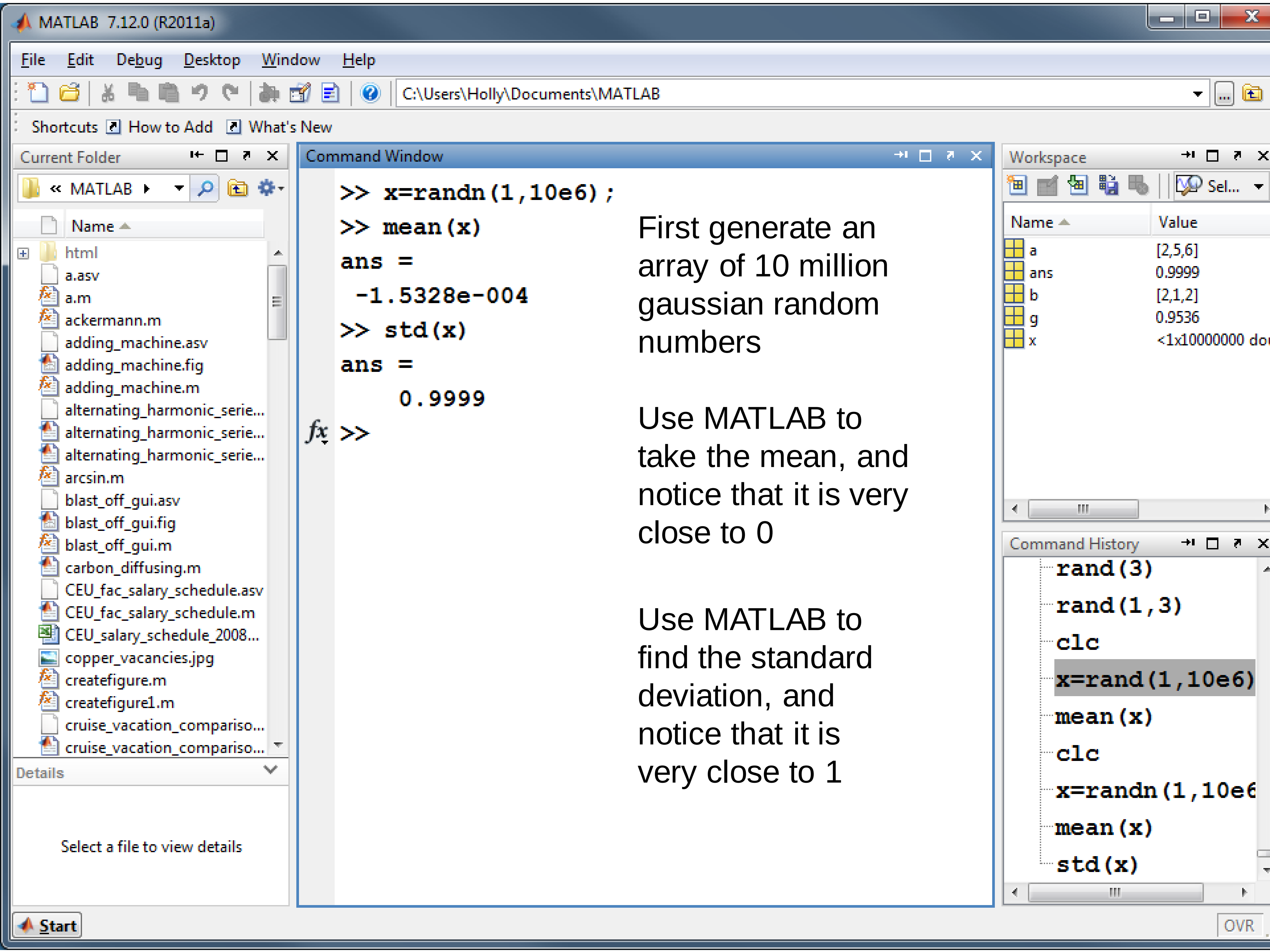
Notice that we created a 1 by 10^7 matrix, which required 2 inputs (rand(1,10e6)). If we had entered a single value (rand(10e6)) the result would have been a $1\text{x}10^7$ by $1\text{x}10^7$ matrix.

Name ▲	Value
a	[2,5,6]
ans	0.5000
b	[2,1,2]
g	0.9536
x	<1x10000000 do

```
size(x)
length(x)
numel(x)
clc
rand(3)
rand(1,3)
clc
x=rand(1,10e6)
mean(x)
```

Gaussian Random numbers

- randn(n)
- Also called a normal distribution
- Generates numbers with a mean of 0 and a standard deviation of 1


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=randn(1,10e6);
>> mean(x)
ans =
 -1.5328e-004
>> std(x)
ans =
     0.9999
fx >>
```

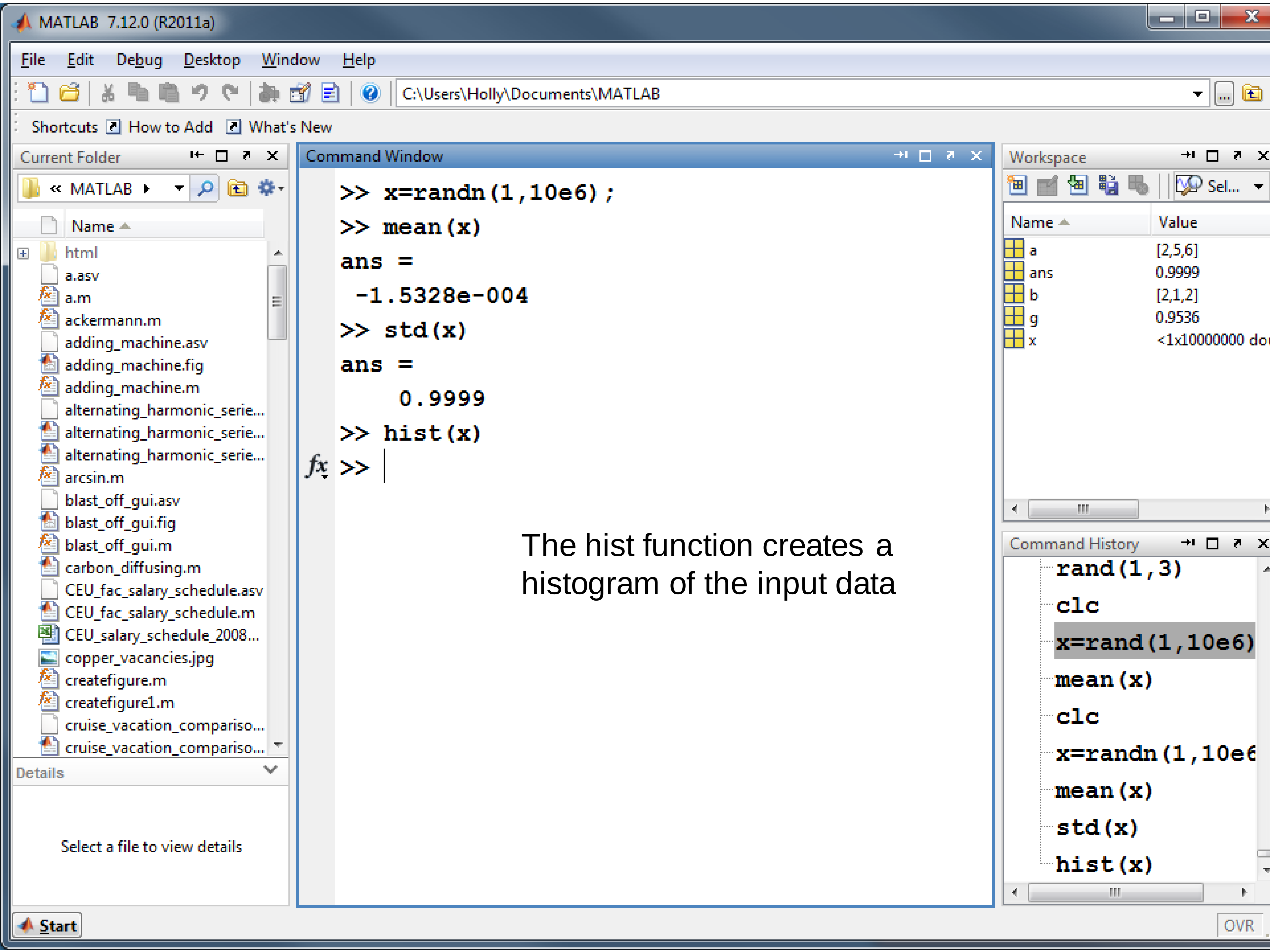
First generate an array of 10 million gaussian random numbers

Use MATLAB to take the mean, and notice that it is very close to 0

Use MATLAB to find the standard deviation, and notice that it is very close to 1

Name	Value
a	[2,5,6]
ans	0.9999
b	[2,1,2]
g	0.9536
x	<1x10000000 do

```
rand(3)
rand(1,3)
clc
x=rand(1,10e6)
mean(x)
clc
x=randn(1,10e6
mean(x)
std(x)
```


MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> x=randn(1,10e6);
>> mean(x)
ans =
 -1.5328e-004
>> std(x)
ans =
    0.9999
>> hist(x)
fx >> 
```

Name	Value
a	[2,5,6]
ans	0.9999
b	[2,1,2]
g	0.9536
x	<1x10000000 do

```
rand(1,3)
clc
x=rand(1,10e6)
mean(x)
clc
x=randn(1,10e6
mean(x)
std(x)
hist(x)
```

The hist function creates a histogram of the input data

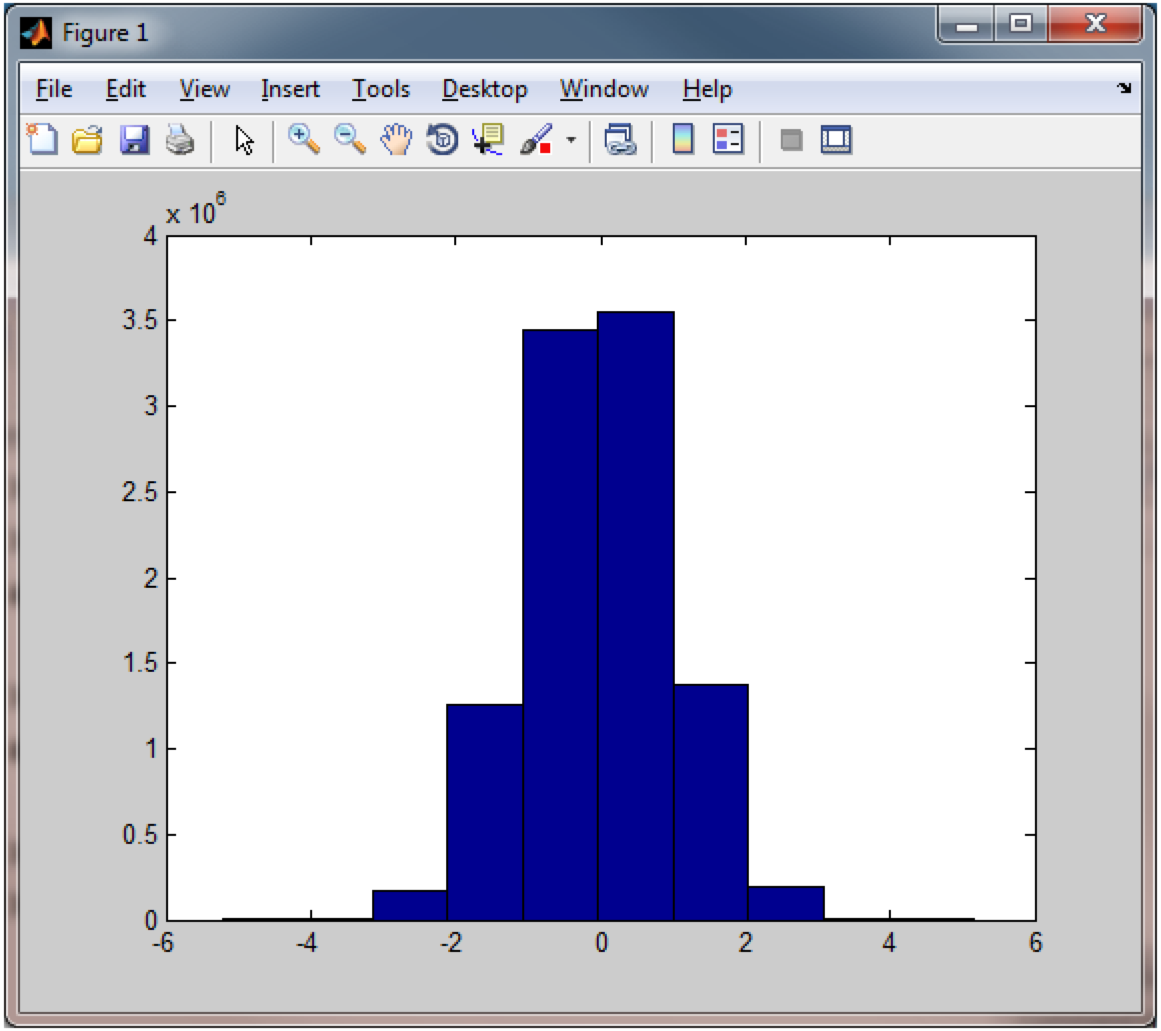


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
x 10^6
4
3.5
3
2.5
2
1.5
1
0.5
0
-6
-4
-2
0
2
4
6

To generate random numbers between other bounds…

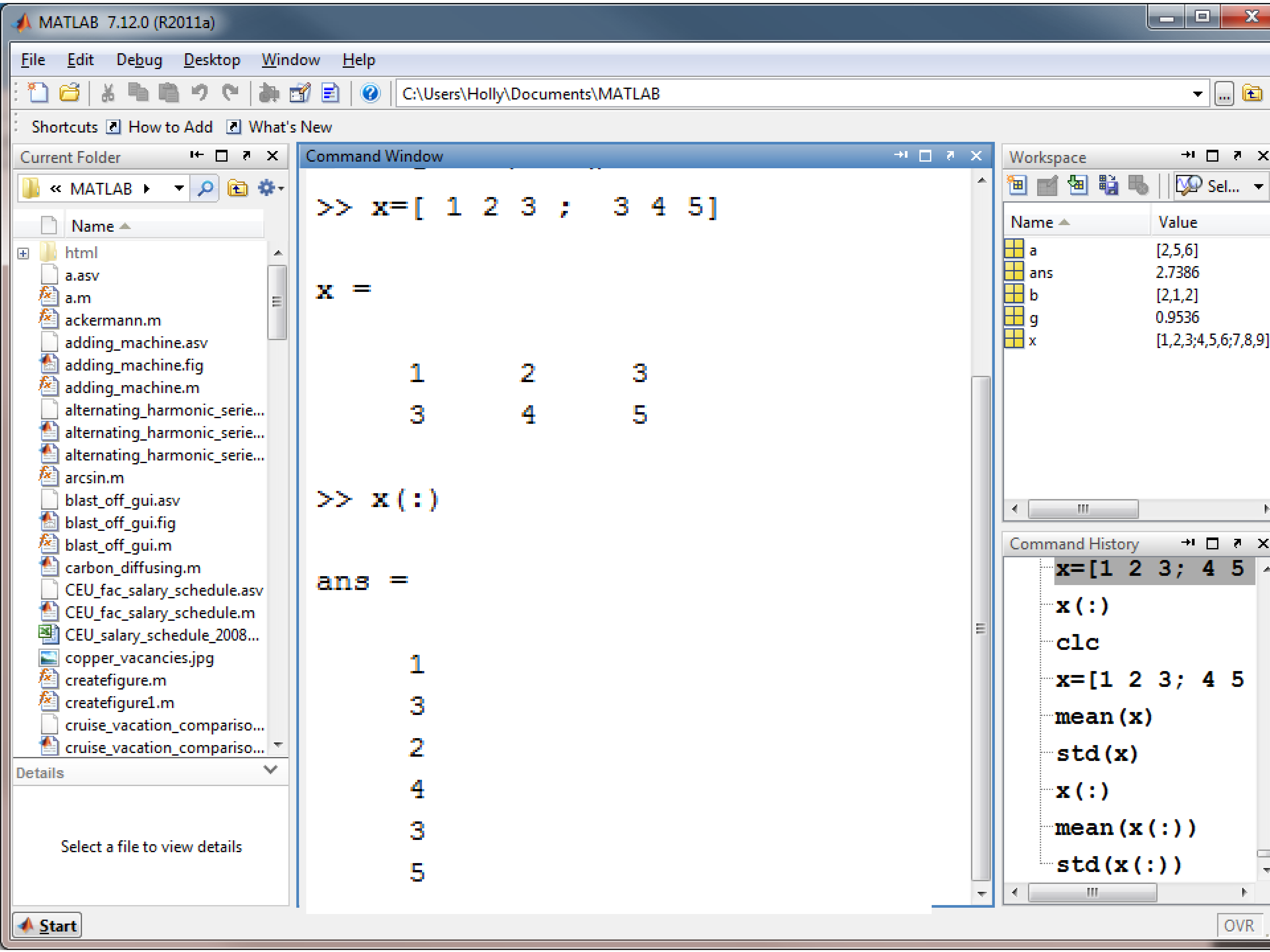
$$x = (b - a) \cdot r + a$$

a and b are the upper and lower bounds

r is the array of random numbers

More about Manipulating Matrices

- M(:)
  - Converts a two dimensional matrix to a single column



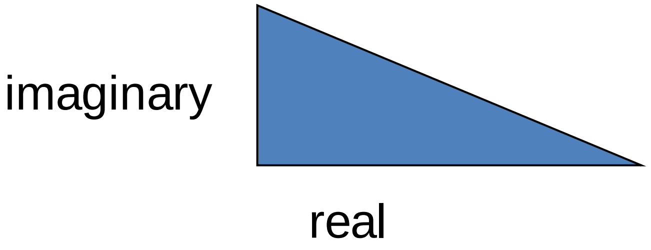
```
>> x=[ 1 2 3 ;  3 4 5]

x =

     1     2     3
     3     4     5


>> x(:)

ans =

     1
     3
     2
     4
     3
     5
```

Name ▲	Value
a	[2,5,6]
ans	2.7386
b	[2,1,2]
g	0.9536
x	[1,2,3;4,5,6;7,8,9]

```
x=[1 2 3; 4 5
x(:)
clc
x=[1 2 3; 4 5
mean(x)
std(x)
x(:)
mean(x(:))
std(x(:))
```

3.7 Complex Numbers

- complex(x,y)
- real(A) used if A is a complex number
- imag(A)
- conj(A) For a complex x, conj(x) = real(x) - j*imag(x)
- abs(A)
- angle(A)



imaginary

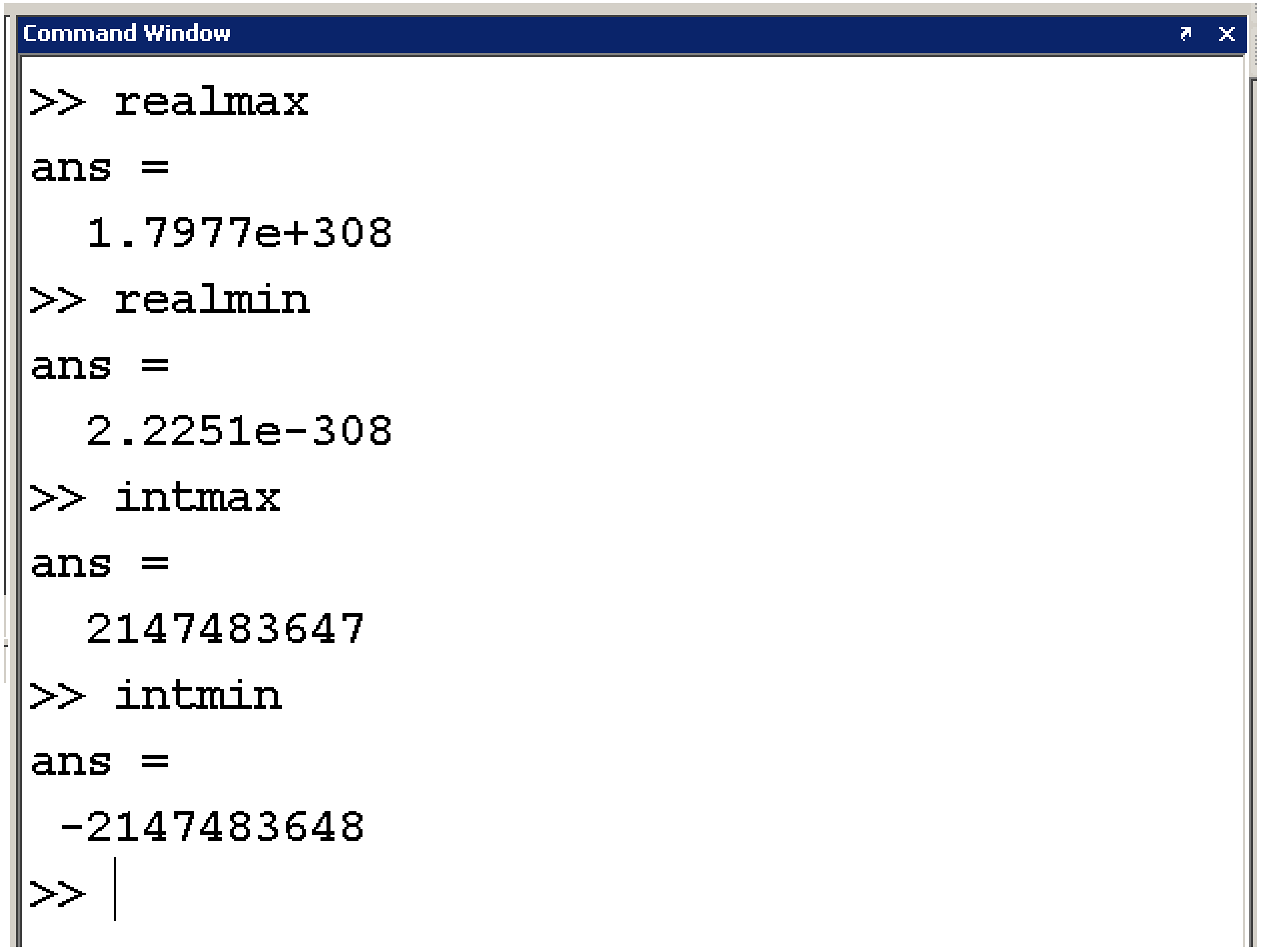
real

3.8 Computational Limits

- MATLAB’s computational range on most computers is:
  - 10^{-308}
  - 10^{308}
- When you divide by 0, the computer returns Inf

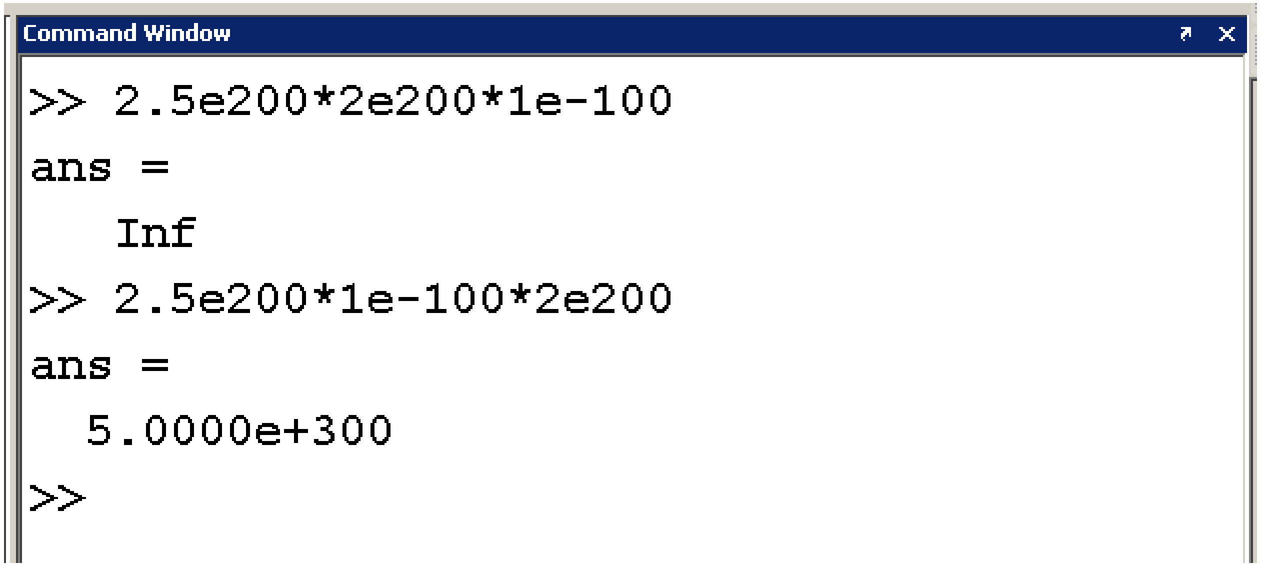
Check the limits on your computer with these commands

- realmax
- realmin
- intmax
- intmin



```
>> realmax
ans =
  1.7977e+308
>> realmin
ans =
  2.2251e-308
>> intmax
ans =
  2147483647
>> intmin
ans =
 -2147483648
>> 
```

When using very large or very small numbers the result may depend on the order of operation



```
>> 2.5e200*2e200*1e-100
ans =
   Inf
>> 2.5e200*1e-100*2e200
ans =
  5.0000e+300
>>
```

3.9 Special Values and Miscellaneous Functions

- pi
- i,j
- Inf
- NaN
- clock
- date
- ans

Hint: The function i is the most common of these functions to be unintentionally renamed by MATLAB users.

Summary

- MATLAB contains a wide array of predefined functions
  - Elementary Math Functions
  - Trigonometric Functions
  - Data Analysis Functions
  - Random Numbers
  - Complex Numbers

Summary

- The colon operator allows you to manipulate matrices
- Computational Limits
- Special Values and Functions

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 4
(Manipulating MATLAB Matrices)

Zeyad T. Shareef

Objectives

After studying this chapter, you should be able to:

- Manipulate matrices
- Extract data from matrices
- Solve problems with two variables
- Explore some of the special matrices built into MATLAB

Section 4.1
Manipulating Matrices

- We’ll start with a brief review
- To define a matrix, type in a list of numbers enclosed in square brackets

Remember that we can define a matrix using the following syntax

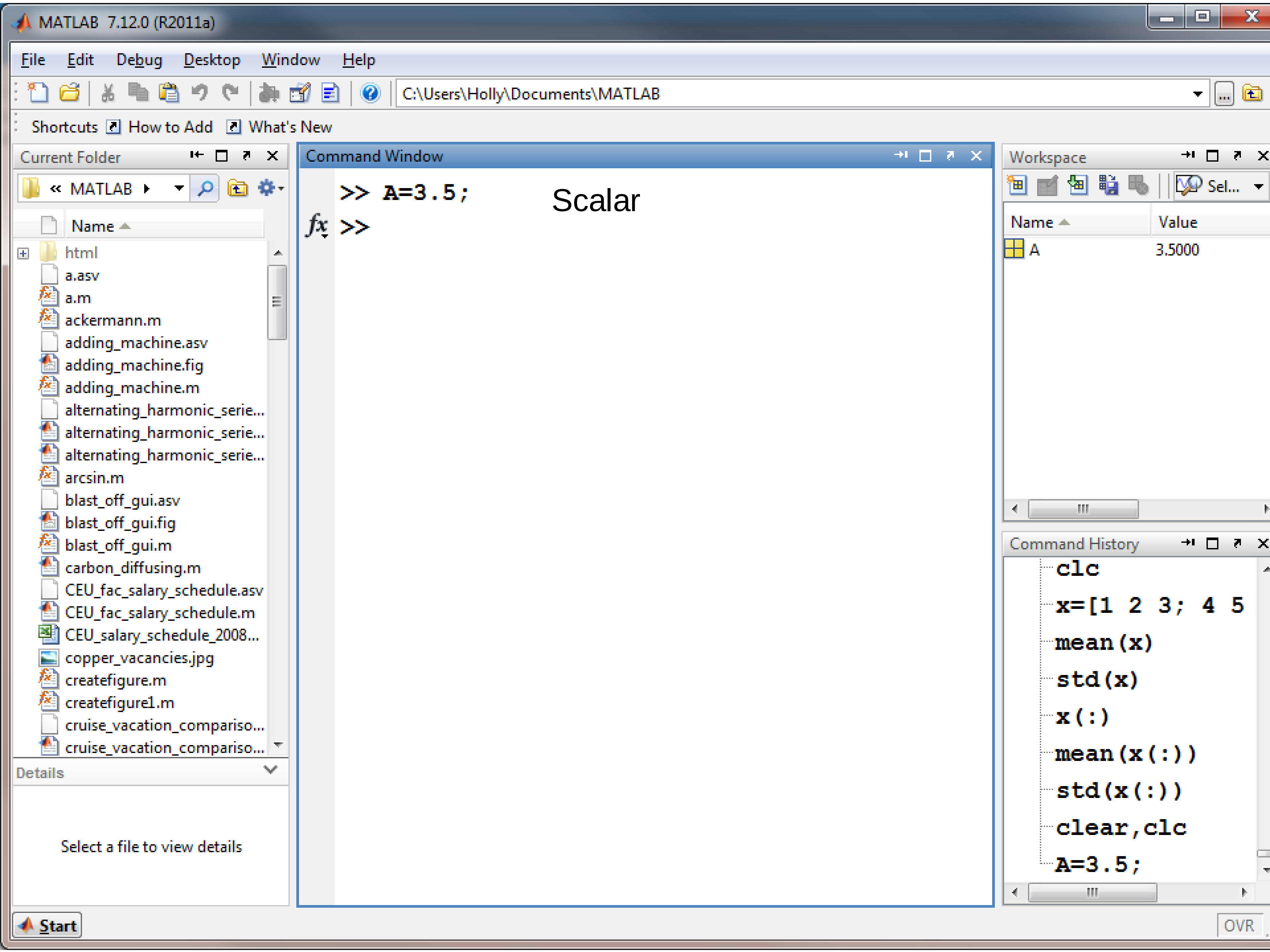
- A=[3.5]
- B=[1.5, 3.1]   or
- B=[1.5 3.1]
- C=[-1, 0, 0; 1, 1, 0; 0, 0, 2];

2-D Matrices can also be entered by listing each row on a separate line

```
C =  [-1,  0, 0
        1,  1, 0
        1, -1, 0
        0,   0, 2]
```

Use an ellipsis to continue a definition onto a new line

```
F = [1, 52, 64, 197, 42, -42, …
     55, 82, 22, 109];
```

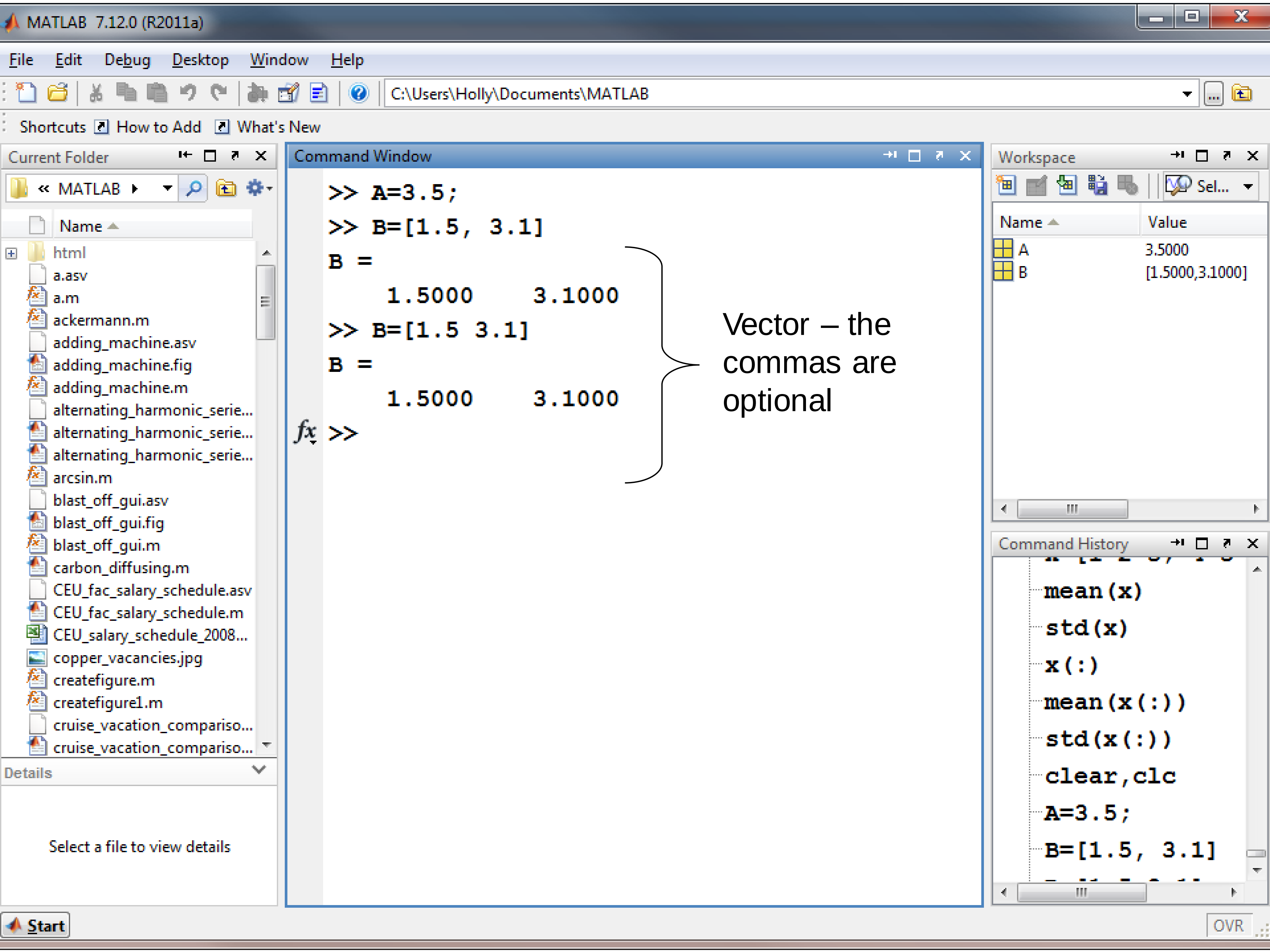

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> A=3.5;
>>
```

Scalar

Name	Value
A	3.5000

```
clc
x=[1 2 3; 4 5
mean(x)
std(x)
x(:)
mean(x(:))
std(x(:))
clear,clc
A=3.5;
```

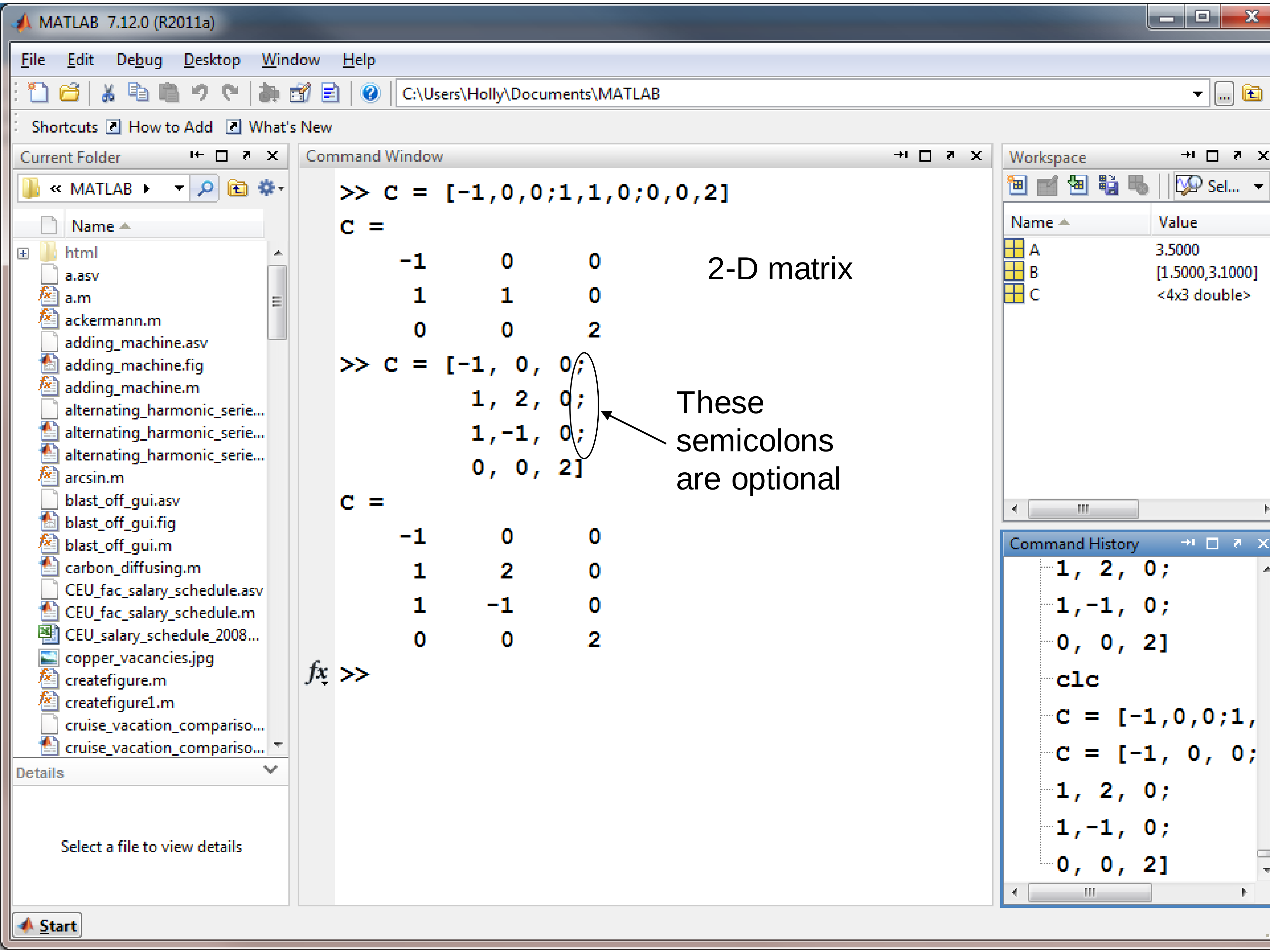

MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB ▸
Name ▲
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_serie...
alternating_harmonic_serie...
alternating_harmonic_serie...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_compariso...
cruise_vacation_compariso...
Details
Select a file to view details
Command Window
Workspace
Sel...
Command History
Start
OVR

```
>> A=3.5;
>> B=[1.5, 3.1]
B =
      1.5000    3.1000
>> B=[1.5 3.1]
B =
      1.5000    3.1000
fx >>
```

Vector – the commas are optional

Name ▲	Value
A	3.5000
B	[1.5000,3.1000]

```
mean(x)
std(x)
x(:)
mean(x(:))
std(x(:))
clear,clc
A=3.5;
B=[1.5, 3.1]
```



- html
- a.asv
- a.m
- ackermann.m
- adding_machine.asv
- adding_machine.fig
- adding_machine.m
- alternating_harmonic_serie...
- alternating_harmonic_serie...
- alternating_harmonic_serie...
- arcsin.m
- blast_off_gui.asv
- blast_off_gui.fig
- blast_off_gui.m
- carbon_diffusing.m
- CEU_fac_salary_schedule.asv
- CEU_fac_salary_schedule.m
- CEU_salary_schedule_2008...
- copper_vacancies.jpg
- createfigure.m
- createfigure1.m
- cruise_vacation_compariso...
- cruise_vacation_compariso...

```
>> C = [-1,0,0;1,1,0;0,0,2]
C =
    -1     0     0
     1     1     0
     0     0     2
>> C = [-1, 0, 0;
        1, 2, 0;
        1,-1, 0;
        0, 0, 2]
C =
    -1     0     0
     1     2     0
     1    -1     0
     0     0     2
>>
```

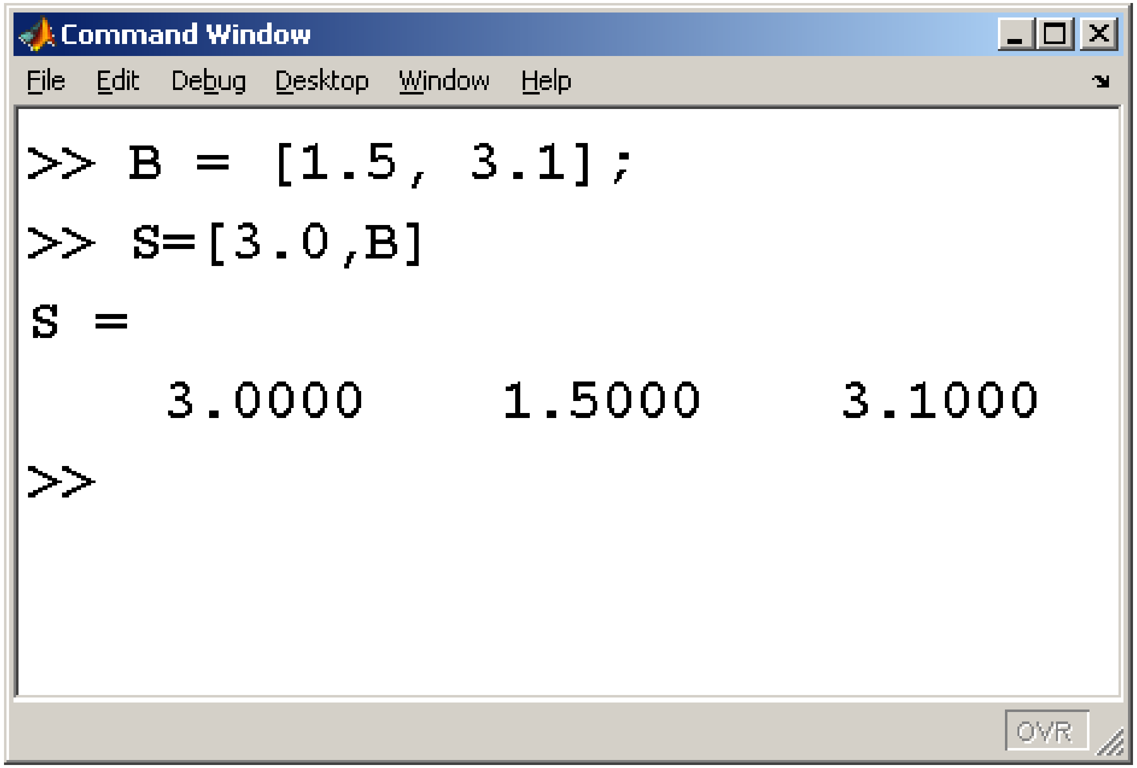
2-D matrix

These semicolons are optional

Name	Value
A	3.5000
B	[1.5000,3.1000]
C	<4x3 double>

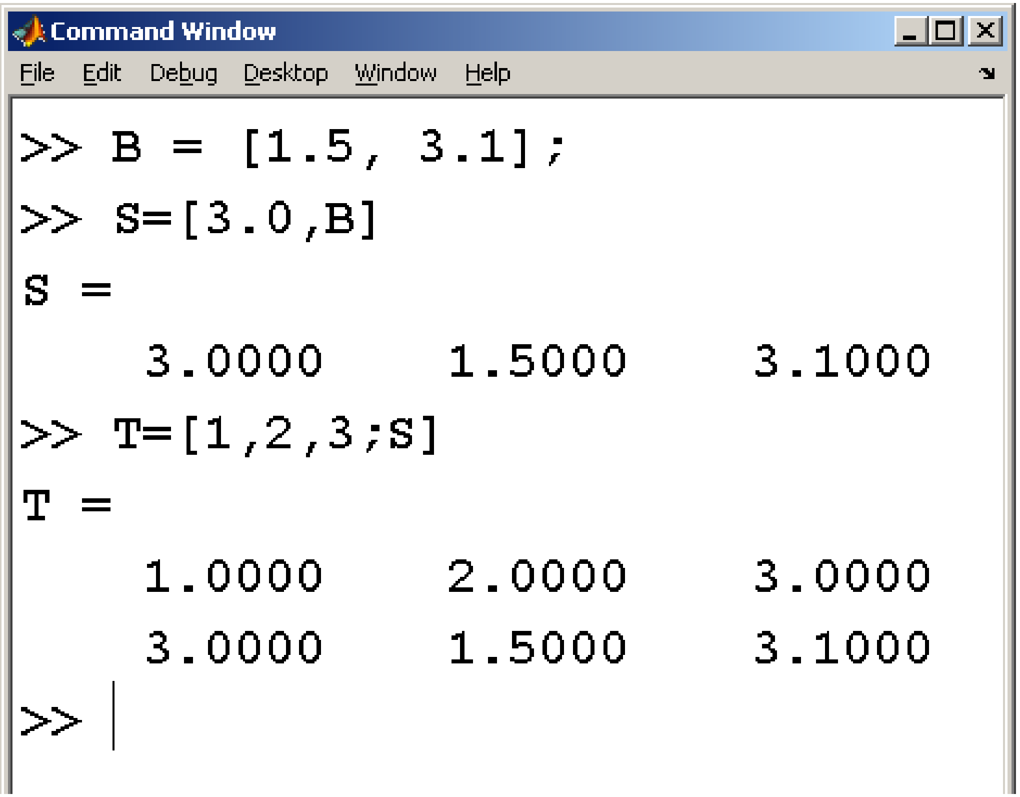
```
1, 2, 0;
1,-1, 0;
0, 0, 2]
clc
C = [-1,0,0;1,
C = [-1, 0, 0;
1, 2, 0;
1,-1, 0;
0, 0, 2]
```

You can define a matrix using other matrices as components



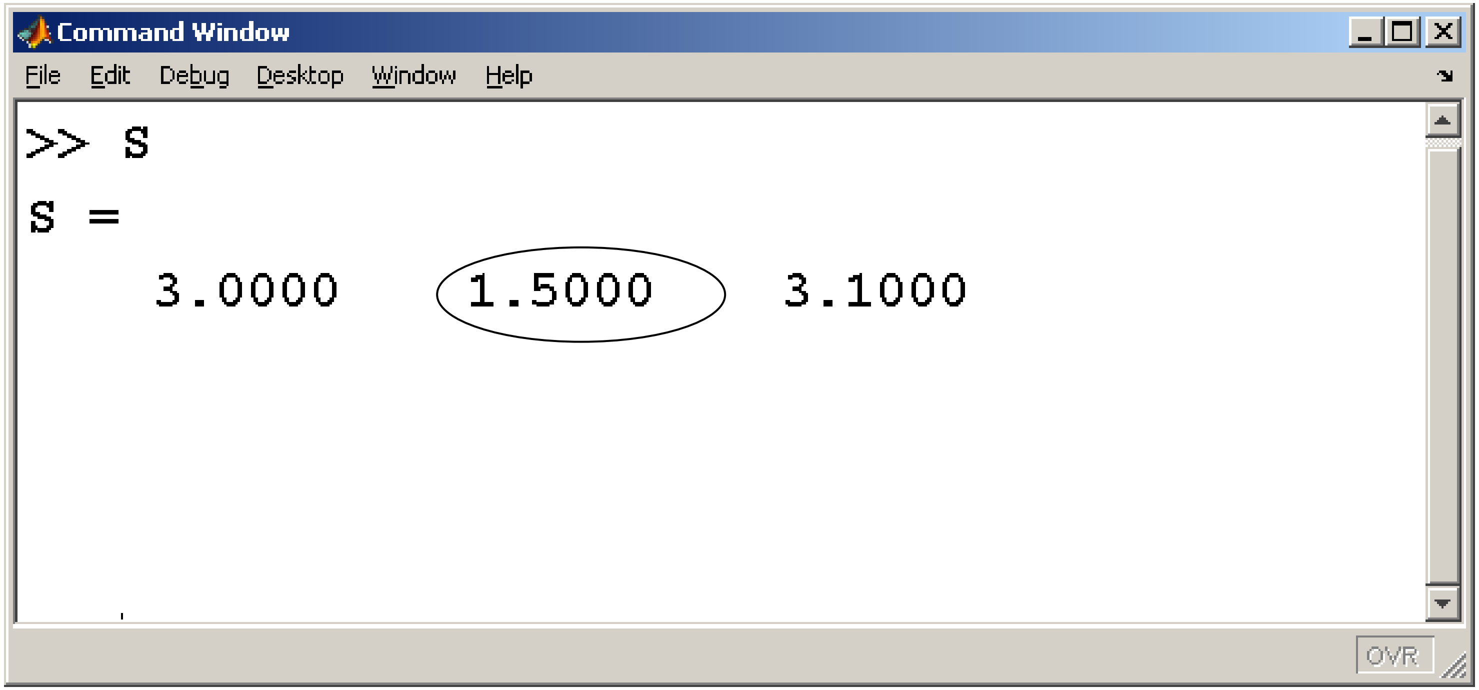
```
>> B = [1.5, 3.1];
>> S=[3.0,B]
S =
     3.0000     1.5000     3.1000
>>
```

Or…



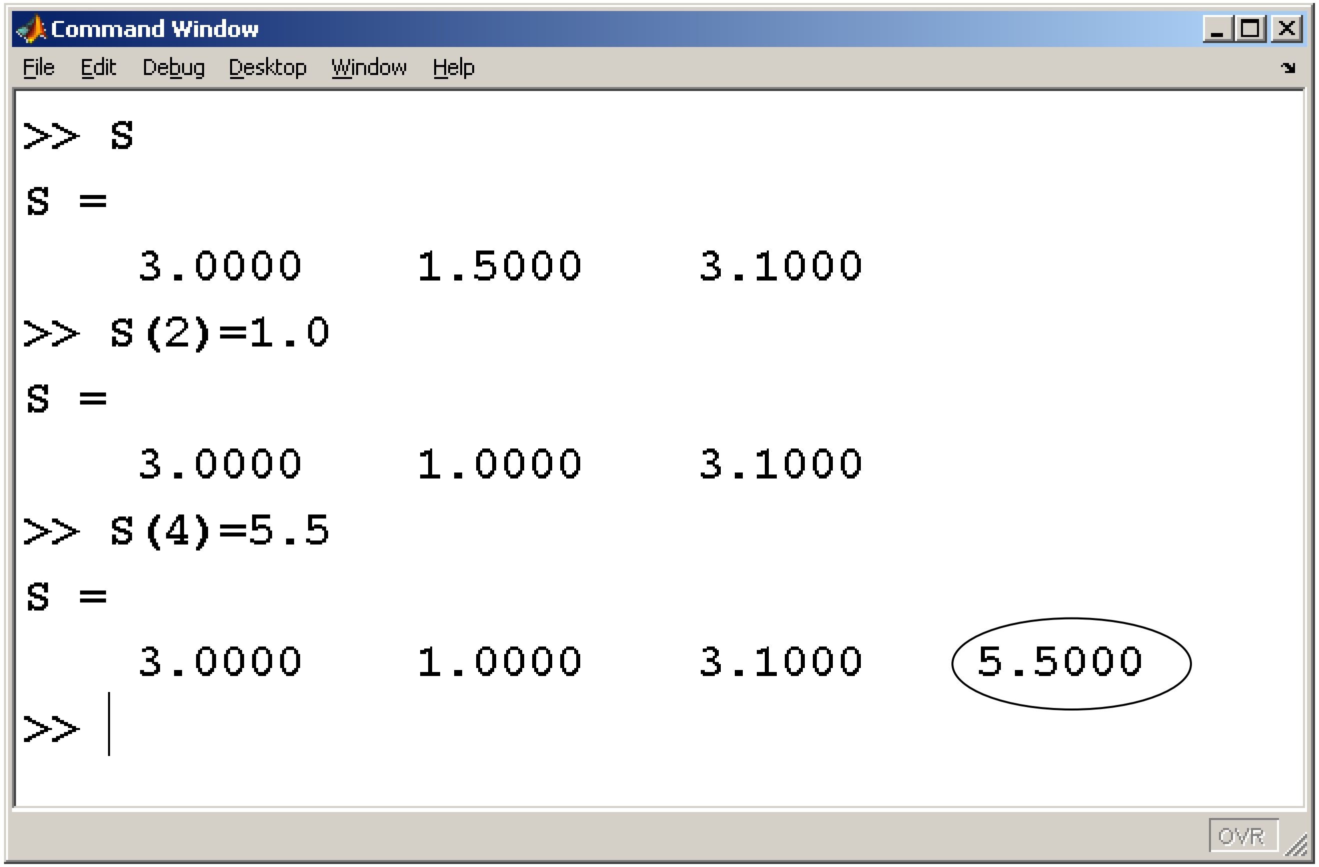
```
>> B = [1.5, 3.1];
>> S=[3.0,B]
S =
    3.0000    1.5000    3.1000
>> T=[1,2,3;S]
T =
    1.0000    2.0000    3.0000
    3.0000    1.5000    3.1000
>> 
```

Indexing Into an Array allows you to change a value

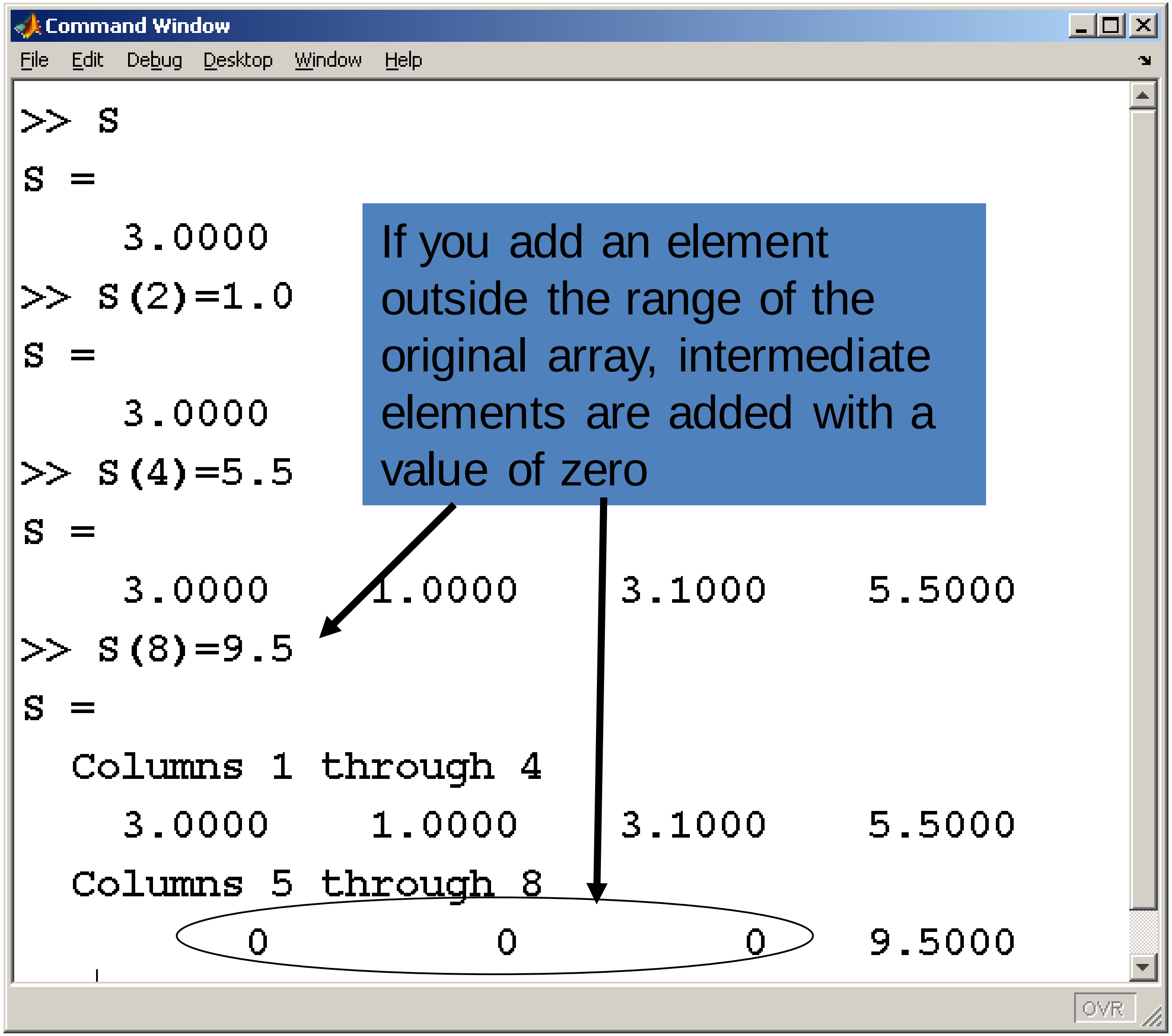


```
>> S

S =

     3.0000     1.5000     3.1000
```

Adding Elements



```
>> S

S =

    3.0000    1.5000    3.1000

>> S(2)=1.0

S =

    3.0000    1.0000    3.1000

>> S(4)=5.5

S =

    3.0000    1.0000    3.1000    5.5000

>> 
```



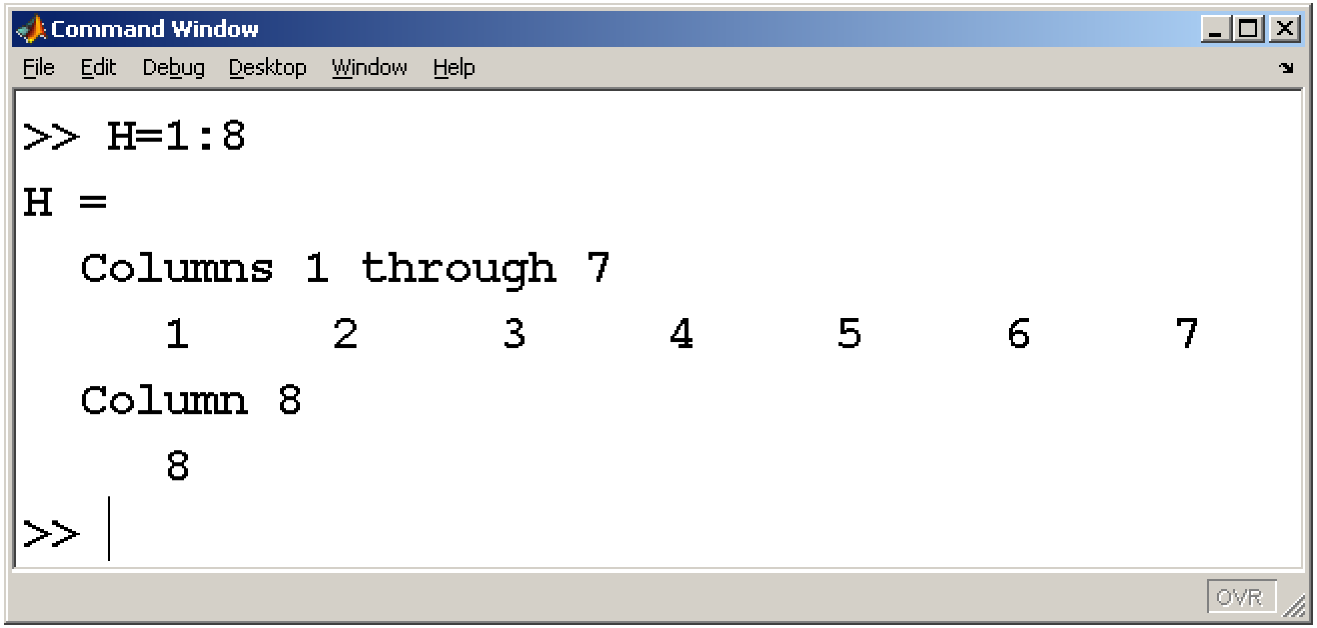
```
>> s
s =
      3.0000
>> s(2)=1.0
s =
      3.0000
>> s(4)=5.5
s =
      3.0000      1.0000      3.1000      5.5000
>> s(8)=9.5
s =
   Columns 1 through 4
      3.0000      1.0000      3.1000      5.5000
   Columns 5 through 8
           0           0           0      9.5000
```

If you add an element outside the range of the original array, intermediate elements are added with a value of zero

Colon Operator

- Used to define new matrices
- Modify existing matrices
- Extract data from existing matrices

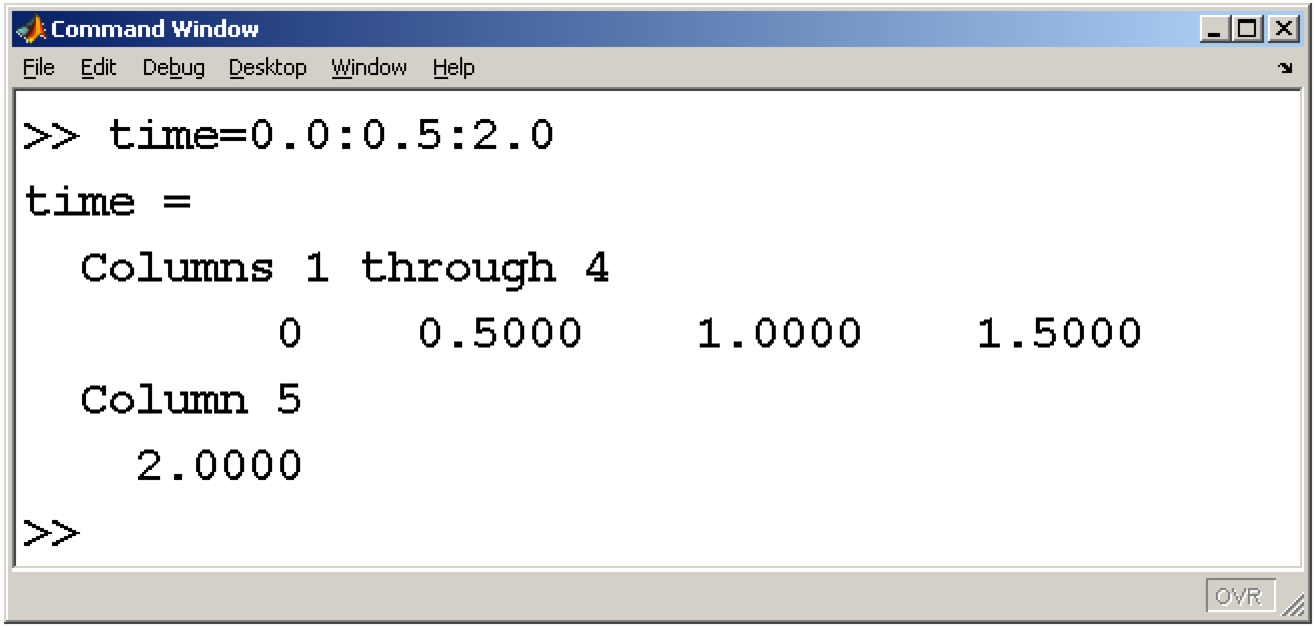
Evenly spaced vector



```
>> H=1:8
H =
  Columns 1 through 7
     1     2     3     4     5     6     7
  Column 8
     8
>> 
```

The default spacing is 1

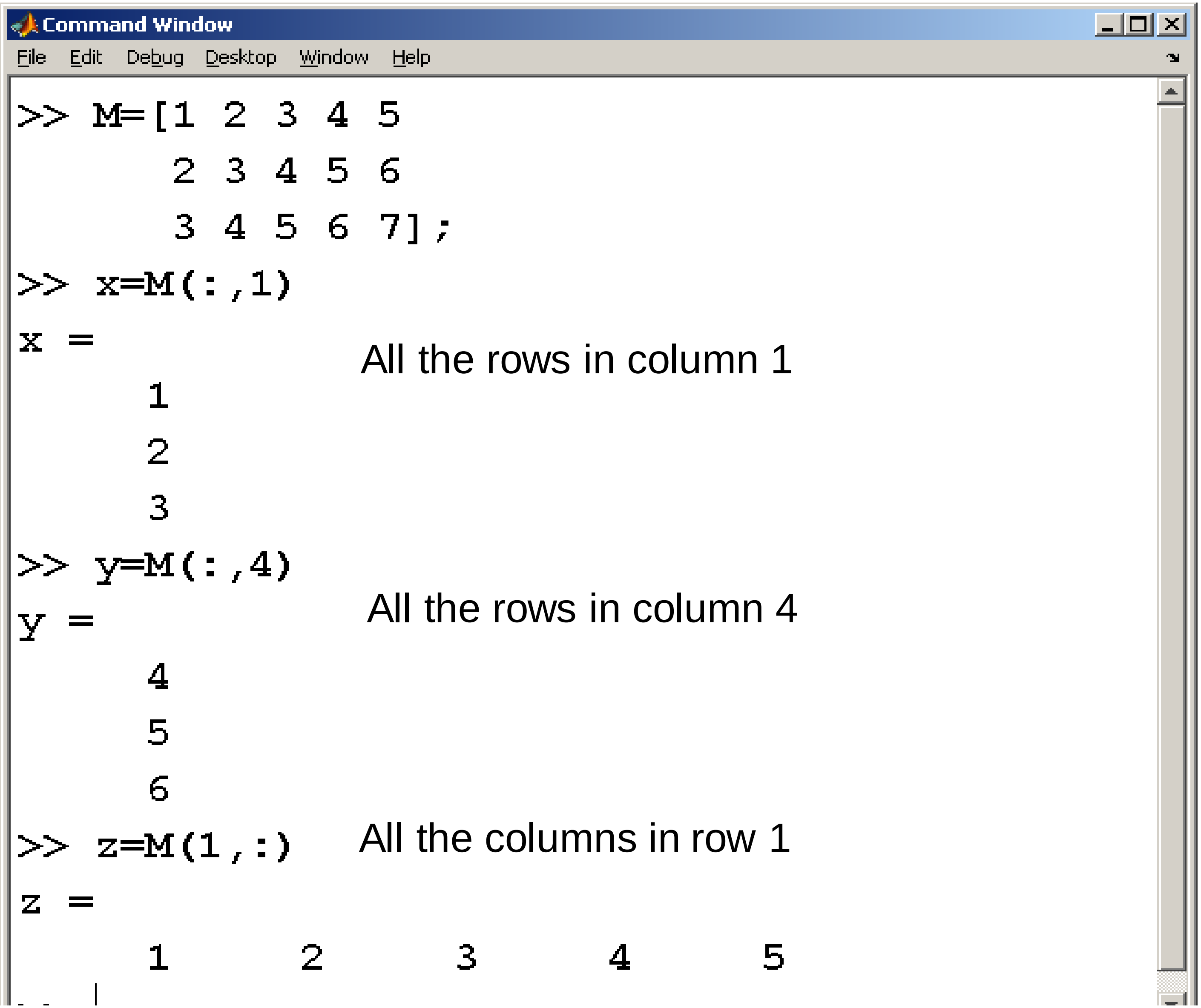
User specified spacing



```
>> time=0.0:0.5:2.0
time =
  Columns 1 through 4
         0    0.5000    1.0000    1.5000
  Column 5
    2.0000
>>
```

The spacing is specified as 0.5

The col



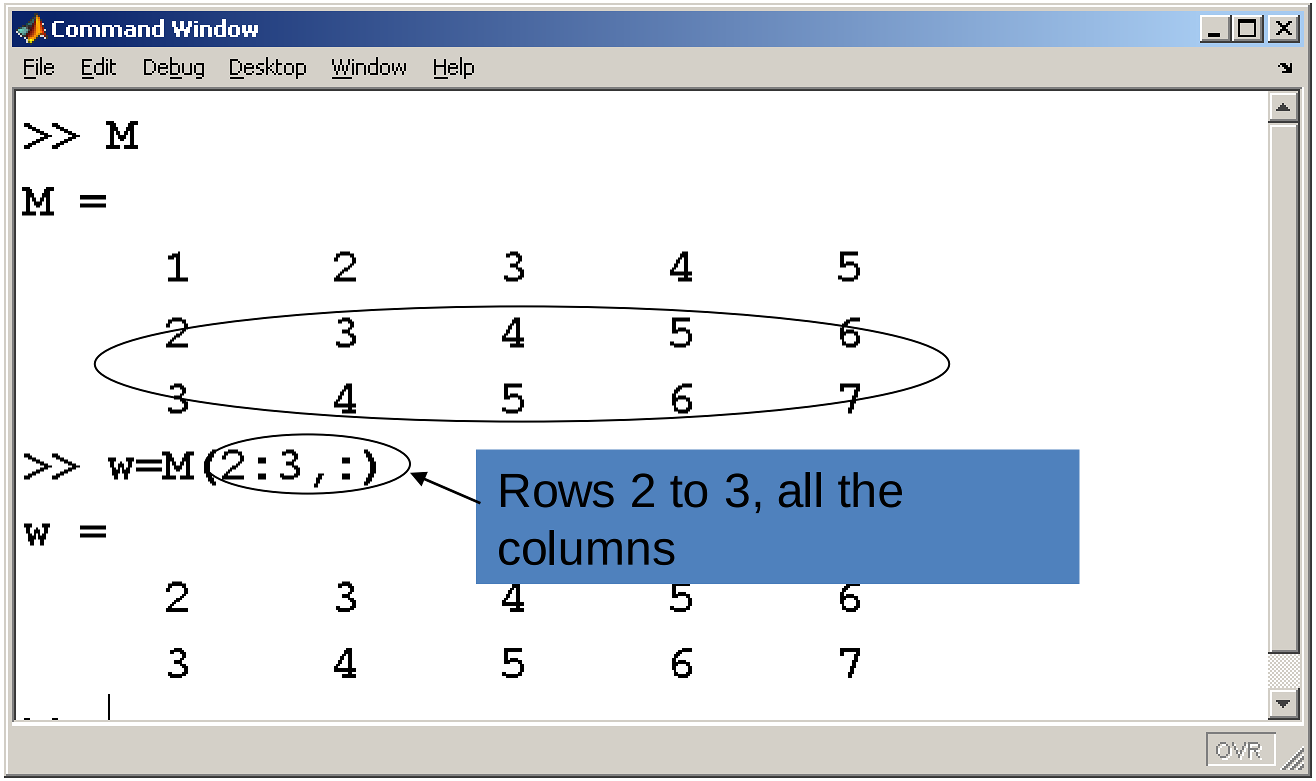
```
>> M=[1 2 3 4 5
      2 3 4 5 6
      3 4 5 6 7];
>> x=M(:,1)

x =

     1
     2
     3

>> y=M(:,4)

y =

     4
     5
     6

>> z=M(1,:)

z =

     1     2     3     4     5
```

All the rows in column 1

All the rows in column 4

All the columns in row 1

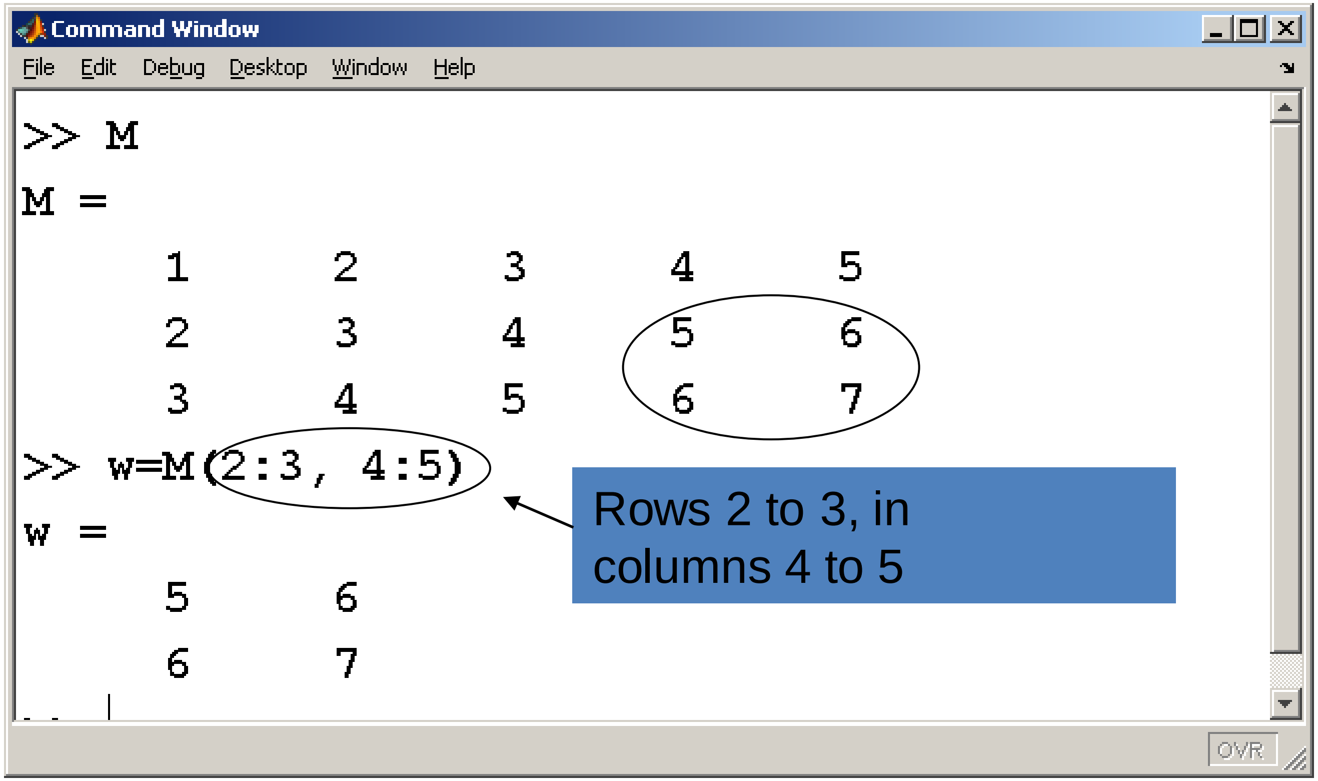
You don’t need to extract an entire row or column



```
>> M

M =

     1     2     3     4     5
     2     3     4     5     6
     3     4     5     6     7

>> w=M(2:3,:)

w =

     2     3     4     5     6
     3     4     5     6     7
```

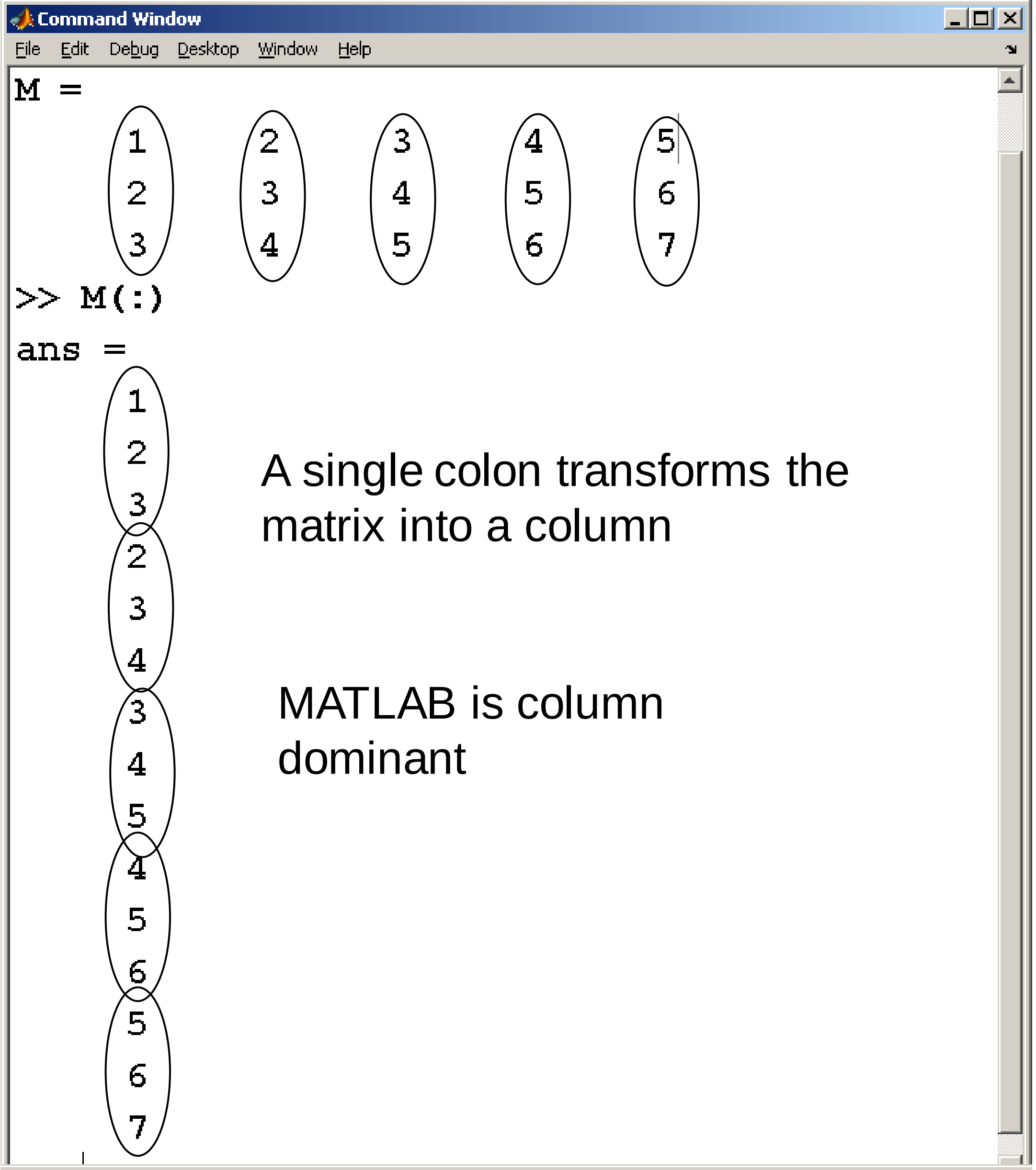
Rows 2 to 3, all the columns

Or…



```
>> M

M =

     1     2     3     4     5
     2     3     4     5     6
     3     4     5     6     7

>> w=M(2:3, 4:5)

w =

     5     6
     6     7
```

Rows 2 to 3, in columns 4 to 5



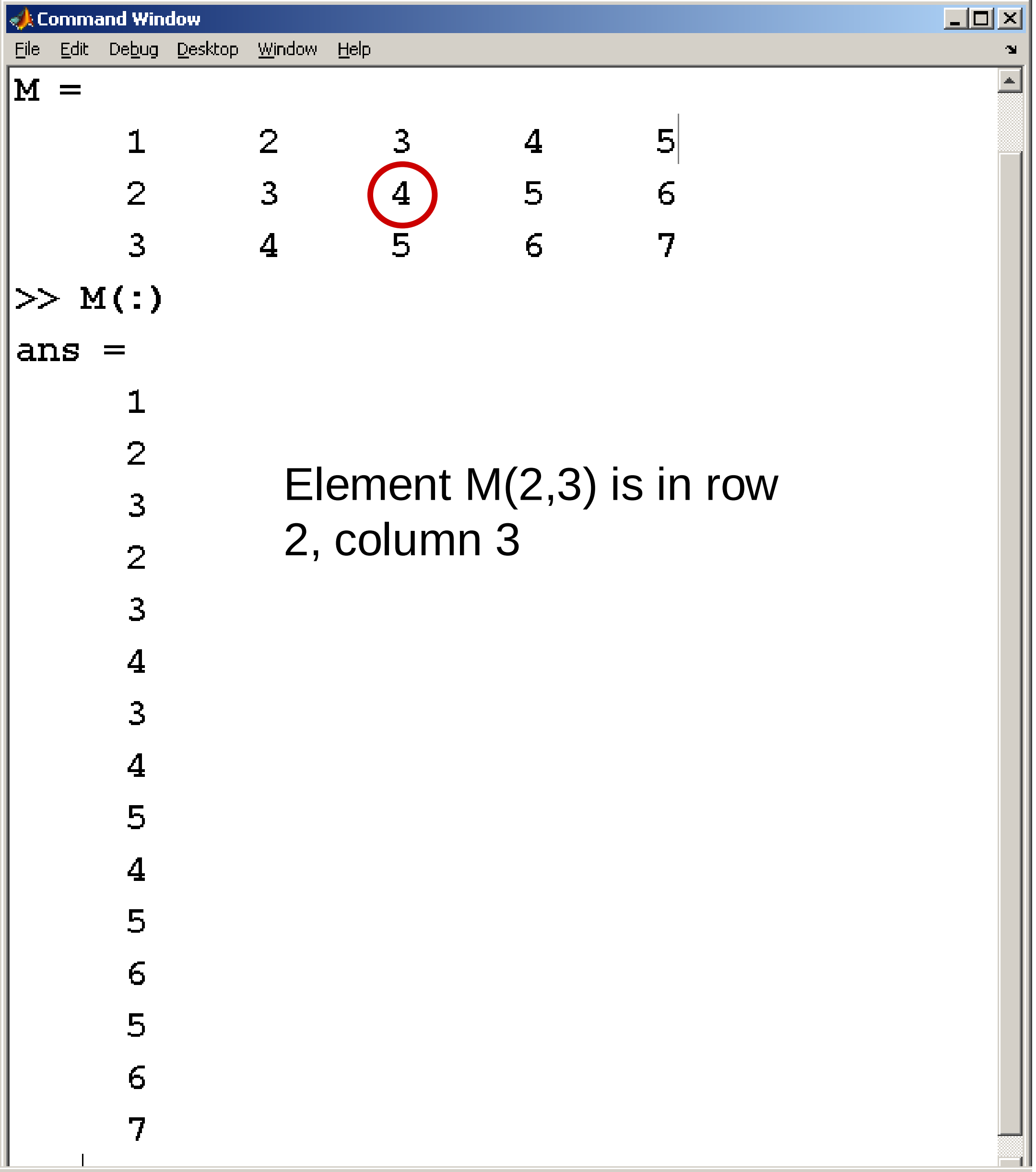
```
M =
        1     2     3     4     5
        2     3     4     5     6
        3     4     5     6     7
>> M(:)
ans =
        1
        2
        3
        2
        3
        4
        3
        4
        5
        4
        5
        6
        5
        6
        7
```

A single colon transforms the matrix into a column

MATLAB is column dominant

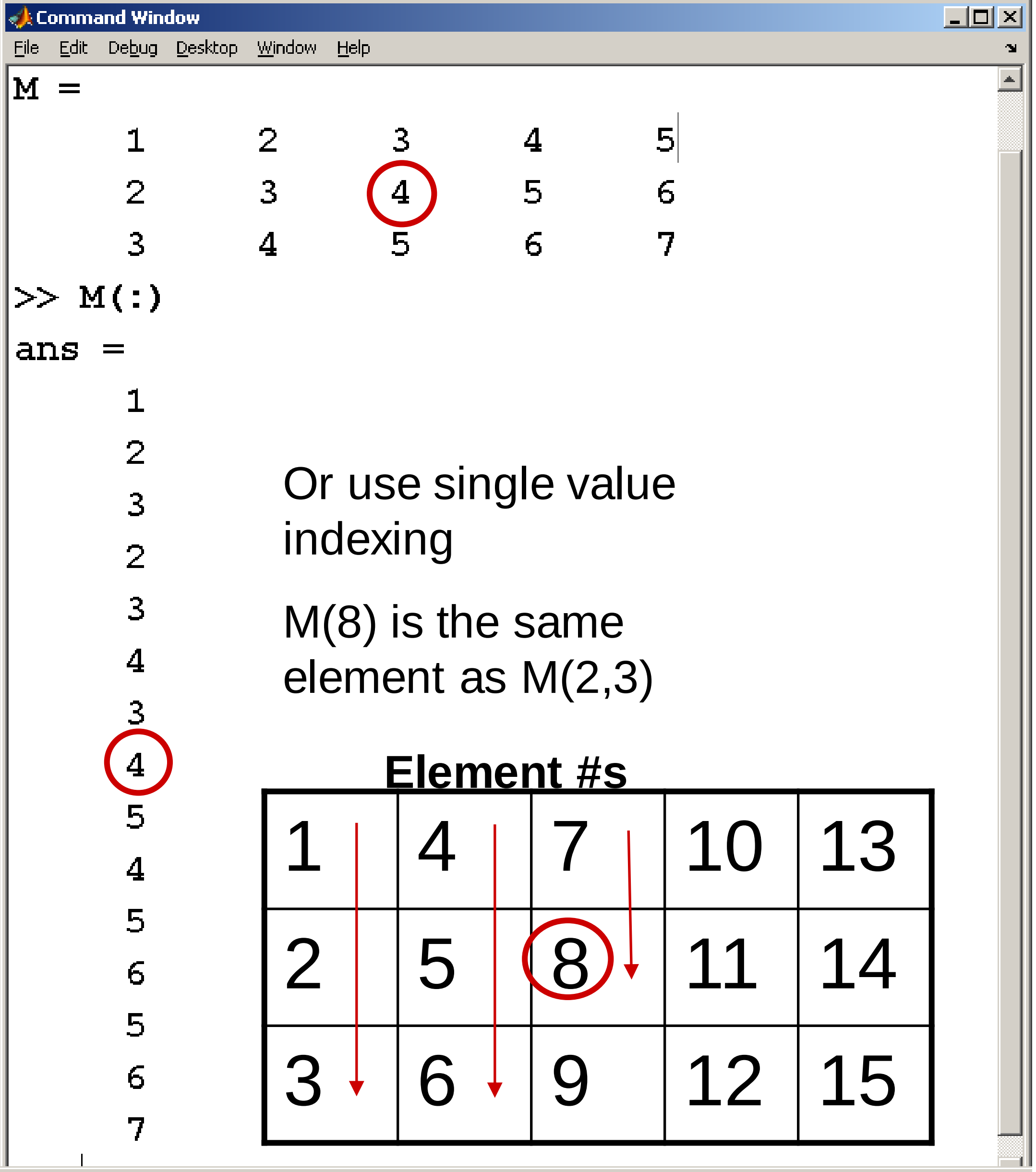
Indexing techniques

- To identify an element in a 2-D matrix, use the row and column number
- For example, element M(2,3)



```
M =

     1     2     3     4     5
     2     3     4     5     6
     3     4     5     6     7

>> M(:)

ans =

     1
     2
     3
     2
     3
     4
     3
     4
     5
     4
     5
     6
     5
     6
     7
```

Element M(2,3) is in row 2, column 3



```
M =
        1         2         3         4         5
        2         3         4         5         6
        3         4         5         6         7
>> M(:)
ans =
        1
        2
        3
        2
        3
        4
        3
        4
        5
        4
        5
        6
        5
        6
        7
```

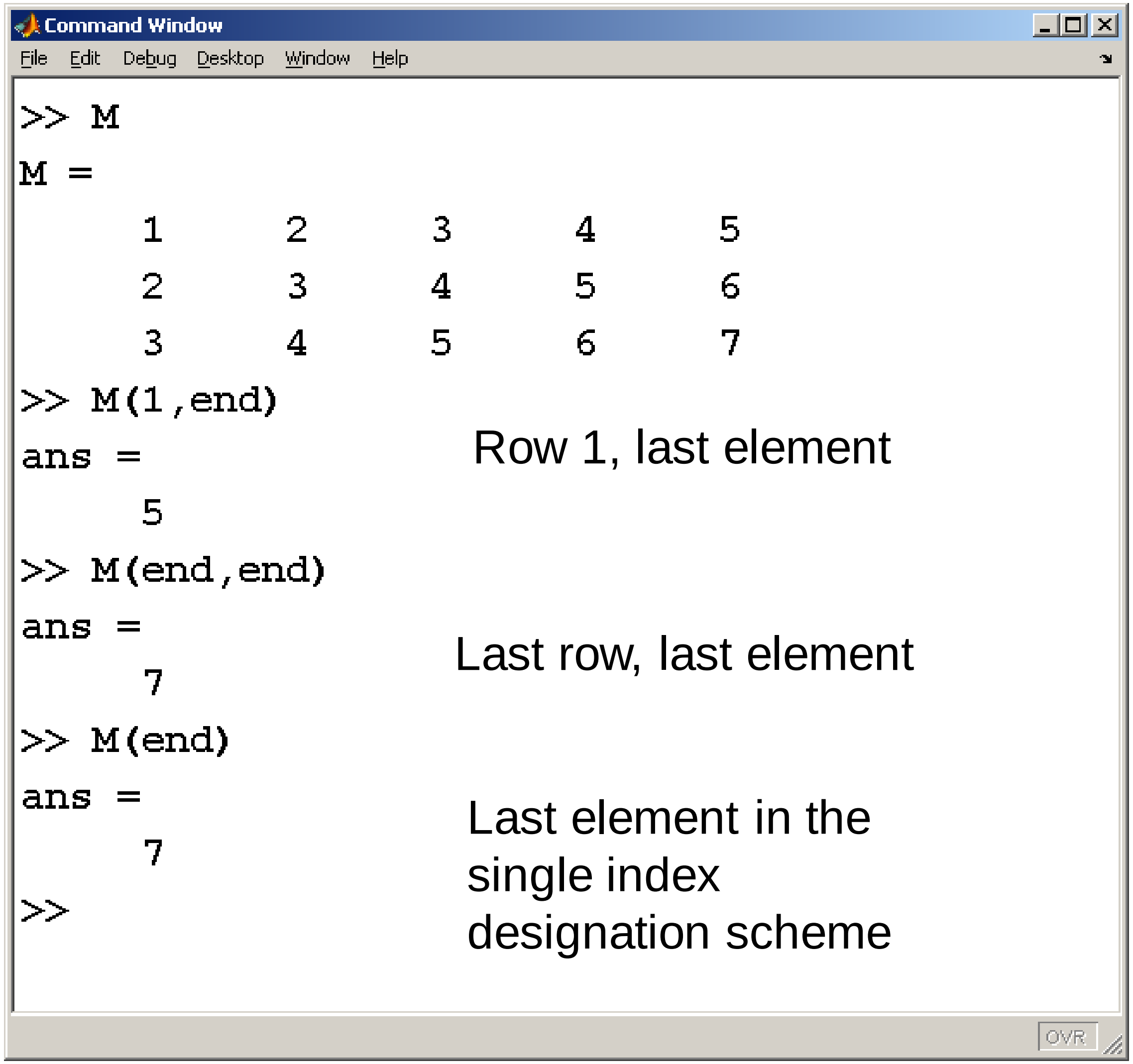
Or use single value indexing

M(8) is the same element as M(2,3)

Element #s

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

The word "end" signifies the last element in the row or column



```
>> M
M =
     1     2     3     4     5
     2     3     4     5     6
     3     4     5     6     7
>> M(1,end)
ans =
     5
>> M(end,end)
ans =
     7
>> M(end)
ans =
     7
>>
```

Row 1, last element

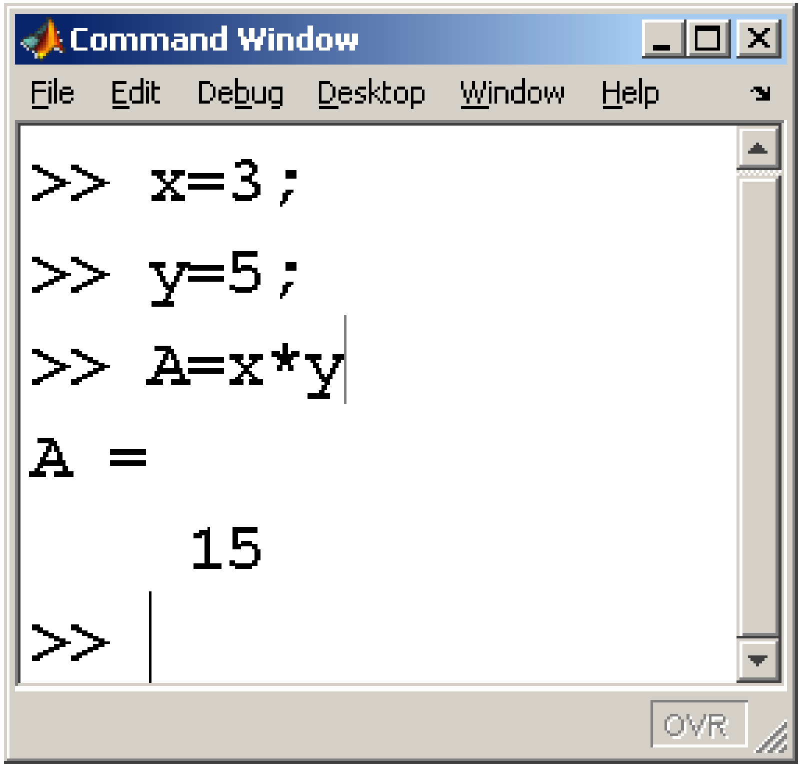
Last row, last element

Last element in the single index designation scheme

Section 4.2
Problems with Two Variables

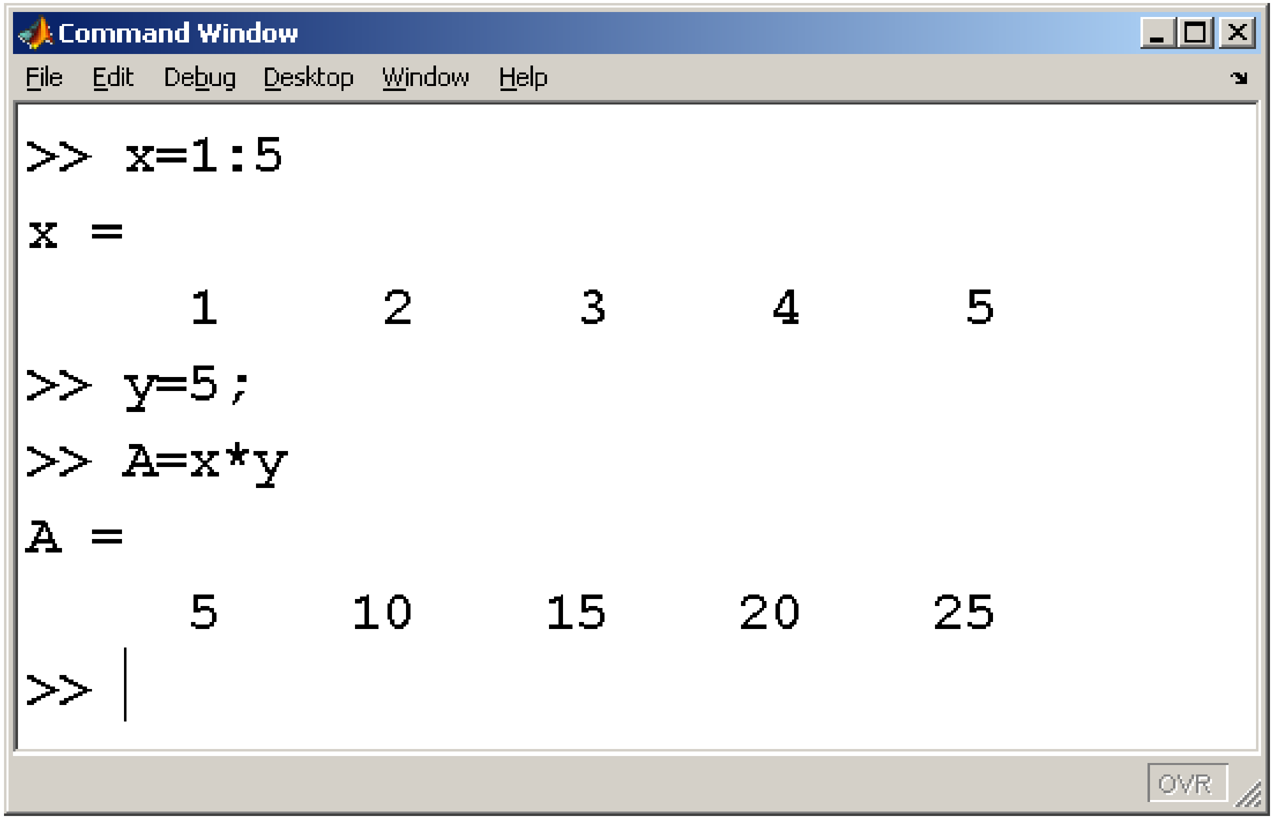
- All of our calculations thus far have only included one variable
- Most physical phenomena can vary with many different factors
- We need a strategy for determining the array of answers that results with a range of values for multiple variables

Two scalars give a scalar result



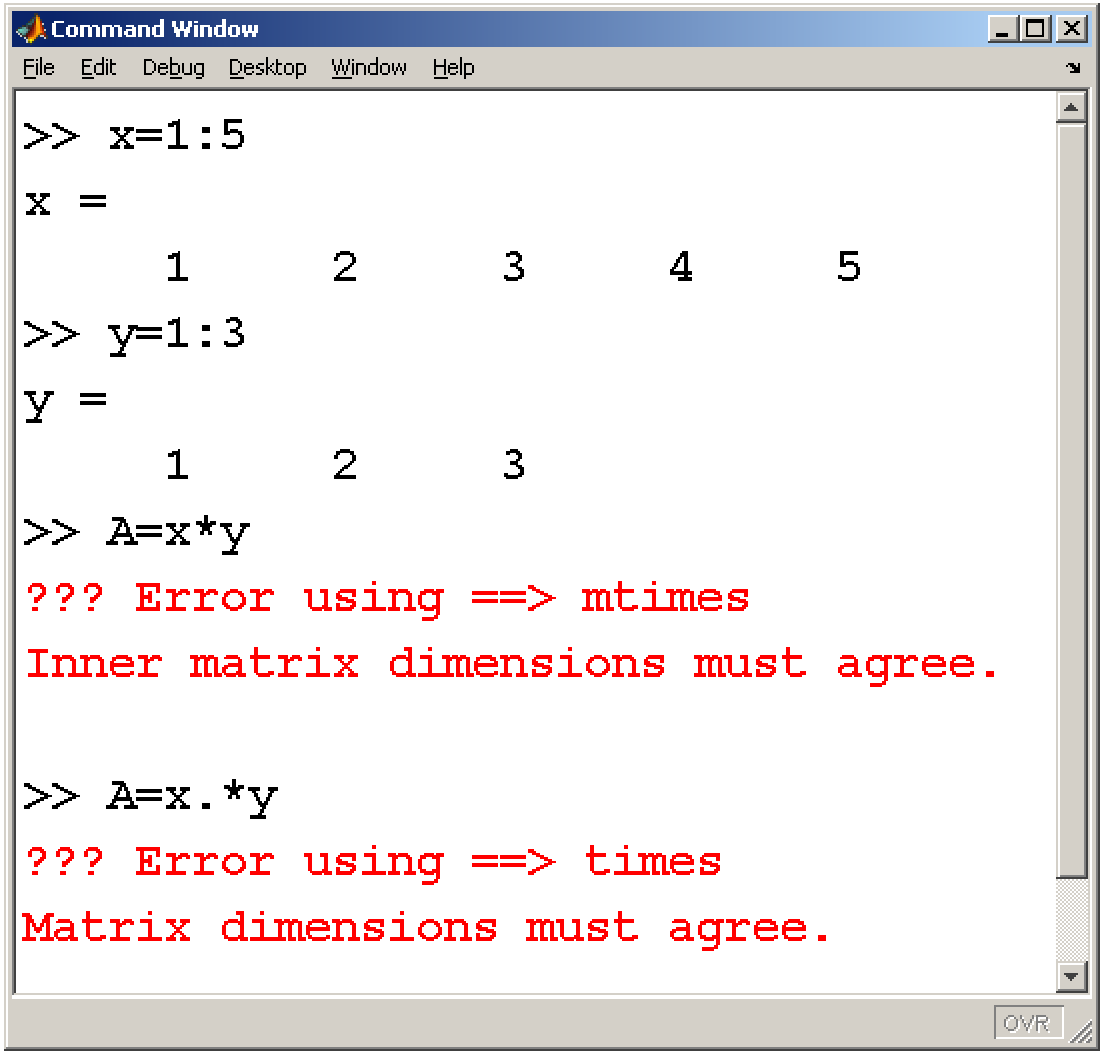
```
>> x=3;
>> y=5;
>> A=x*y

A =

    15

>>
```

A scalar and a vector give a vector result



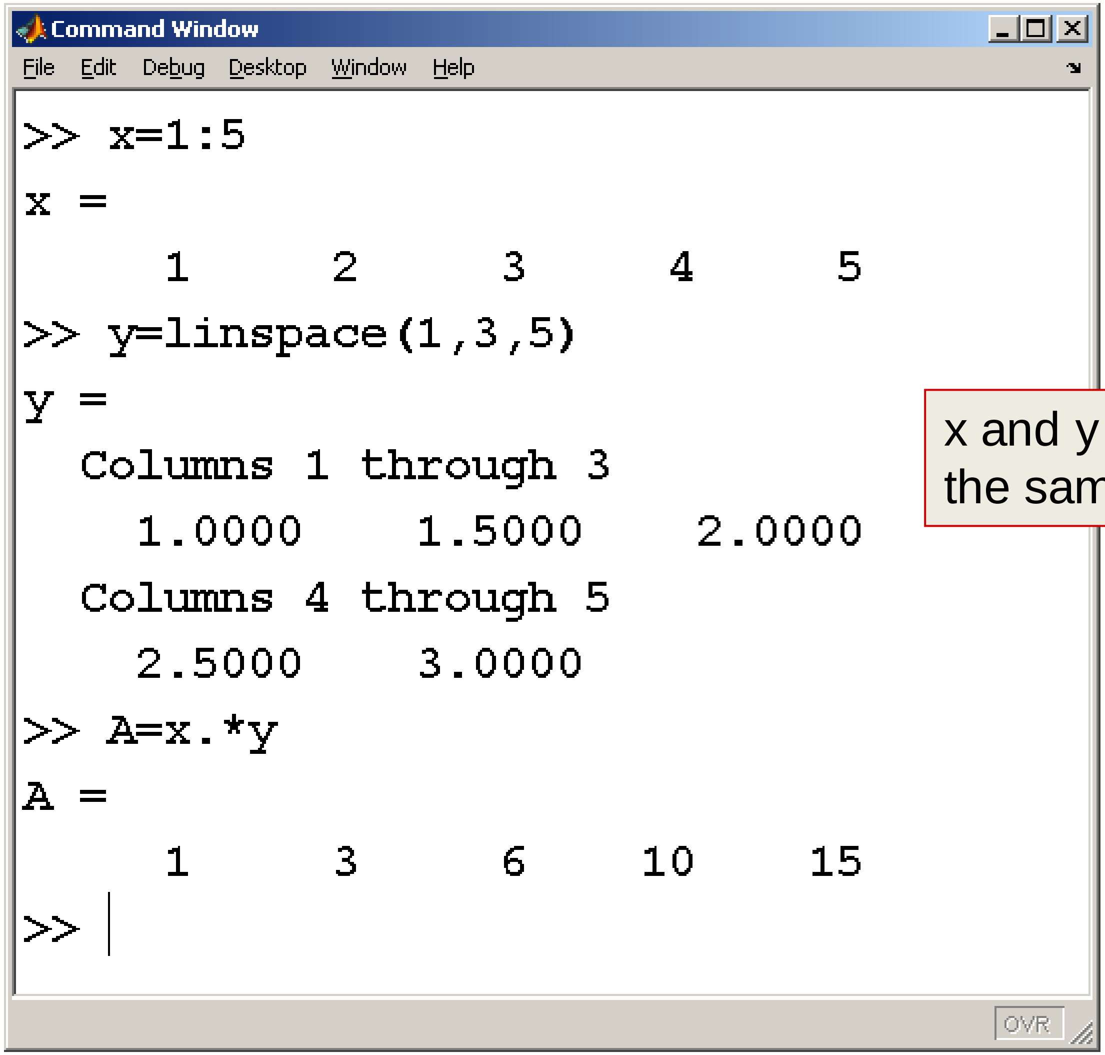
```
>> x=1:5

x =

     1     2     3     4     5

>> y=5;
>> A=x*y

A =

     5    10    15    20    25

>> 
```

When you multiply two vectors together, they must have the same number of elements



```
>> x=1:5
x =
     1     2     3     4     5
>> y=1:3
y =
     1     2     3
>> A=x*y
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> A=x.*y
??? Error using ==> times
Matrix dimensions must agree.
```

Array multiplication gives a result the same size as the input arrays



```
>> x=1:5
x =
     1     2     3     4     5
>> y=linspace(1,3,5)
y =
  Columns 1 through 3
    1.0000    1.5000    2.0000
  Columns 4 through 5
    2.5000    3.0000
>> A=x.*y
A =
     1     3     6    10    15
>> 
```

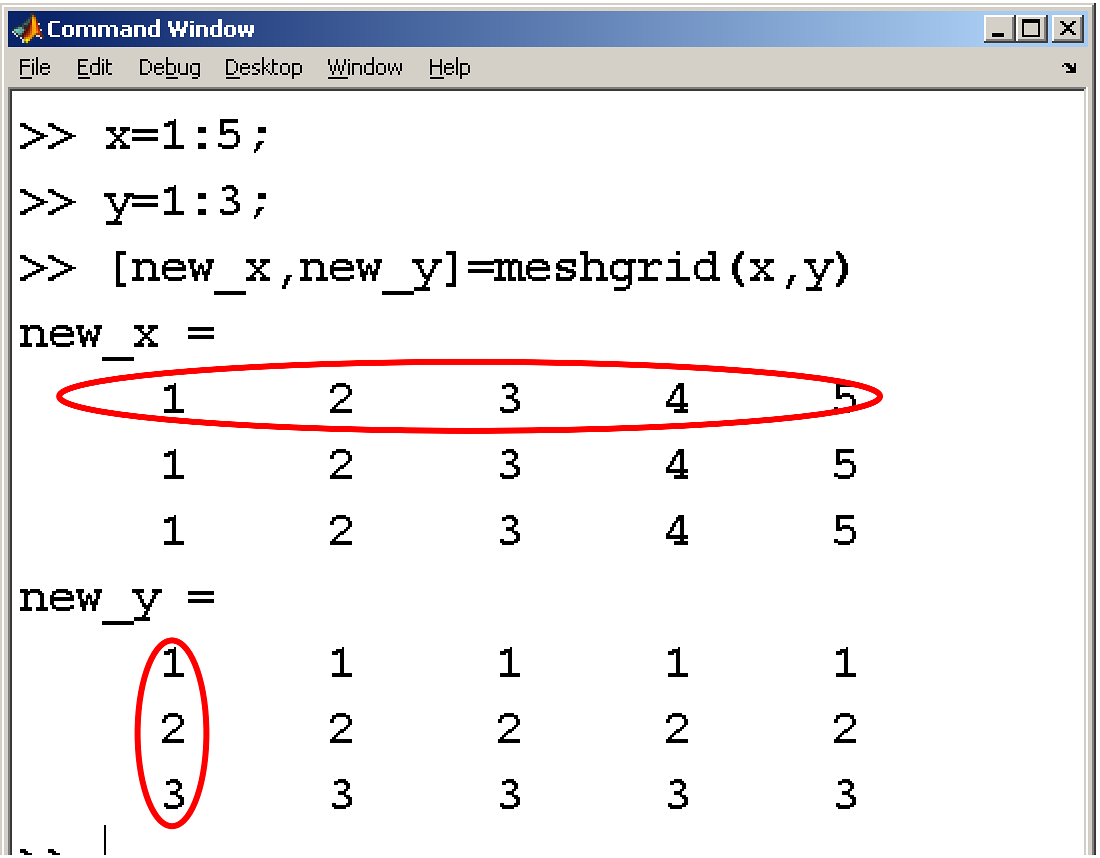
x and y must be the same size

Results of an element by element (array) multiplication

x

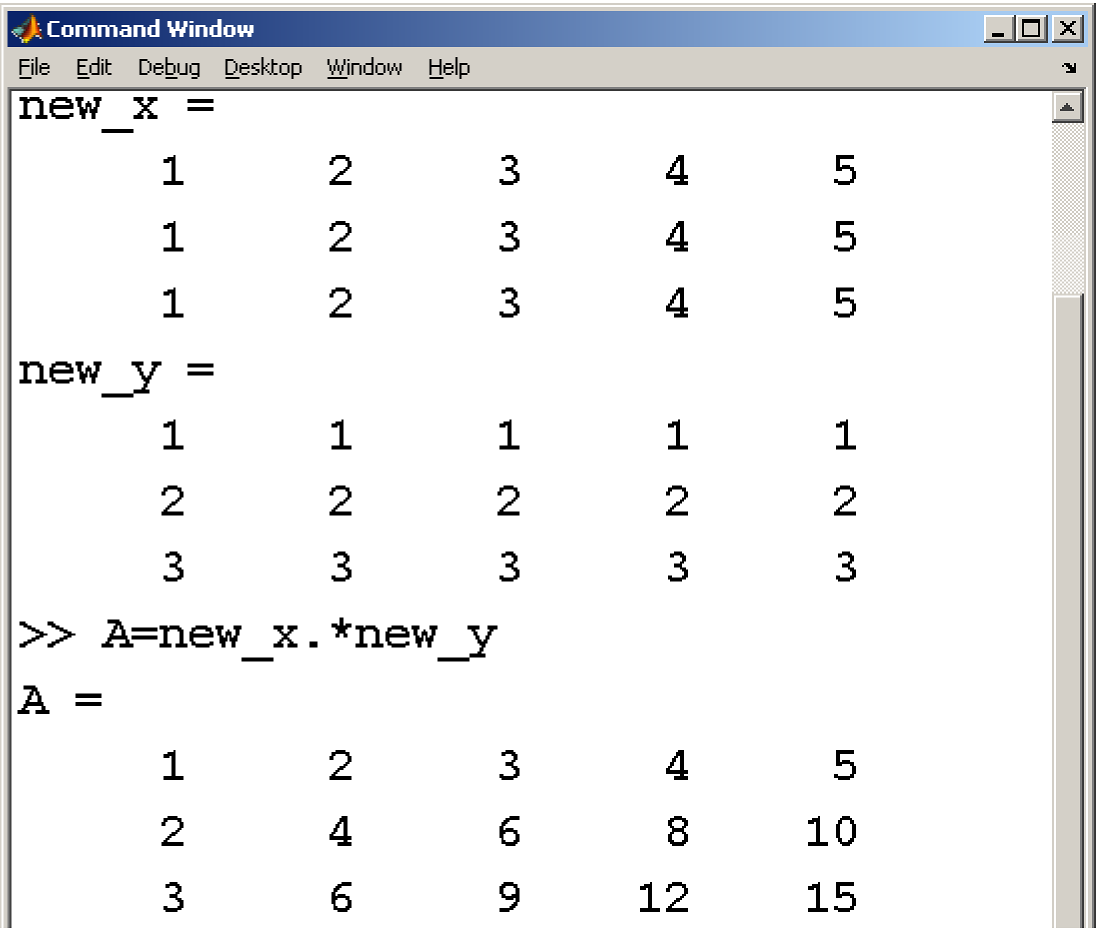
		1	2	3	4	5
y	1.0	1				
	1.5		3			
	2.0			6		
	2.5				10	
	3.0			?		15

The meshgrid function maps two vectors onto a 2-D grid



```
>> x=1:5;
>> y=1:3;
>> [new_x,new_y]=meshgrid(x,y)
new_x =
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
new_y =
     1     1     1     1     1
     2     2     2     2     2
     3     3     3     3     3
>>
```

Now the arrays are the same size, and can be multiplied

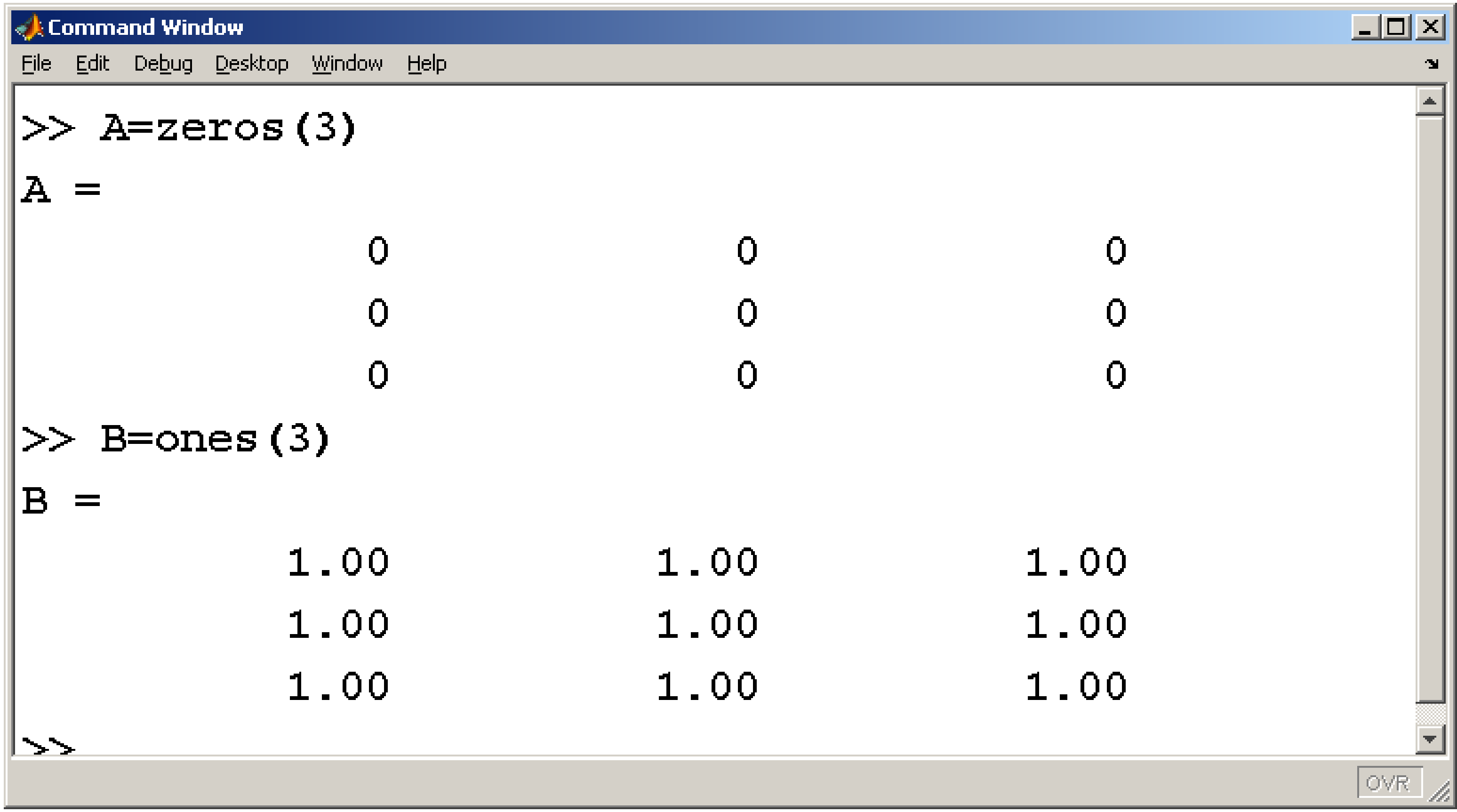


```
new_x =

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

new_y =

     1     1     1     1     1
     2     2     2     2     2
     3     3     3     3     3

>> A=new_x.*new_y

A =

     1     2     3     4     5
     2     4     6     8    10
     3     6     9    12    15
```

Section 4.3
Special Matrices

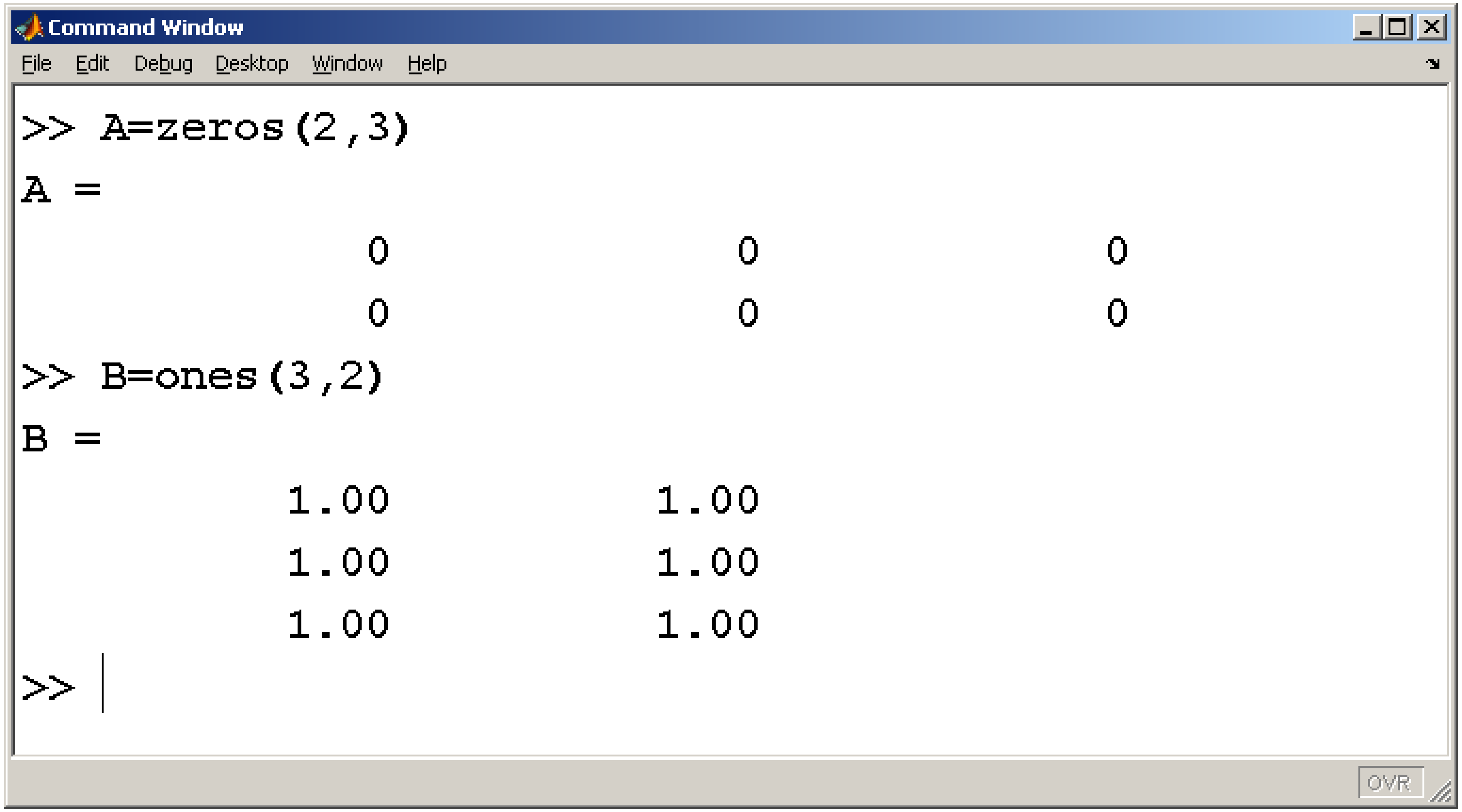
- zeros
  - Creates a matrix of all zeros
- ones
  - Creates a matrix of all ones
- diag
  - Extracts a diagonal or creates an identity matrix
- magic
  - Creates a “magic” matrix

With a single input a square matrix is created with the zeros or ones function



```
>> A=zeros(3)
A =
             0             0             0
             0             0             0
             0             0             0
>> B=ones(3)
B =
          1.00          1.00          1.00
          1.00          1.00          1.00
          1.00          1.00          1.00
>>
```

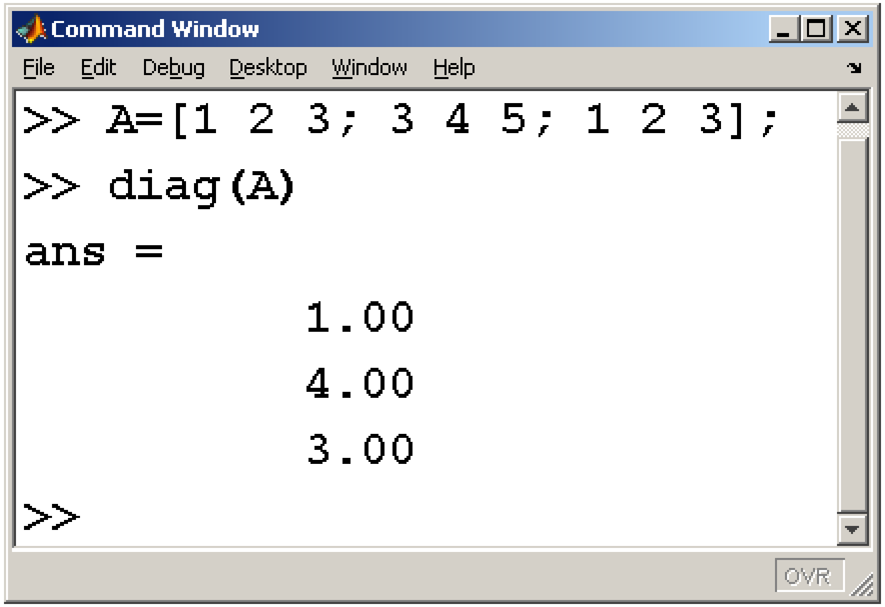
Two input arguments specify the number of rows and columns



```
>> A=zeros(2,3)

A =

             0             0             0
             0             0             0

>> B=ones(3,2)

B =

          1.00          1.00
          1.00          1.00
          1.00          1.00

>> 
```

The diag function

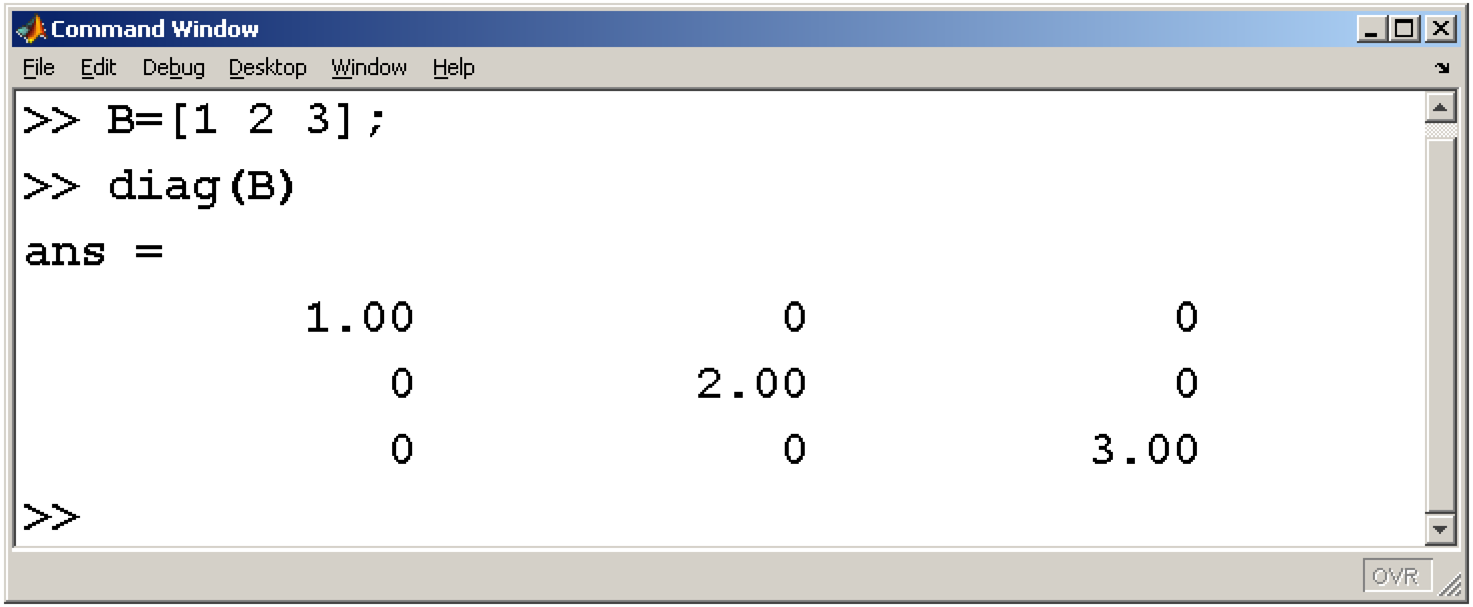
When the input argument to the diag function is a square matrix, the diagonal is returned



```
>> A=[1 2 3; 3 4 5; 1 2 3];
>> diag(A)
ans =
          1.00
          4.00
          3.00
>>
```

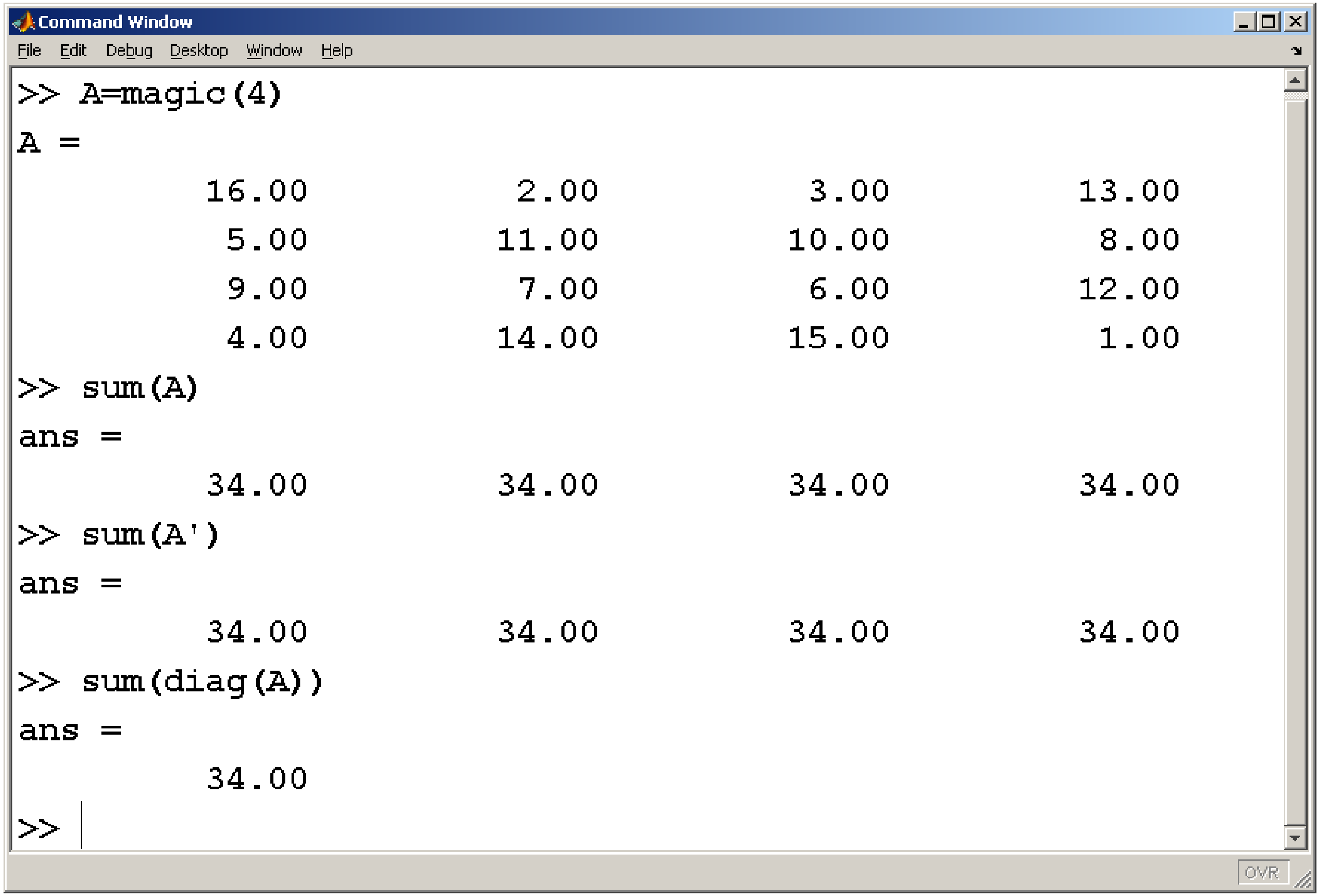
The diag function

When the input is a vector, it is used as the diagonal of an identity matrix



```
>> B=[1 2 3];
>> diag(B)
ans =
         1.00              0              0
            0           2.00              0
            0              0           3.00
>>
```

Magic Matrices



```
>> A=magic(4)
A =
          16.00          2.00          3.00         13.00
           5.00         11.00         10.00          8.00
           9.00          7.00          6.00         12.00
           4.00         14.00         15.00          1.00
>> sum(A)
ans =
          34.00         34.00         34.00         34.00
>> sum(A')
ans =
          34.00         34.00         34.00         34.00
>> sum(diag(A))
ans =
          34.00
>> 
```

Summary

- Matrices can be created by combining other matrices
- Portions of existing matrices can be extracted to form smaller matrices

Summary – The colon operator

- The colon operator
  - can be used to create evenly spaced matrices
  - can be used to extract portions of existing matrices
  - can be used to transform a 2-D matrix into a single column

Summary - Meshgrid

- Meshgrid is an extremely useful function that can be used to map vectors into two dimensional matrices
  - This makes it possible to perform array calculations with vectors of unequal size

Summary – Special Matrices

- zeros – creates a matrix composed of all zeros
- ones – creates a matrix composed of all ones
- diag – extracts the diagonal from a square matrix or can be used to create a square matrix identity matrix
- magic – creates a “magic matrix”

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming SCE2304

Lecture 5 (Plotting)

Zeyad T. Shareef

Objectives

After studying this lecture, you should be able to:

- Create and label two dimensional plots
- Adjust the appearance of your plots
- Divide the plotting window into subplots
- Create three dimensional plots
- Use the interactive plotting tools

Section 5.1
5.1.1 Two Dimensional Plots

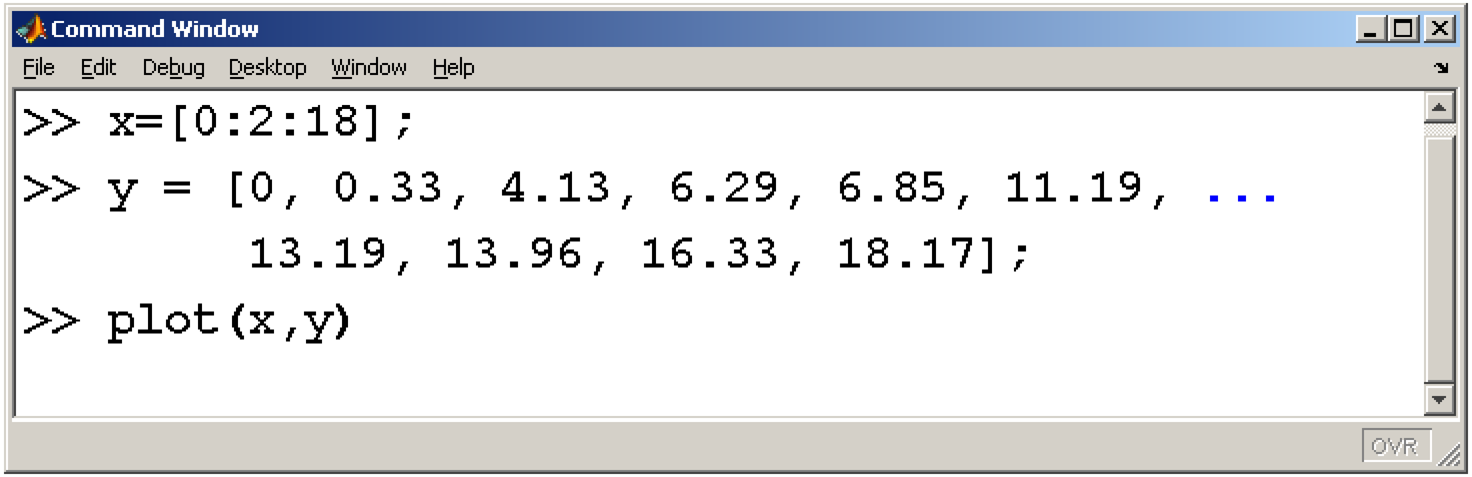
- The x-y plot is the most commonly used plot by engineers
- The independent variable is usually called x
- The dependent variable is usually called y

Consider this x-y data

time, sec	Distance, Ft
0	0
2	0.33
4	4.13
6	6.29
8	6.85
10	11.19
12	13.19
14	13.96
16	16.33
18	18.17

Time is the independent variable and distance is the dependent variable

Define x and y and call the plot function



```
>> x=[0:2:18];
>> y = [0, 0.33, 4.13, 6.29, 6.85, 11.19, ...
        13.19, 13.96, 16.33, 18.17];
>> plot(x,y)
```

You can use any variable name that is convenient for the dependent and independent variables

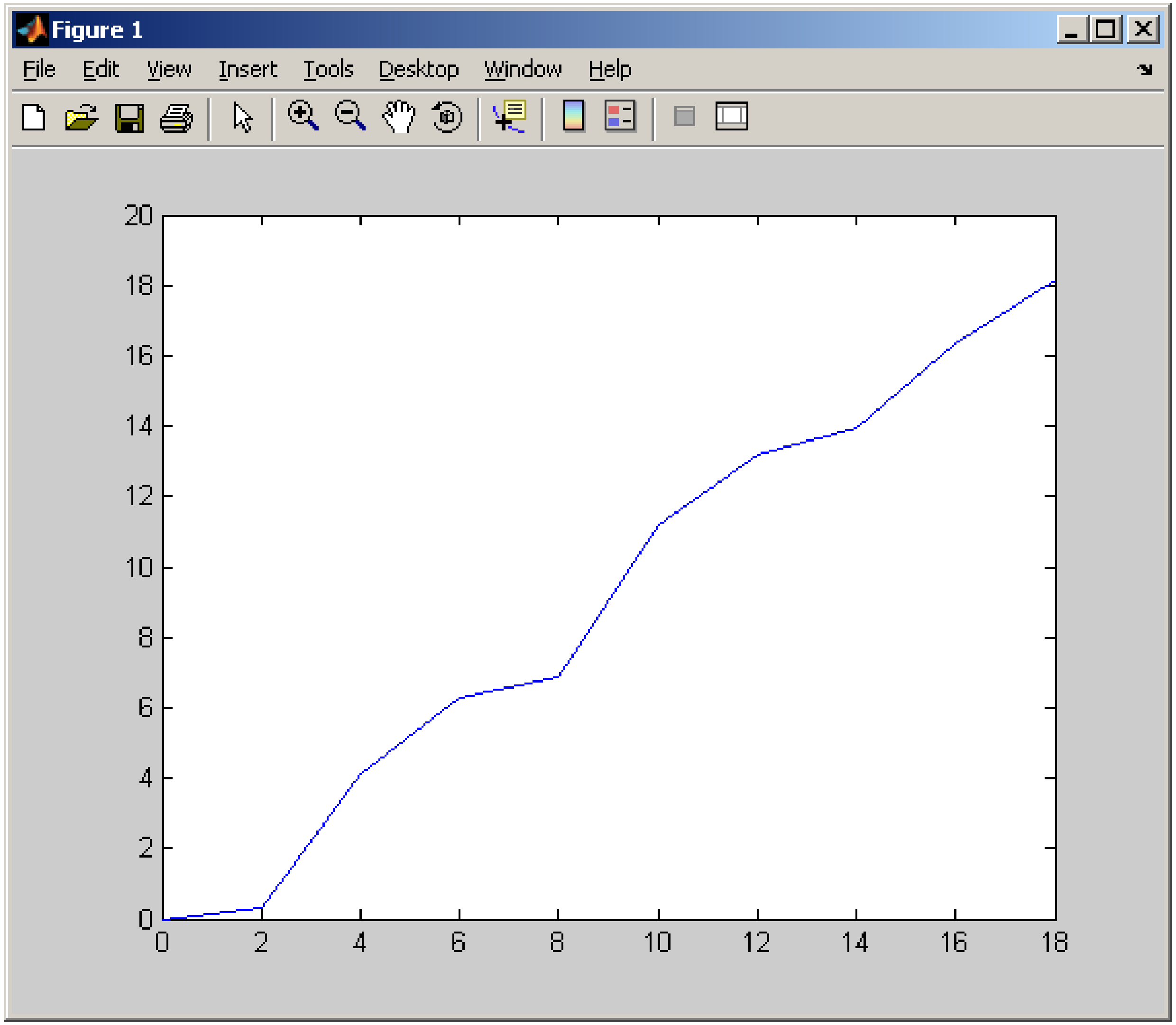
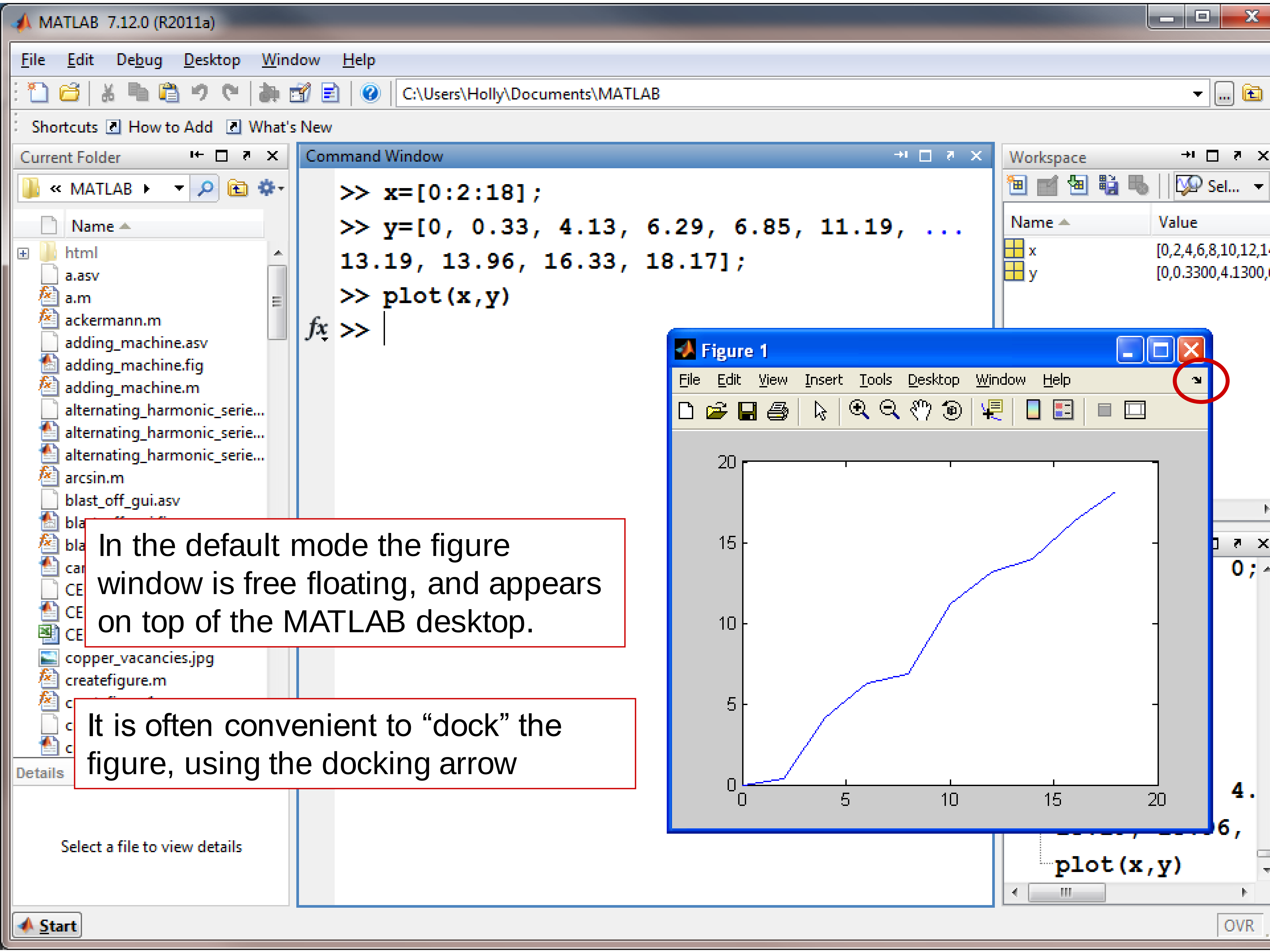


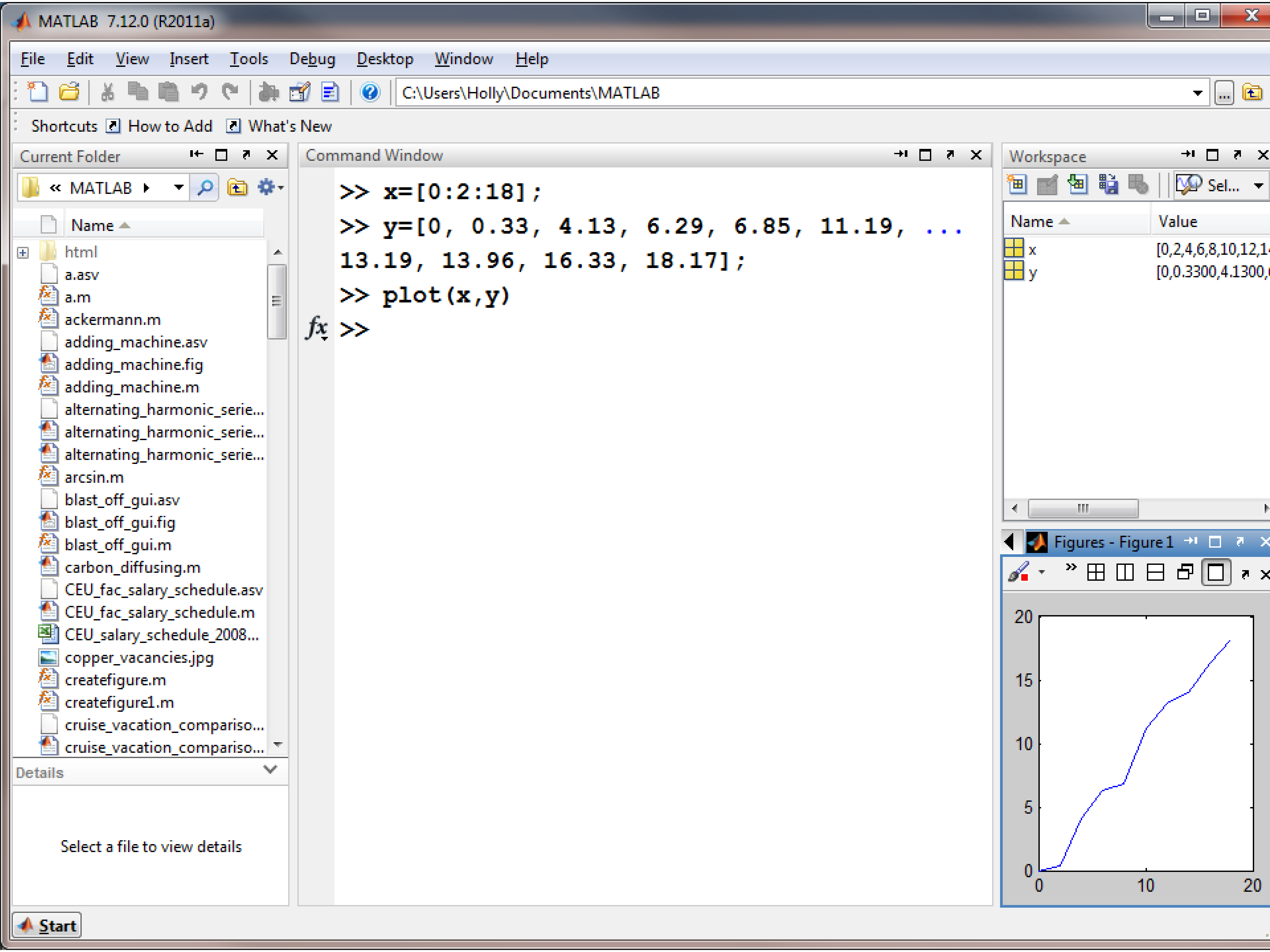
Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
20
18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18



```
>> x=[0:2:18];
>> y=[0, 0.33, 4.13, 6.29, 6.85, 11.19, ...
13.19, 13.96, 16.33, 18.17];
>> plot(x,y)
>>
```

In the default mode the figure window is free floating, and appears on top of the MATLAB desktop.

It is often convenient to “dock” the figure, using the docking arrow



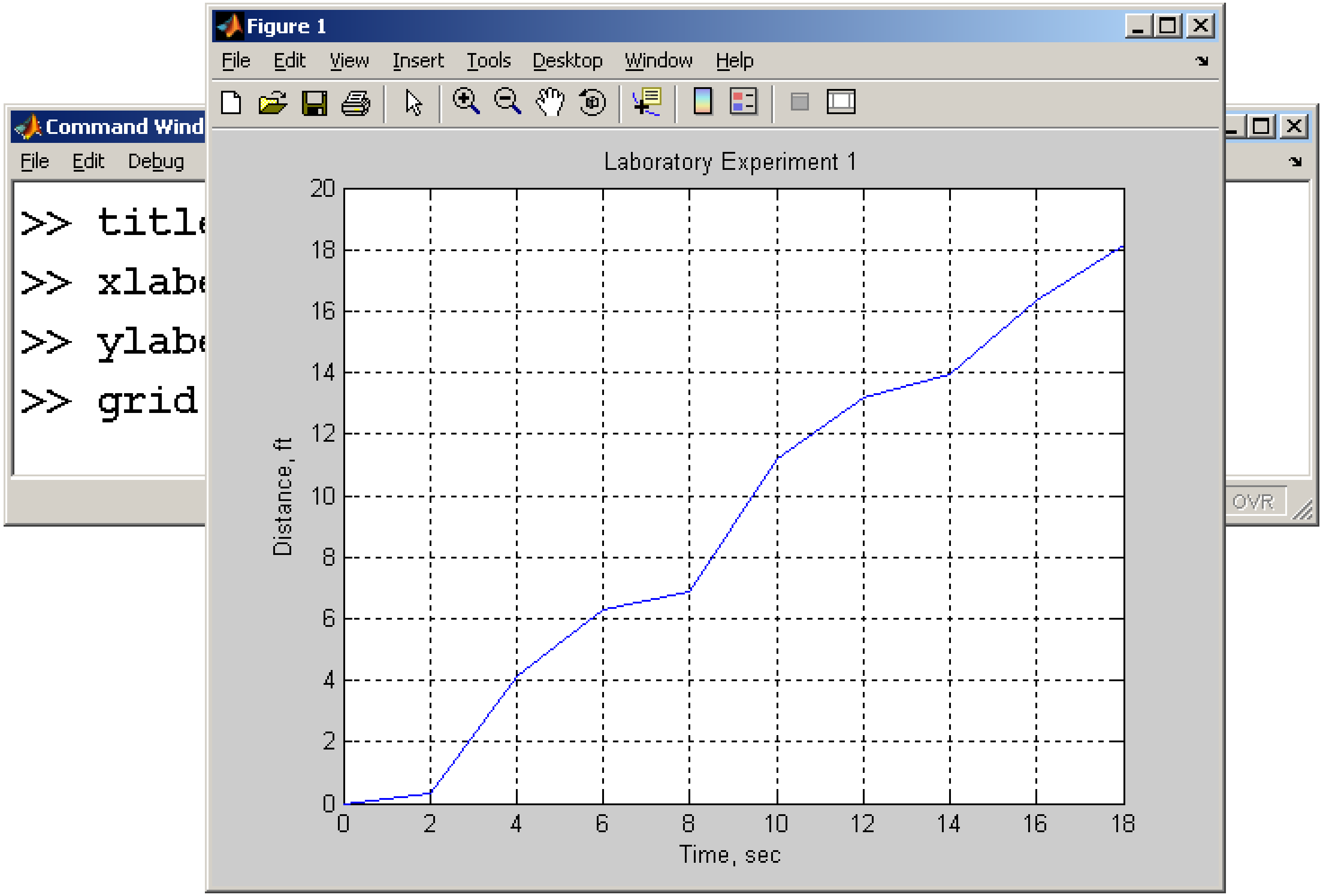
```
>> x=[0:2:18];
>> y=[0, 0.33, 4.13, 6.29, 6.85, 11.19, ...
13.19, 13.96, 16.33, 18.17];
>> plot(x,y)
>>
```

Name ▲	Value
x	[0,2,4,6,8,10,12,1
y	[0,0.3300,4.1300,

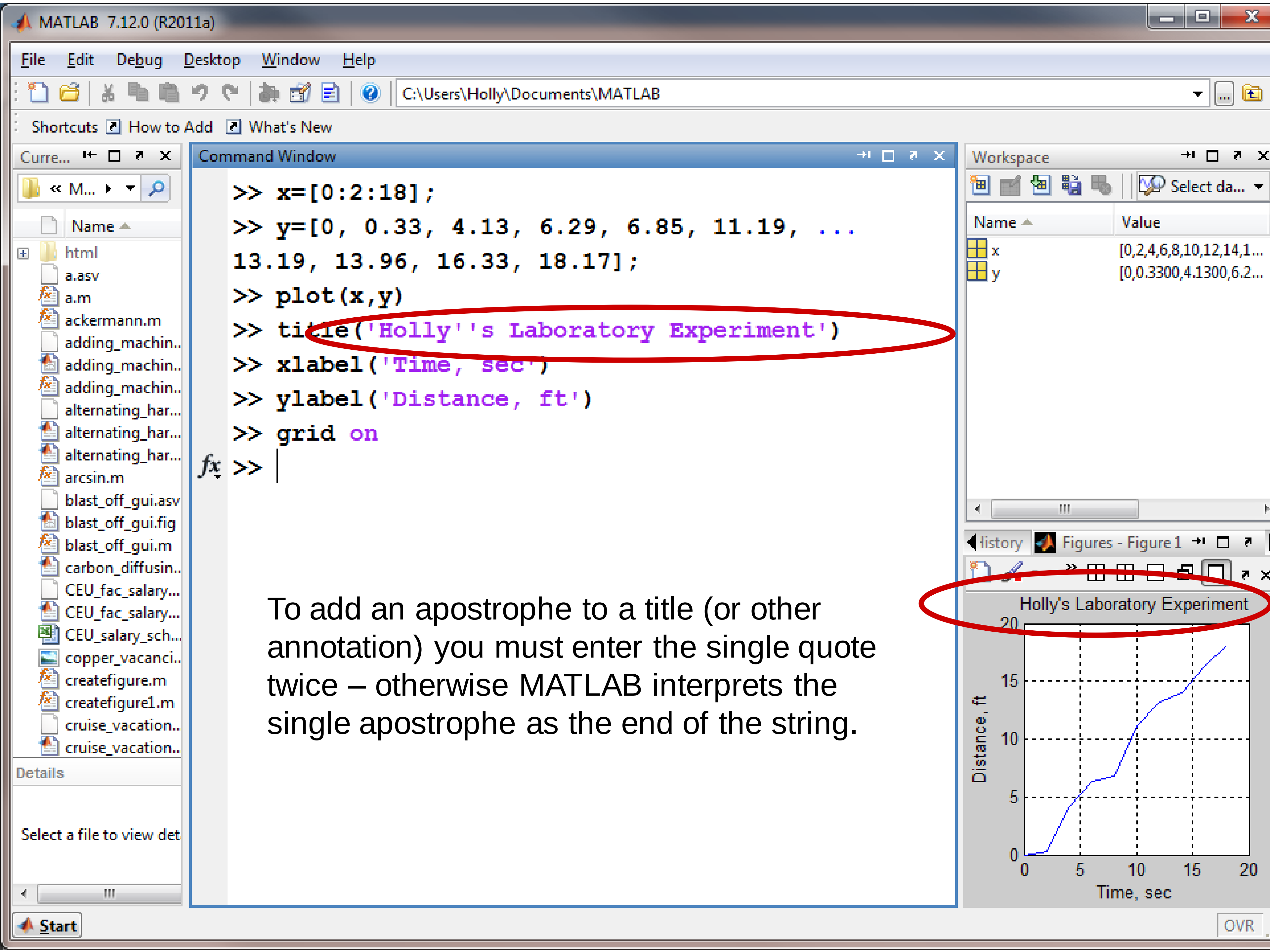
Engineers always add …

- Title
- X axis label, complete with units
- Y axis label, complete with units

- Often it is useful to add a grid



```
>> titl
>> xlabe
>> ylabe
>> grid
```



```
>> x=[0:2:18];
>> y=[0, 0.33, 4.13, 6.29, 6.85, 11.19, ...
13.19, 13.96, 16.33, 18.17];
>> plot(x,y)
>> title('Holly''s Laboratory Experiment')
>> xlabel('Time, sec')
>> ylabel('Distance, ft')
>> grid on
>>
```

To add an apostrophe to a title (or other annotation) you must enter the single quote twice – otherwise MATLAB interprets the single apostrophe as the end of the string.

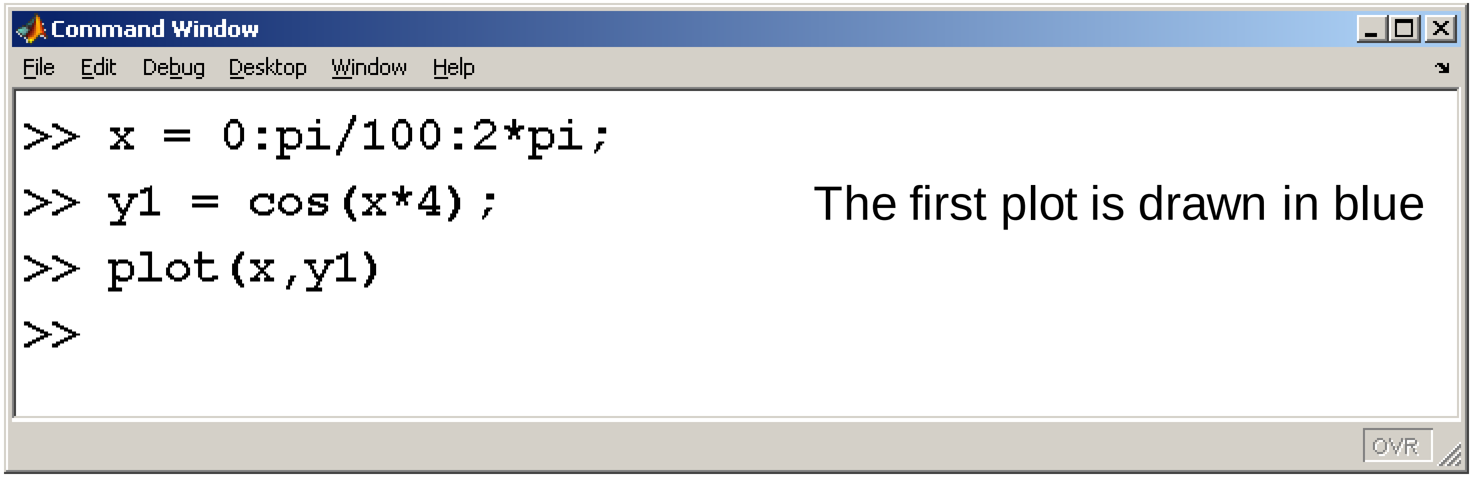
Creating multiple plots

- MATLAB overwrites the figure window every time you request a new plot
- To open a new figure window, use the figure function – for example

```
figure(2)
```

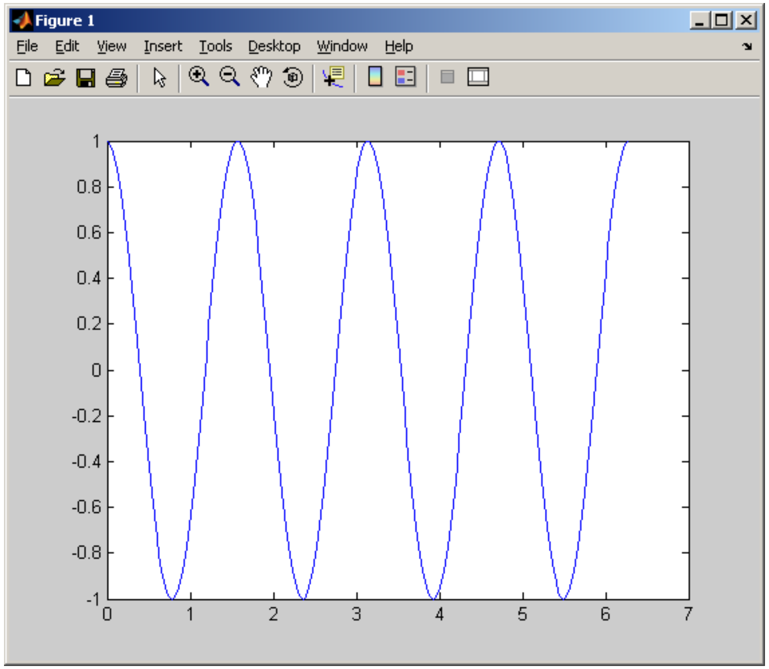
Plots with multiple lines

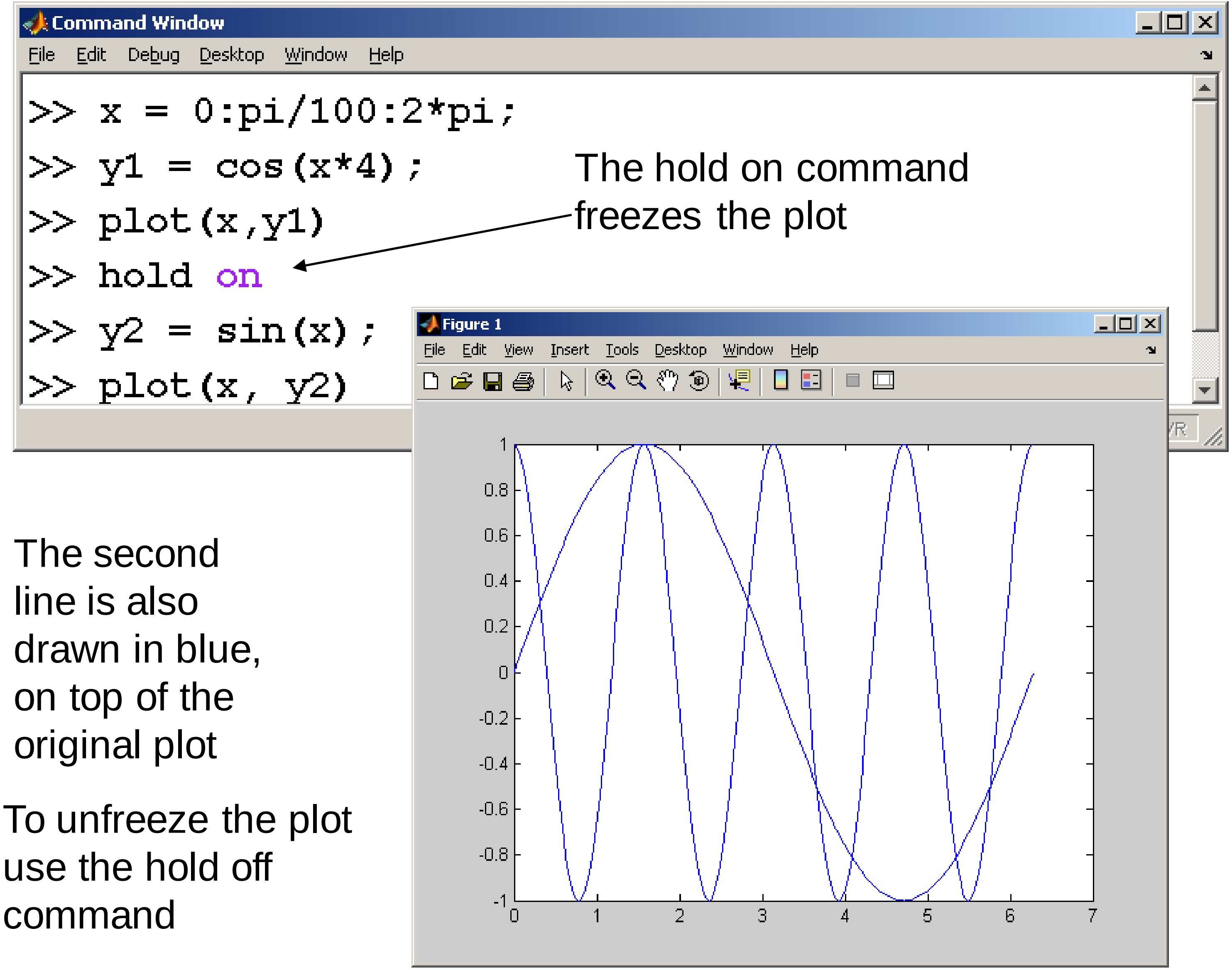
- hold on
  - Freezes the current plot, so that an additional plot can be overlaid
- When you use this approach, the additional line is drawn in blue – the default drawing color



```
>> x = 0:pi/100:2*pi;
>> y1 = cos(x*4);
>> plot(x,y1)
>>
```

The first plot is drawn in blue





```
>> x = 0:pi/100:2*pi;
>> y1 = cos(x*4);
>> plot(x,y1)
>> hold on
>> y2 = sin(x);
>> plot(x, y2)
```

The hold on command freezes the plot

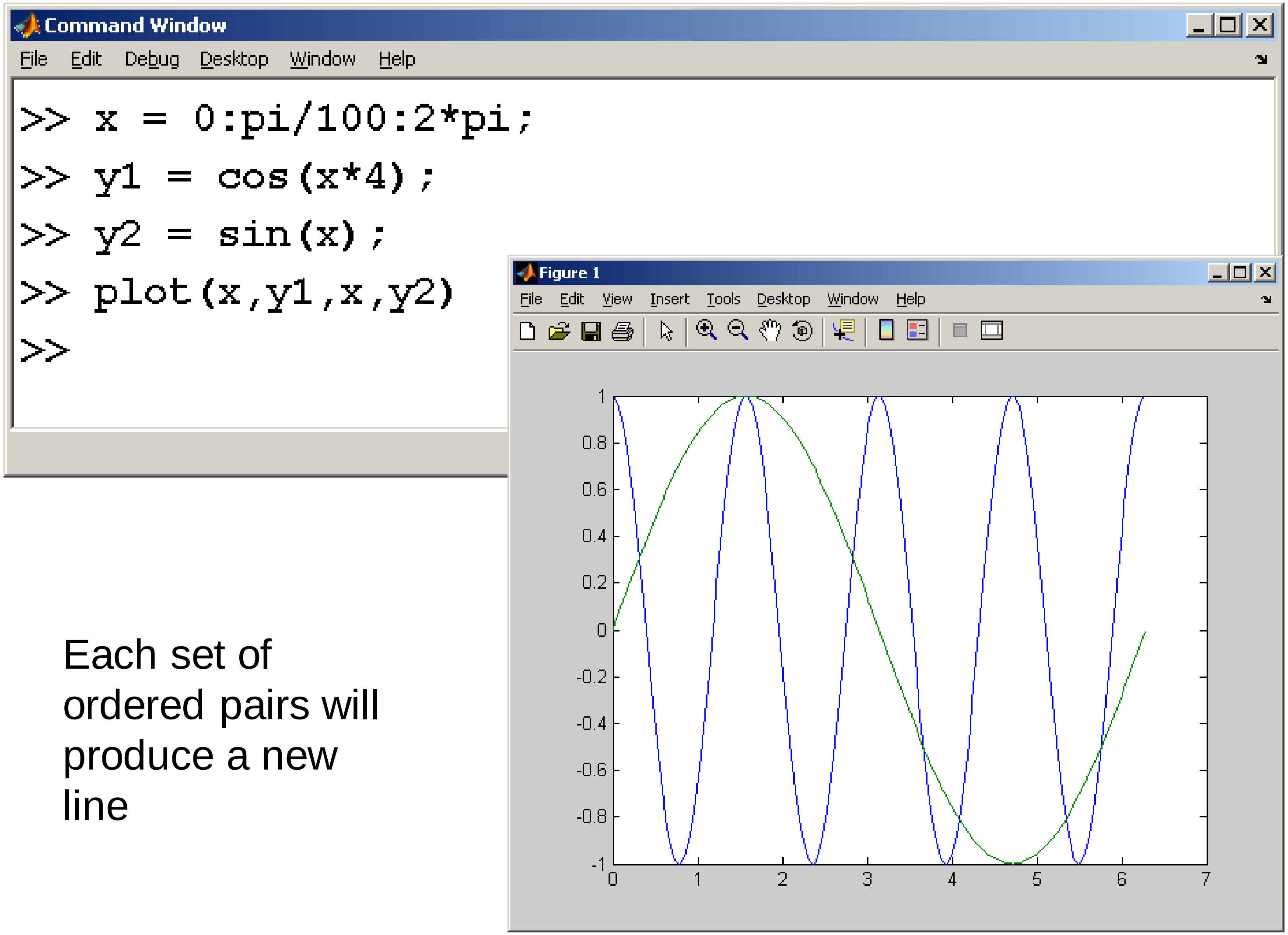
Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

The second line is also drawn in blue, on top of the original plot

To unfreeze the plot use the hold off command

- You can also create multiple lines on a single graph with one command

- Using this approach each line defaults to a different color



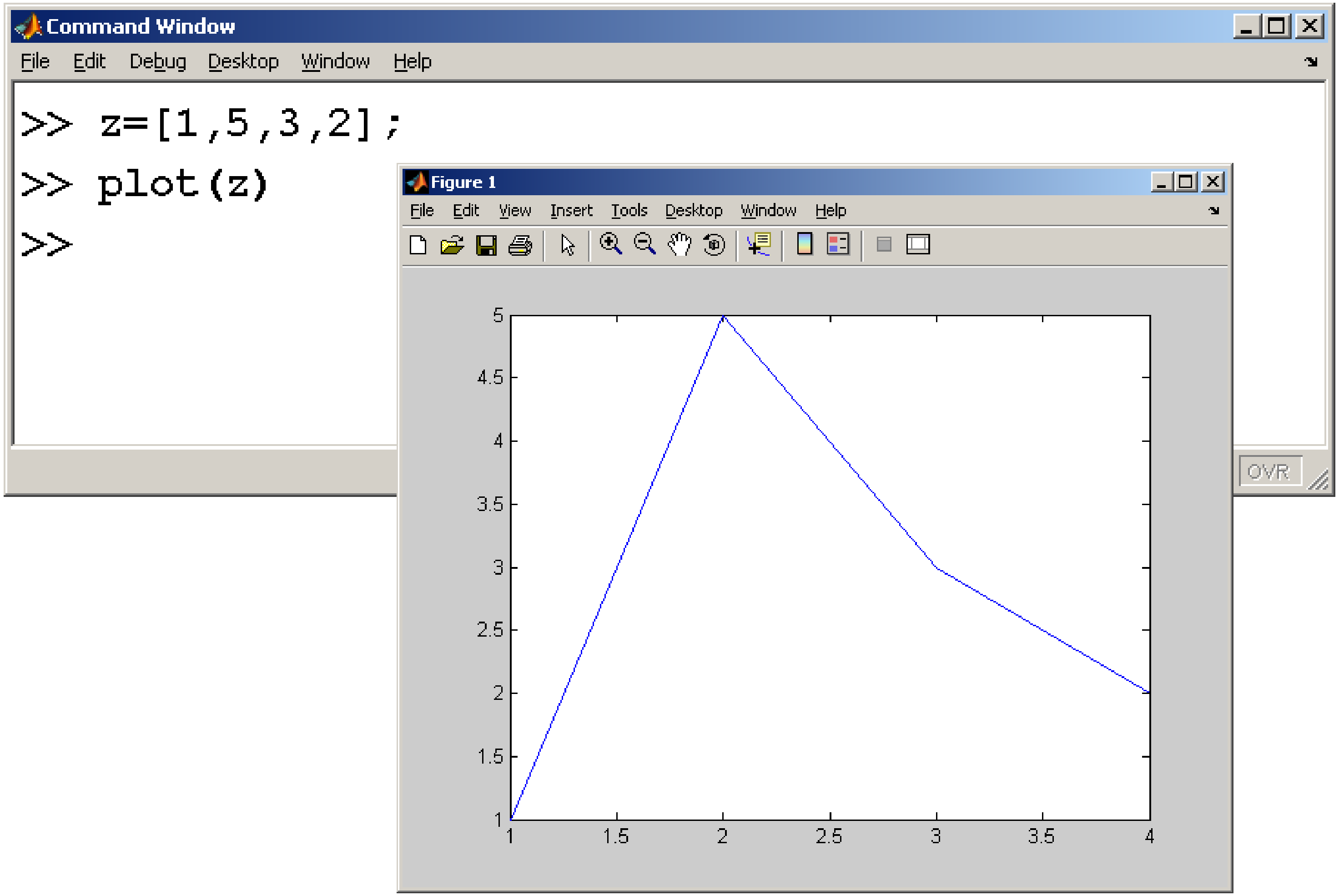
Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

```
>> x = 0:pi/100:2*pi;
>> y1 = cos(x*4);
>> y2 = sin(x);
>> plot(x,y1,x,y2)
>>
```

Each set of ordered pairs will produce a new line

Variations

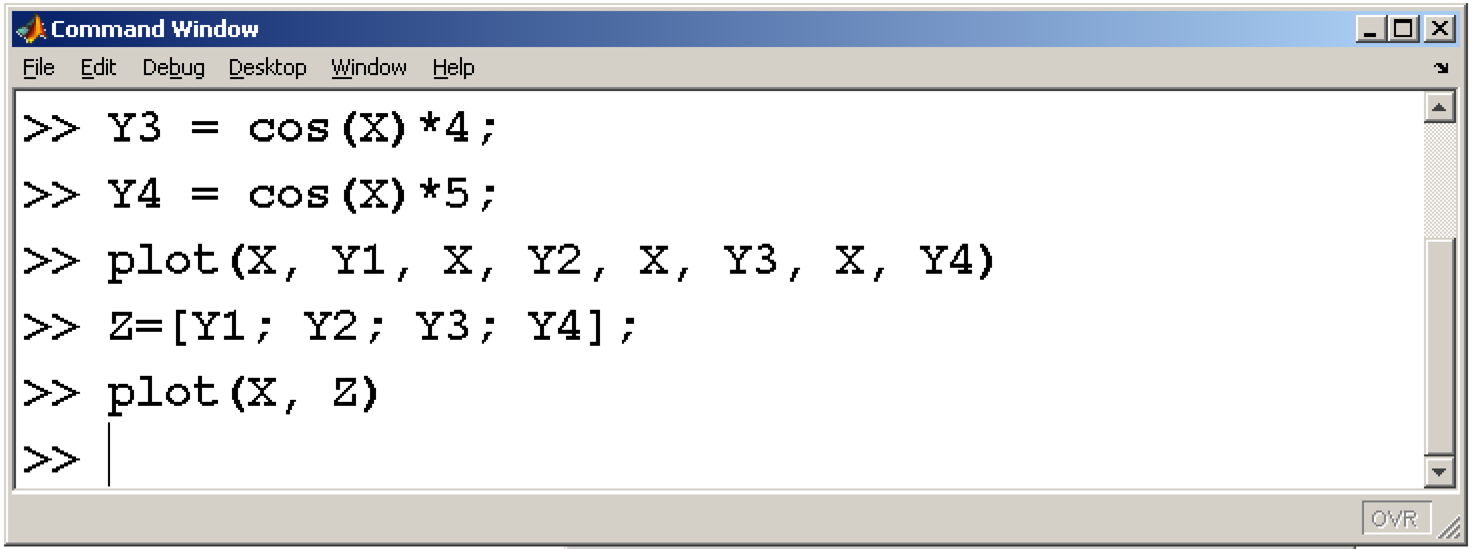
- If you use the plot command with a single matrix, MATLAB plots the values versus the index number
- Usually this type of data is plotted on a bar graph
- When plotted on an x-y grid, it is often called a line graph


Command Window
File
Edit
Debug
Desktop
Window
Help
Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
5
4.5
4
3.5
3
2.5
2
1.5
1
1
1.5
2
2.5
3
3.5
4
OVR

```
>> z=[1,5,3,2];
>> plot(z)
>>
```

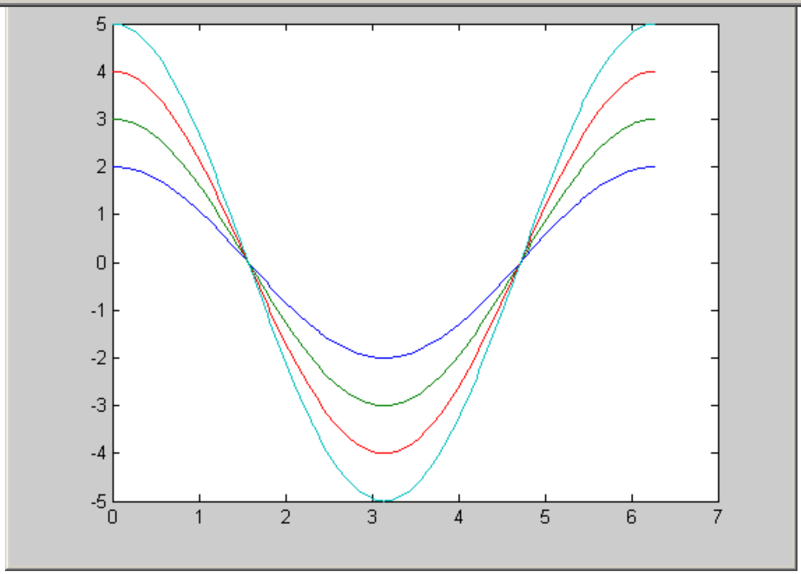
If you want to create multiple plots, all with the same x value you can…

- Use alternating sets of ordered pairs
  - plot(x, y1, x, y2, x, y3, x, y4)
- Or group the y values into a matrix
  - z=[y1, y2, y3, y4]
  - plot(x,z)

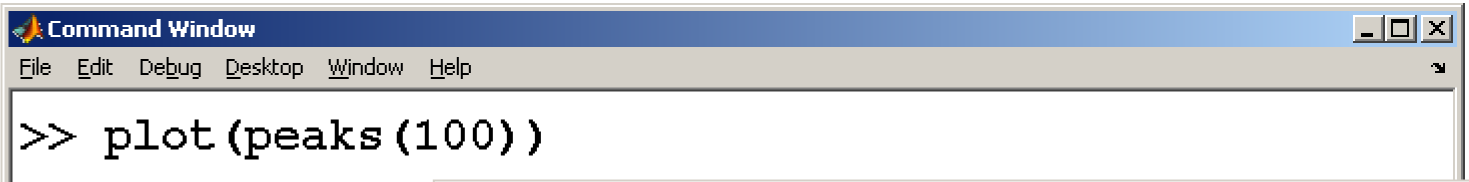


```
>> Y3 = cos(X)*4;
>> Y4 = cos(X)*5;
>> plot(X, Y1, X, Y2, X, Y3, X, Y4)
>> Z=[Y1; Y2; Y3; Y4];
>> plot(X, Z)
>> 
```

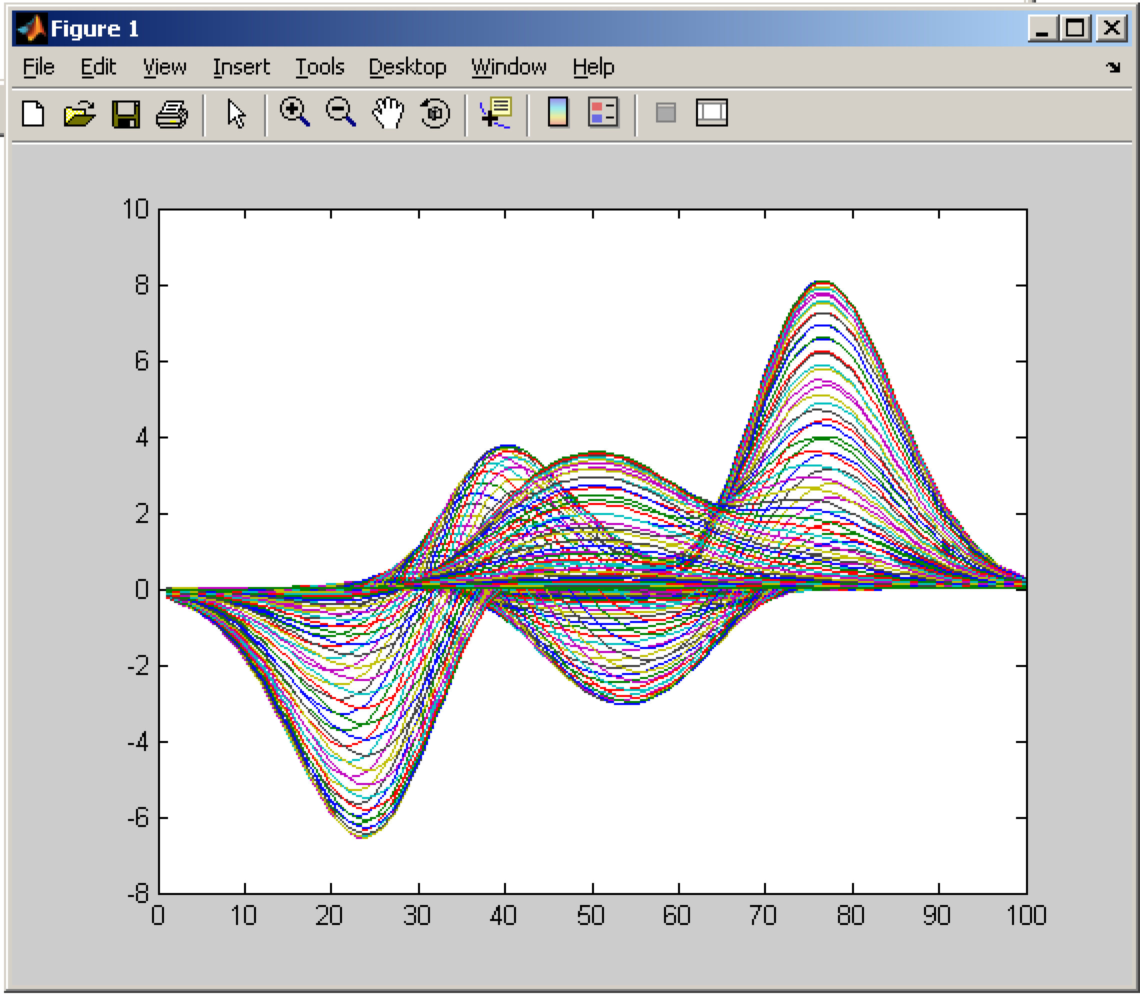
Alternating sets of ordered pairs



Matrix of Y values



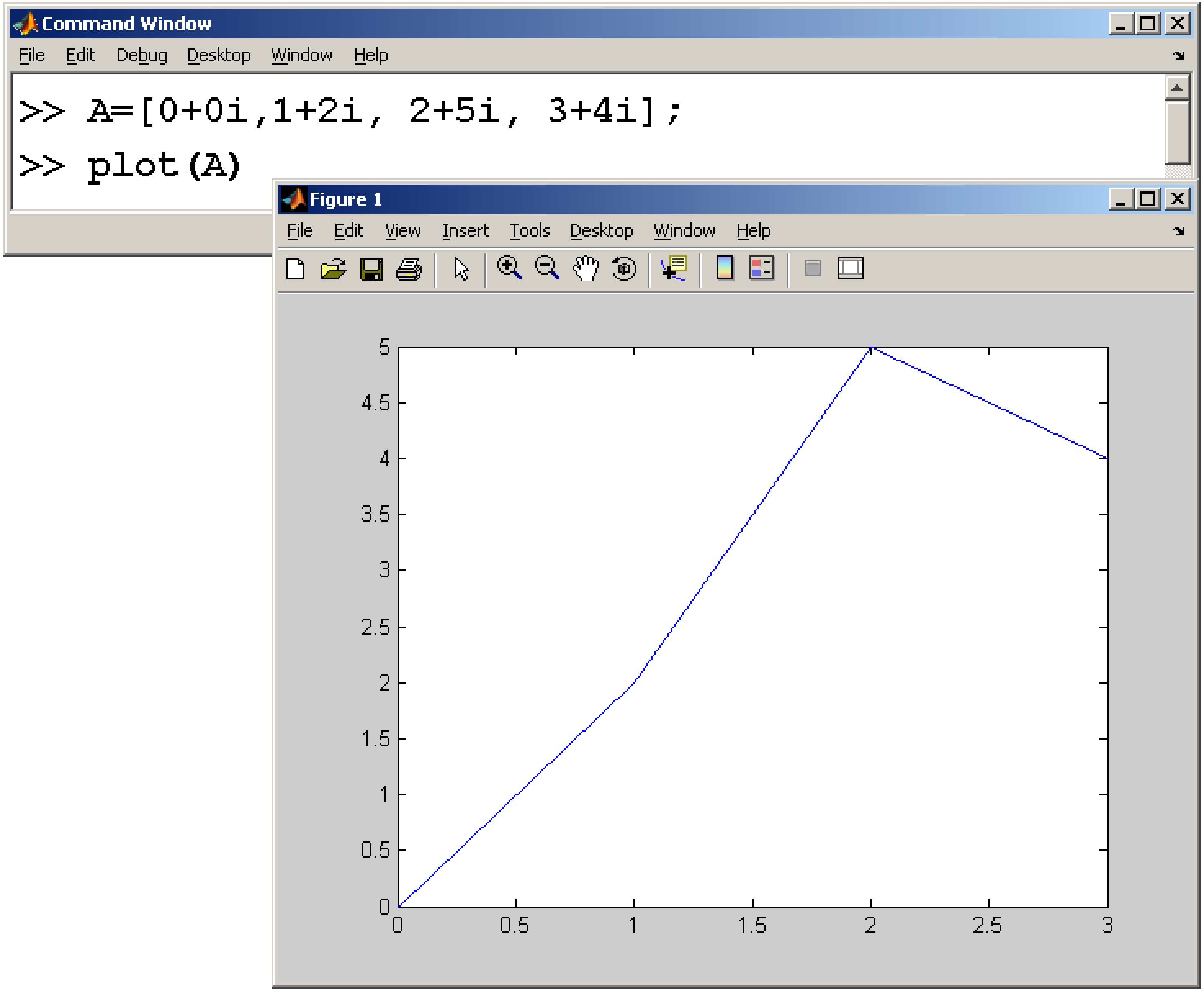
```
>> plot(peaks(100))
>>
```



The peaks(100) function creates a 100x100 array of values. Since this is a plot of a single variable, we get 100 different line plots

Plots of Complex Arrays

- If the input to the plot command is a single array of complex numbers, MATLAB plots the real component on the x-axis and the imaginary component on the y-axis

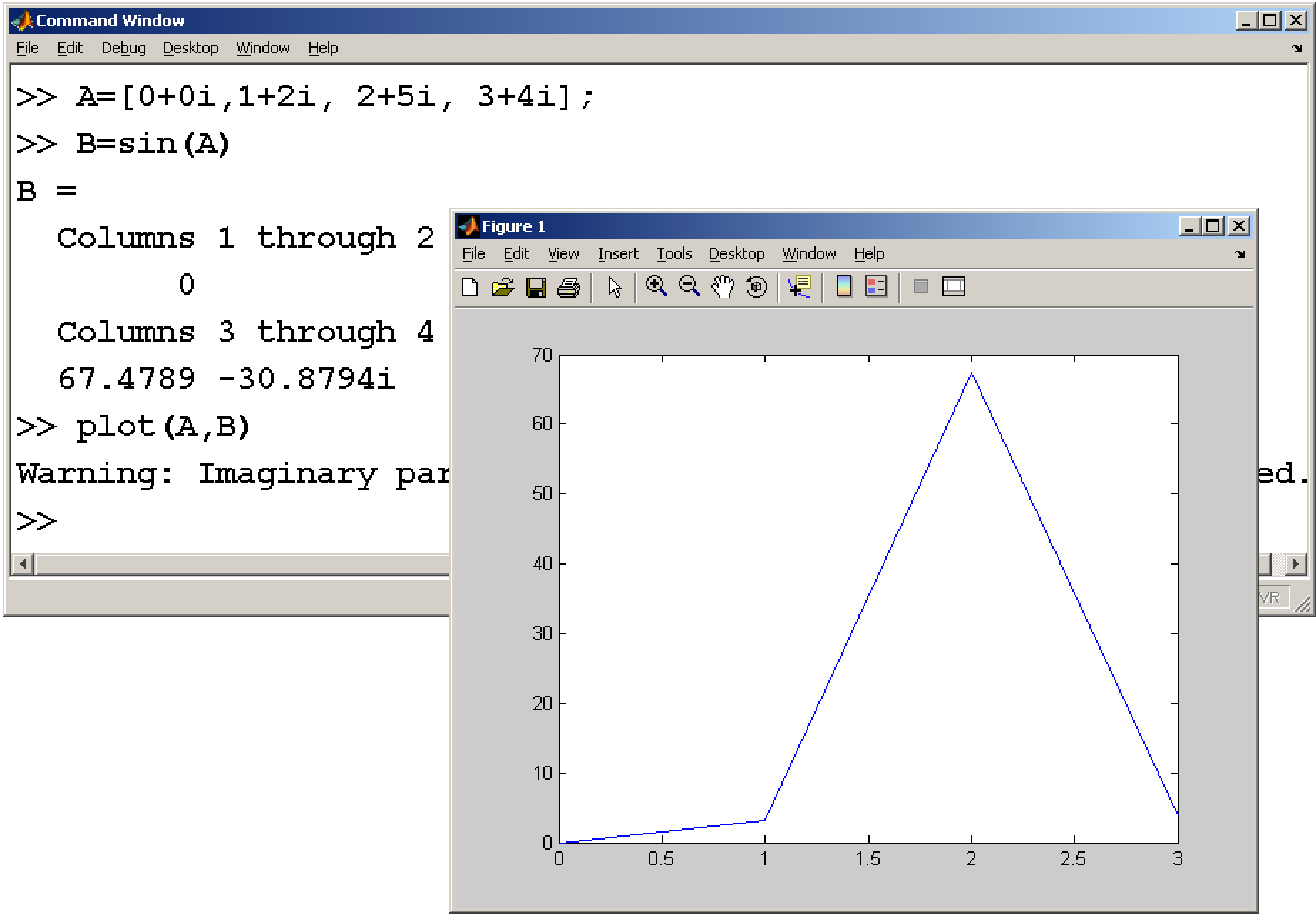


```
>> A=[0+0i,1+2i, 2+5i, 3+4i];
>> plot(A)
```

Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3

Multiple arrays of complex numbers

- If you try to use two arrays of complex numbers in the plot function, the imaginary components are ignored



Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
70
60
50
40
30
20
10
0
0
0.5
1
1.5
2
2.5
3
VR

```
>> A=[0+0i,1+2i, 2+5i, 3+4i];
>> B=sin(A)
B =
   Columns 1 through 2
         0
   Columns 3 through 4
   67.4789 -30.8794i
>> plot(A,B)
Warning: Imaginary par                                        ed.
>>
```

5.1.2 Line, Color and Mark Style

- You can change the appearance of your plots by selecting user defined
  - line styles
  - color
  - mark styles
- Try using help plot for a list of available styles

Available choices

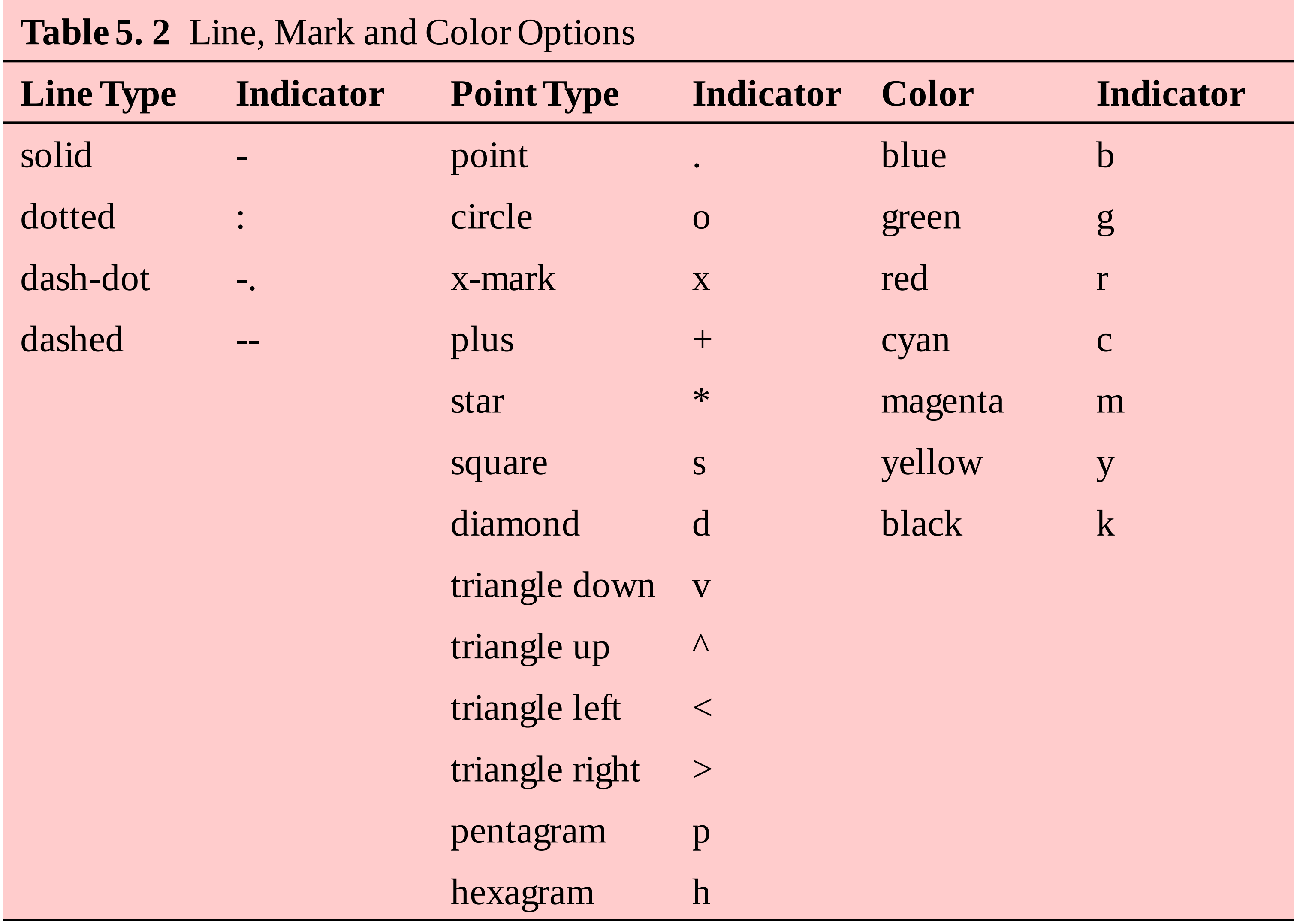


Table 5. 2 Line, Mark and Color Options

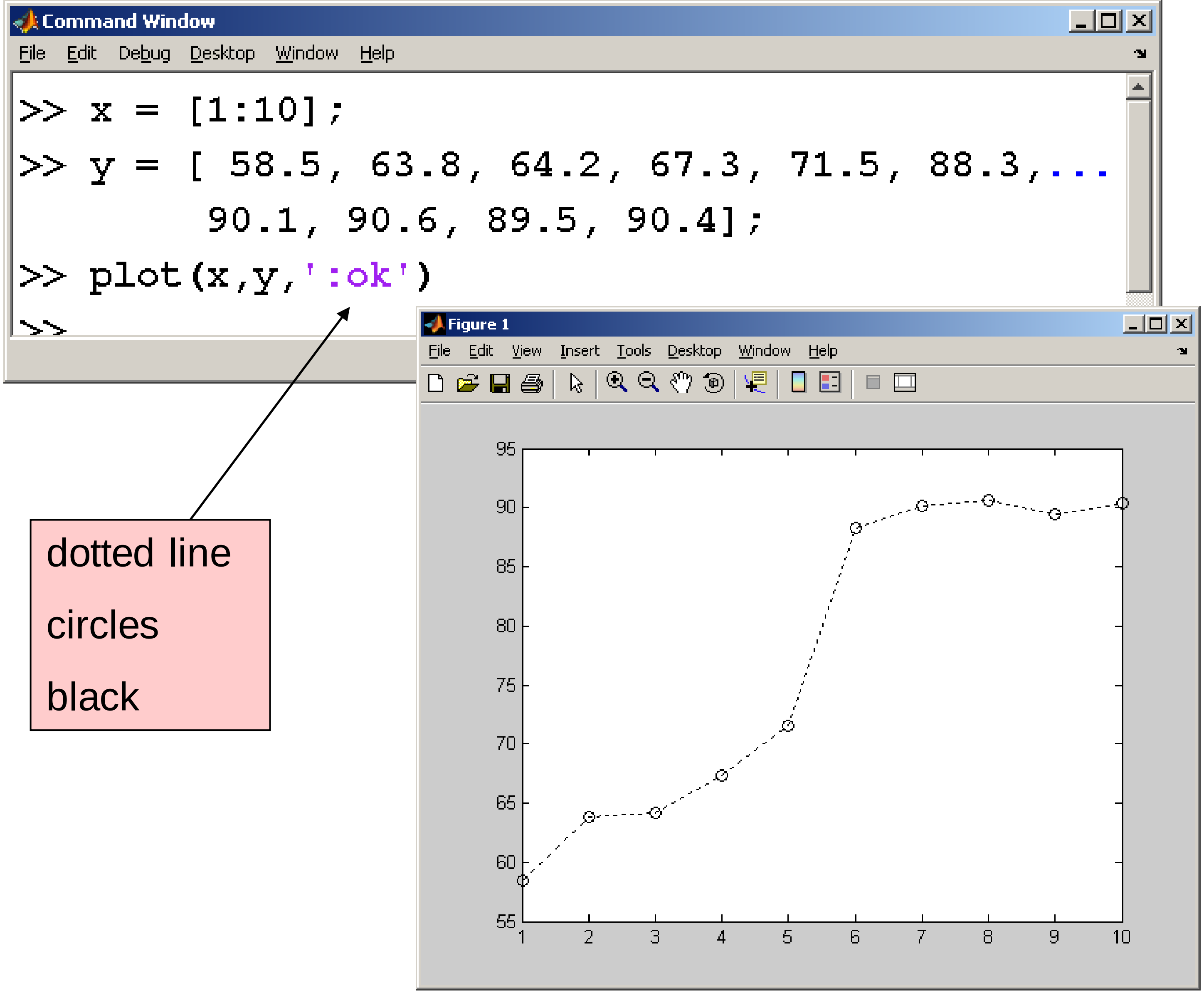
Line Type	**Indicator**	**Point Type**	**Indicator**	**Color**	**Indicator**
solid	-	point	.	blue	b
dotted	:	circle	o	green	g
dash-dot	-.	x-mark	x	red	r
dashed	--	plus	+	cyan	c
		star	*	magenta	m
		square	s	yellow	y
		diamond	d	black	k
		triangle down	v		
		triangle up	^		
		triangle left	<		
		triangle right	>		
		pentagram	p		
		hexagram	h		

Specify your choices in a string

- For example
- plot(x,y,':ok')
  - strings are identified with a tick mark
  - if you don’t specify style, a default is used
    - line style – none
    - mark style – none
    - color - blue

plot(x,y,':ok')

- In this command
  - the : means use a dotted line
  - the o means use a circle to mark each point
  - the letter k indicates that the graph should be drawn in black
    - (b indicates blue)

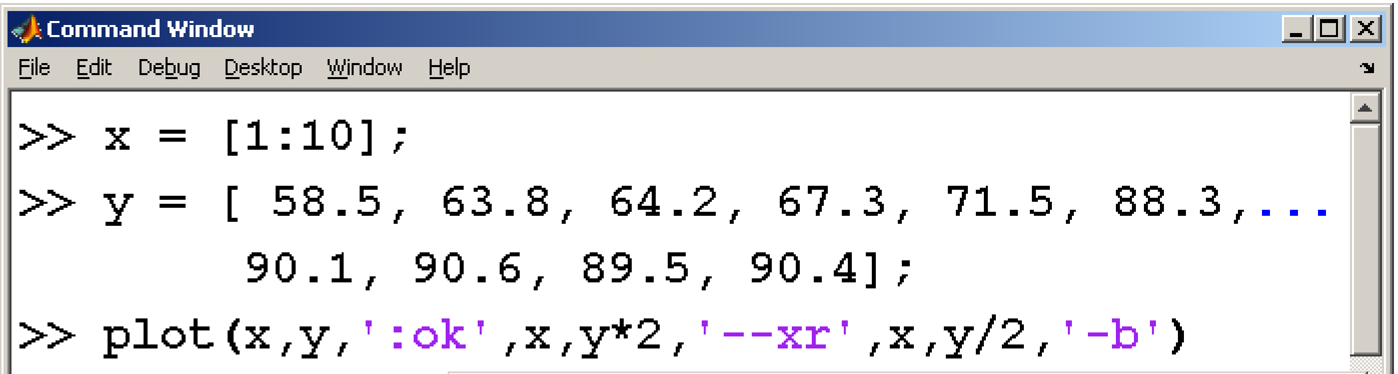

Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
95
90
85
80
75
70
65
60
55
1
2
3
4
5
6
7
8
9
10

```
>> x = [1:10];
>> y = [ 58.5, 63.8, 64.2, 67.3, 71.5, 88.3,...
         90.1, 90.6, 89.5, 90.4];
>> plot(x,y,':ok')
>>
```

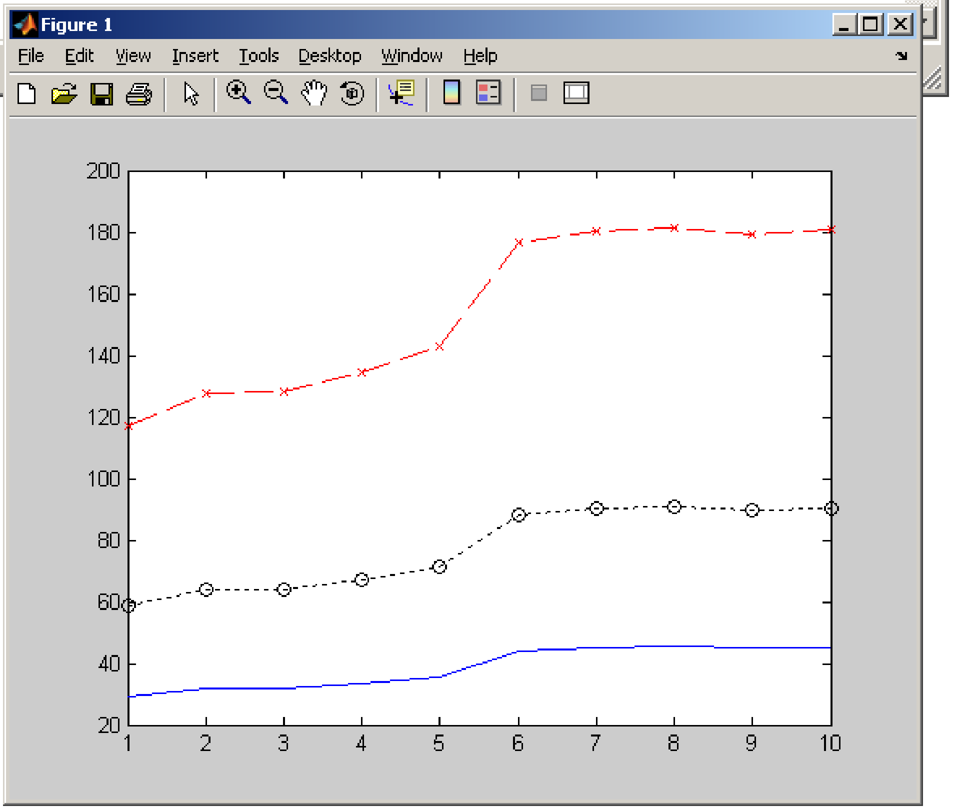
dotted line

circles

black



```
>> x = [1:10];
>> y = [ 58.5, 63.8, 64.2, 67.3, 71.5, 88.3,...
        90.1, 90.6, 89.5, 90.4];
>> plot(x,y,':ok',x,y*2,'--xr',x,y/2,'-b')
>>
```



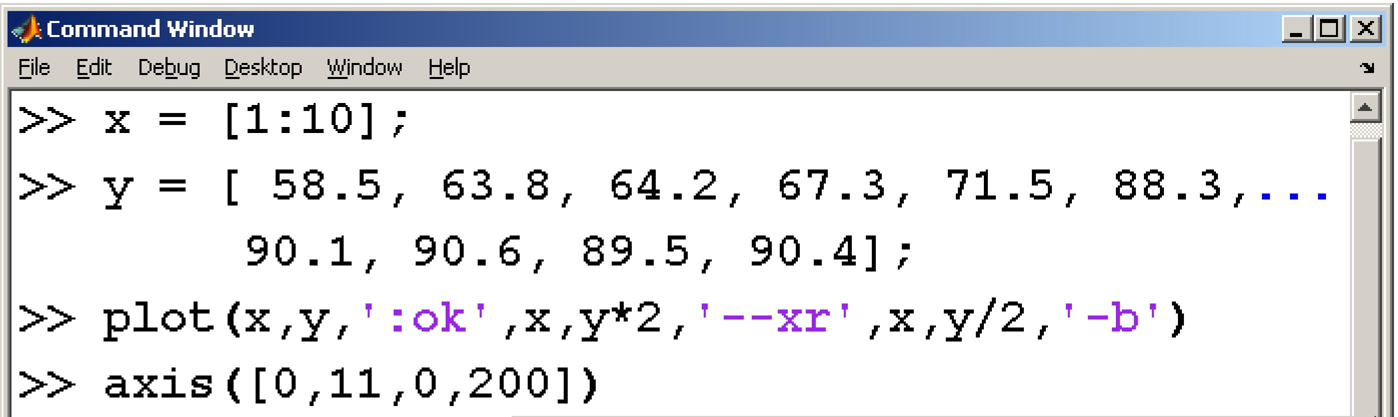
specify the drawing parameters for each line after the ordered pairs that define the line

Axis scaling

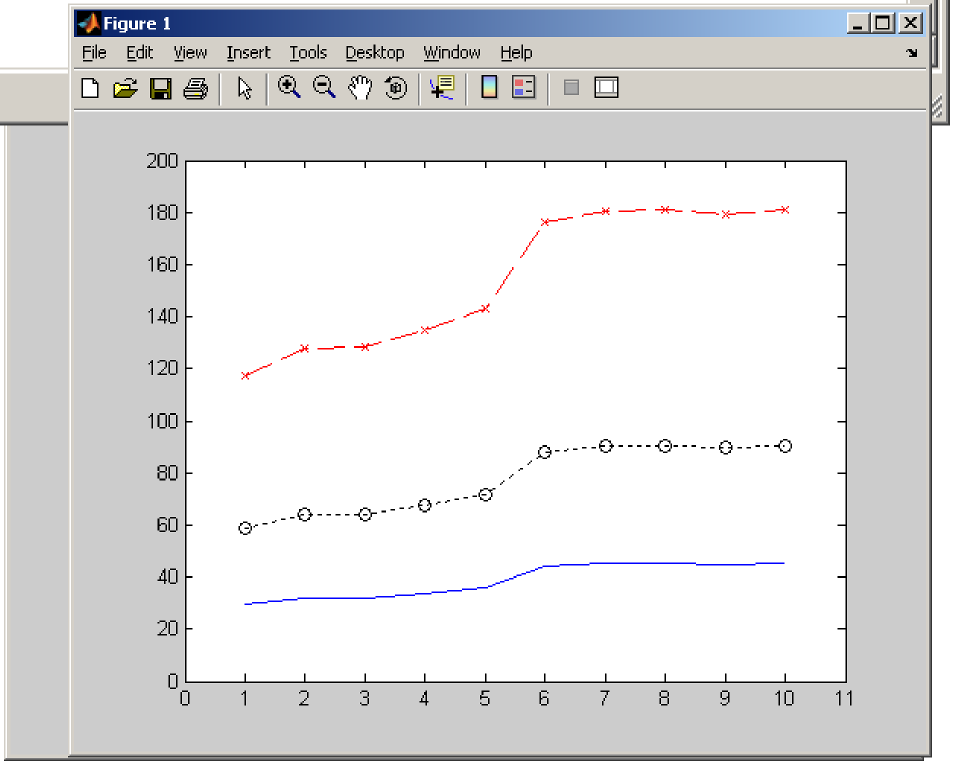
- MATLAB automatically scales each plot to completely fill the graph
- If you want to specify a different axis – use the axis command

```
axis([xmin,xmax,ymin,ymax])
```

- Lets change the axes on the graph we just looked at



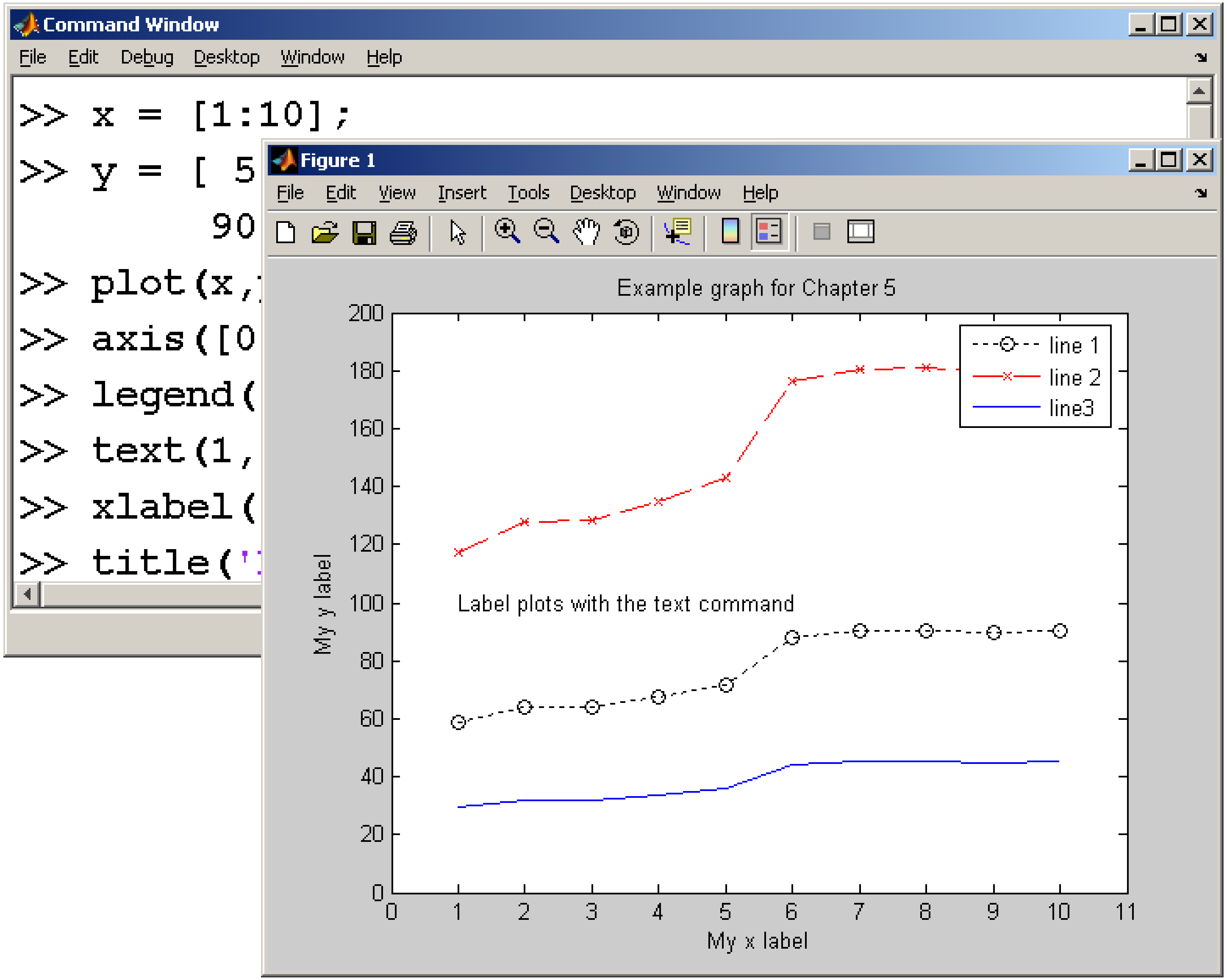
```
>> x = [1:10];
>> y = [ 58.5, 63.8, 64.2, 67.3, 71.5, 88.3,...
        90.1, 90.6, 89.5, 90.4];
>> plot(x,y,':ok',x,y*2,'--xr',x,y/2,'-b')
>> axis([0,11,0,200])
>>
```



Use the axis function to override the automatic scaling

Additional Annotations

- You can also add
  - legends
  - textbox
- Of course, you should always add
  - title
  - axis labels



```
>> x = [1:10];
>> y = [ 5
        90
>> plot(x,
>> axis([0
>> legend(
>> text(1,
>> xlabel(
>> title('
```

Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
Example graph for Chapter 5
line 1
line 2
line3
Label plots with the text command
My y label
My x label
0
20
40
60
80
100
120
140
160
180
200
0
1
2
3
4
5
6
7
8
9
10
11

Improving your labels

You can use Greek letters in your labels by putting a backslash (\) before the name of the letter. For example:

title(‘\alpha \beta \gamma’)

creates the plot title

α β γ

Superscripts and Subscripts

To create a superscript use curly brackets

```
title('x^{2}')
```

gives

x^2

To create a subscript use an underscore

```
title('x_2')
```

gives

x_2

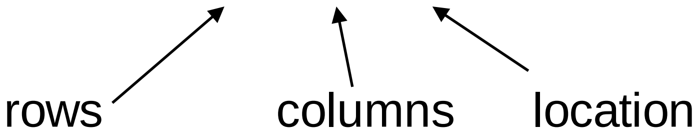
Tex Markup Language

- These label improvements use the Tex Markup Language
- Use the Help feature to find out more!

Section 5.2
Subplots

- The **subplot** command allows you to subdivide the graphing window into a grid of m rows and n columns

subplot(m,n,p)

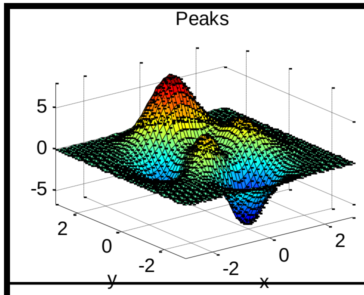


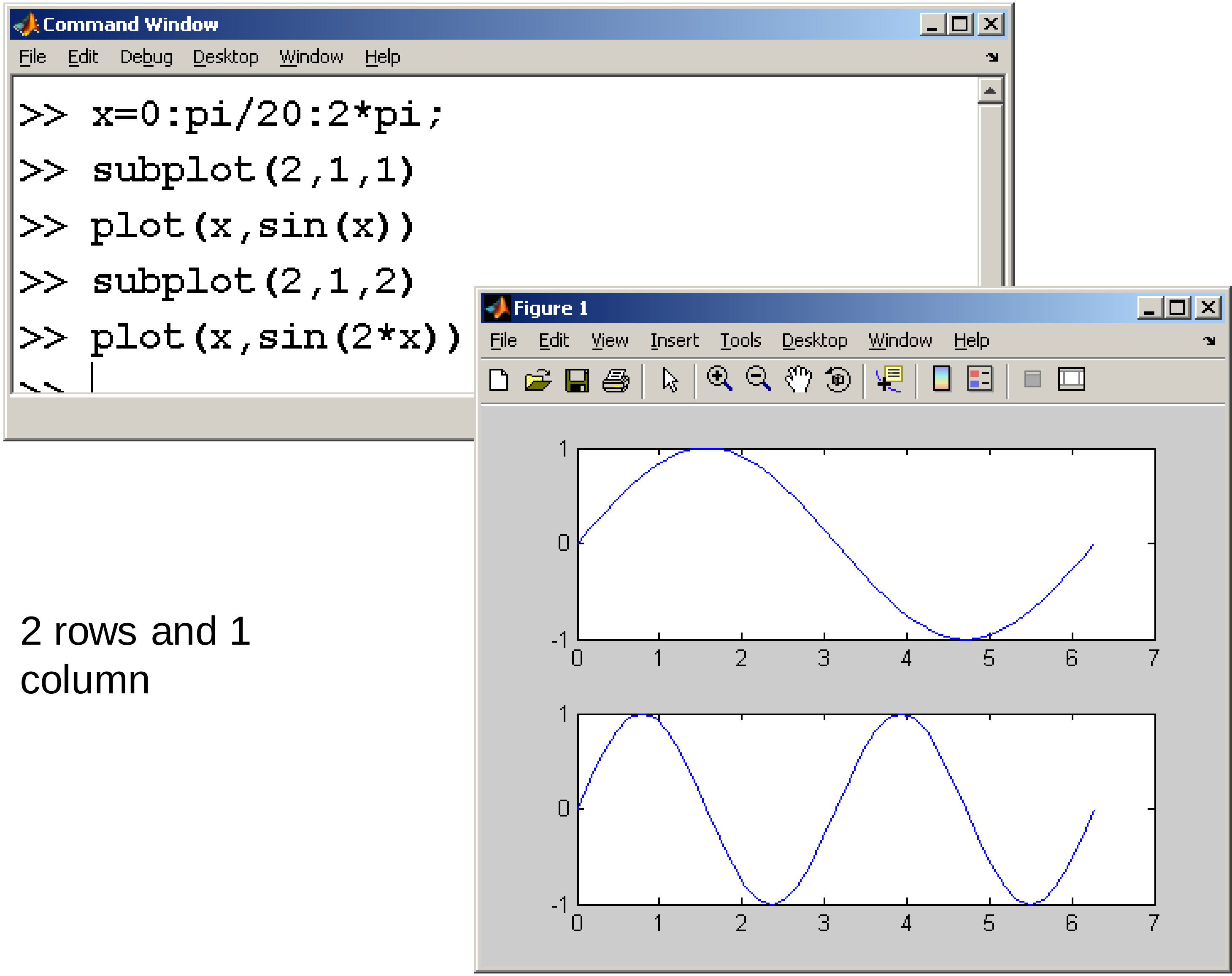
rows columns location

subplot(2,2,1)

2 columns

2 rows

	2
3	4



Command Window
File Edit Debug Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0
-1
0 1 2 3 4 5 6 7
1
0
-1
0 1 2 3 4 5 6 7

```
>> x=0:pi/20:2*pi;
>> subplot(2,1,1)
>> plot(x,sin(x))
>> subplot(2,1,2)
>> plot(x,sin(2*x))
>>
```

2 rows and 1 column

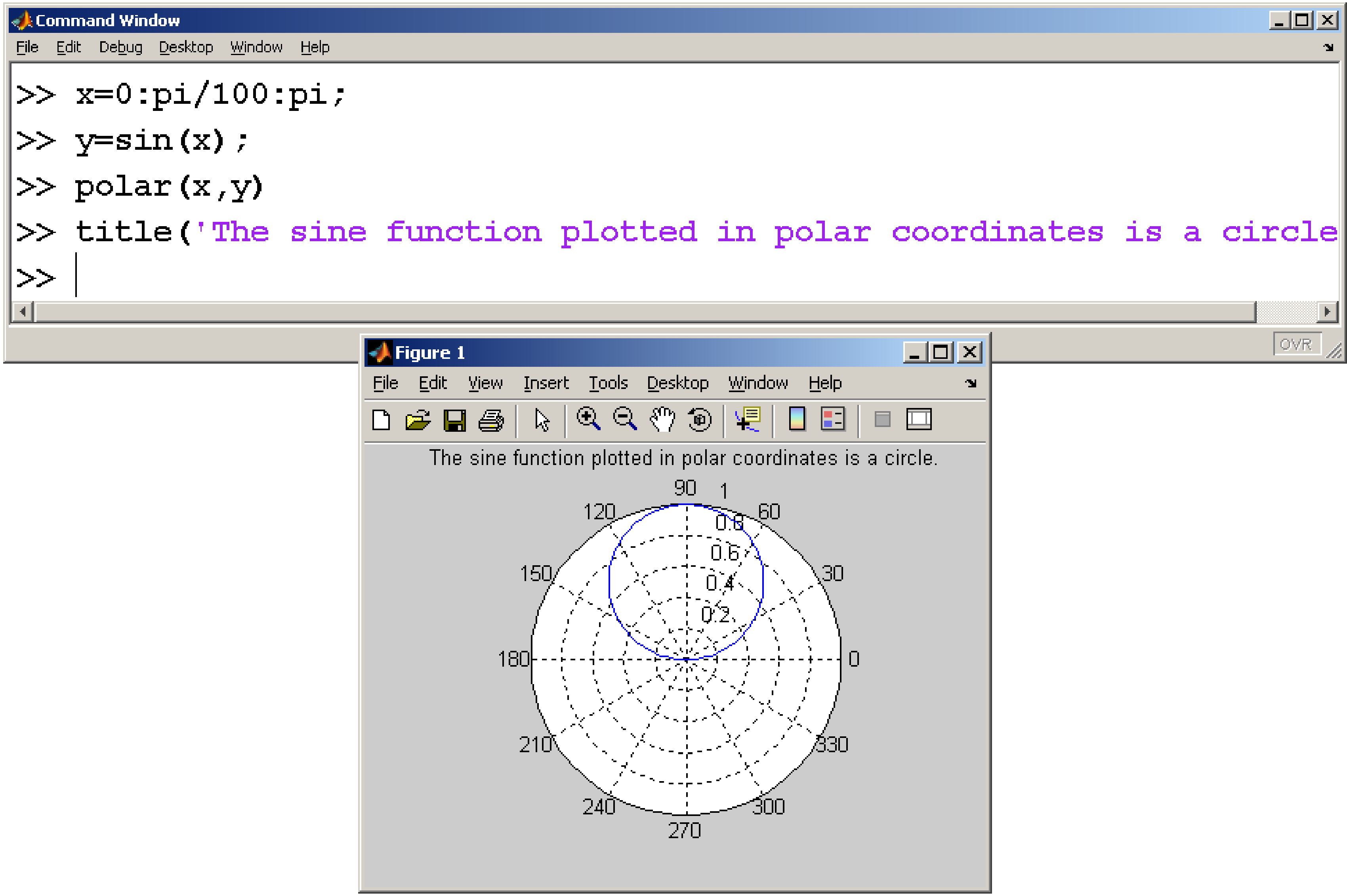
Section 5.3
Other Types of 2-D Plots

- Polar Plots
- Logarithmic Plots
- Bar Graphs
- Pie Charts
- Histograms
- X-Y graphs with 2 y axes
- Function Plots

Polar Plots

- Some functions are easier to specify using polar coordinates than by using rectangular coordinates
- For example, the equation of a circle is
  - y=sin(x)

  in polar coordinates

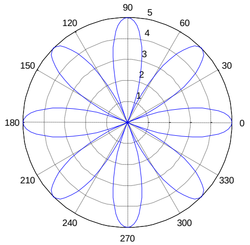


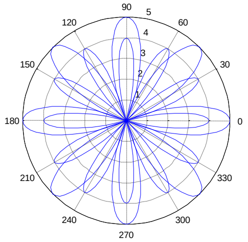
```
>> x=0:pi/100:pi;
>> y=sin(x);
>> polar(x,y)
>> title('The sine function plotted in polar coordinates is a circle
>> 
```

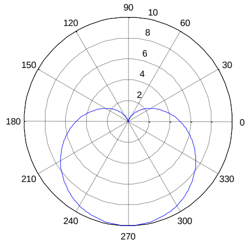
Command Window
File Edit Debug Desktop Window Help
OVR
Figure 1
File Edit View Insert Tools Desktop Window Help
The sine function plotted in polar coordinates is a circle.
90
1
120
60
0.8
0.6
150
30
0.4
0.2
180
0
210
330
240
300
270

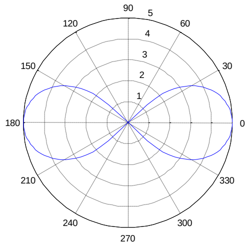
Practice Exercise 5.3

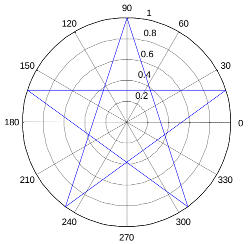
- Try these exercises to create some interesting shapes





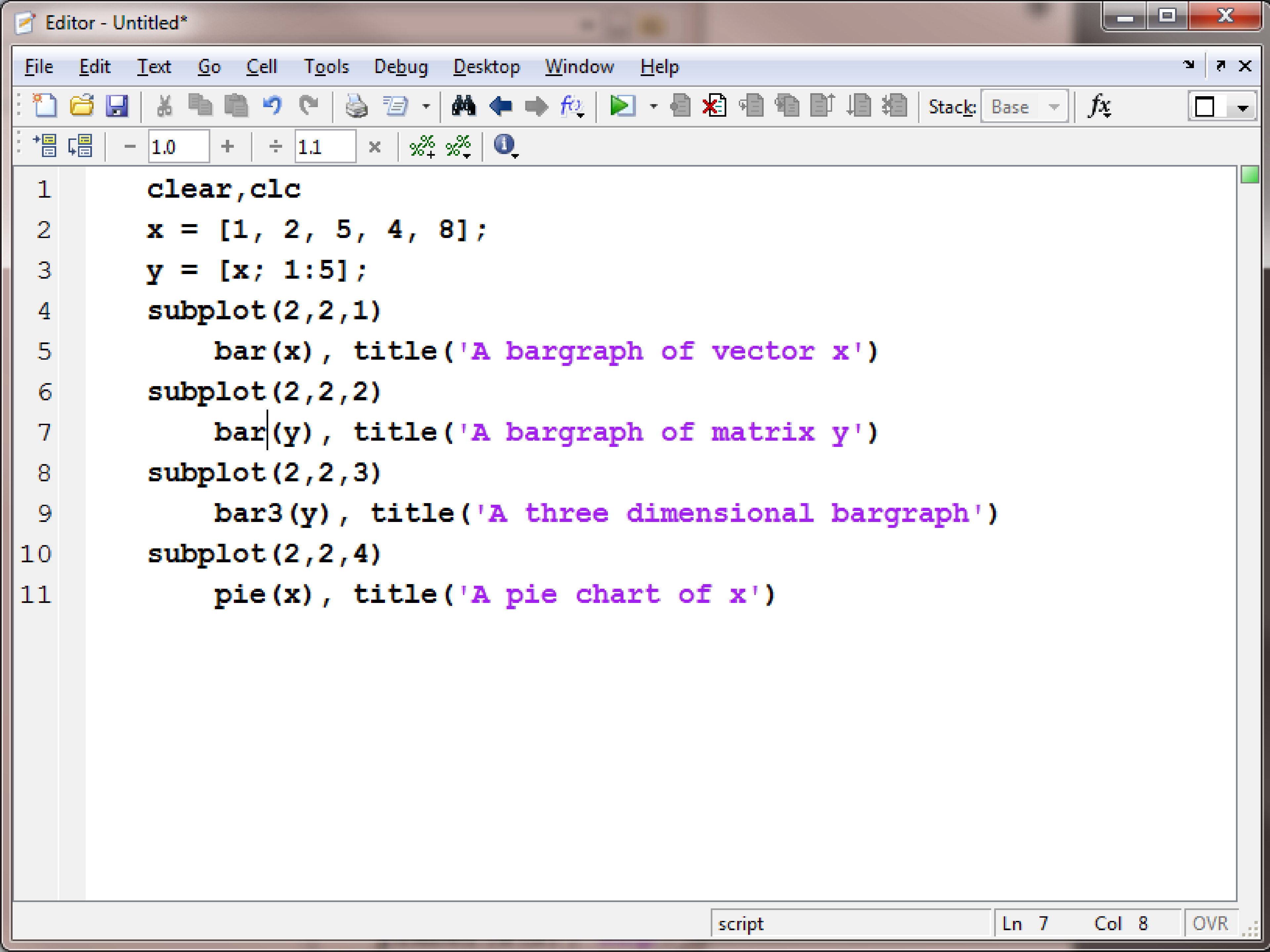






Bar Graphs and Pie Charts

- MATLAB includes a whole family of bar graphs and pie charts
  - bar(x) – vertical bar graph
  - barh(x) – horizontal bar graph
  - bar3(x) – 3-D vertical bar graph
  - bar3h(x) – 3-D horizontal bar graph
  - pie(x) – pie chart
  - pie3(x) – 3-D pie chart



```
clear,clc
x = [1, 2, 5, 4, 8];
y = [x; 1:5];
subplot(2,2,1)
    bar(x), title('A bargraph of vector x')
subplot(2,2,2)
    bar(y), title('A bargraph of matrix y')
subplot(2,2,3)
    bar3(y), title('A three dimensional bargraph')
subplot(2,2,4)
    pie(x), title('A pie chart of x')
```

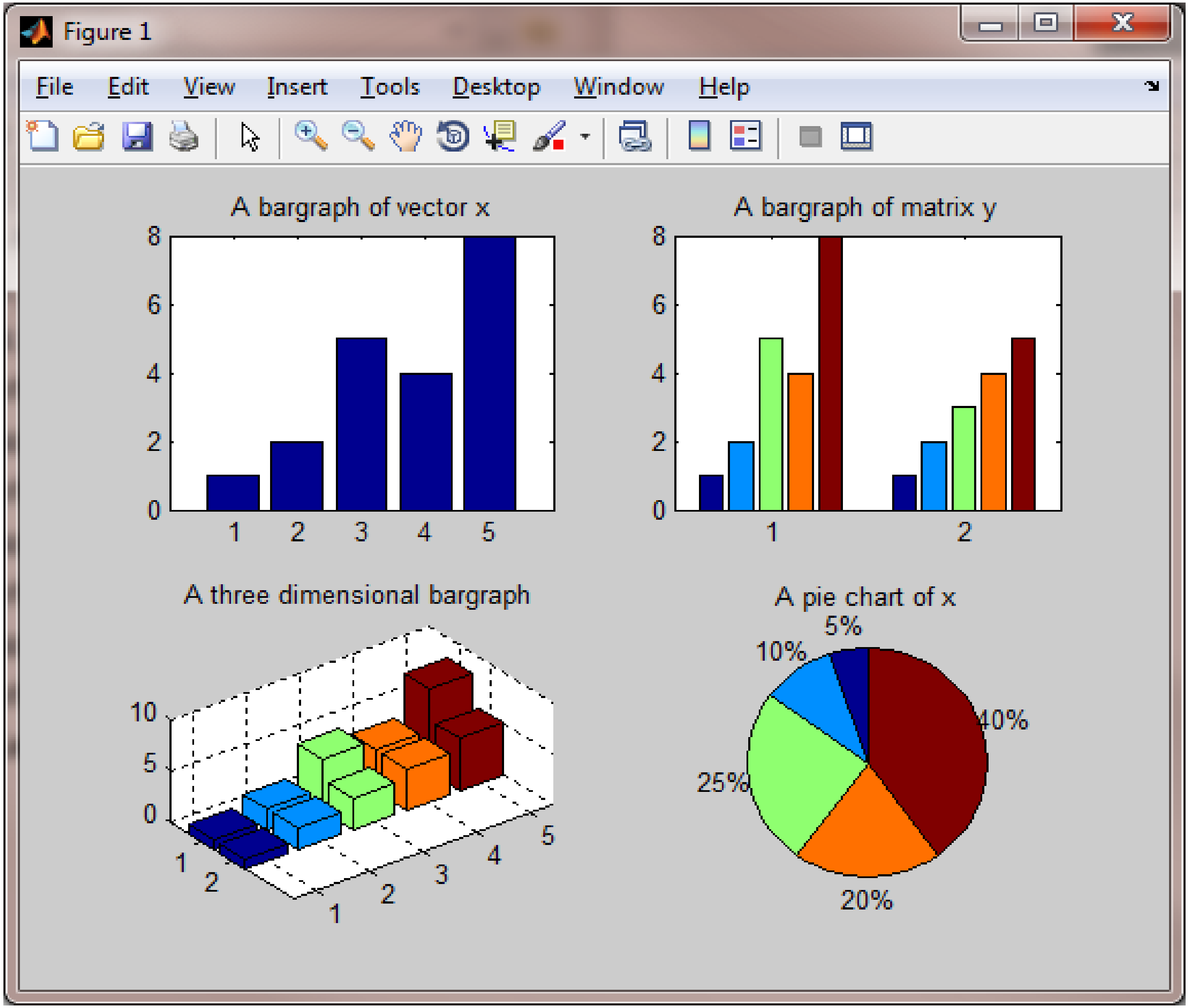
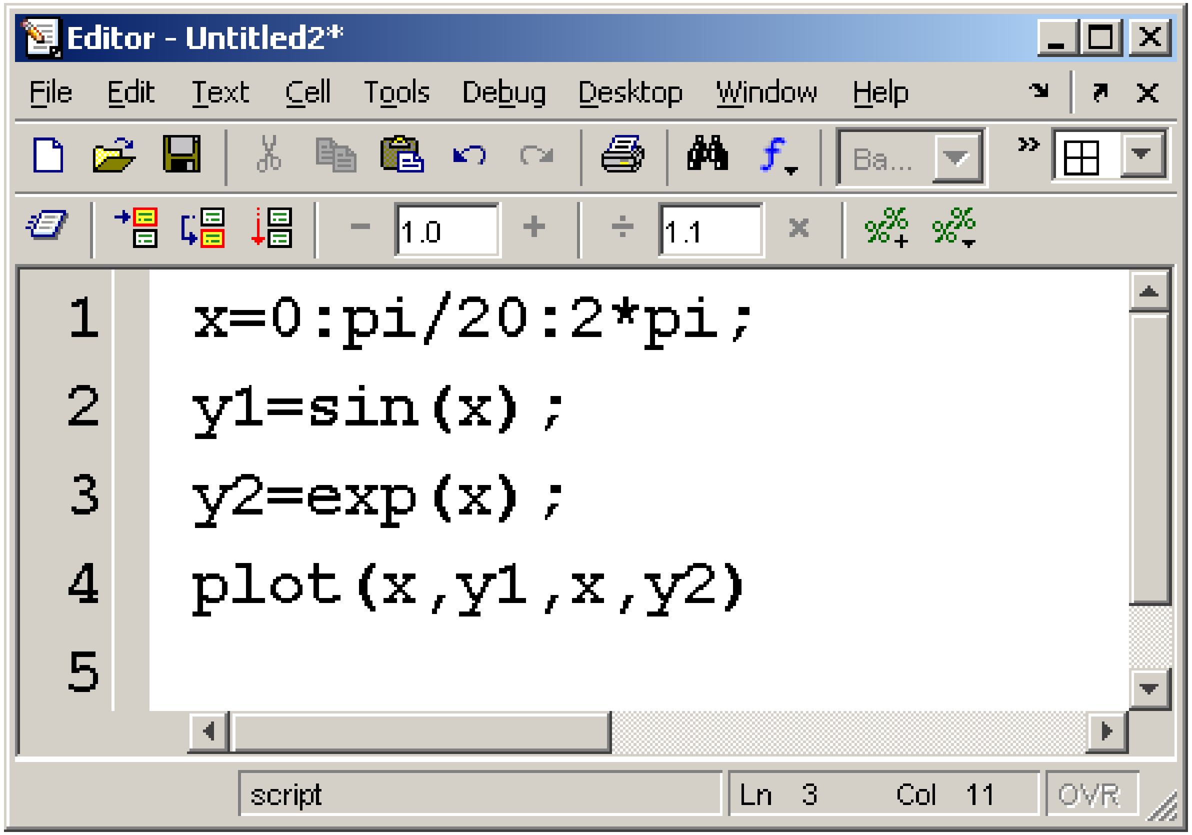


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
A bargraph of vector x
8
6
4
2
0
1
2
3
4
5
A bargraph of matrix y
8
6
4
2
0
1
2
A three dimensional bargraph
10
5
0
1
2
1
2
3
4
5
A pie chart of x
5%
10%
40%
25%
20%

X-Y Graphs with Two Y Axes

- Sometimes it is useful to overlay two *x-y* plots onto the same figure. However, if the order of magnitude of the *y* values are quite different, it may be difficult to see how the data behave.

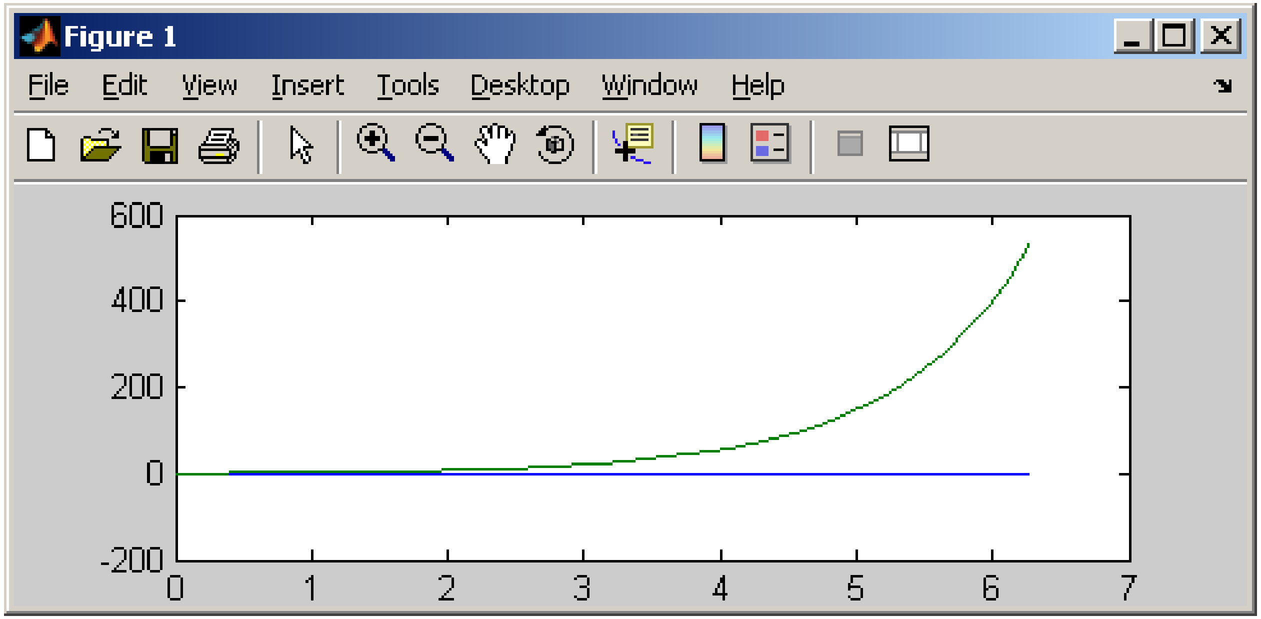
For example

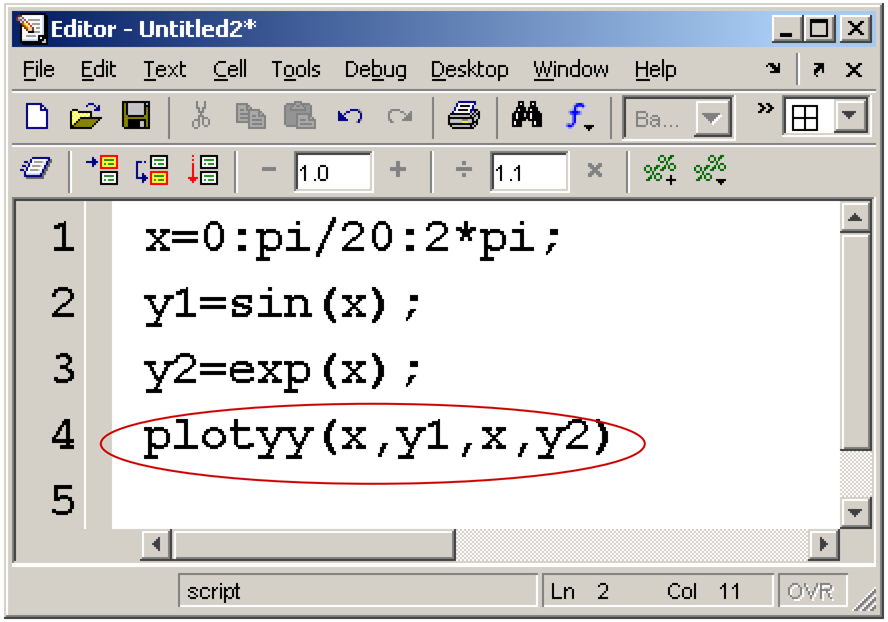


```
x=0:pi/20:2*pi;
y1=sin(x);
y2=exp(x);
plot(x,y1,x,y2)
```

Scaling Depends on the largest value plotted

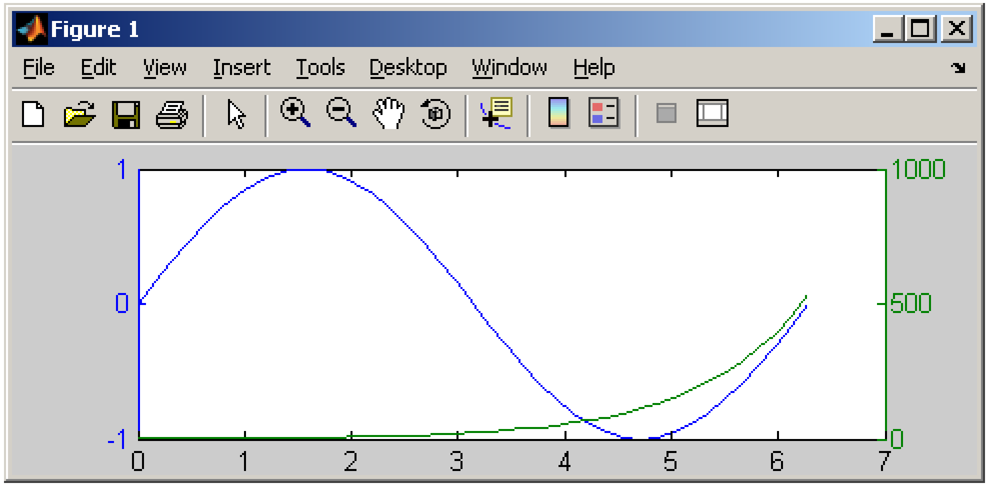
- Its difficult to see how the blue line behaves, because the scale isn't appropriate



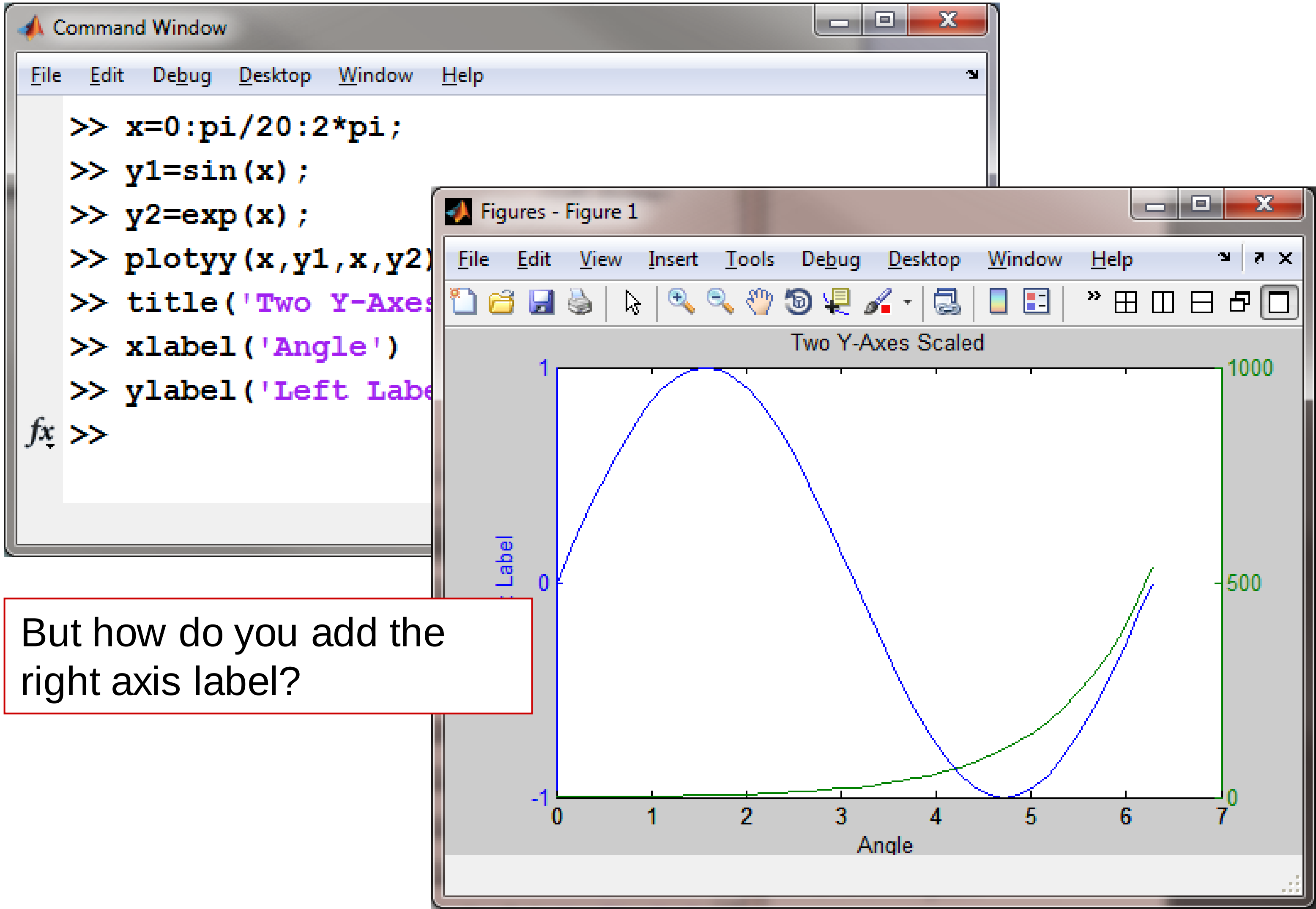


```
x=0:pi/20:2*pi;
y1=sin(x);
y2=exp(x);
plotyy(x,y1,x,y2)
```

The plotyy function allows you to use two scales on a single graph



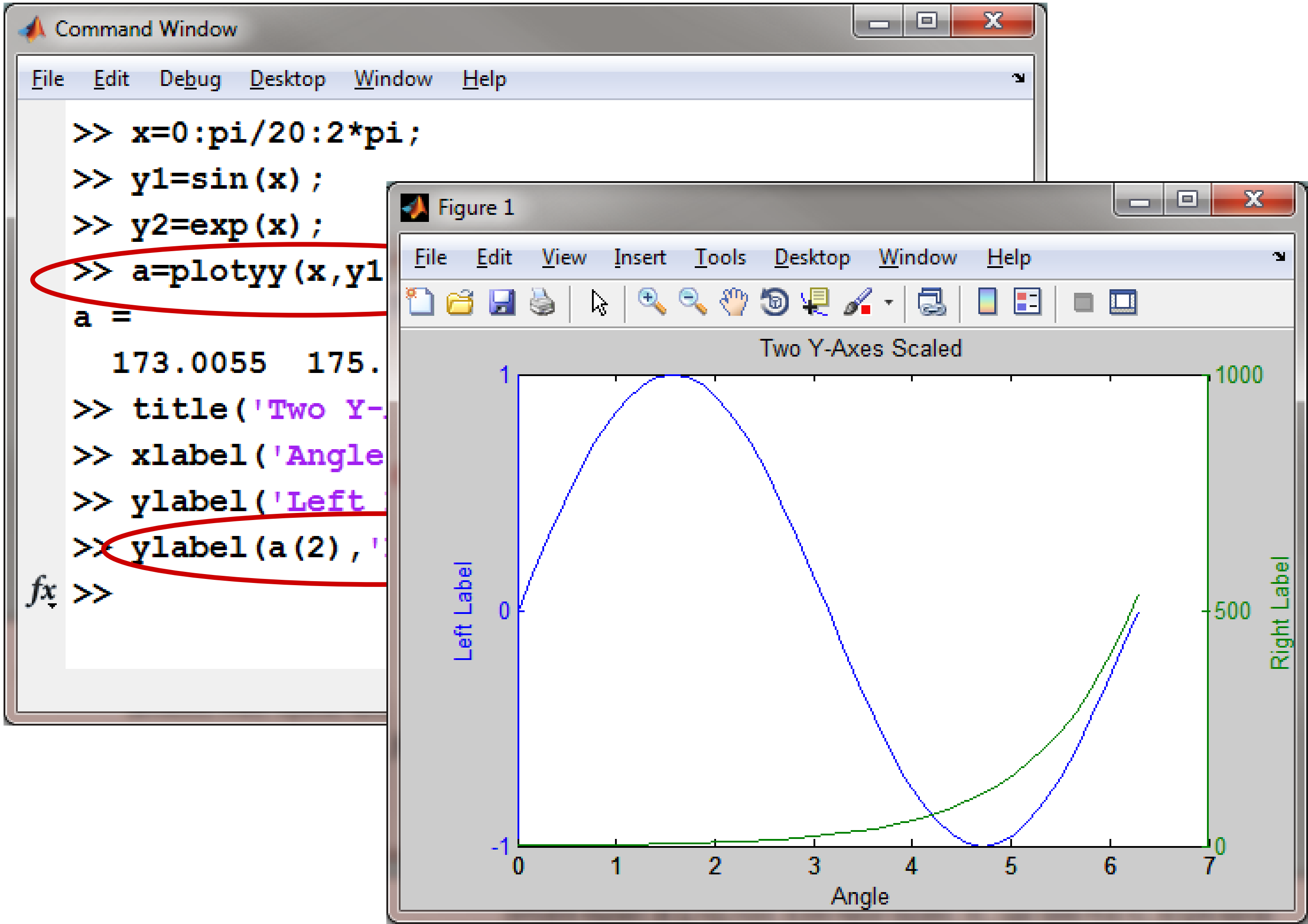
Adding Labels


Command Window
File Edit Debug Desktop Window Help
Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Two Y-Axes Scaled
1
0
-1
1000
500
0
0 1 2 3 4 5 6 7
Angle
Label

```
>> x=0:pi/20:2*pi;
>> y1=sin(x);
>> y2=exp(x);
>> plotyy(x,y1,x,y2)
>> title('Two Y-Axes
>> xlabel('Angle')
>> ylabel('Left Labe
fx >>
```

But how do you add the right axis label?

Give the plot a name – also called a ‘handle’



```
>> x=0:pi/20:2*pi;
>> y1=sin(x);
>> y2=exp(x);
>> a=plotyy(x,y1
a =
  173.0055   175.
>> title('Two Y-
>> xlabel('Angle
>> ylabel('Left '
>> ylabel(a(2),''
>>
```

5.3 Function Plots

- Function plots allow you to use a function as input to a plot command, instead of a set of ordered pairs of x-y values

```
fplot('sin(x)',[-2*pi,2*pi])
```

function input as a string

range of the independent variable – in this case x

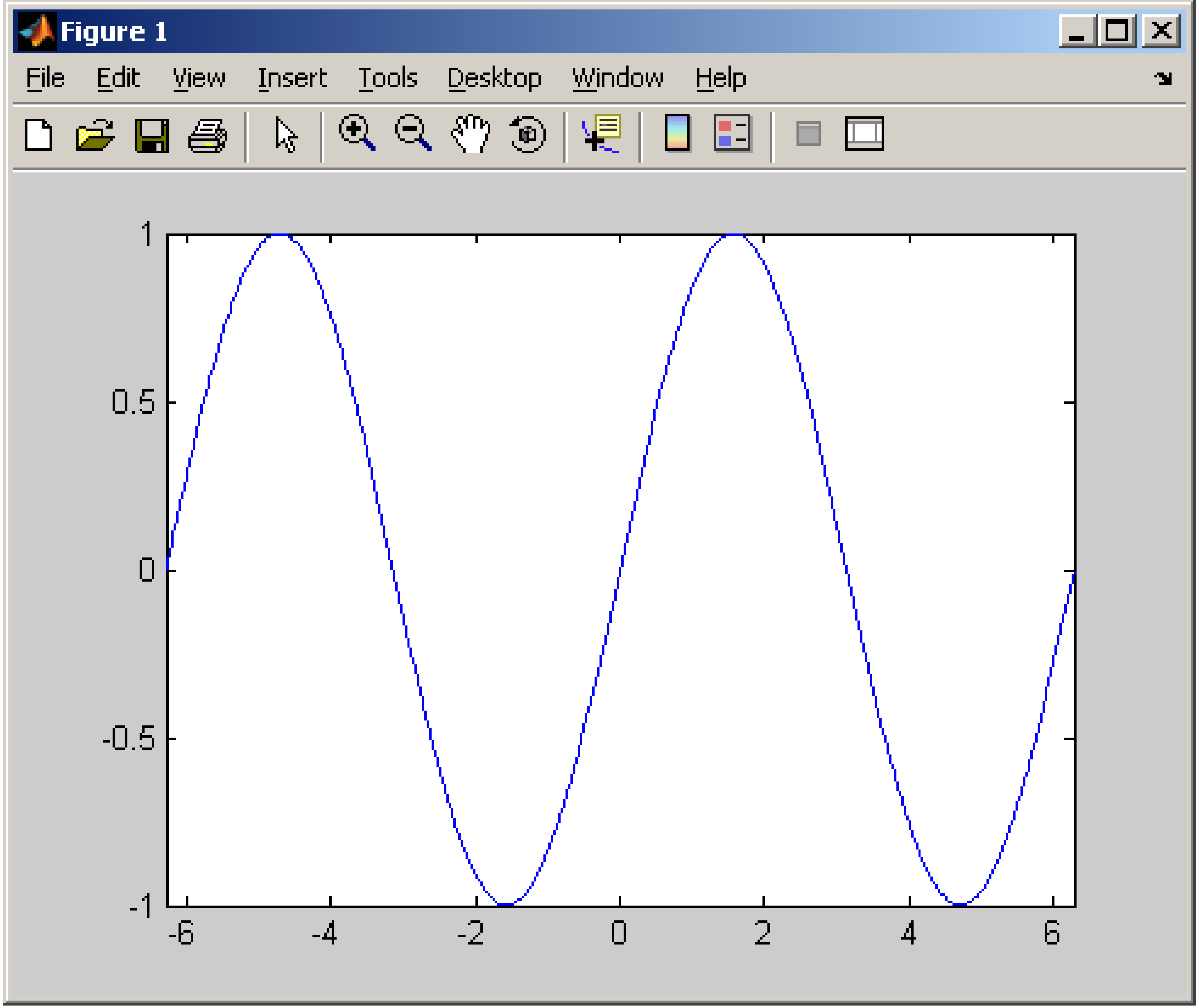


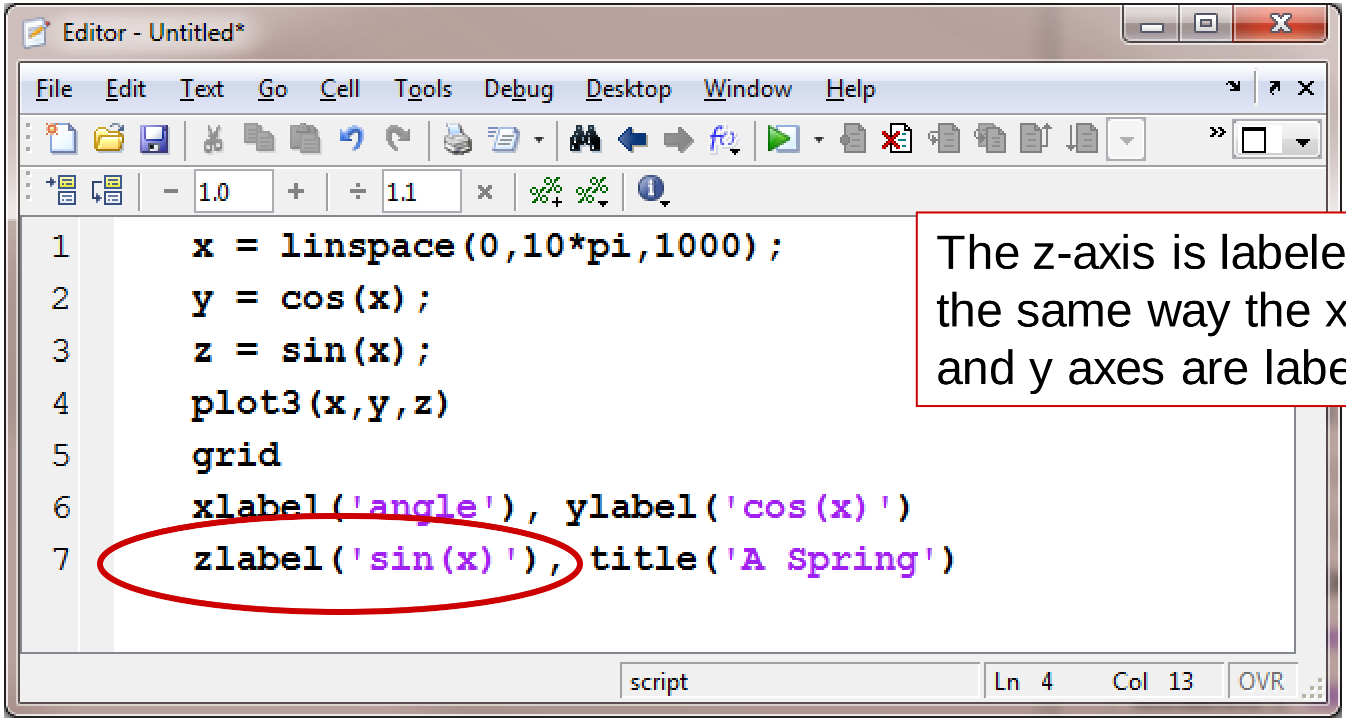
Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
1
0.5
0
-0.5
-1
-6
-4
-2
0
2
4
6

Section 5.4
Three Dimensional Plotting

- Line plots
- Surface plots
- Contour plots

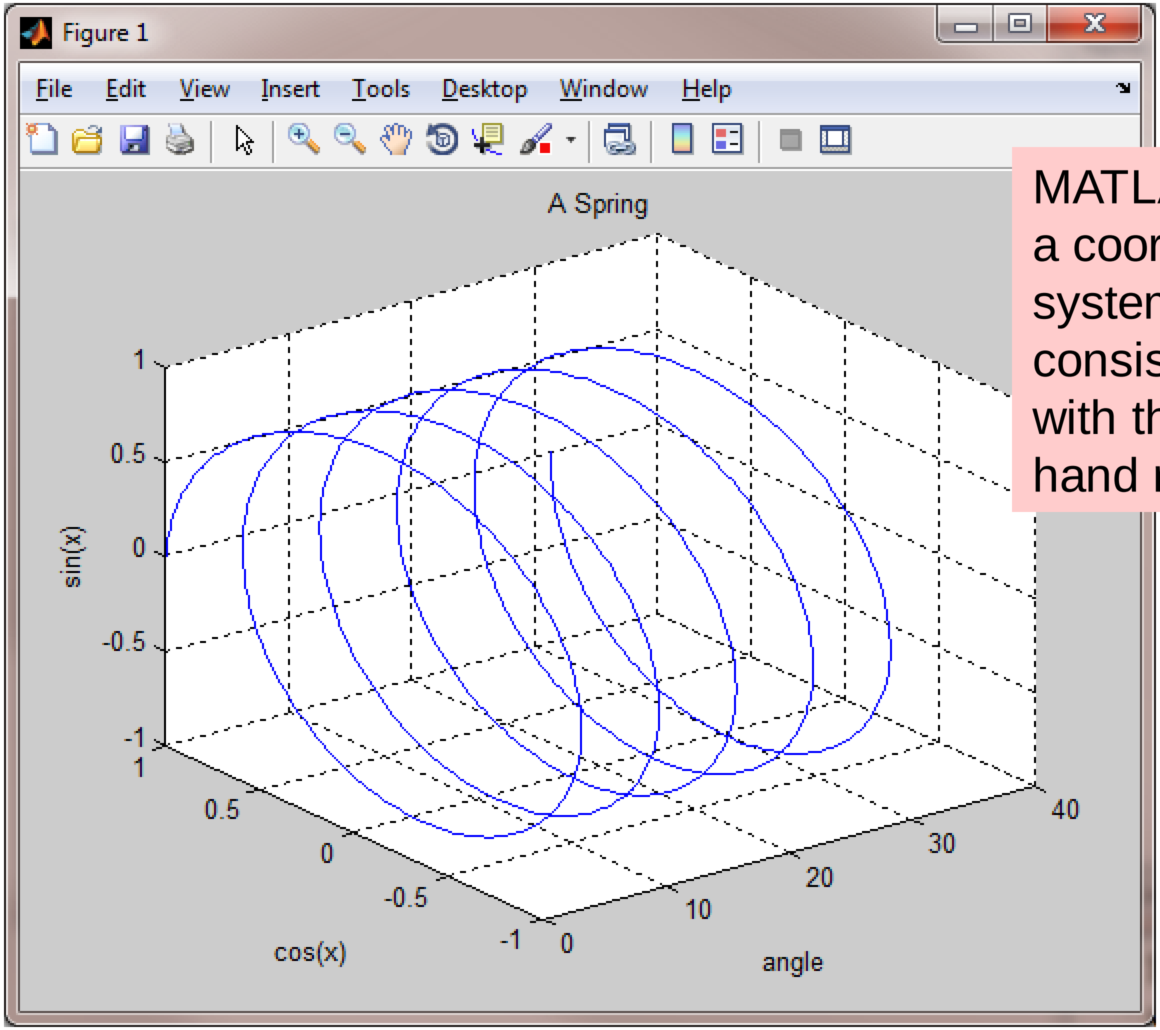
5.4.1 Three Dimensional Line Plots

- These plots require a set of order triples (x-y-z values) as input



```
x = linspace(0,10*pi,1000);
y = cos(x);
z = sin(x);
plot3(x,y,z)
grid
xlabel('angle'), ylabel('cos(x)')
zlabel('sin(x)'), title('A Spring')
```

The z-axis is labeled the same way the x and y axes are labeled



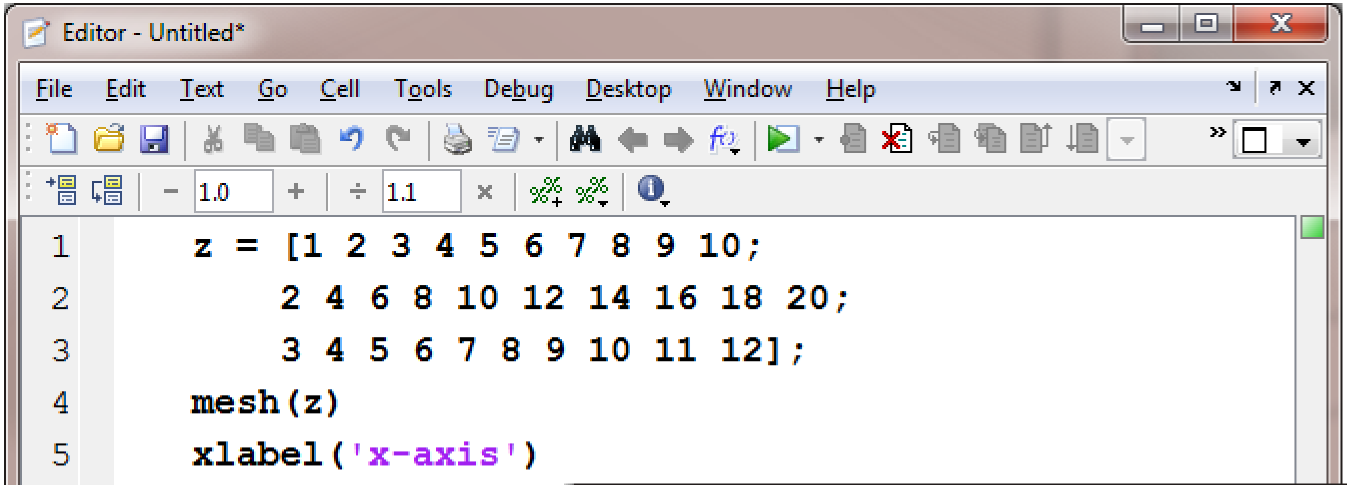
MATLAB uses a coordinate system consistent with the right hand rule

5.4.2 Surface Plots

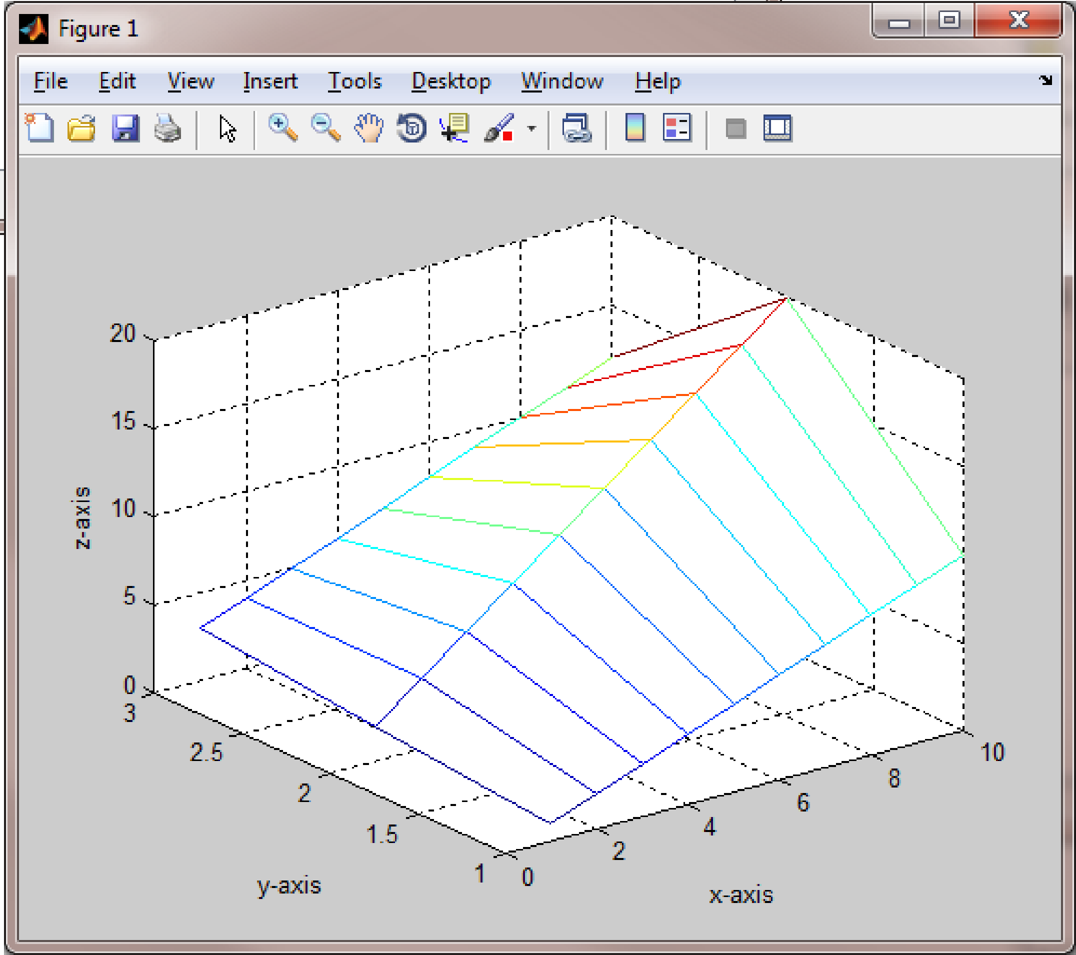
- Represent x-y-z data as a surface
  - mesh - meshplot
  - surf – surface plot

Both Mesh and Surf

- Can be used to good effect with a single two dimensional matrix



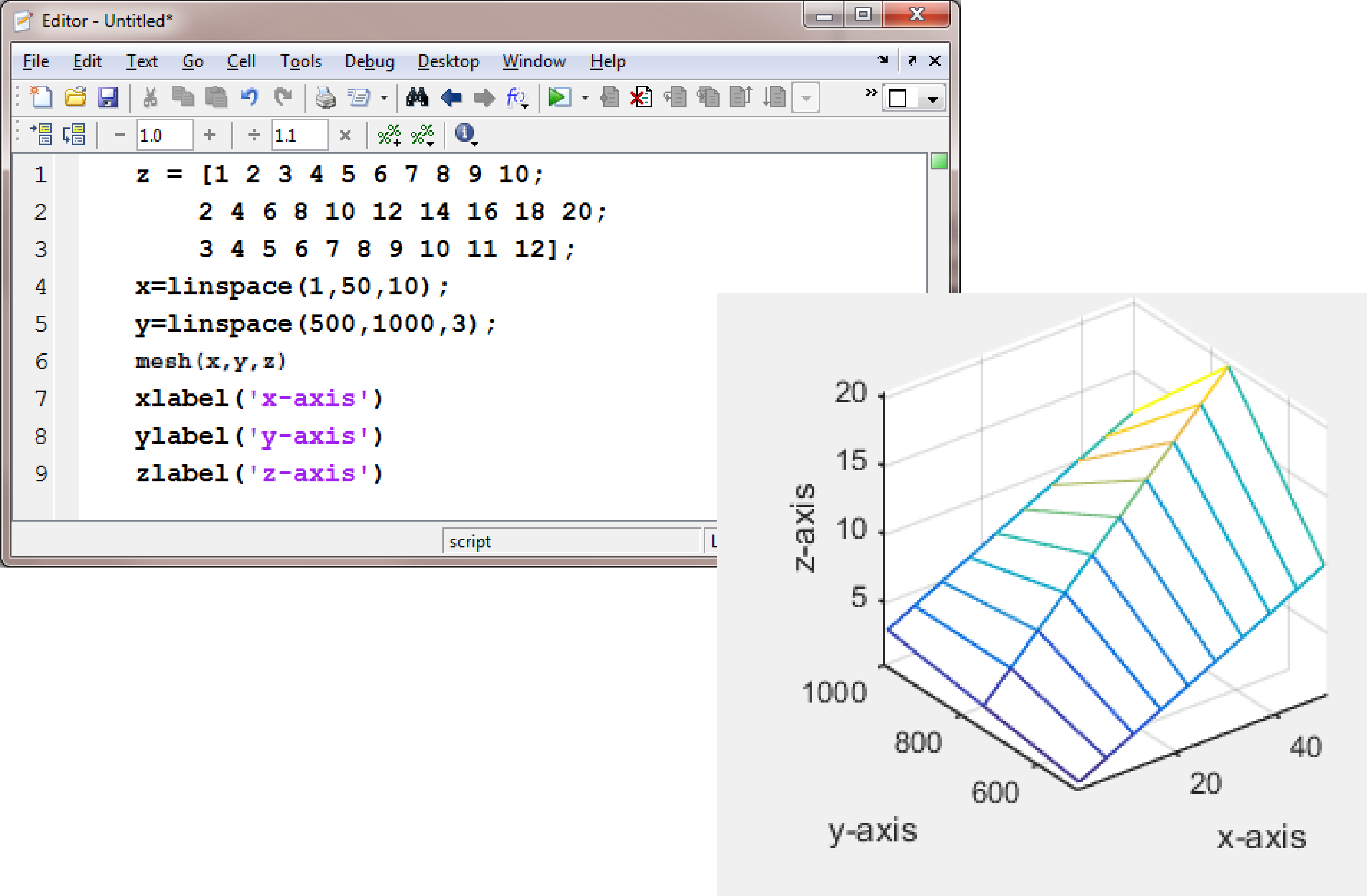
```
z = [1 2 3 4 5 6 7 8 9 10;
     2 4 6 8 10 12 14 16 18 20;
     3 4 5 6 7 8 9 10 11 12];
mesh(z)
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```



The x and y coordinates are the matrix index numbers

Using mesh with 3 variables

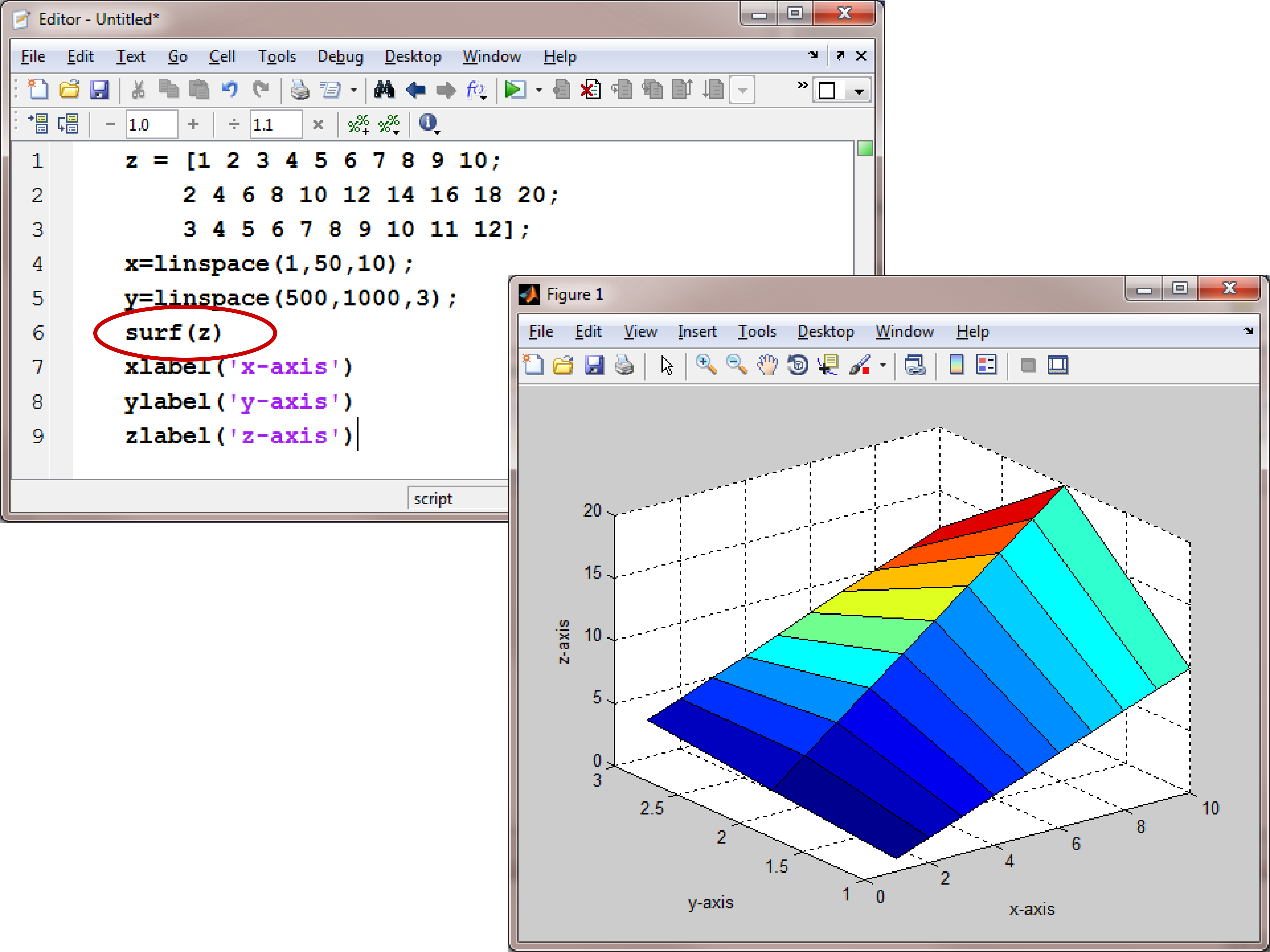
- If we know the values of x and y that correspond to our z values, we can plot against those values instead of the index numbers



```
z = [1 2 3 4 5 6 7 8 9 10;
    2 4 6 8 10 12 14 16 18 20;
    3 4 5 6 7 8 9 10 11 12];
x=linspace(1,50,10);
y=linspace(500,1000,3);
mesh(x,y,z)
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

Surf plots

- surf plots are similar to mesh plots
  - they create a 3-D colored surface instead of an open mesh
  - syntax is the same

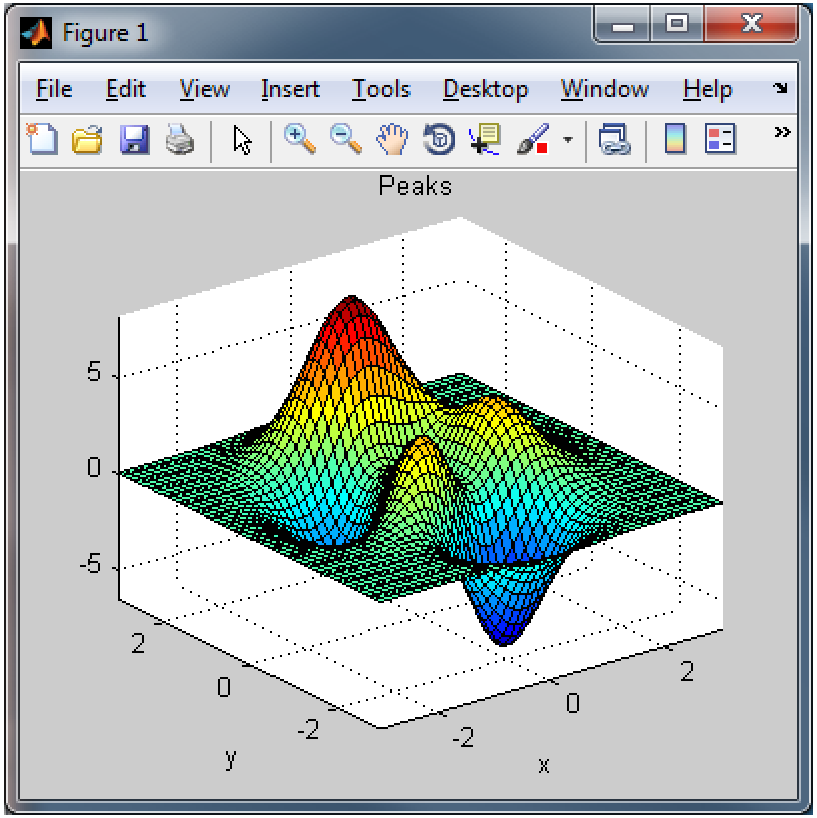


```
z = [1 2 3 4 5 6 7 8 9 10;
     2 4 6 8 10 12 14 16 18 20;
     3 4 5 6 7 8 9 10 11 12];
x=linspace(1,50,10);
y=linspace(500,1000,3);
surf(z)
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

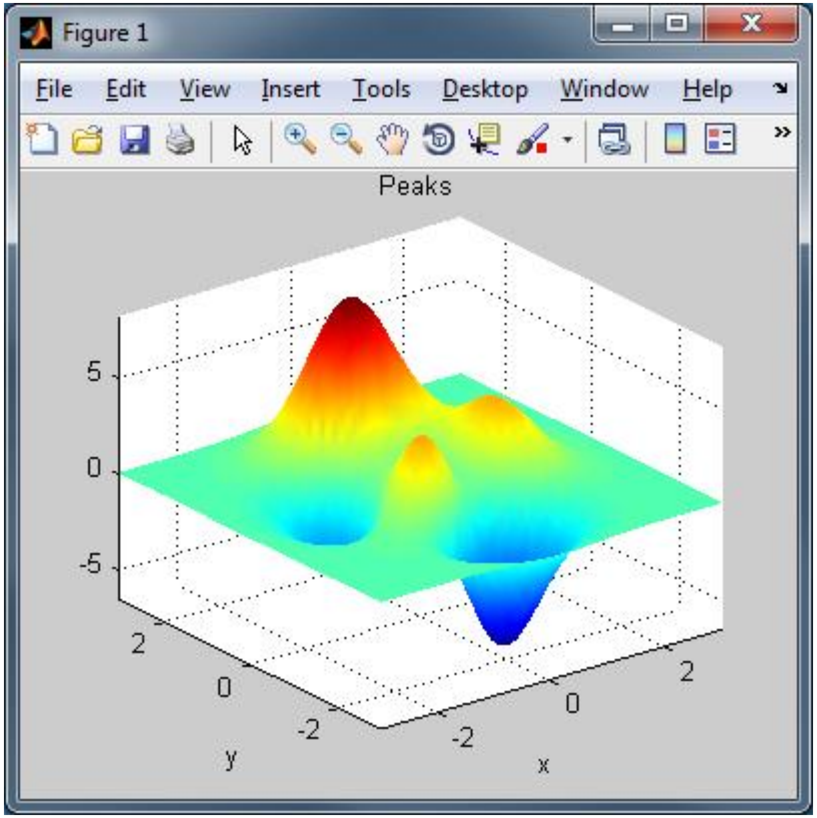
Shading

- There are several shading options
  - shading interp
  - shading flat
  - faceted flat is the default
- You can also adjust the color scheme with the color map function

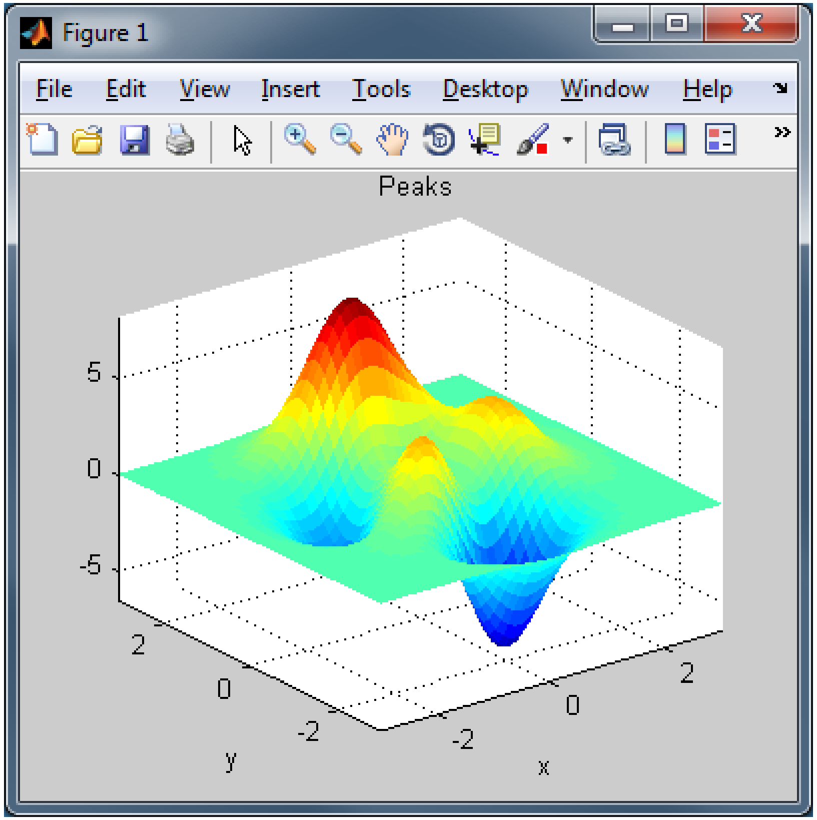
Default shading



Shading Interp



Shading flat



Colormaps

autumn

spring

summer

winter

jet (default)

bone

colorcube

cool

copper

flag

hot

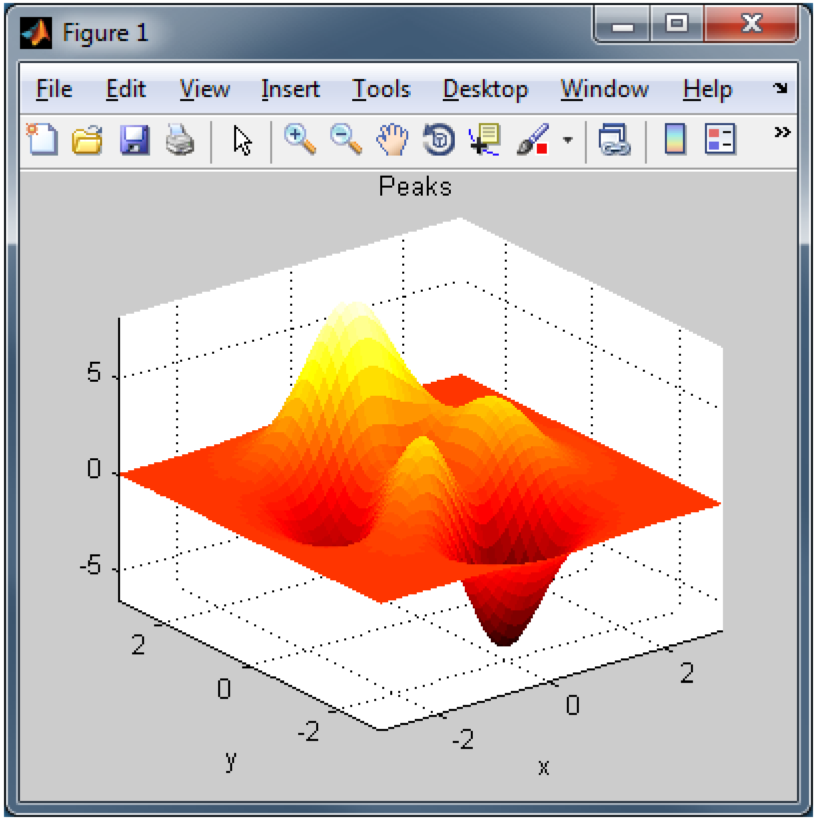
hsv

pink

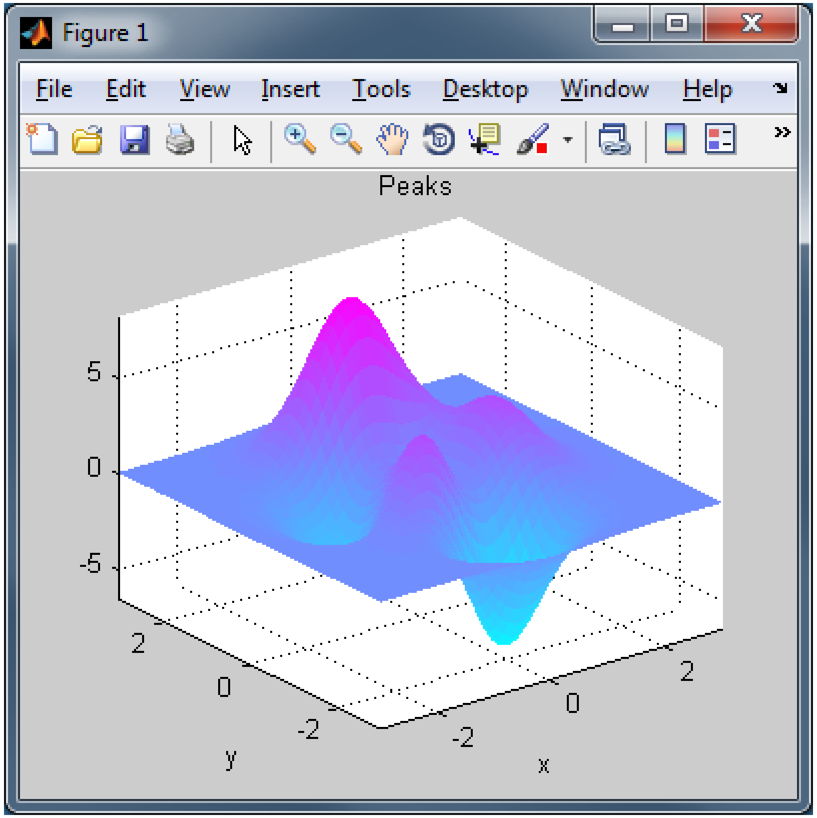
prism

white

Colormap hot



Colormap cool



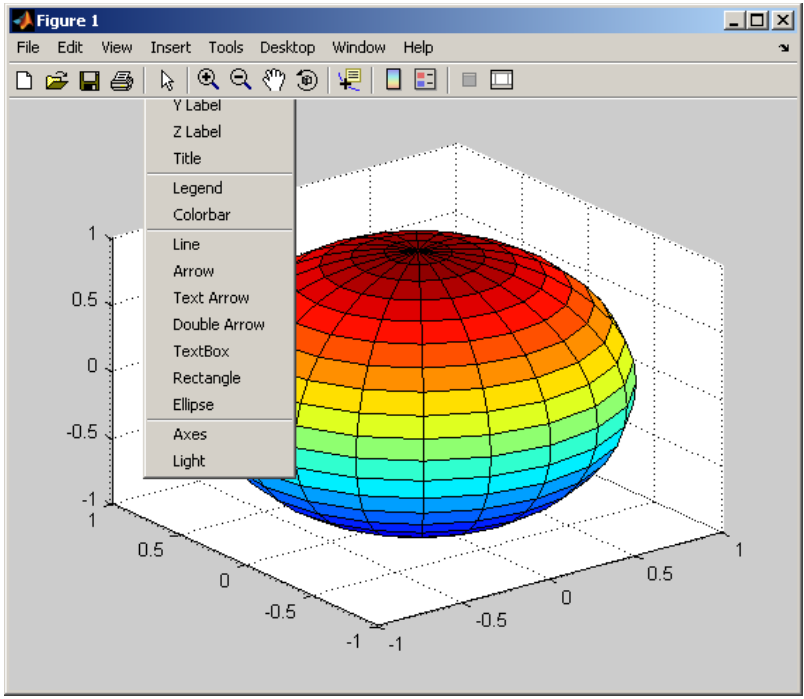
Section 5.5

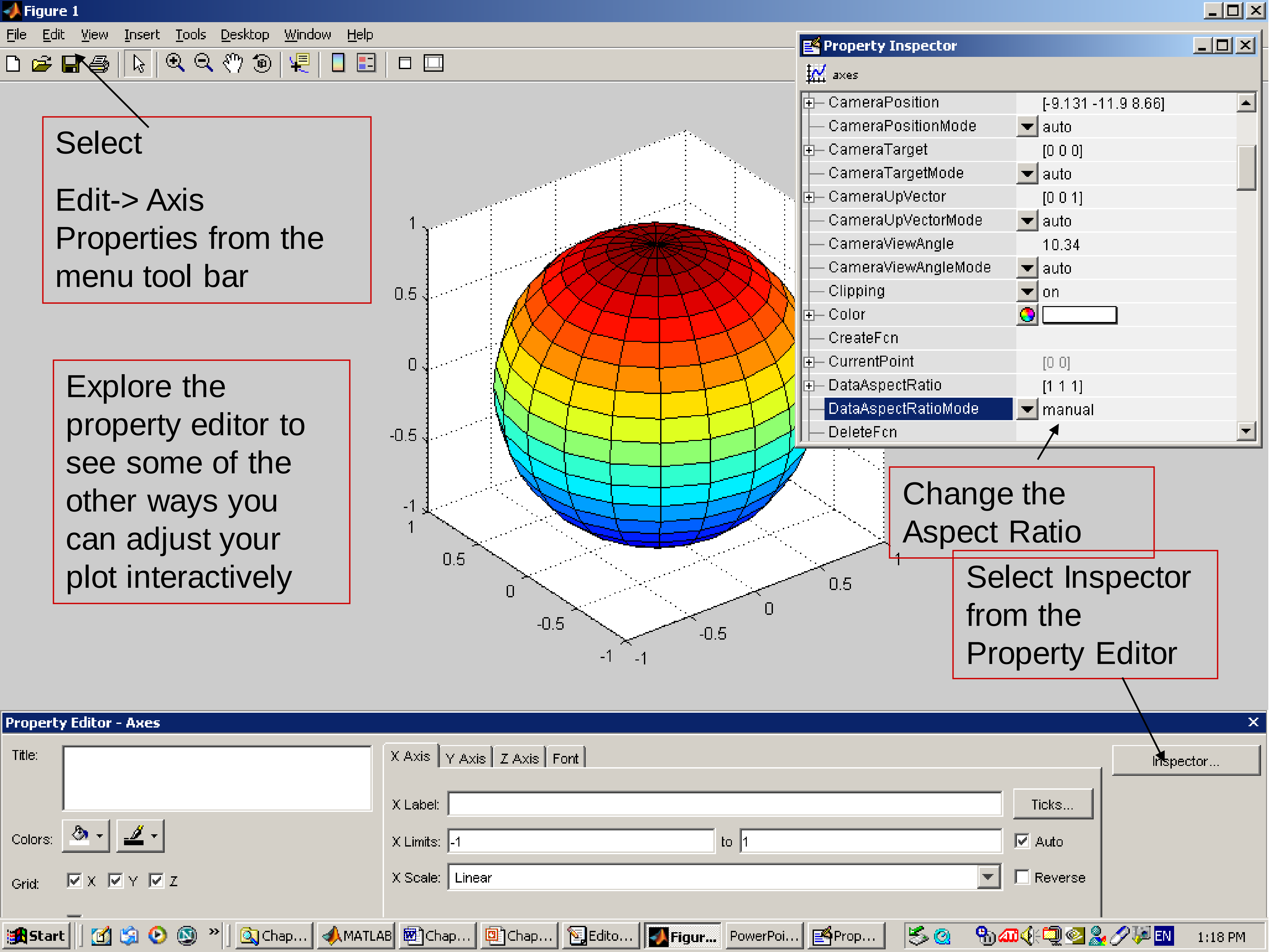
Editing Plots from the Menu Bar

- In addition to controlling the way your plots look by using MATLAB commands, you can also edit a plot once you've created it using the menu bar
- Another demonstration function built into MATLAB is sphere

Once you've created a plot you can adjust it using the menu bar

- In this picture the insert menu has been selected
- Notice you can use it to add labels, legends, a title and other annotations





Select

Edit-> Axis Properties from the menu tool bar

Explore the property editor to see some of the other ways you can adjust your plot interactively

Change the Aspect Ratio

Select Inspector from the Property Editor

Property Inspector

axes

CameraPosition	[-9.131 -11.9 8.66]
CameraPositionMode	auto
CameraTarget	[0 0 0]
CameraTargetMode	auto
CameraUpVector	[0 0 1]
CameraUpVectorMode	auto
CameraViewAngle	10.34
CameraViewAngleMode	auto
Clipping	on
Color	
CreateFcn	
CurrentPoint	[0 0]
DataAspectRatio	[1 1 1]
DataAspectRatioMode	manual
DeleteFcn	

Title:

Colors:

Grid: ☑ X ☑ Y ☑ Z

X Label:

X Limits: -1 to 1 ☑ Auto

X Scale: Linear ☐ Reverse

- If you adjust a figure interactively, you’ll lose your improvements when you rerun your program

Saving your plots

- Rerun your M-file to recreate a plot
- Save the figure from the file menu using the save as… option
  - You'll be presented with several choices of file format such as
    - jpeg
    - emg (enhanced metafile) etc
- Right-click on the figure and select copy – then paste it into another document

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 6
(User Defined Functions)

Zeyad T. Shareef

Objectives

- Create and use MATLAB functions with both single and multiple inputs and outputs
- Learn how to store and access functions in a user defined toolbox
- Create and use anonymous functions
- Create and use function handles

Section 6.1
Creating Function M-files

- User defined functions are stored as separate M-files
- To use them, they must be in the current directory

Syntax

- All functions have a similar syntax, whether they are built-in functions or user-defined functions
  - Name
  - Input
  - Result

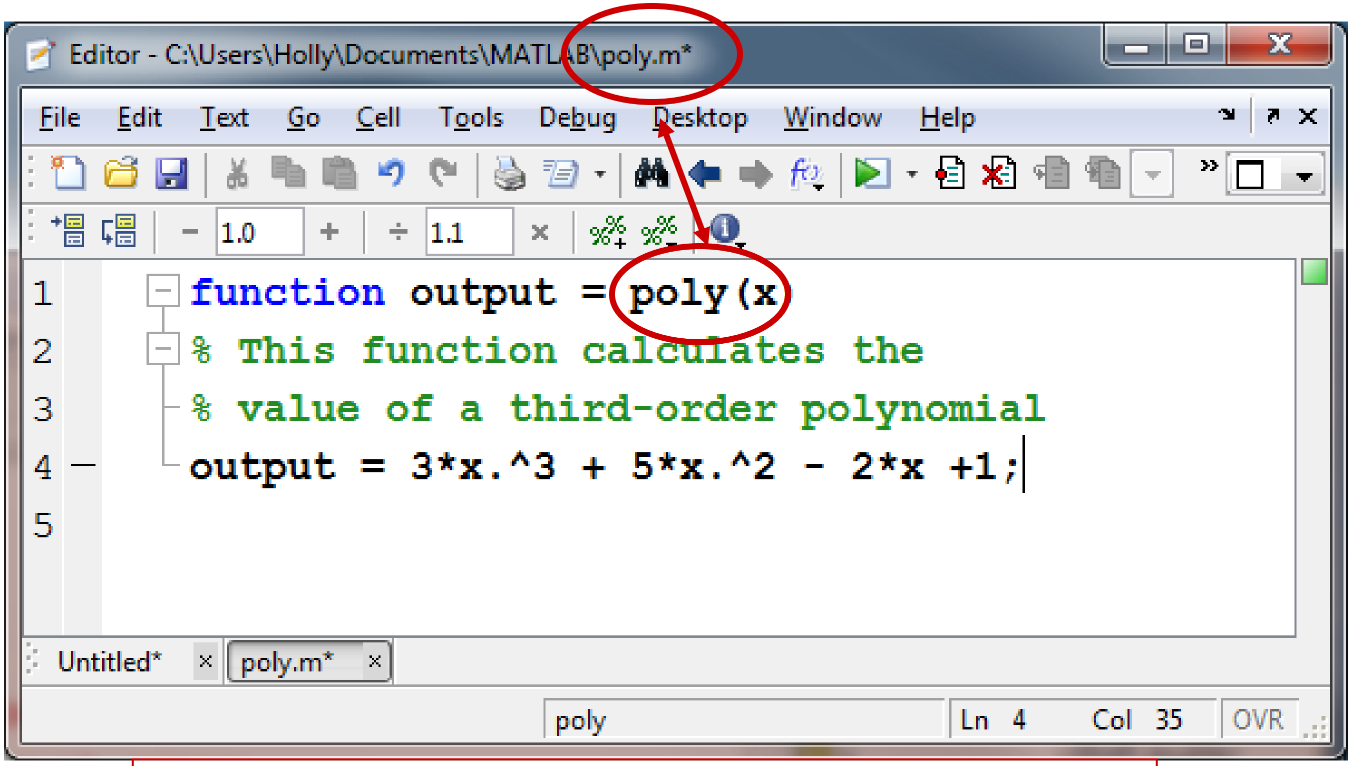
A=cos(x)

User-defined functions must start with a function definition line

- The line contains…
  - The word ‘function’
  - A variable that defines the function output
  - A function name
  - A variable used for the input argument

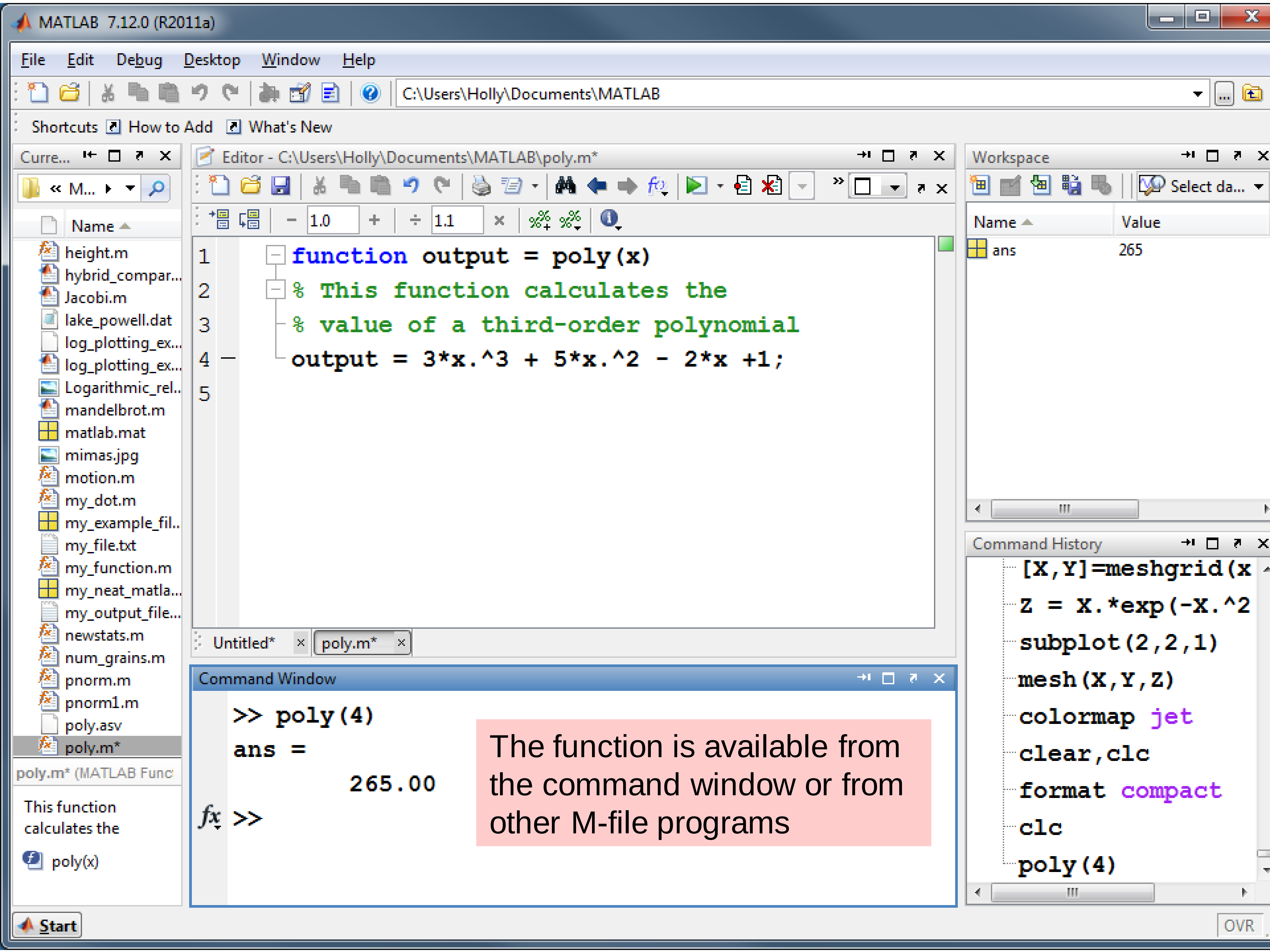
```
function output = poly(x)
```

A simple function



```
function output = poly(x)
% This function calculates the
% value of a third-order polynomial
output = 3*x.^3 + 5*x.^2 - 2*x +1;
```

The function name must be the same as the file name



```
function output = poly(x)
% This function calculates the
% value of a third-order polynomial
output = 3*x.^3 + 5*x.^2 - 2*x +1;
```

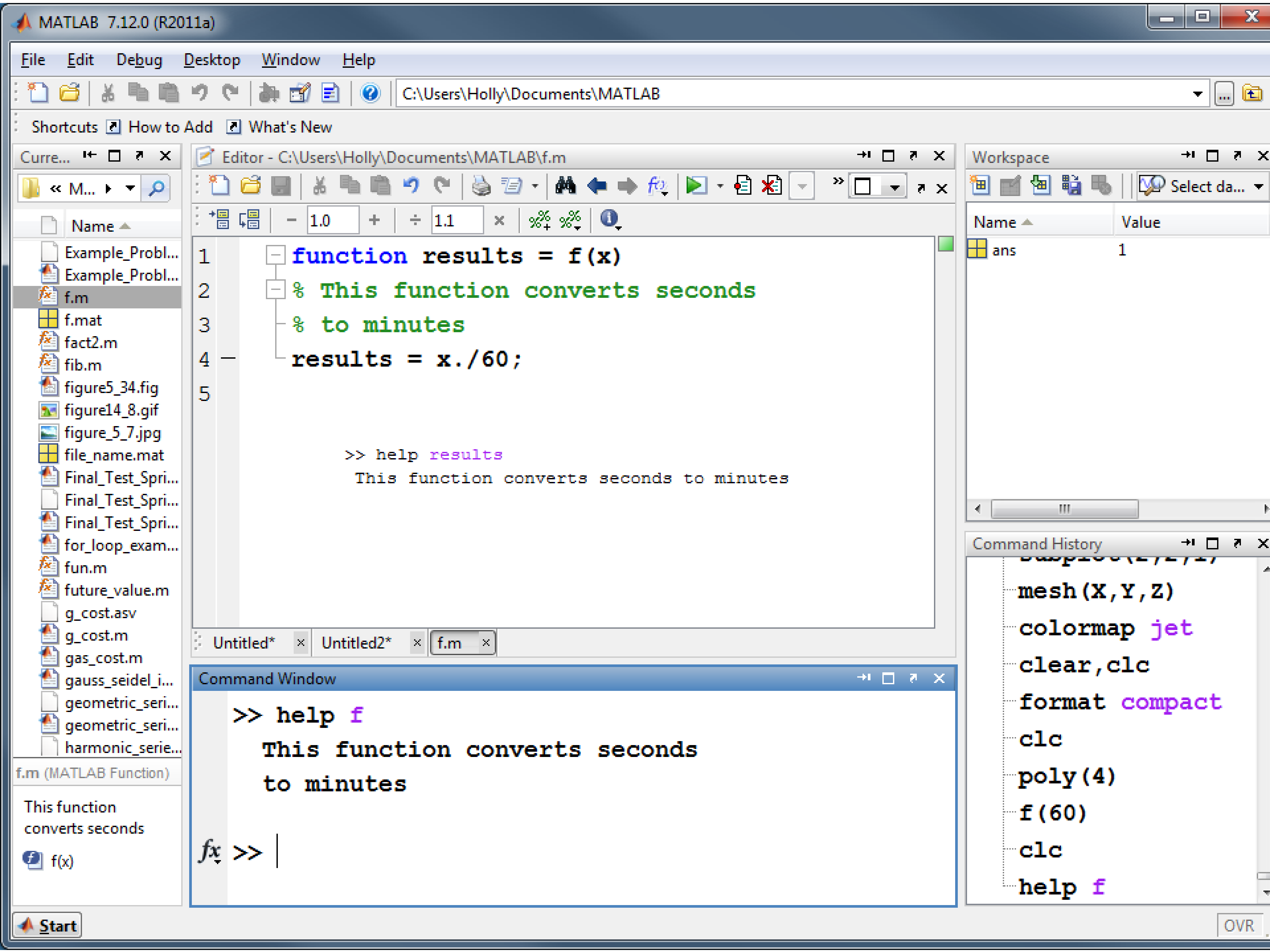
```
>> poly(4)
ans =
          265.00
>>
```

The function is available from the command window or from other M-file programs

```
[X,Y]=meshgrid(x
Z = X.*exp(-X.^2
subplot(2,2,1)
mesh(X,Y,Z)
colormap jet
clear,clc
format compact
clc
poly(4)
```

Comments

- You should comment functions liberally, just as you would any computer code
- The comment lines immediately after the first line are returned when you query the help function



```
function results = f(x)
% This function converts seconds
% to minutes
results = x./60;
```

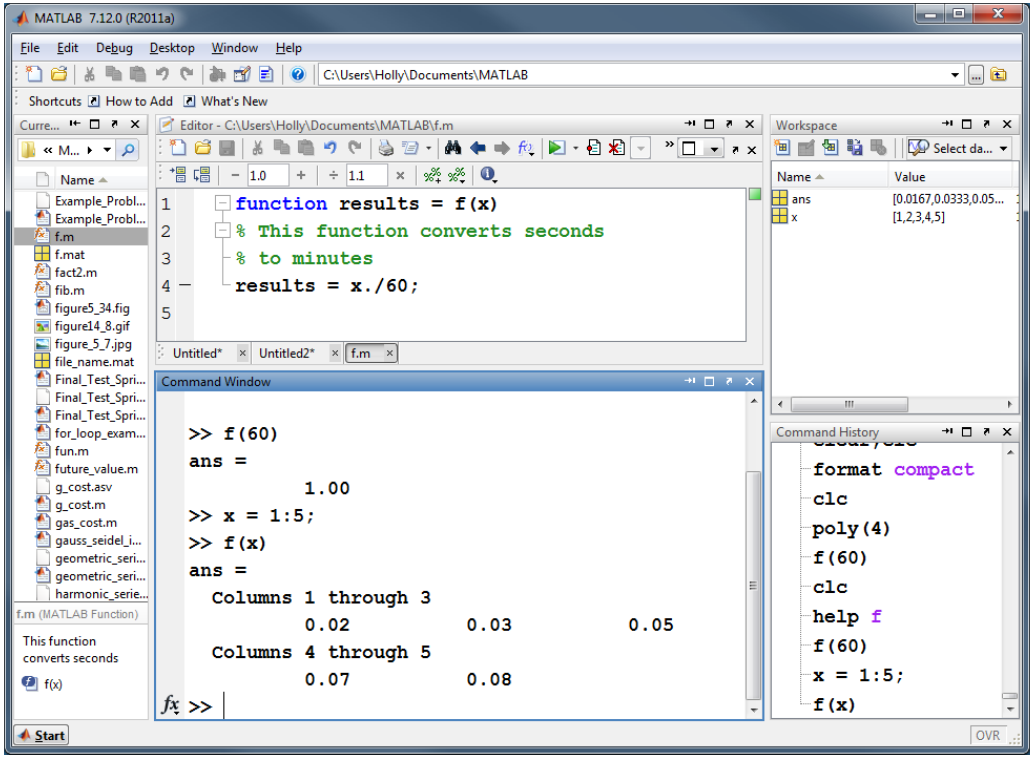
```
>> help results
 This function converts seconds to minutes
```

```
>> help f
  This function converts seconds
  to minutes

>>
```

```
mesh(X,Y,Z)
colormap jet
clear,clc
format compact
clc
poly(4)
f(60)
clc
help f
```

Functions can accept…

- numeric values
- variables
- scalars
- arrays



```
function results = f(x)
% This function converts seconds
% to minutes
results = x./60;
```

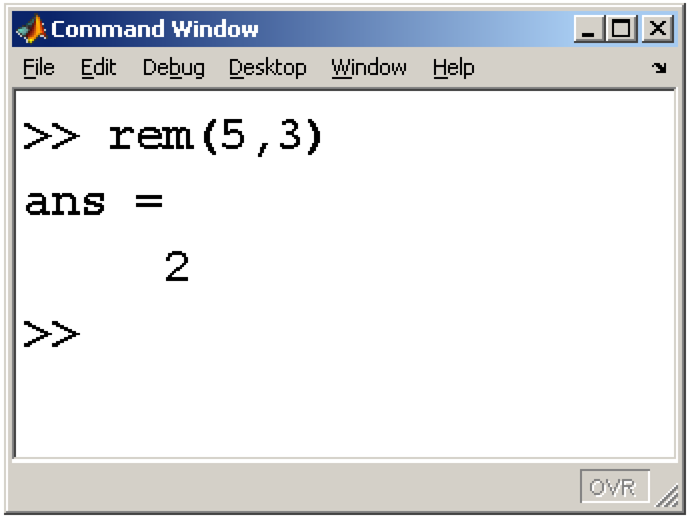
Name	Value
ans	[0.0167,0.0333,0.05...
x	[1,2,3,4,5]

```
>> f(60)
ans =
          1.00
>> x = 1:5;
>> f(x)
ans =
  Columns 1 through 3
          0.02          0.03          0.05
  Columns 4 through 5
          0.07          0.08
>>
```

```
format compact
clc
poly(4)
f(60)
clc
help f
f(60)
x = 1:5;
f(x)
```

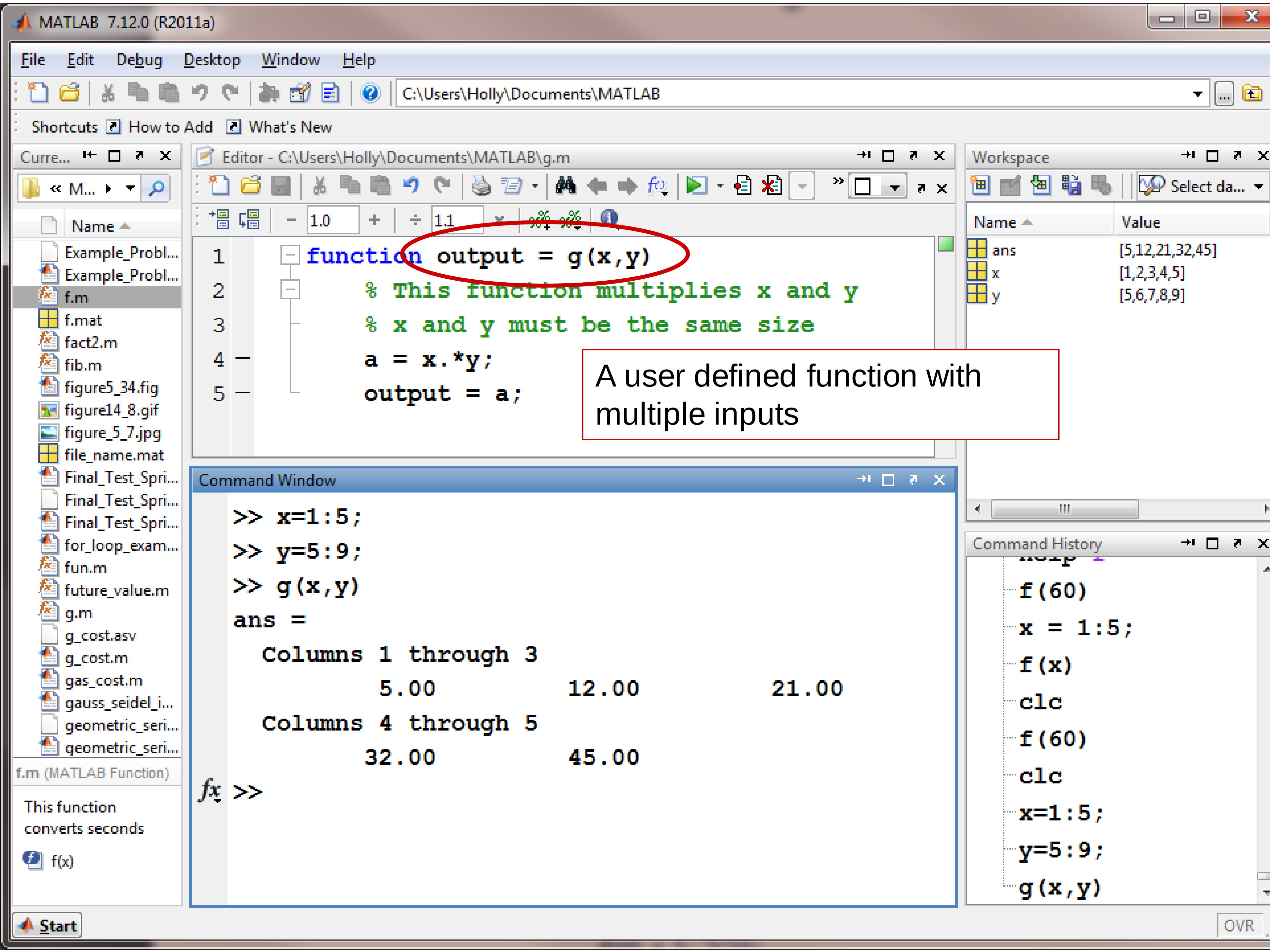
Functions with Multiple Inputs and Outputs

- Recall the remainder function



```
>> rem(5,3)
ans =
     2
>>
```

This function has two inputs



```
function output = g(x,y)
    % This function multiplies x and y
    % x and y must be the same size
    a = x.*y;
    output = a;
```

A user defined function with multiple inputs

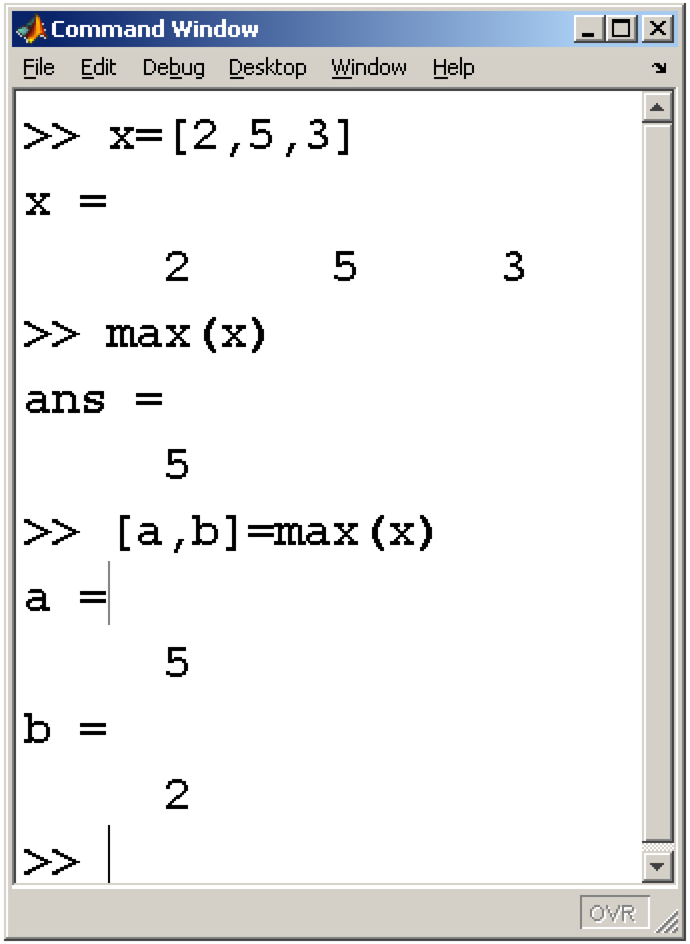
```
>> x=1:5;
>> y=5:9;
>> g(x,y)
ans =
  Columns 1 through 3
          5.00         12.00         21.00
  Columns 4 through 5
         32.00         45.00
>>
```

Name	Value
ans	[5,12,21,32,45]
x	[1,2,3,4,5]
y	[5,6,7,8,9]

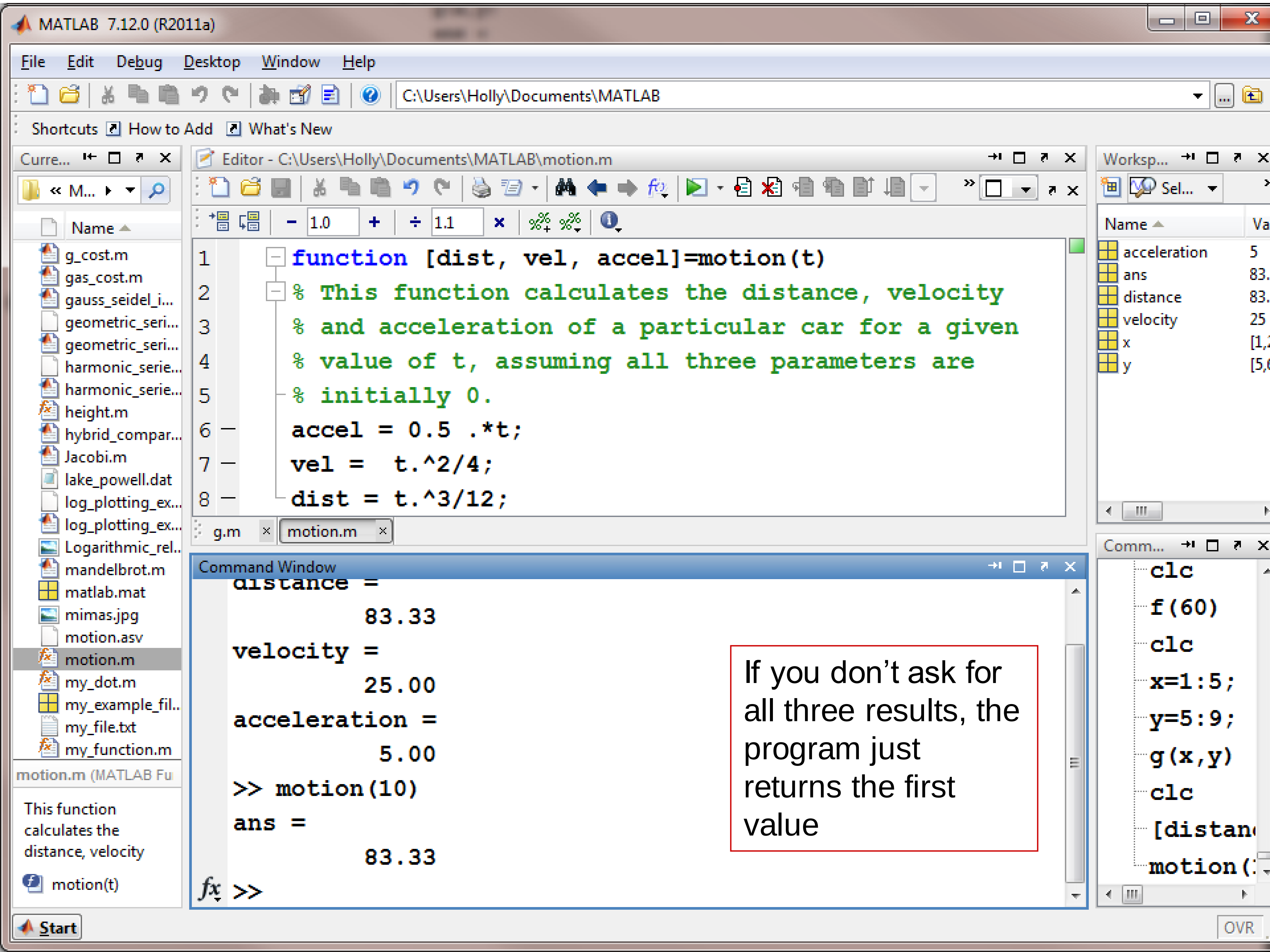
```
f(60)
x = 1:5;
f(x)
clc
f(60)
clc
x=1:5;
y=5:9;
g(x,y)
```

Functions with Multiple Outputs

- Recall the max function
- It returns two results



```
>> x=[2,5,3]
x =
     2     5     3
>> max(x)
ans =
     5
>> [a,b]=max(x)
a =
     5
b =
     2
>>
```



```
function [dist, vel, accel]=motion(t)
% This function calculates the distance, velocity
% and acceleration of a particular car for a given
% value of t, assuming all three parameters are
% initially 0.
accel = 0.5 .*t;
vel =  t.^2/4;
dist = t.^3/12;
```

Name	Va
acceleration	5
ans	83.
distance	83.
velocity	25
x	[1,
y	[5,

```
distance =

       83.33

velocity =

       25.00

acceleration =

        5.00

>> motion(10)

ans =

       83.33

>>
```

This function calculates the distance, velocity

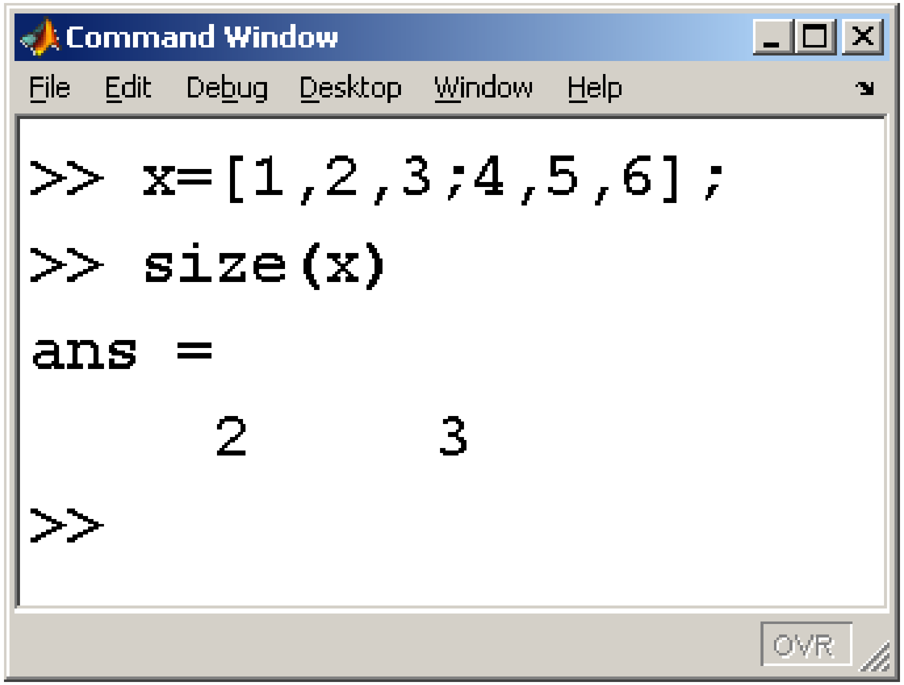
motion(t)

```
clc
f(60)
clc
x=1:5;
y=5:9;
g(x,y)
clc
[distan
motion(
```

If you don’t ask for all three results, the program just returns the first value

Recall the size function

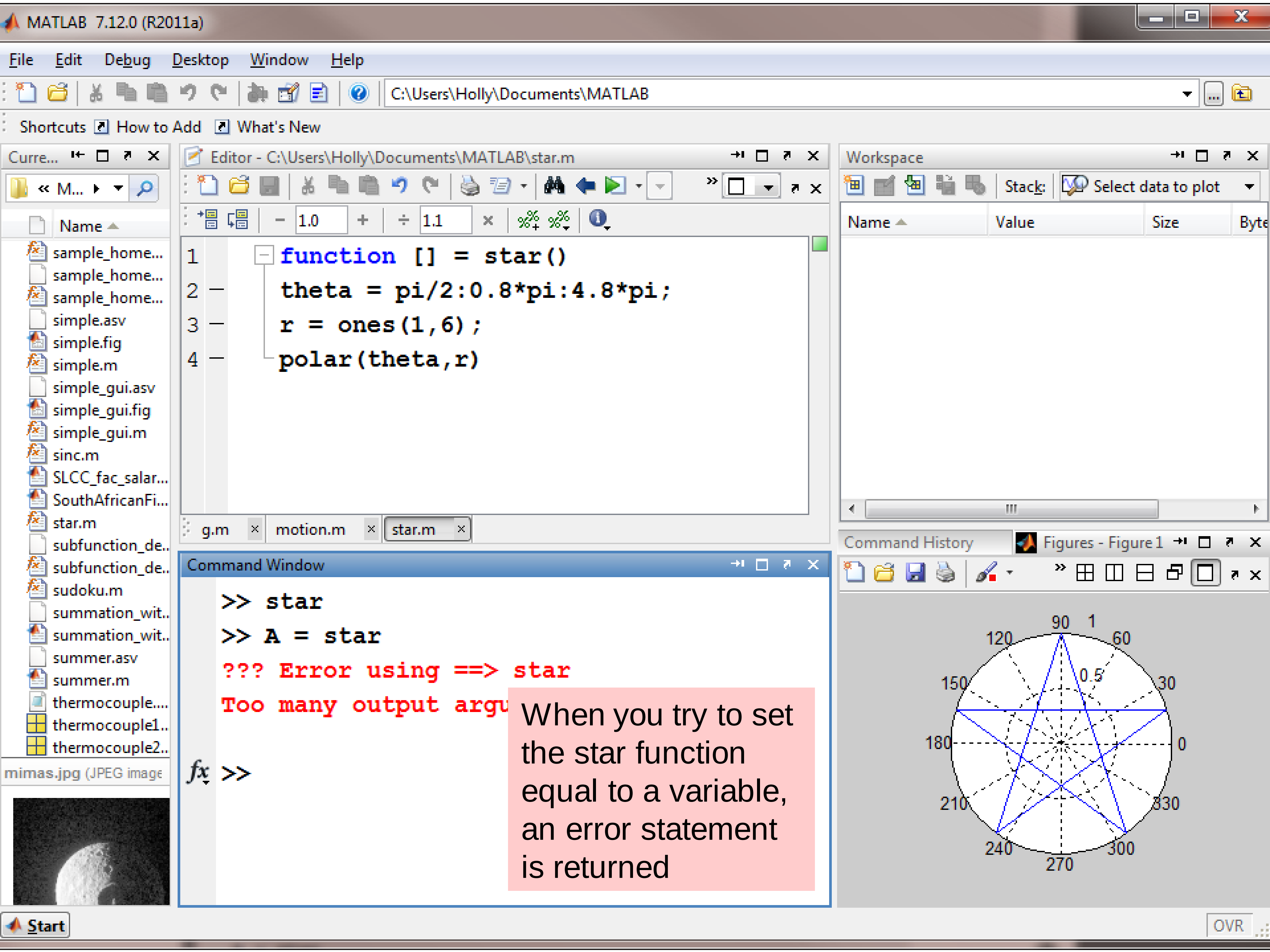
At first this function looks like it returns two values – but it really only returns a single array with two elements



```
>> x=[1,2,3;4,5,6];
>> size(x)
ans =
     2     3
>>
```

Functions with no input or no output

- It isn’t always necessary to define an output



```
function [] = star()
theta = pi/2:0.8*pi:4.8*pi;
r = ones(1,6);
polar(theta,r)
```

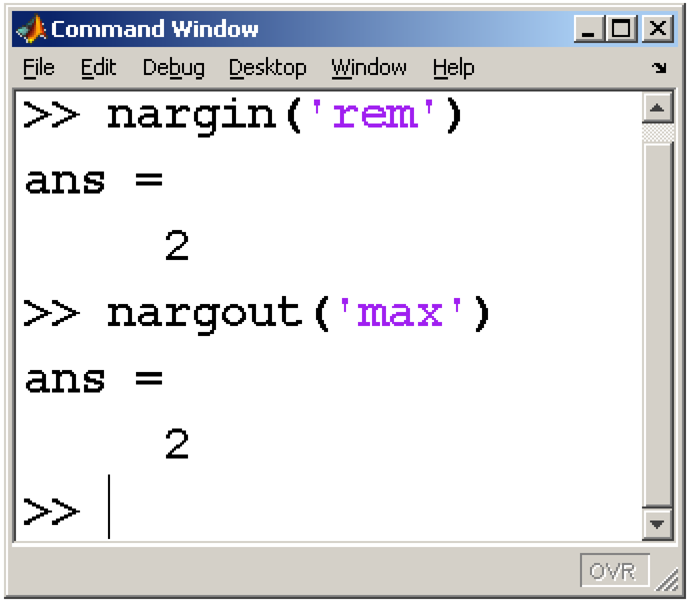
```
>> star
>> A = star
??? Error using ==> star
Too many output argu

>>
```

When you try to set the star function equal to a variable, an error statement is returned

Determining the number of input and output arguments

- nargin
  - determines the number of input arguments
- nargout
  - determines the number of output arguments

The input to these functions is represented using a string



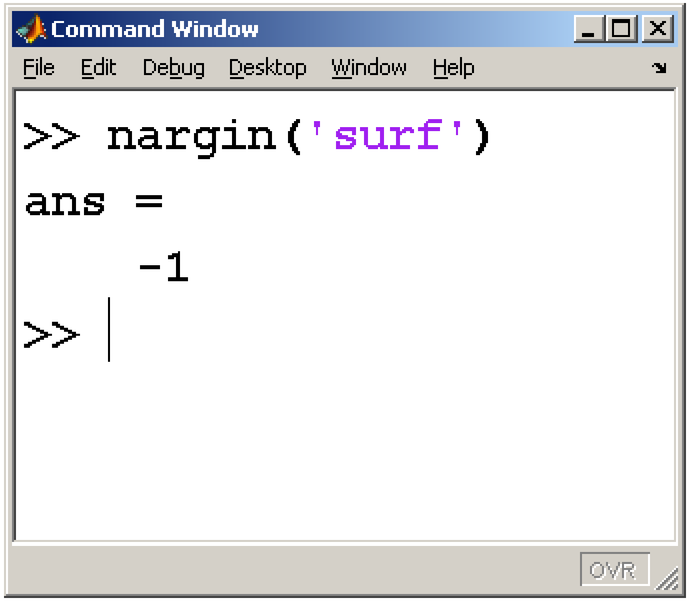
```
>> nargin('rem')
ans =
     2
>> nargout('max')
ans =
     2
>> 
```

You can use these functions in your programming to make your functions more versatile

- For example the surf function accepts a variable number of arguments
- surf(z) plots the 2-D matrix z against the index numbers
- surf(x,y,z) plots the 2-D matrix z against the x and y coordinates

When a variable number of arguments is allowed…

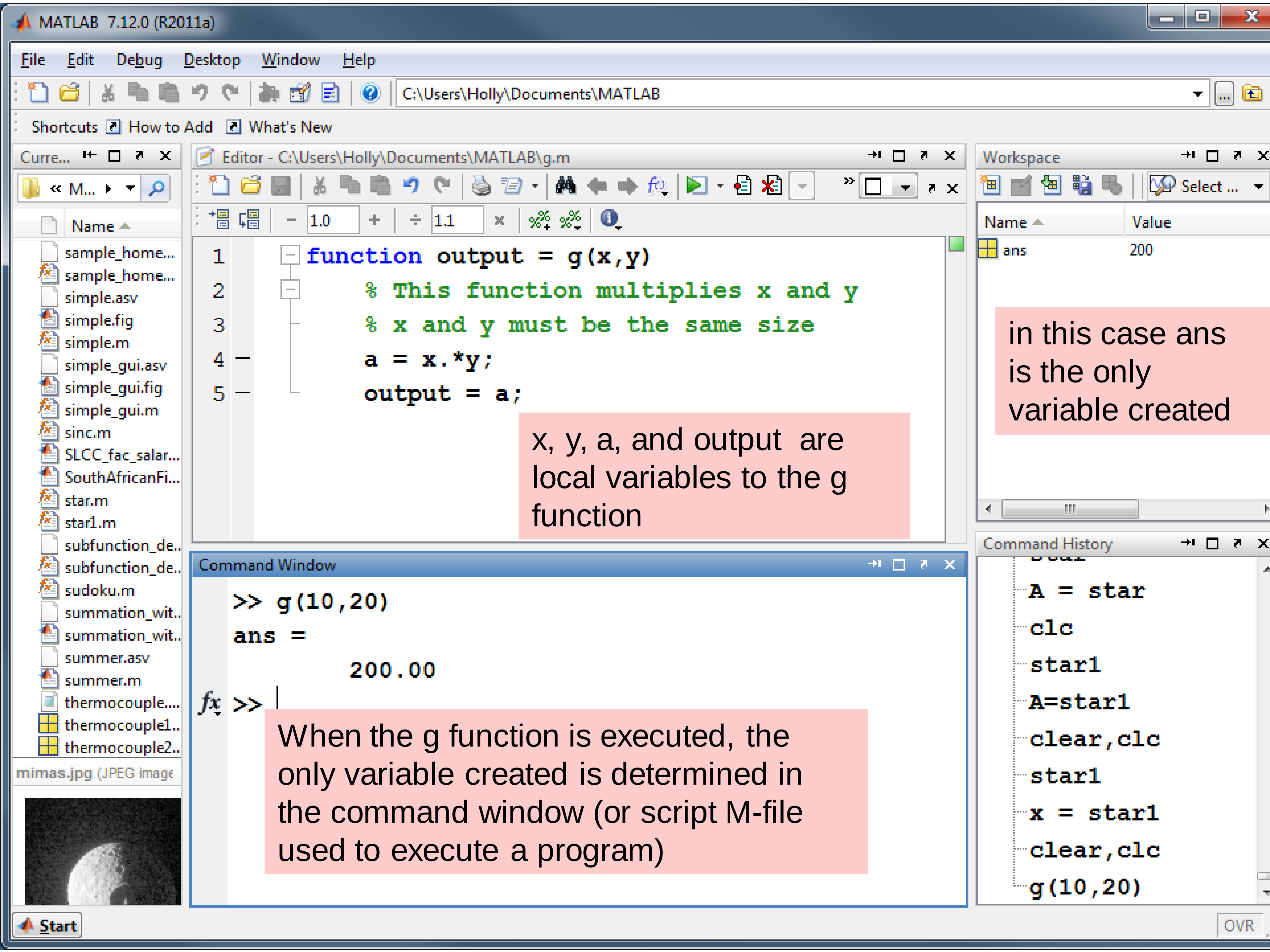
- nargin returns -1



```
>> nargin('surf')
ans =
    -1
>> 
```

Local Variables

- Variables defined in an M-file function, only have meaning inside that program
- if I set x=1 in the command window, it is not equal to 1 in the function
- If I set y=2 in a function, it is not equal to 2 in the workspace window
- The only way to communicate between functions and the workspace, is through the function input and output arguments



```
function output = g(x,y)
    % This function multiplies x and y
    % x and y must be the same size
    a = x.*y;
    output = a;
```

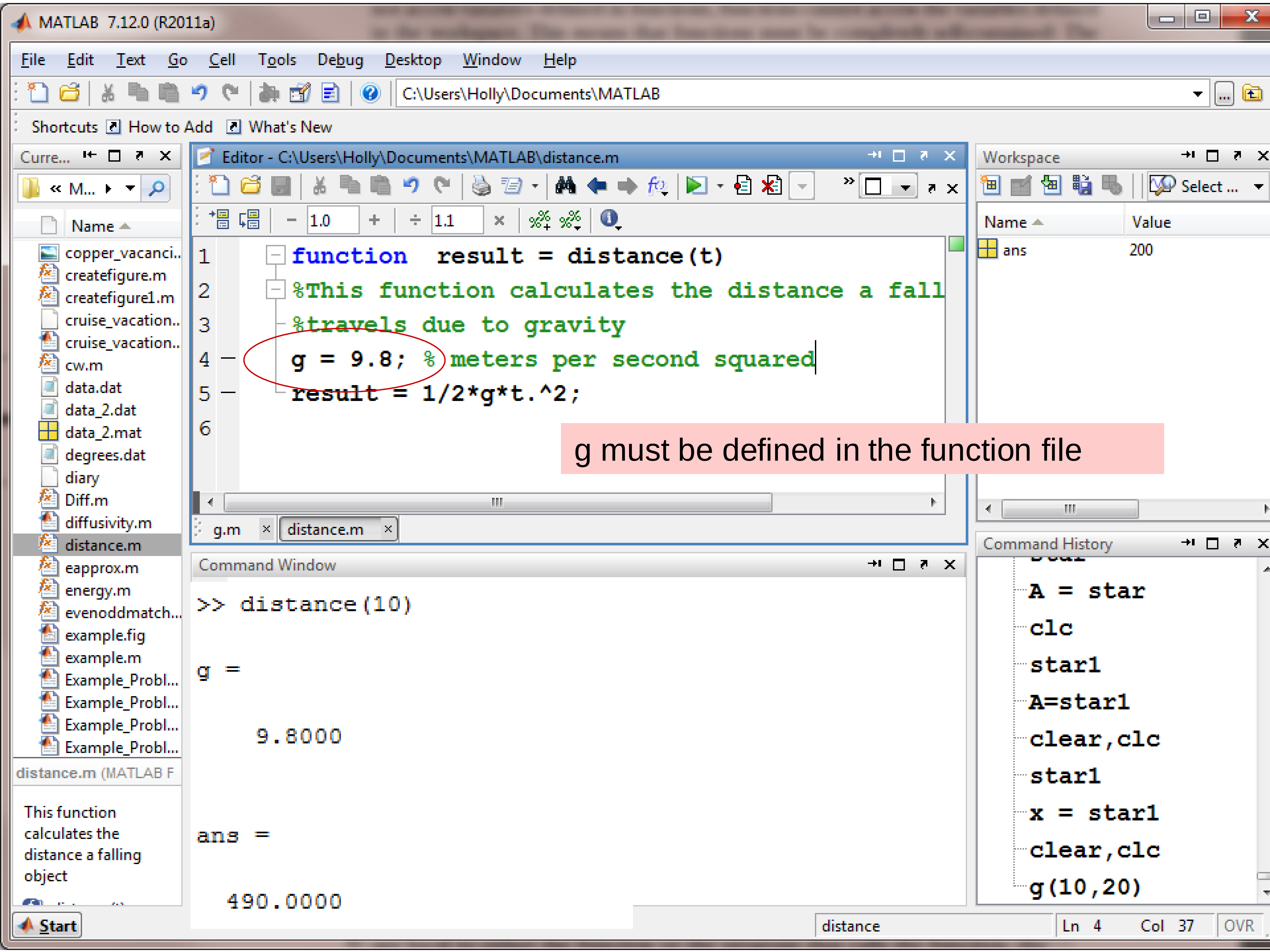
```
>> g(10,20)
ans =
        200.00
>> 
```

```
A = star
clc
star1
A=star1
clear,clc
star1
x = star1
clear,clc
g(10,20)
```

x, y, a, and output are local variables to the g function

in this case ans is the only variable created

When the g function is executed, the only variable created is determined in the command window (or script M-file used to execute a program)

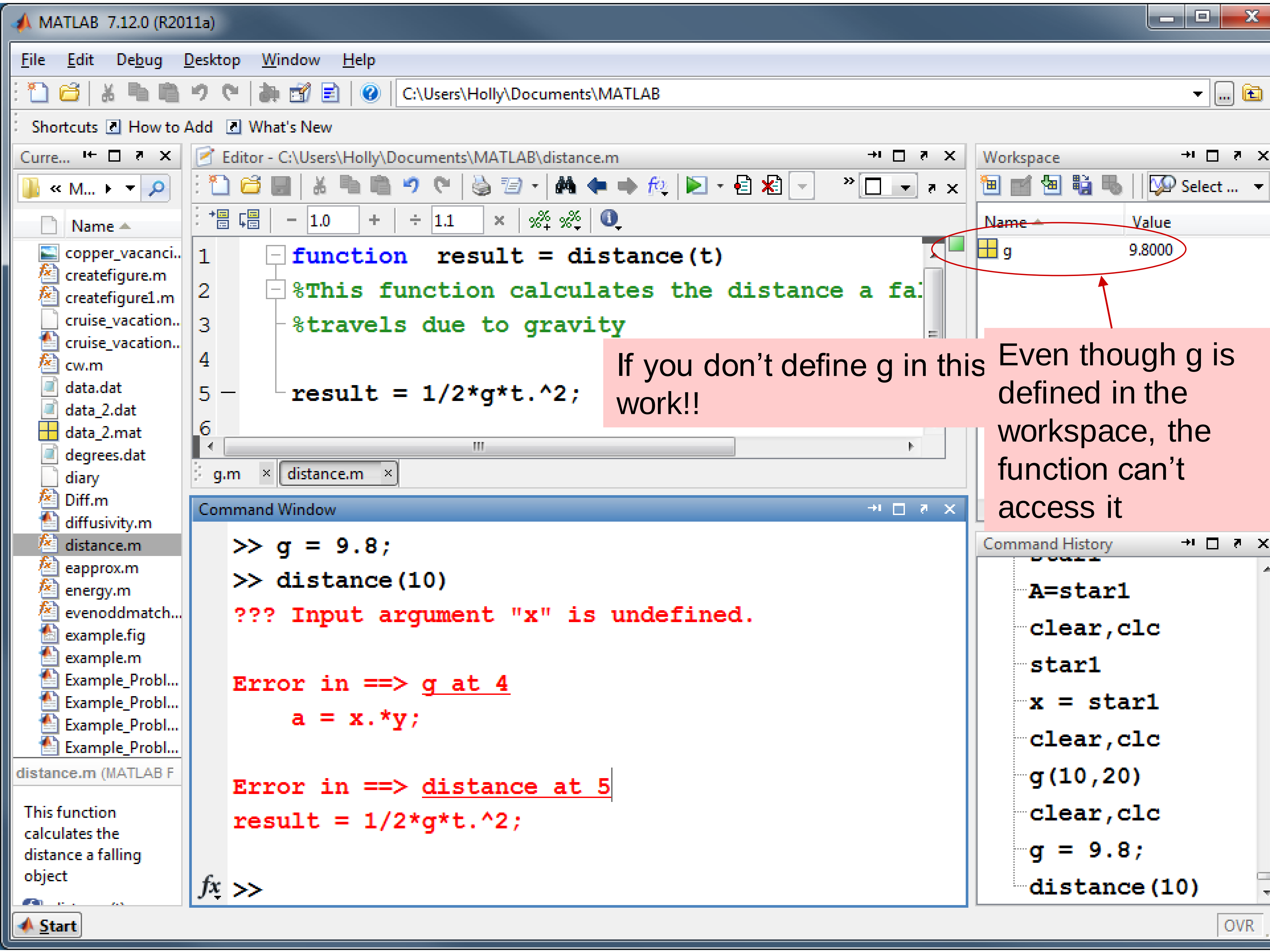


```
function  result = distance(t)
%This function calculates the distance a fall
%travels due to gravity
g = 9.8; % meters per second squared
result = 1/2*g*t.^2;
```

g must be defined in the function file

```
>> distance(10)

g =

    9.8000


ans =

  490.0000
```

```
A = star
clc
star1
A=star1
clear,clc
star1
x = star1
clear,clc
g(10,20)
```



Name	Value
g	9.8000

```
function  result = distance(t)
%This function calculates the distance a fa
%travels due to gravity

result = 1/2*g*t.^2;
```

If you don’t define g in this work!!

Even though g is defined in the workspace, the function can’t access it

```
>> g = 9.8;
>> distance(10)
??? Input argument "x" is undefined.

Error in ==> g at 4
    a = x.*y;

Error in ==> distance at 5
result = 1/2*g*t.^2;

>>
```

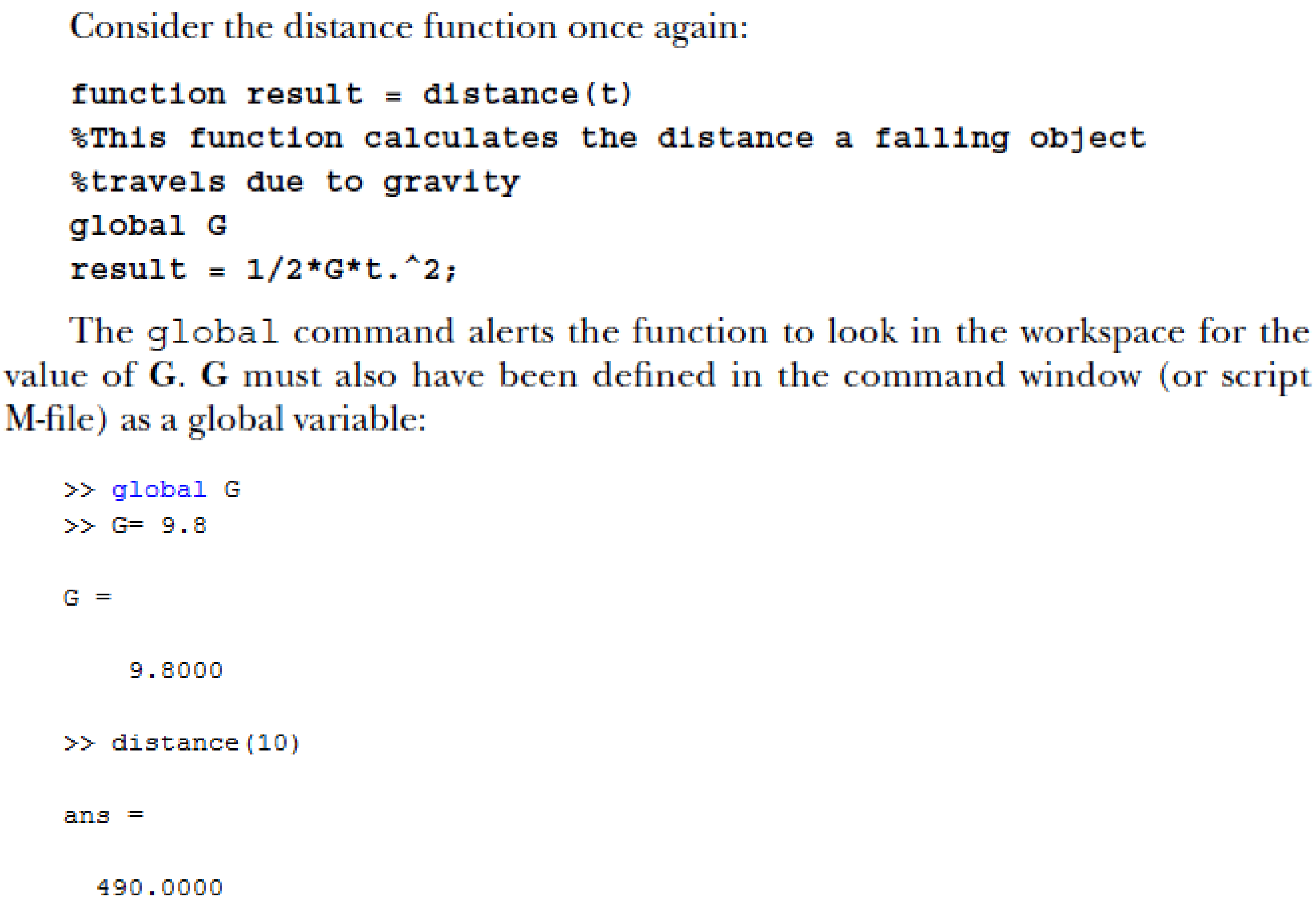
```
A=star1
clear,clc
star1
x = star1
clear,clc
g(10,20)
clear,clc
g = 9.8;
distance(10)
```

Global Variables

- Although it is possible to define global variables

It is a bad idea!!

Global Variables



Consider the distance function once again:

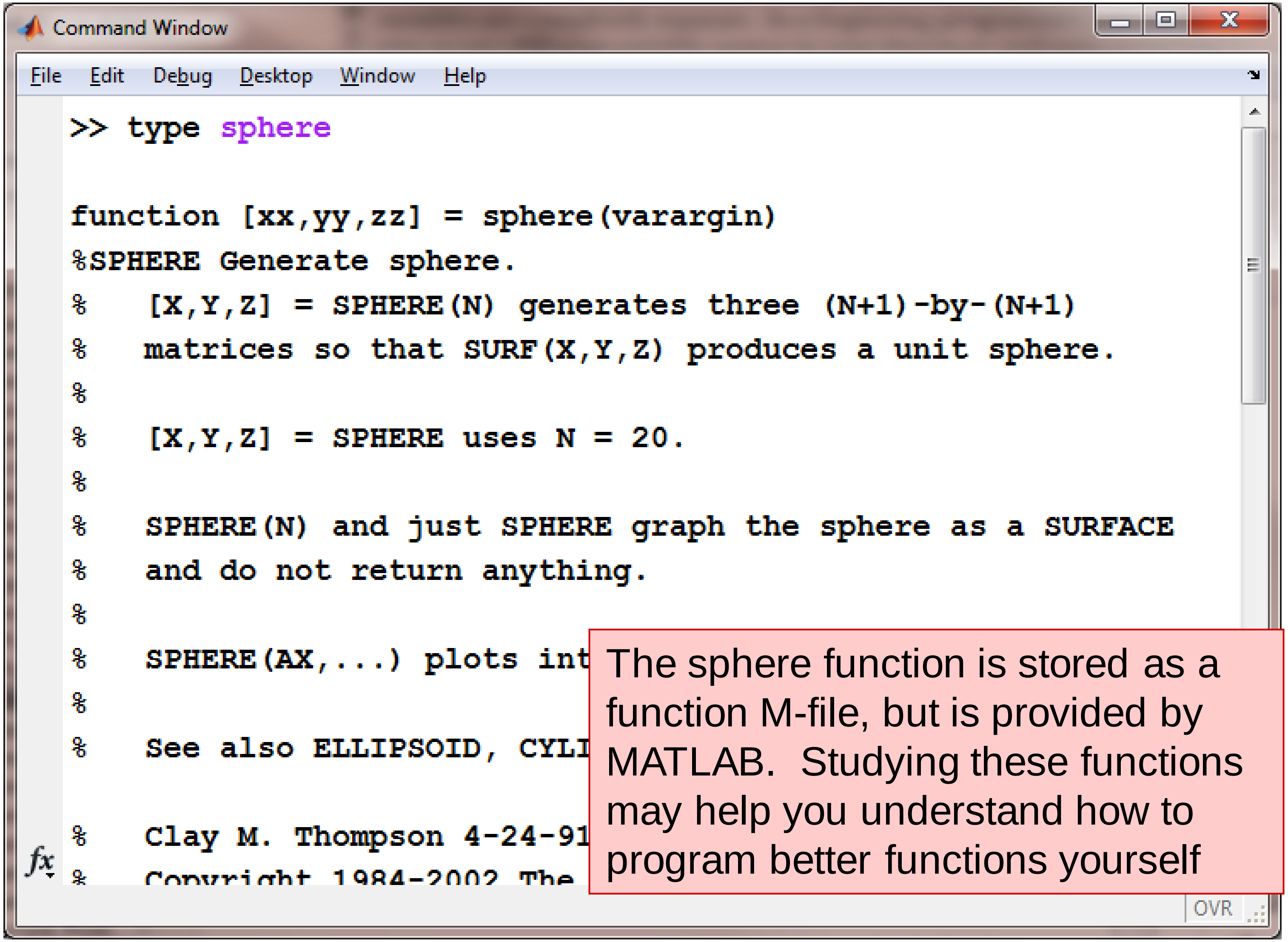
```
function result = distance(t)
%This function calculates the distance a falling object
%travels due to gravity
global G
result = 1/2*G*t.^2;
```

The `global` command alerts the function to look in the workspace for the value of **G**. **G** must also have been defined in the command window (or script M-file) as a global variable:

```
>> global G
>> G= 9.8

G =

    9.8000

>> distance(10)

ans =

  490.0000
```

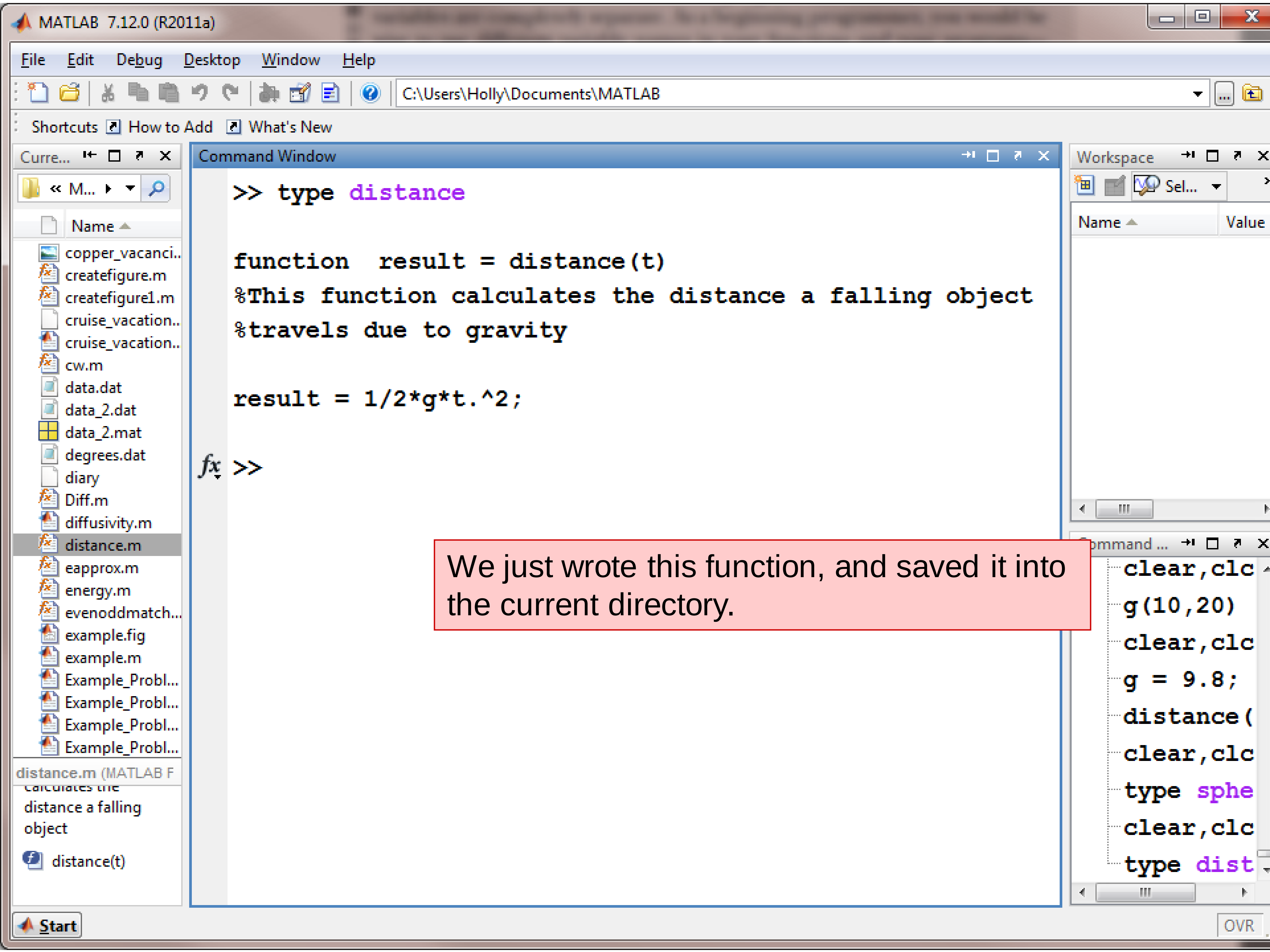
Accessing M-file Code

- functions provided with MATLAB consist of two types
    - The first is built in, and the code is not accessible to us
    - The second type consists of groups of function M-files – just like the ones we’ve been writing
- Use the type function to see the code in function M-files



```
>> type sphere

function [xx,yy,zz] = sphere(varargin)
%SPHERE Generate sphere.
%   [X,Y,Z] = SPHERE(N) generates three (N+1)-by-(N+1)
%   matrices so that SURF(X,Y,Z) produces a unit sphere.
%
%   [X,Y,Z] = SPHERE uses N = 20.
%
%   SPHERE(N) and just SPHERE graph the sphere as a SURFACE
%   and do not return anything.
%
%   SPHERE(AX,...) plots int
%
%   See also ELLIPSOID, CYLI

%   Clay M. Thompson 4-24-91
%   Copyright 1984-2002 The
```

The sphere function is stored as a function M-file, but is provided by MATLAB. Studying these functions may help you understand how to program better functions yourself



```
>> type distance

function  result = distance(t)
%This function calculates the distance a falling object
%travels due to gravity

result = 1/2*g*t.^2;

>>
```

Command History:

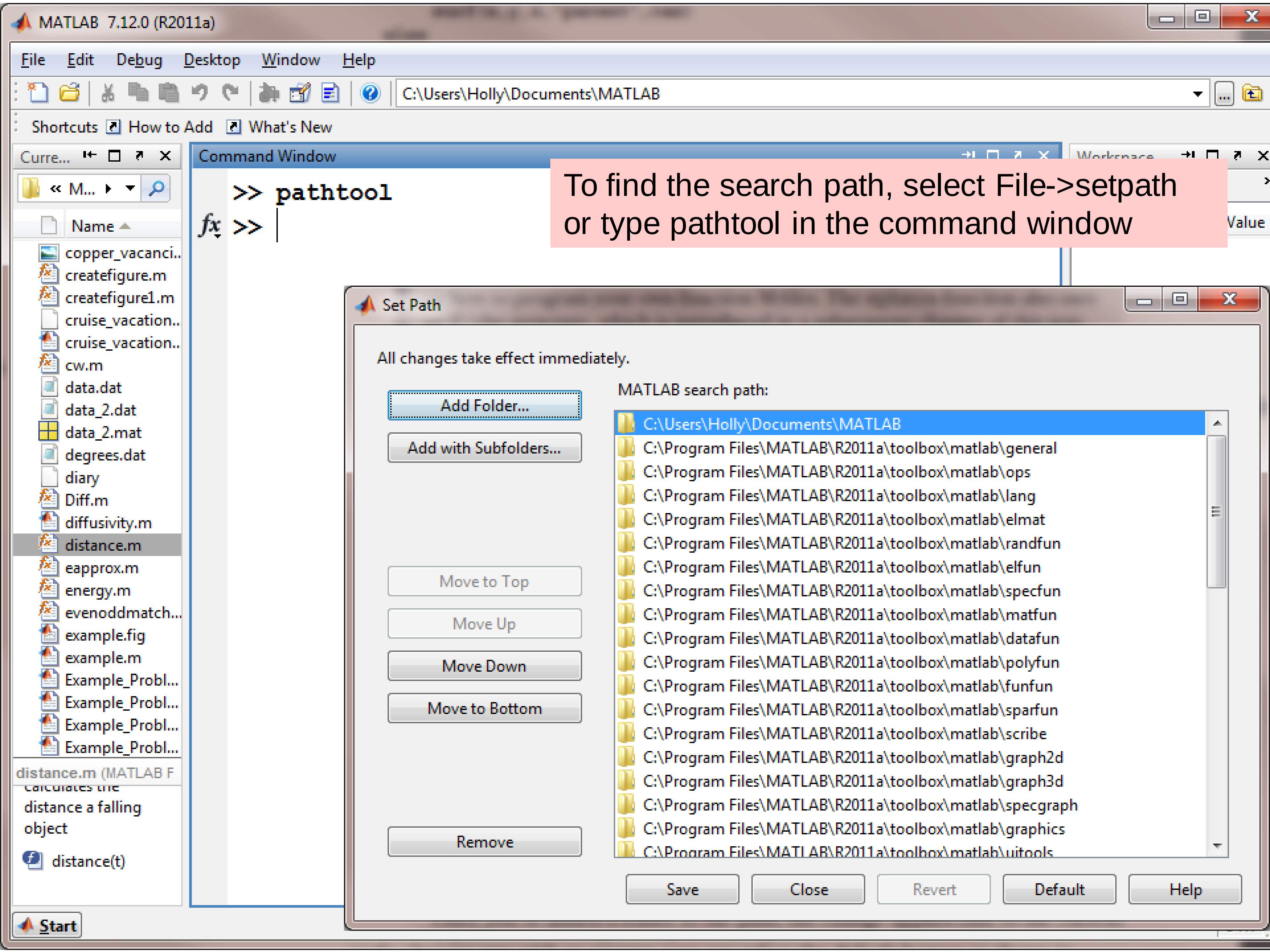
```
clear,clc
g(10,20)
clear,clc
g = 9.8;
distance(
clear,clc
type sphe
clear,clc
type dist
```

We just wrote this function, and saved it into the current directory.

Section 6.2

Creating Your Own Toolbox of Functions

- When you call a function MATLAB searches for it along a predetermined path
  - First it looks in the current directory
  - Then it follows a search path determined by your installation of the program



To find the search path, select File->setpath or type pathtool in the command window

```
>> pathtool
>>
```

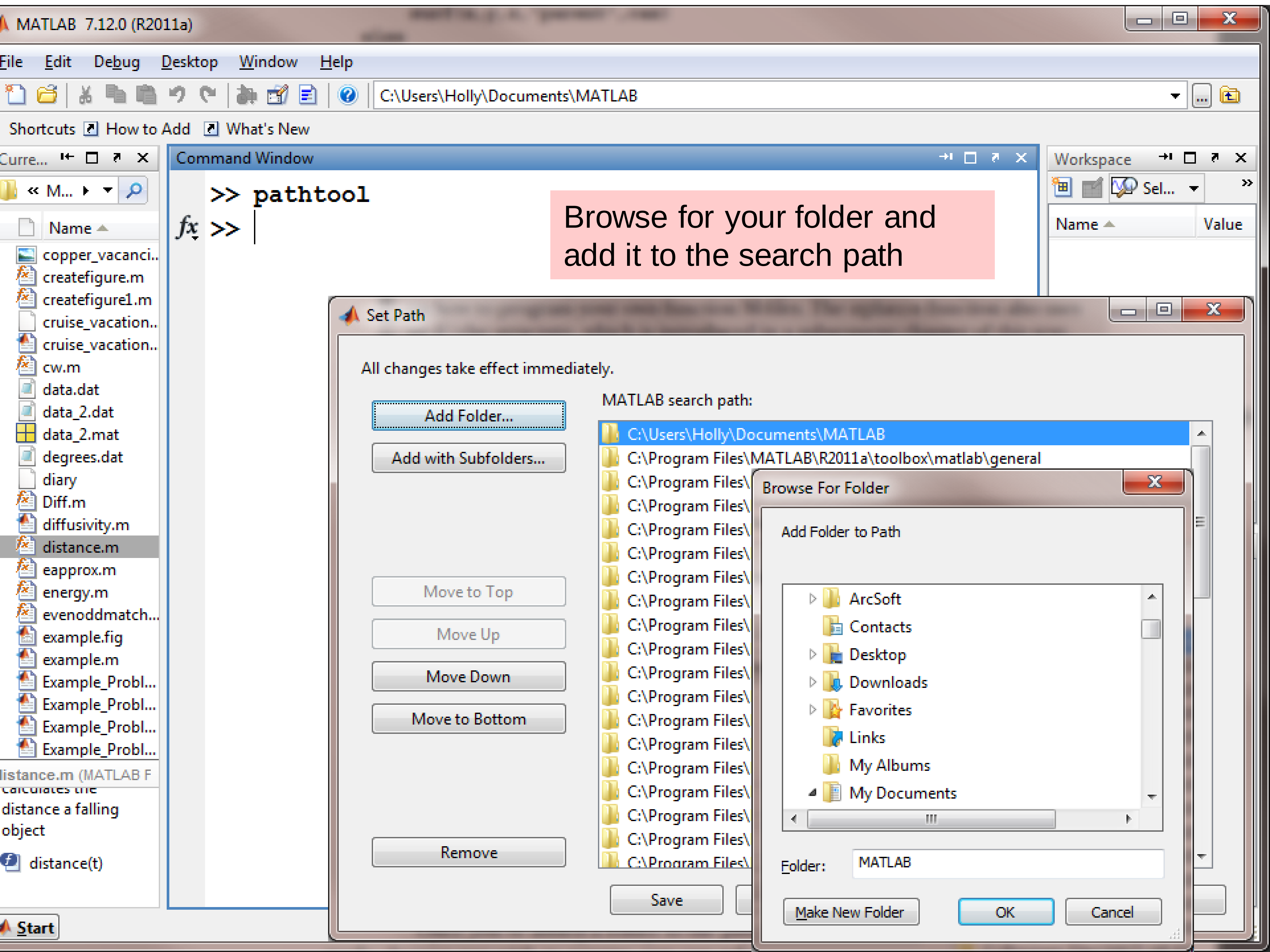
All changes take effect immediately.

MATLAB search path:

```
C:\Users\Holly\Documents\MATLAB
C:\Program Files\MATLAB\R2011a\toolbox\matlab\general
C:\Program Files\MATLAB\R2011a\toolbox\matlab\ops
C:\Program Files\MATLAB\R2011a\toolbox\matlab\lang
C:\Program Files\MATLAB\R2011a\toolbox\matlab\elmat
C:\Program Files\MATLAB\R2011a\toolbox\matlab\randfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\elfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\specfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\matfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\datafun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\polyfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\funfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\sparfun
C:\Program Files\MATLAB\R2011a\toolbox\matlab\scribe
C:\Program Files\MATLAB\R2011a\toolbox\matlab\graph2d
C:\Program Files\MATLAB\R2011a\toolbox\matlab\graph3d
C:\Program Files\MATLAB\R2011a\toolbox\matlab\specgraph
C:\Program Files\MATLAB\R2011a\toolbox\matlab\graphics
C:\Program Files\MATLAB\R2011a\toolbox\matlab\uitools
```

Create your own toolbox

- Once you’ve created a set of functions, you’d like to be able to access regularly, group them into a directory (folder) and add them to the search path using the pathtool



C:\Users\Holly\Documents\MATLAB

calculates the distance a falling object

distance(t)

```
>> pathtool
>>
```

Browse for your folder and add it to the search path

All changes take effect immediately.

MATLAB search path:

C:\Users\Holly\Documents\MATLAB
C:\Program Files\MATLAB\R2011a\toolbox\matlab\general
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\
C:\Program Files\

Add Folder to Path

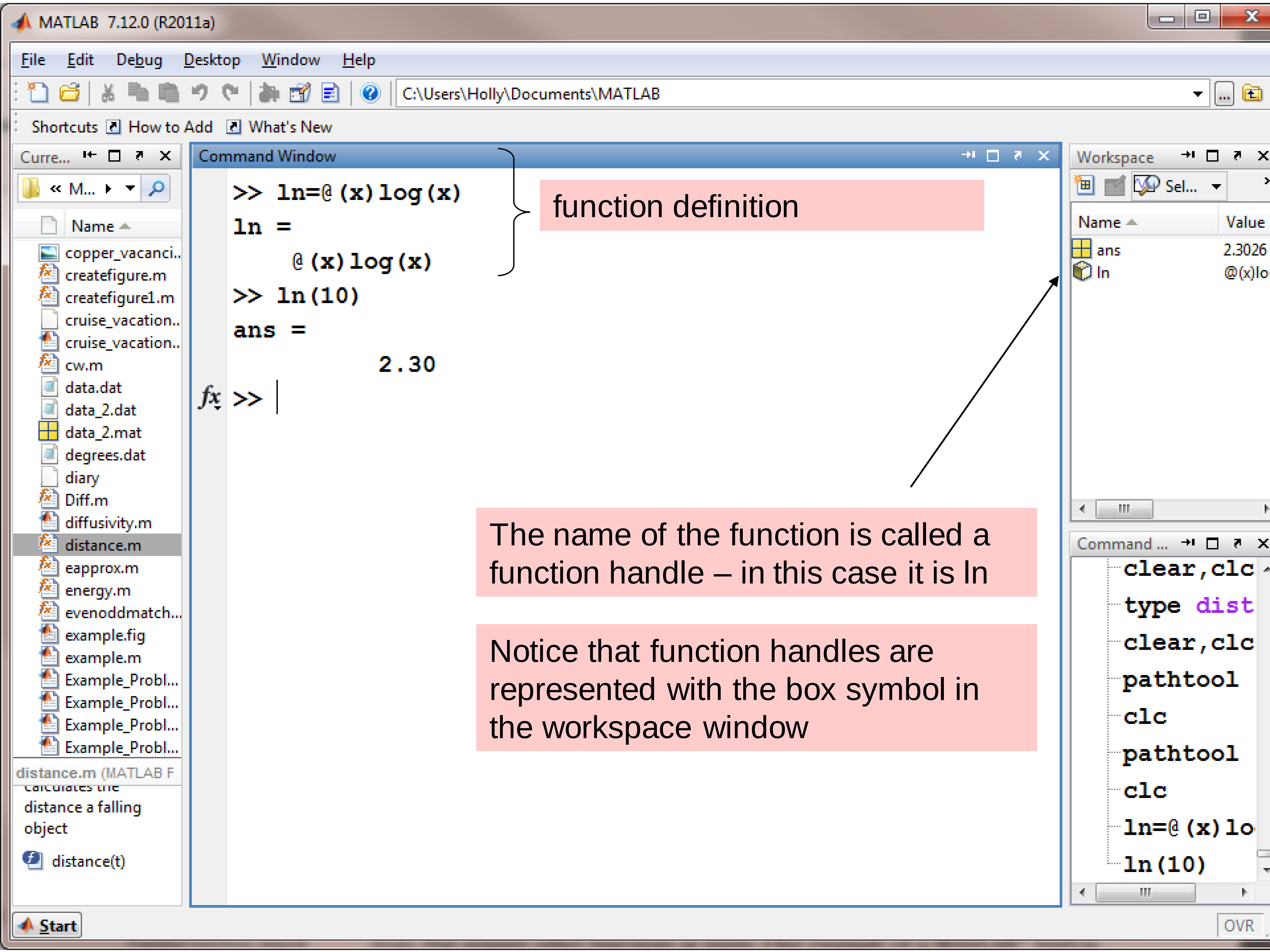
Folder: MATLAB

Section 6.3
Anonymous Functions

- Normally if you go to the trouble of creating a function, you want to store it for future use.
- Anonymous functions are defined inside a script M-file or in the command window, and are only available while they are stored in the workspace window – much like variables

Define anonymous functions in a script M-file

- Suppose you’d like to define a function for natural log called ln
- ln=@(x) log(x)
  - The @ symbol alerts MATLAB that ln is a function
  - The function input is next, inside parentheses
  - Finally the function is defined



```
>> ln=@(x)log(x)
ln =
    @(x)log(x)
>> ln(10)
ans =
          2.30
>>
```

function definition

Name	Value
ans	2.3026
ln	@(x)lo

```
clear,clc
type dist
clear,clc
pathtool
clc
pathtool
clc
ln=@(x)lo
ln(10)
```

The name of the function is called a function handle – in this case it is ln

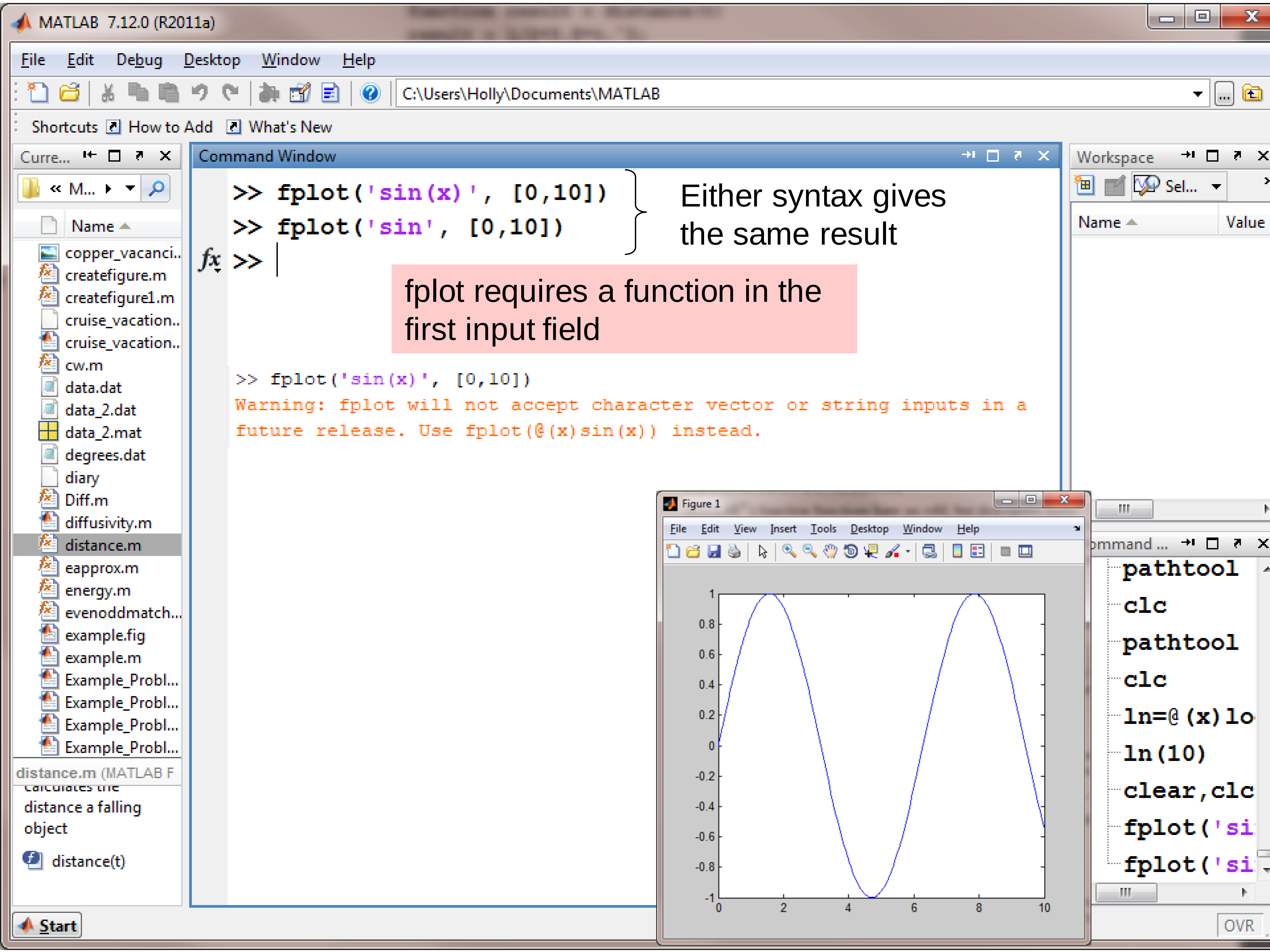
Notice that function handles are represented with the box symbol in the workspace window

Saving Anonymous Functions

- Anonymous functions can be saved as a .mat file – just like anything else listed in the workspace window
- Retrieve anonymous functions using the load command

Section 6.4
Function Functions

- Some functions accept other functions as input
- An example is the fplot function described in chapter 5 or the nargin and nargout functions described in this chapter

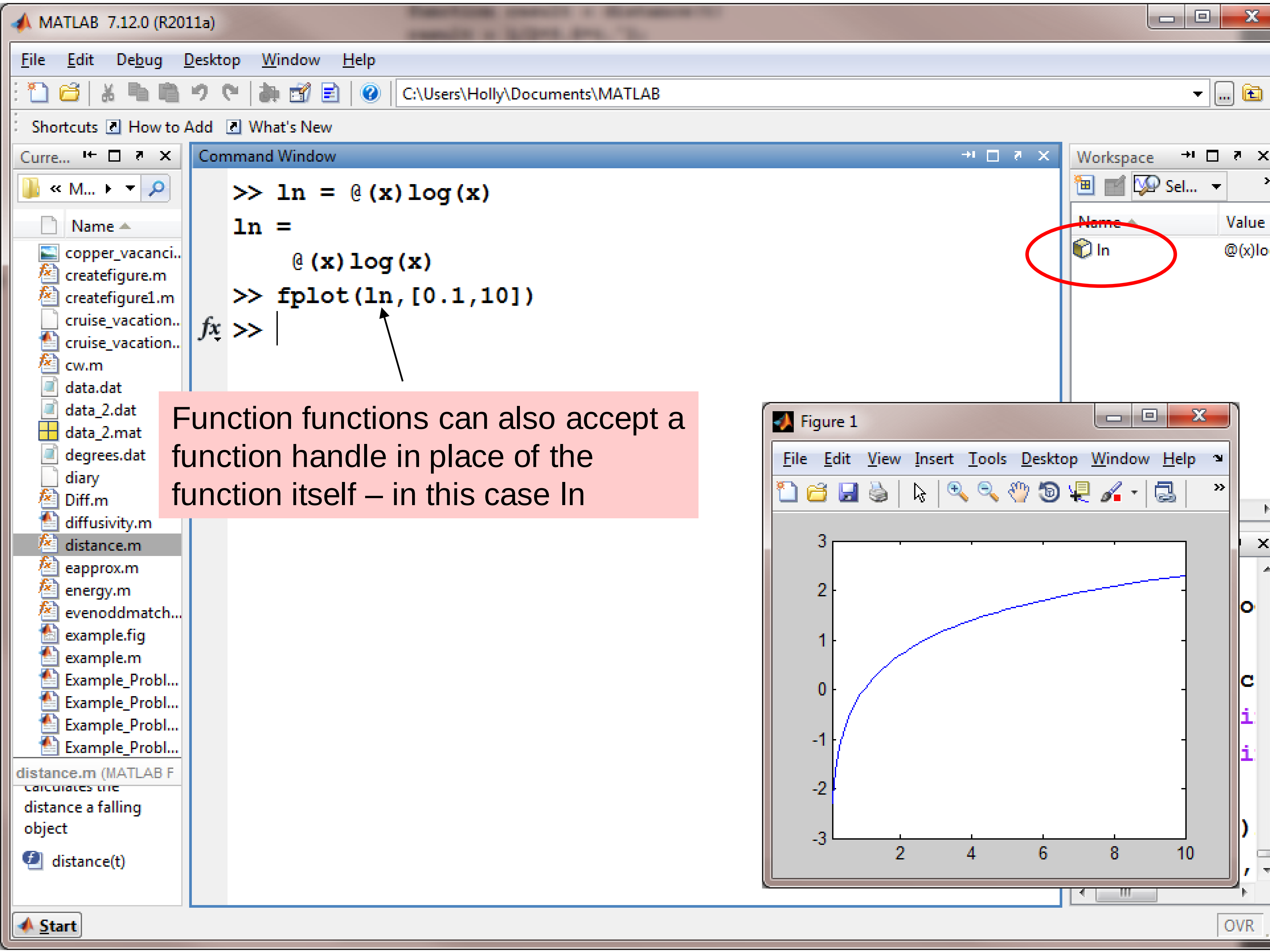


```
>> fplot('sin(x)', [0,10])
>> fplot('sin', [0,10])
>>
```

Either syntax gives the same result

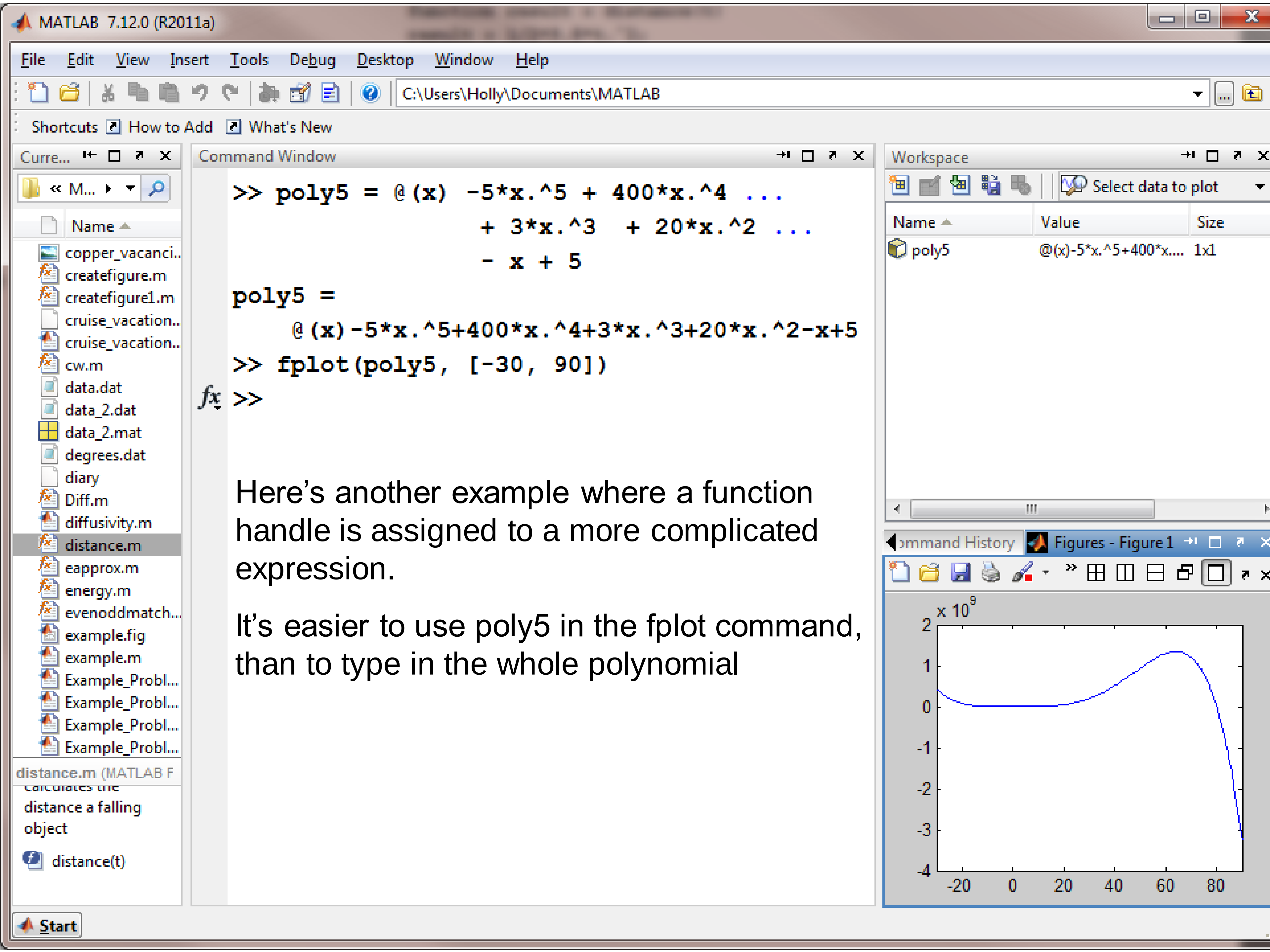
fplot requires a function in the first input field

```
>> fplot('sin(x)', [0,10])
Warning: fplot will not accept character vector or string inputs in a
future release. Use fplot(@(x)sin(x)) instead.
```



```
>> ln = @(x)log(x)
ln =
     @(x)log(x)
>> fplot(ln,[0.1,10])
>> 
```

Function functions can also accept a function handle in place of the function itself – in this case ln

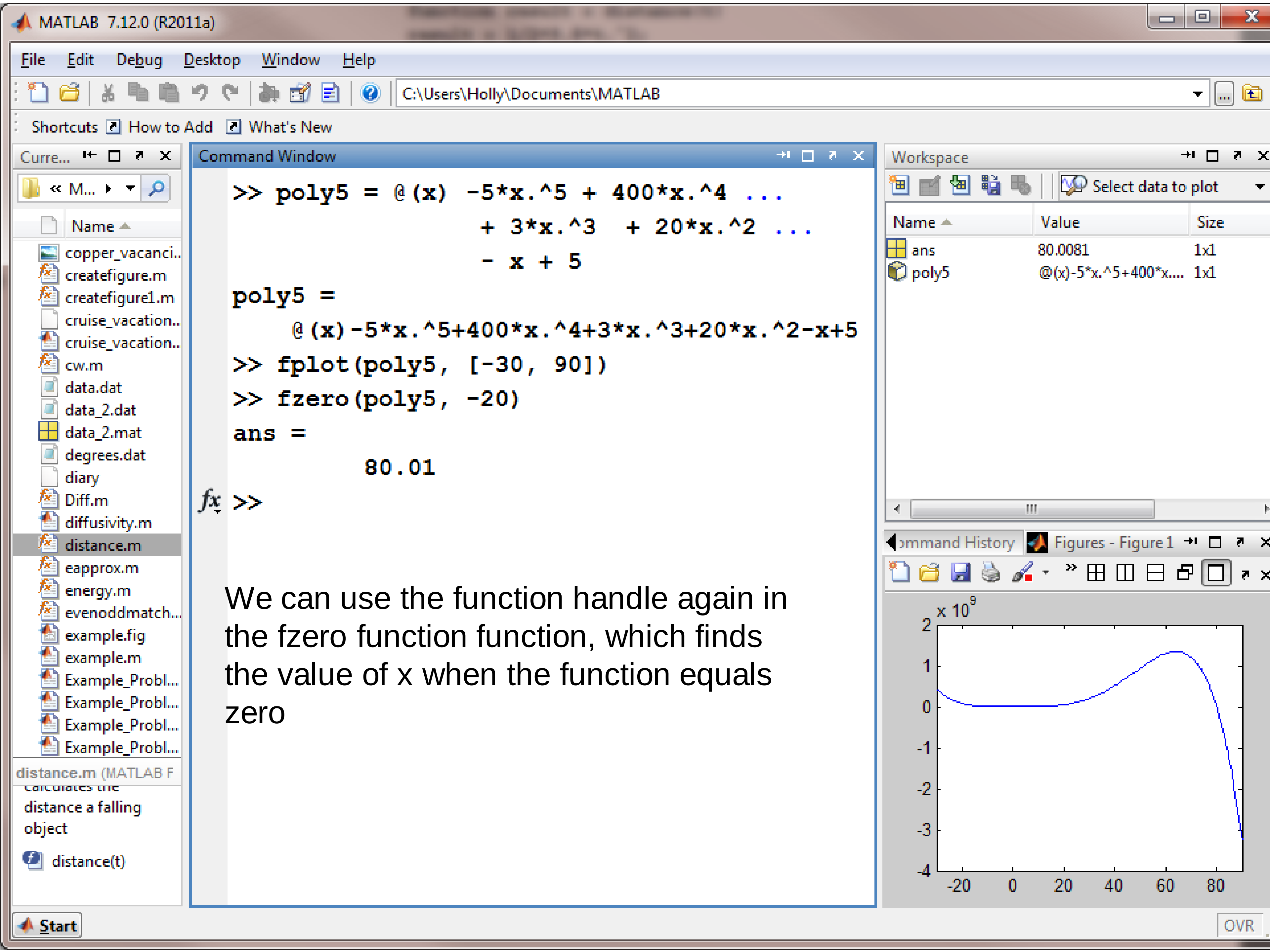


```
>> poly5 = @(x) -5*x.^5 + 400*x.^4 ...
                + 3*x.^3  + 20*x.^2 ...
                - x + 5

poly5 =
    @(x)-5*x.^5+400*x.^4+3*x.^3+20*x.^2-x+5
>> fplot(poly5, [-30, 90])
>>
```

Name	Value	Size
poly5	@(x)-5*x.^5+400*x....	1x1

Here’s another example where a function handle is assigned to a more complicated expression.

It’s easier to use poly5 in the fplot command, than to type in the whole polynomial



```
>> poly5 = @(x) -5*x.^5 + 400*x.^4 ...
              + 3*x.^3  + 20*x.^2 ...
              - x + 5

poly5 =
    @(x)-5*x.^5+400*x.^4+3*x.^3+20*x.^2-x+5
>> fplot(poly5, [-30, 90])
>> fzero(poly5, -20)
ans =
         80.01
>>
```

Name	Value	Size
ans	80.0081	1x1
poly5	@(x)-5*x.^5+400*x....	1x1

We can use the function handle again in the fzero function function, which finds the value of x when the function equals zero

Summary – Function M-Files

- Function M-files must start with a definition line containing
  - the word function
  - a variable that defines the function output
  - a function name
  - a variable used for the input argument

Summary – Function M-files

- Function M-files must be stored in the current directory or in a user defined toolbox
- The function name must also be the file name

Summary - IO

- Multiple Inputs are allowed
- Multiple Outputs are allowed
- Some functions require no input
- Some functions produce no outputs

Summary - Comments

- Functions should contain ample comments to document the code
- The comments directly after the function definition are used by the help feature to describe the function

Summary - Toolboxes

- Numerous toolboxes are provided by MATLAB
- Others are available from the user community
- Individual users can define their own toolboxes
- The pathtool is used to define the search path so that MATLAB can find the toolboxes

Summary – Anonymous Functions

- Anonymous functions are defined in a MATLAB session or M-file
- They only exist during the current session
- They are useful as input to function functions

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 7
(Logical Functions & Repetition Structures)

Zeyad T. Shareef

Objectives

After studying this lecture, you should be able to:

- Understand how MATLAB interprets relational and logical operators
- Use the find function
- Understand the appropriate uses of the if/else family of commands
- Understand the switch/case structure

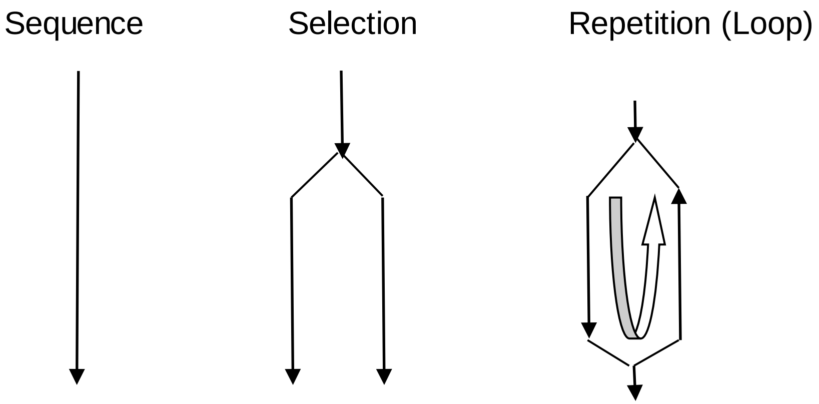
Objectives (cont.)

After studying this lecture, you should be able to:

- Write and use for loops
- Write and use while loops
- Create midpoint break structures
- Measure the time required to execute program components
- Understand how to improve program execution times

Structures

- Sequence
- Selection
- Repetition



7.1 Relational and Logical Operators

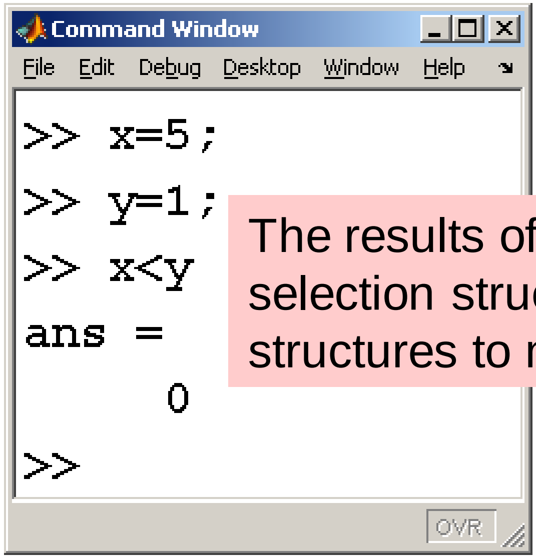
- Sequence and Repetition structures require comparisons to work
- Relational operators make those comparisons
- Logical operators allow us to combine the comparisons

Relational Operators

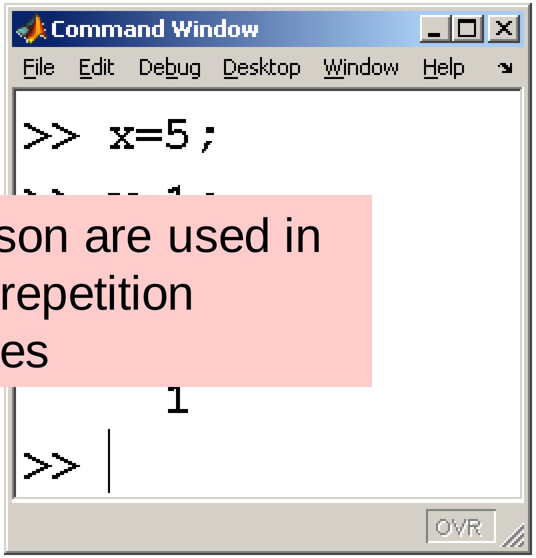
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Comparisons are either true or false

- Most computer programs use the number 1 for true and 0 for false



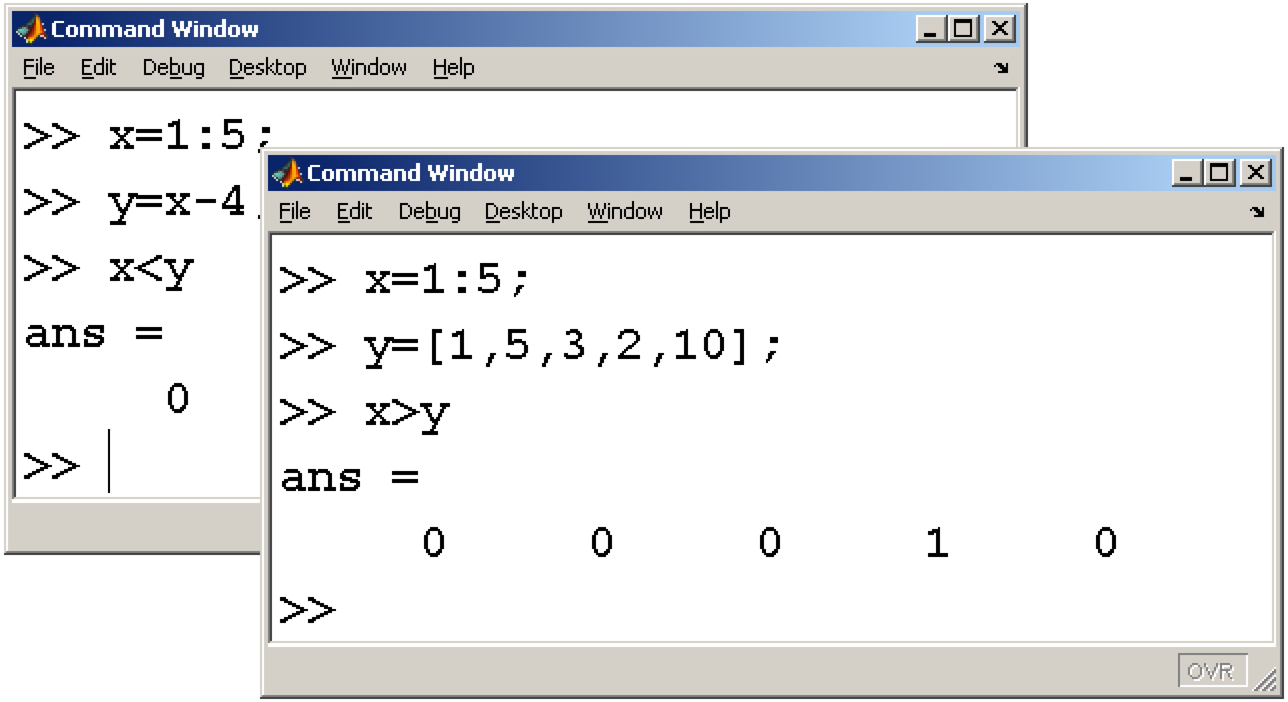
```
>> x=5;
>> y=1;
>> x<y
ans =
     0
>>
```



```
>> x=5;

     1
>>
```

The results of a comparison are used in selection structures and repetition structures to make choices

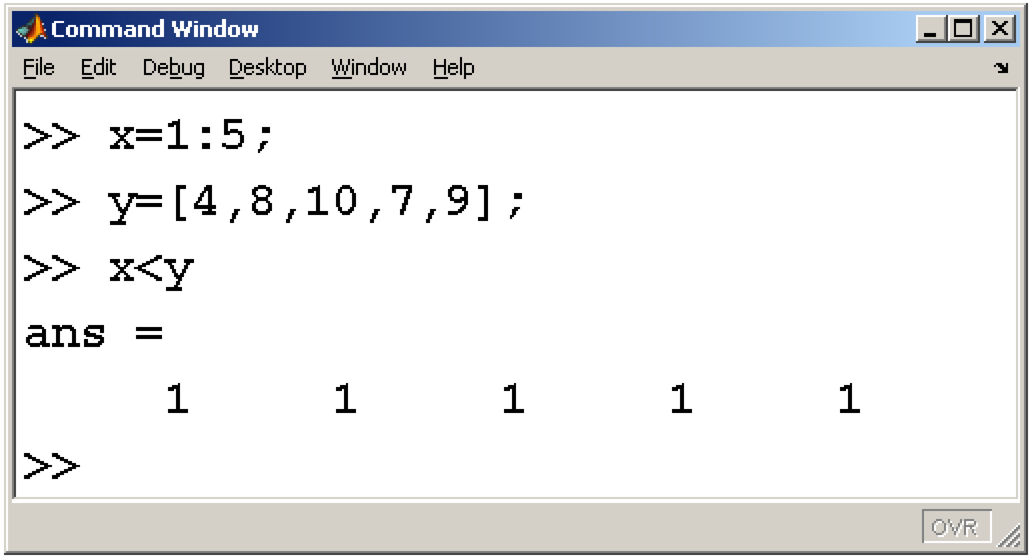
MATLAB compares corresponding elements and determines if the result is true or false for each


Command Window
File Edit Debug Desktop Window Help
Command Window
File Edit Debug Desktop Window Help
OVR

```
>> x=1:5;
>> y=x-4.
>> x<y
ans =
     0
>> 
```

```
>> x=1:5;
>> y=[1,5,3,2,10];
>> x>y
ans =
     0     0     0     1     0
>>
```

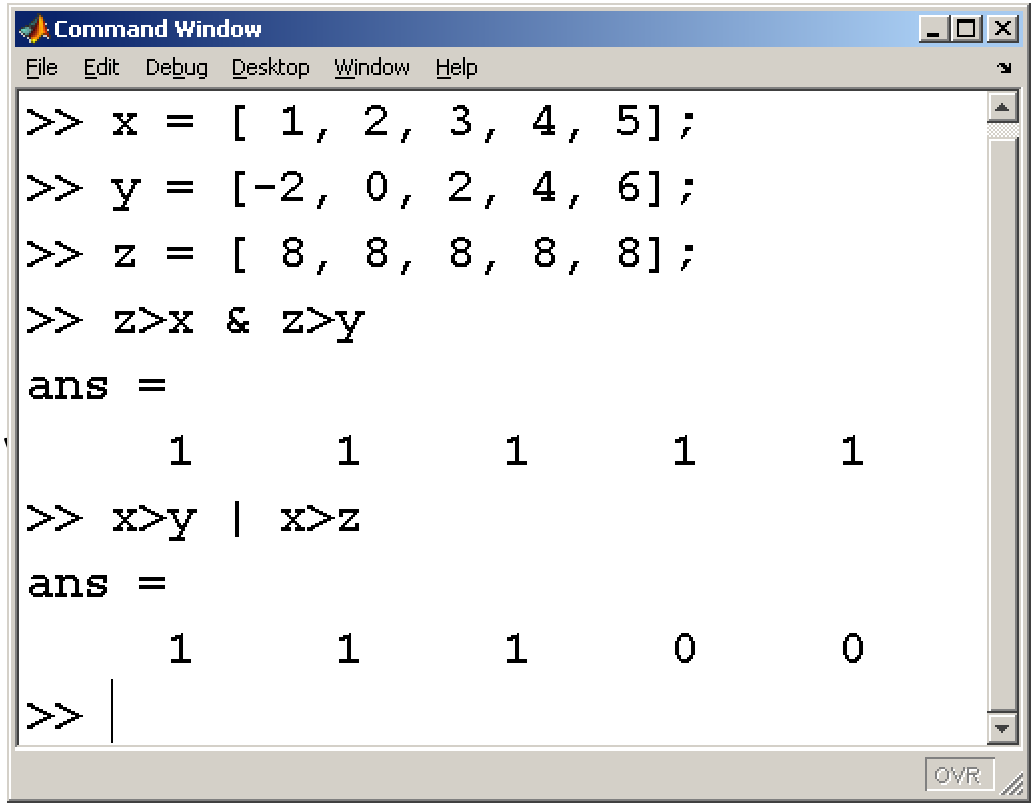
In order for MATLAB to decide a comparison is true for an entire matrix, it must be true for every element in the matrix



```
>> x=1:5;
>> y=[4,8,10,7,9];
>> x<y
ans =
     1     1     1     1     1
>>
```

Logical Operators

&	and
~	not
|	or
xor	exclusi



```
>> x = [ 1, 2, 3, 4, 5];
>> y = [-2, 0, 2, 4, 6];
>> z = [ 8, 8, 8, 8, 8];
>> z>x & z>y
ans =
     1     1     1     1     1
>> x>y | x>z
ans =
     1     1     1     0     0
>> 
```

7.2 Flow Charts and Pseudo-Code

- As you write more complicated programs it becomes more and more important to plan your code before you write it
- Flow charts – graphical approach
- Pseudo-code – verbal description

Pseudo-code

- Outline a set of statements describing the steps you will take to solve a problem
- Convert these steps into comments in an M-file
- Insert the appropriate MATLAB code into the file between the comment lines

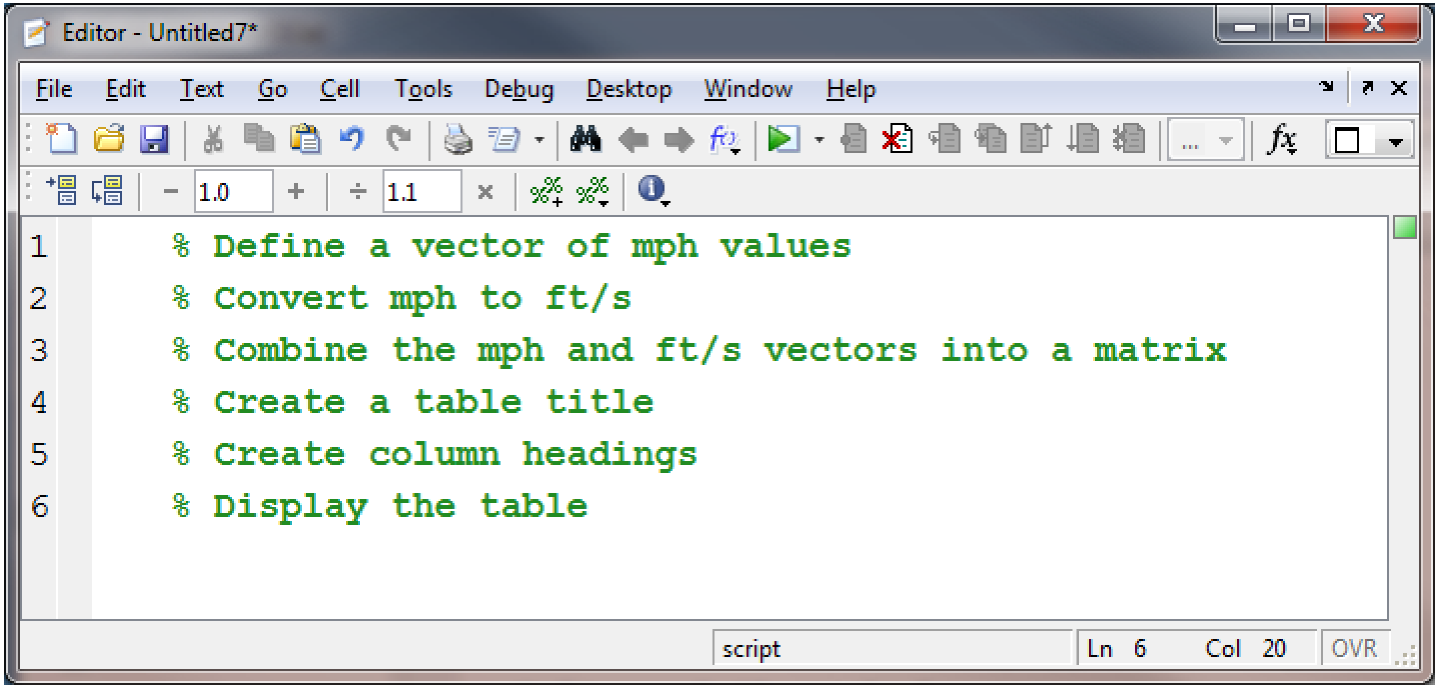
Pseudo-code Example

- You’ve been asked to create a program to convert miles/hr to ft/s. The output should be a table, complete with title and column headings

Outline the steps

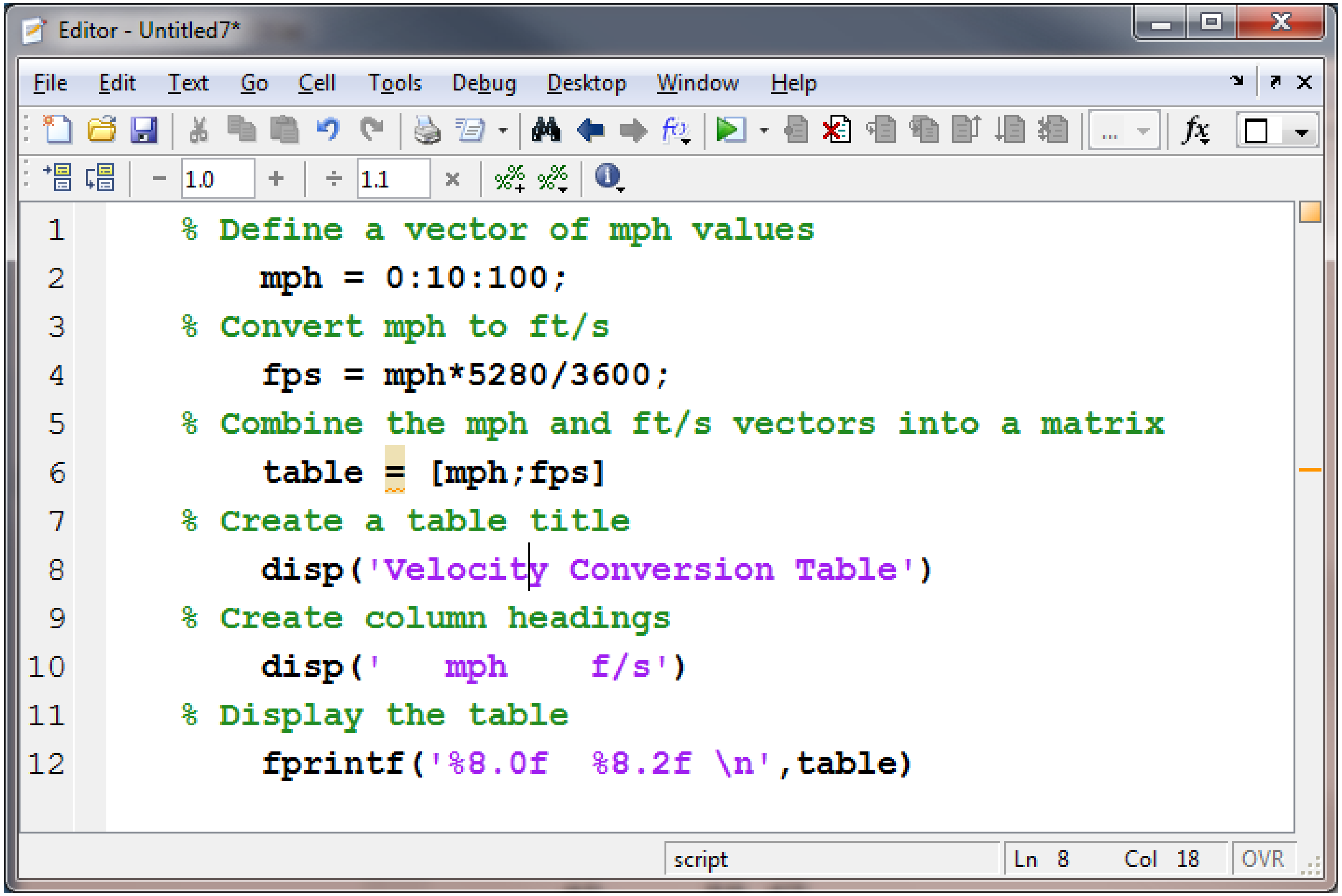
- Define a vector of mph values
- Convert mph to ft/s
- Combine the mph and ft/s vectors into a matrix
- Create a table title
- Create column headings
- Display the table

Convert the steps to M-file comments



```
% Define a vector of mph values
% Convert mph to ft/s
% Combine the mph and ft/s vectors into a matrix
% Create a table title
% Create column headings
% Display the table
```

Insert the MATLAB code between the comments

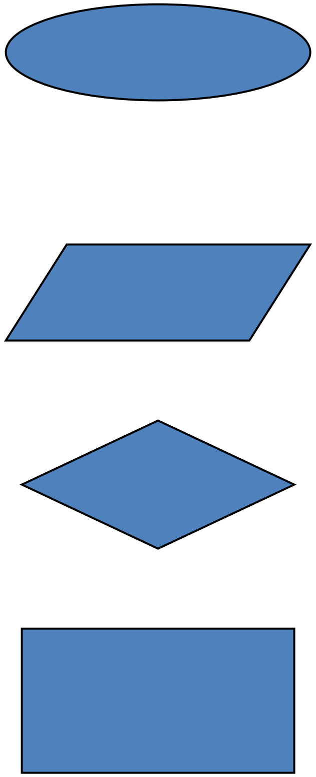


```
    % Define a vector of mph values
        mph = 0:10:100;
    % Convert mph to ft/s
        fps = mph*5280/3600;
    % Combine the mph and ft/s vectors into a matrix
        table = [mph;fps]
    % Create a table title
        disp('Velocity Conversion Table')
    % Create column headings
        disp('   mph    f/s')
    % Display the table
        fprintf('%8.0f  %8.2f \n',table)
```

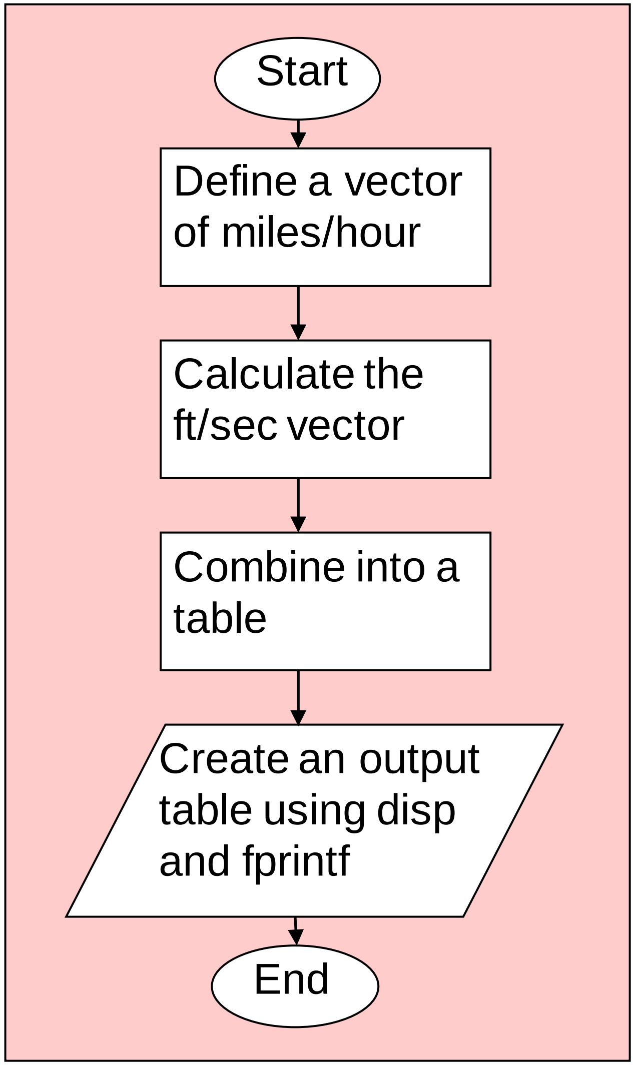
Flow Charting

- Especially appropriate for more complicated programs
- Create a big picture graphically
- Convert to pseudo-code

Simple Flow Chart Symbols



- An oval indicates the beginning of a section of code
- A parallelogram indicates an input or output
- A diamond indicates a decision point
- Calculations are placed in rectangles


Start
Define a vector of miles/hour
Calculate the ft/sec vector
Combine into a table
Create an output table using disp and fprintf
End

This flowchart represents the mph to ft/s problem

7.3 Logical Functions

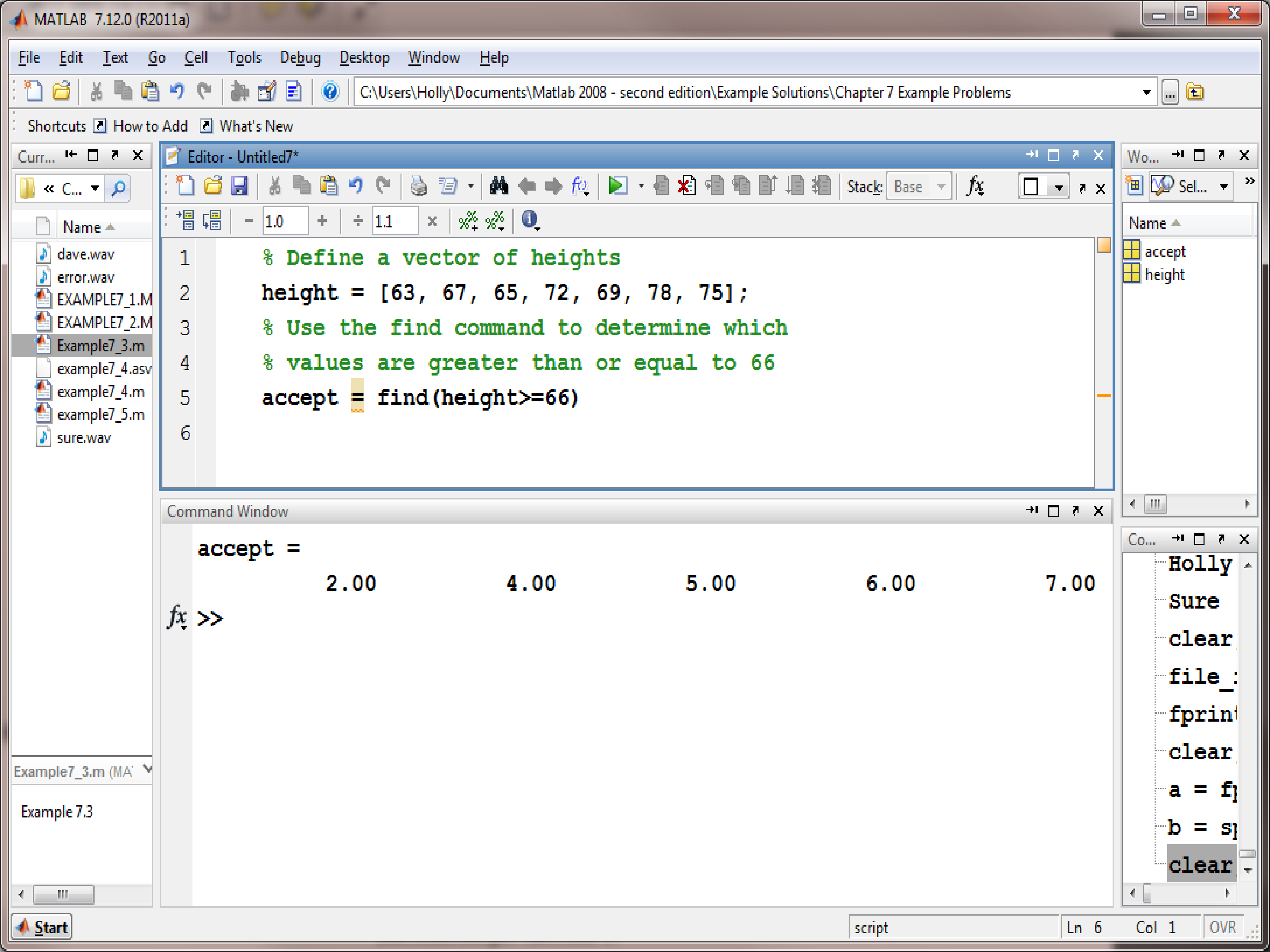
- MATLAB offers traditional programming selection structures
  - if
  - if/else
  - switch/case
- And… a series of logical functions that perform many of the same tasks

find

- The find command searches a matrix and identifies which elements in that matrix meet a given criteria.

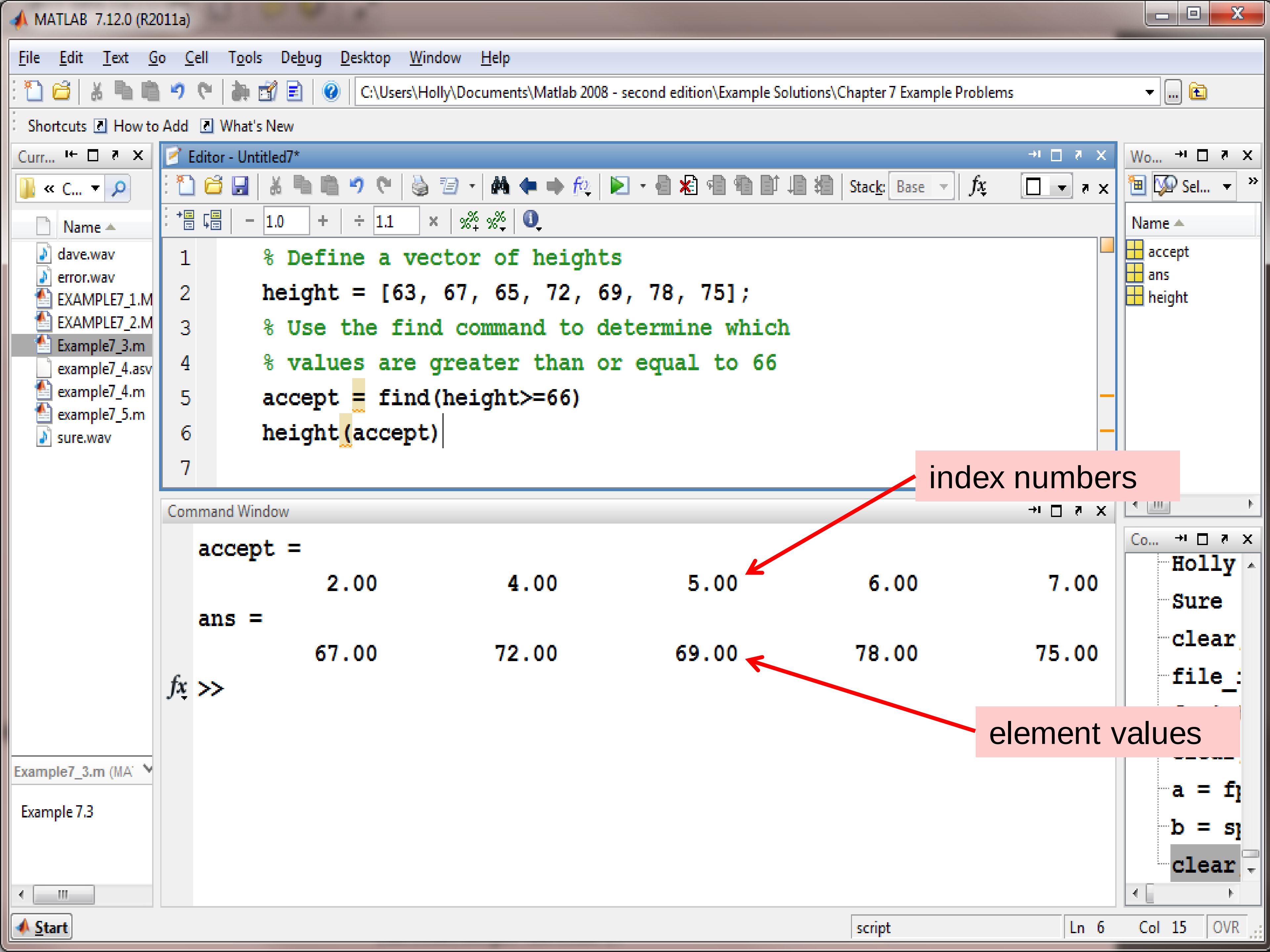
For example…

- An academy requires applicants to be at least 66” tall
- Consider this list of applicant heights
- 63”, 67”, 65”, 72”, 69”, 78”, 75”
- Which applicants meet the criteria?



```
% Define a vector of heights
height = [63, 67, 65, 72, 69, 78, 75];
% Use the find command to determine which
% values are greater than or equal to 66
accept = find(height>=66)
```

```
accept =
          2.00          4.00          5.00          6.00          7.00
>>
```

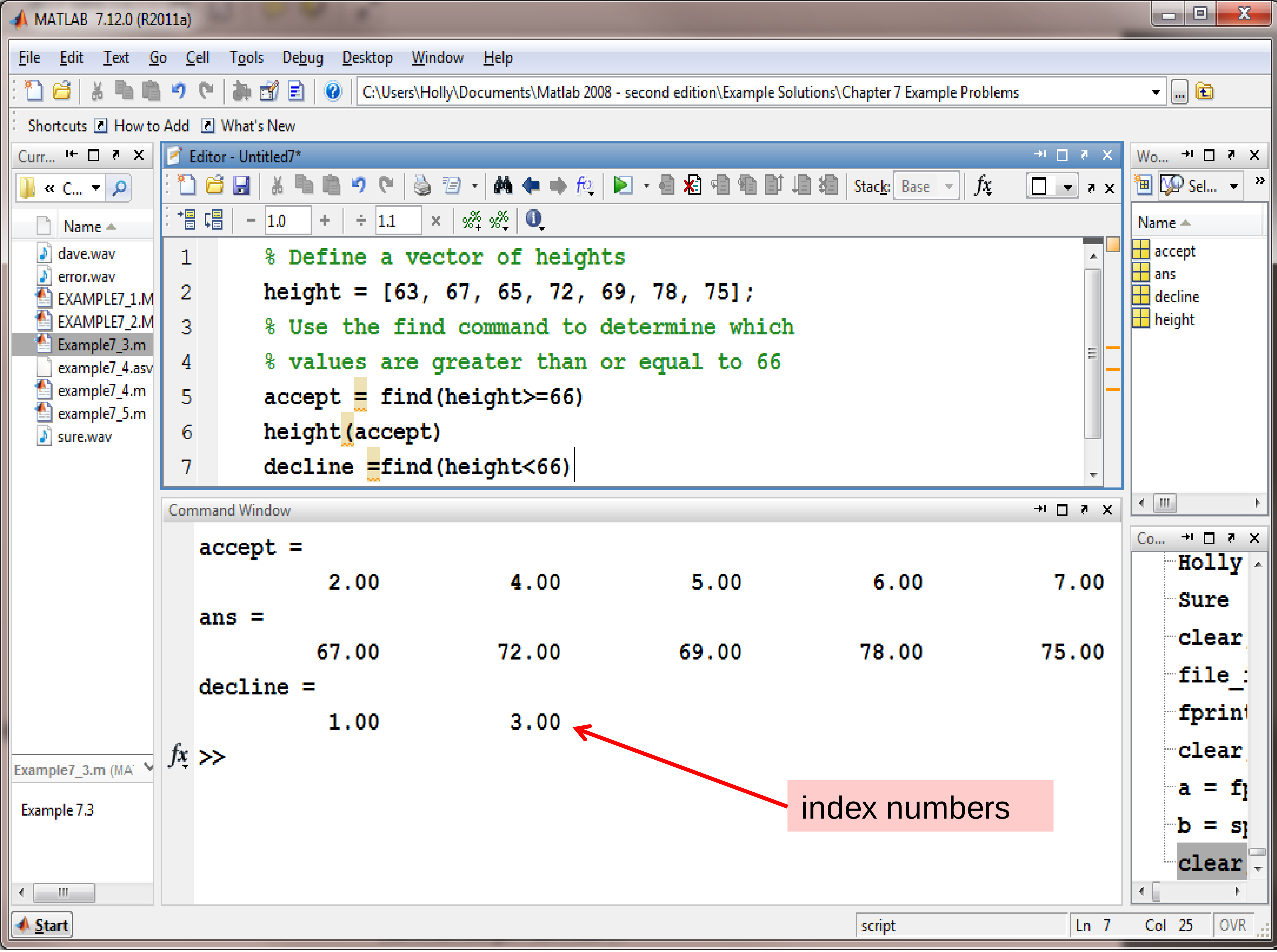


```
% Define a vector of heights
height = [63, 67, 65, 72, 69, 78, 75];
% Use the find command to determine which
% values are greater than or equal to 66
accept = find(height>=66)
height(accept)
```

```
accept =
          2.00          4.00          5.00          6.00          7.00
ans =
         67.00         72.00         69.00         78.00         75.00
>>
```

index numbers

element values

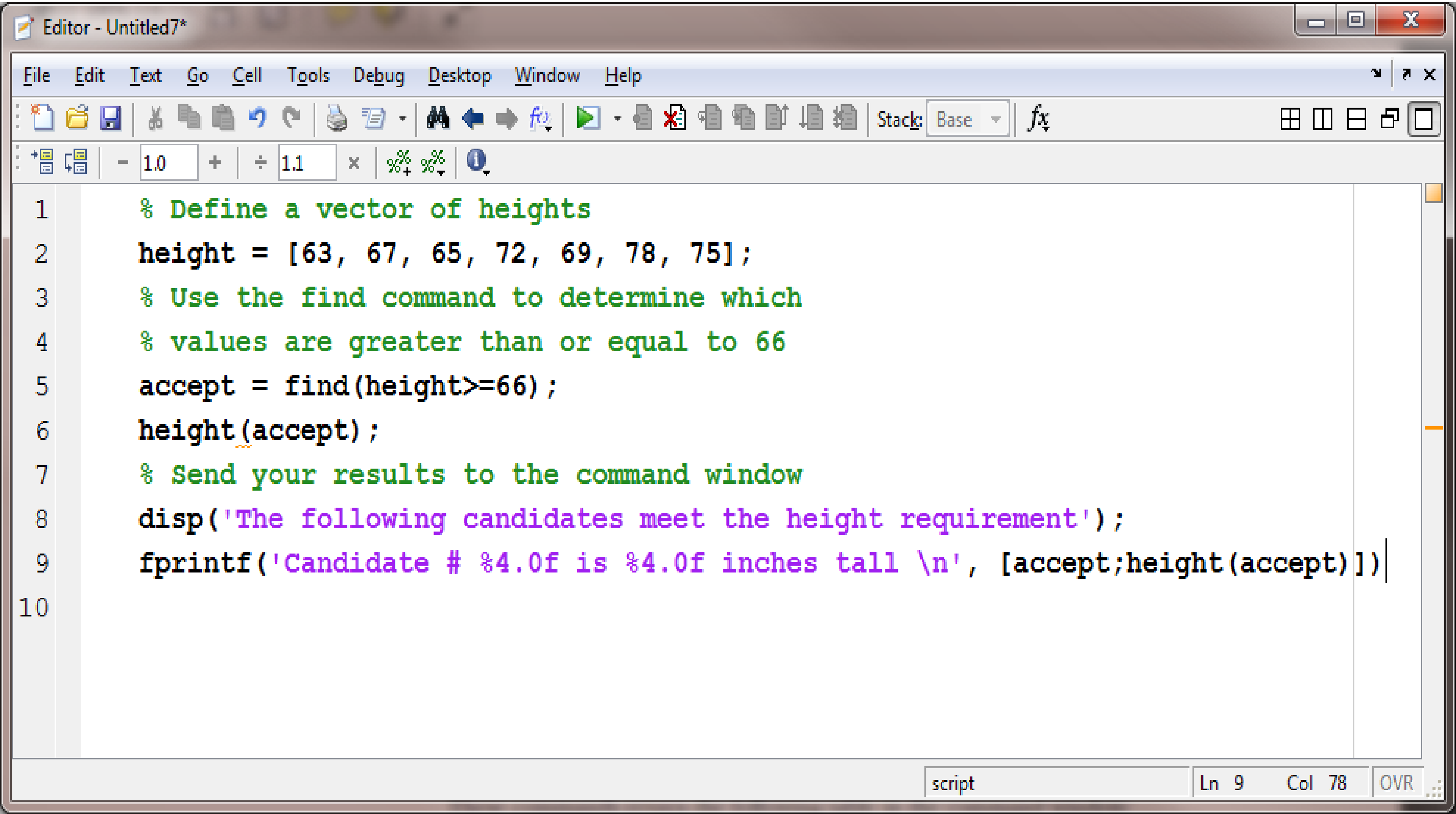


```
% Define a vector of heights
height = [63, 67, 65, 72, 69, 78, 75];
% Use the find command to determine which
% values are greater than or equal to 66
accept = find(height>=66)
height(accept)
decline =find(height<66)
```

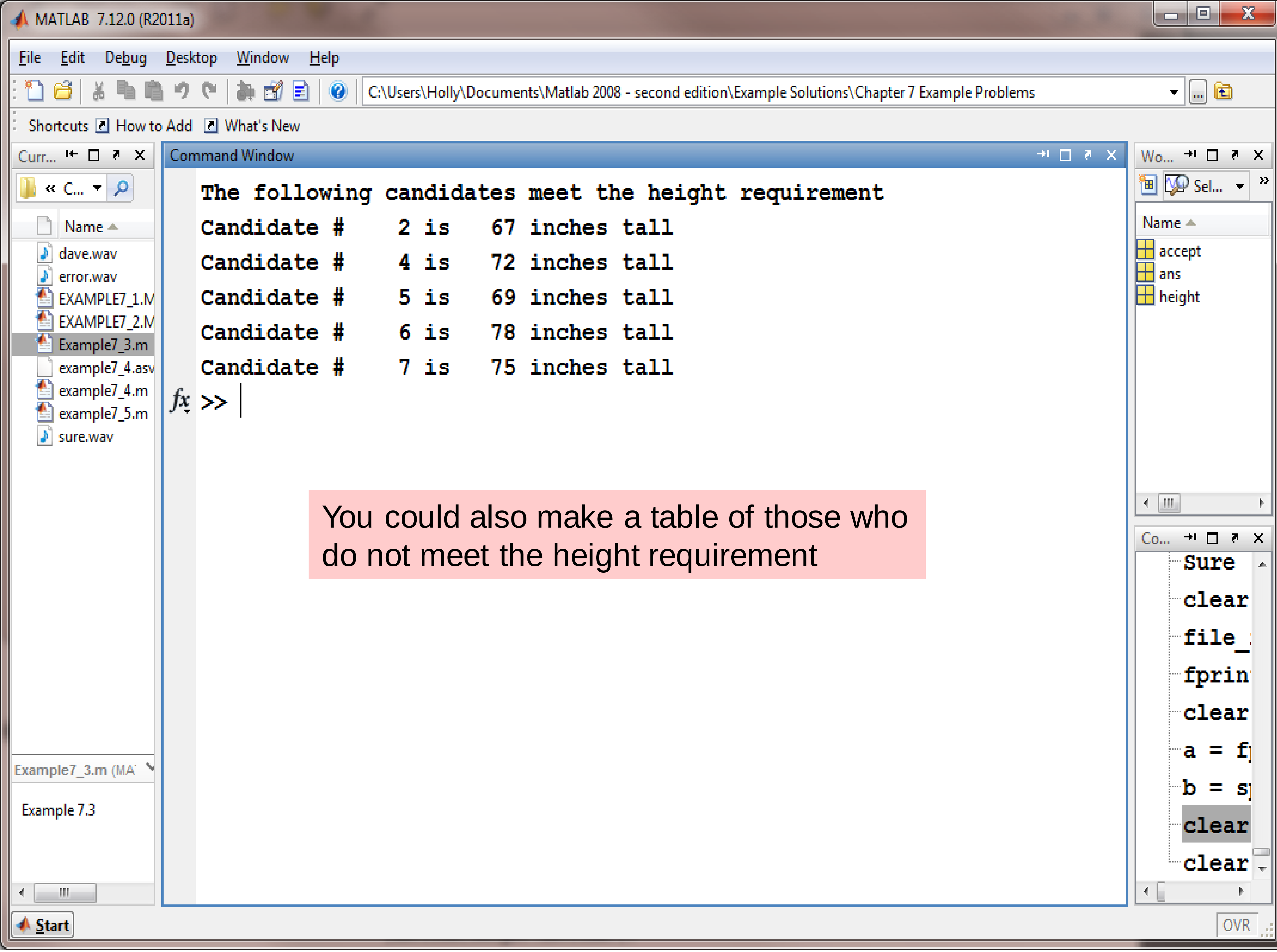
```
accept =
            2.00          4.00          5.00          6.00          7.00
ans =
           67.00         72.00         69.00         78.00         75.00
decline =
            1.00          3.00
fx >>
```

index numbers

You could use the disp and fprintf functions in this program to create a more readable report



```
% Define a vector of heights
height = [63, 67, 65, 72, 69, 78, 75];
% Use the find command to determine which
% values are greater than or equal to 66
accept = find(height>=66);
height(accept);
% Send your results to the command window
disp('The following candidates meet the height requirement');
fprintf('Candidate # %4.0f is %4.0f inches tall \n', [accept;height(accept)])
```



```
The following candidates meet the height requirement
Candidate #    2 is   67 inches tall
Candidate #    4 is   72 inches tall
Candidate #    5 is   69 inches tall
Candidate #    6 is   78 inches tall
Candidate #    7 is   75 inches tall
>>
```

You could also make a table of those who do not meet the height requirement

By combining relational and logical operators you can create fairly complicated search criteria

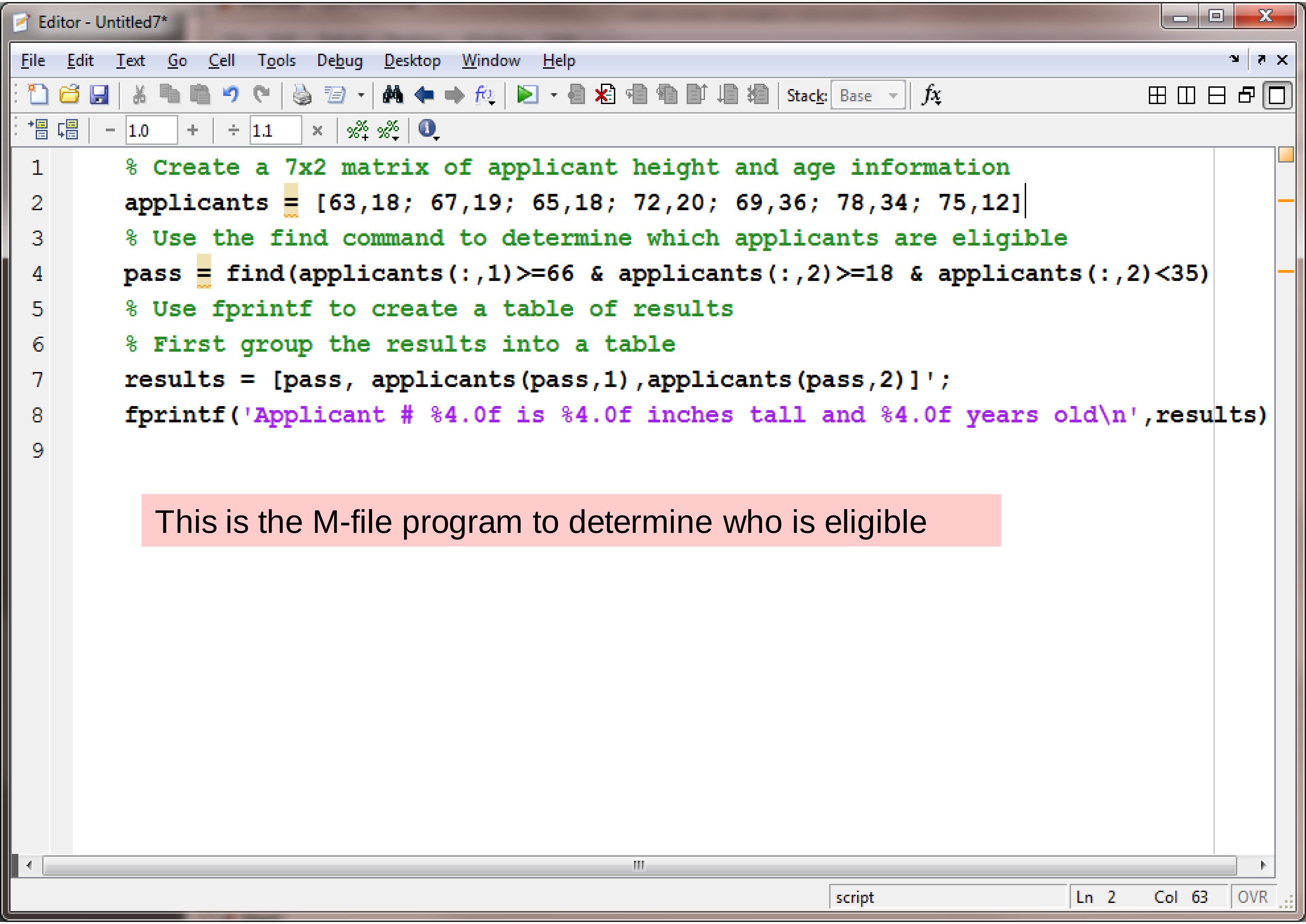
- Assume applicants must be at least 18 years old and less than 35 years old
- They must also meet the height requirement

Applicant pool

Height Inches	Age years
63	18
67	19
65	18
72	20
69	36
78	34
75	12

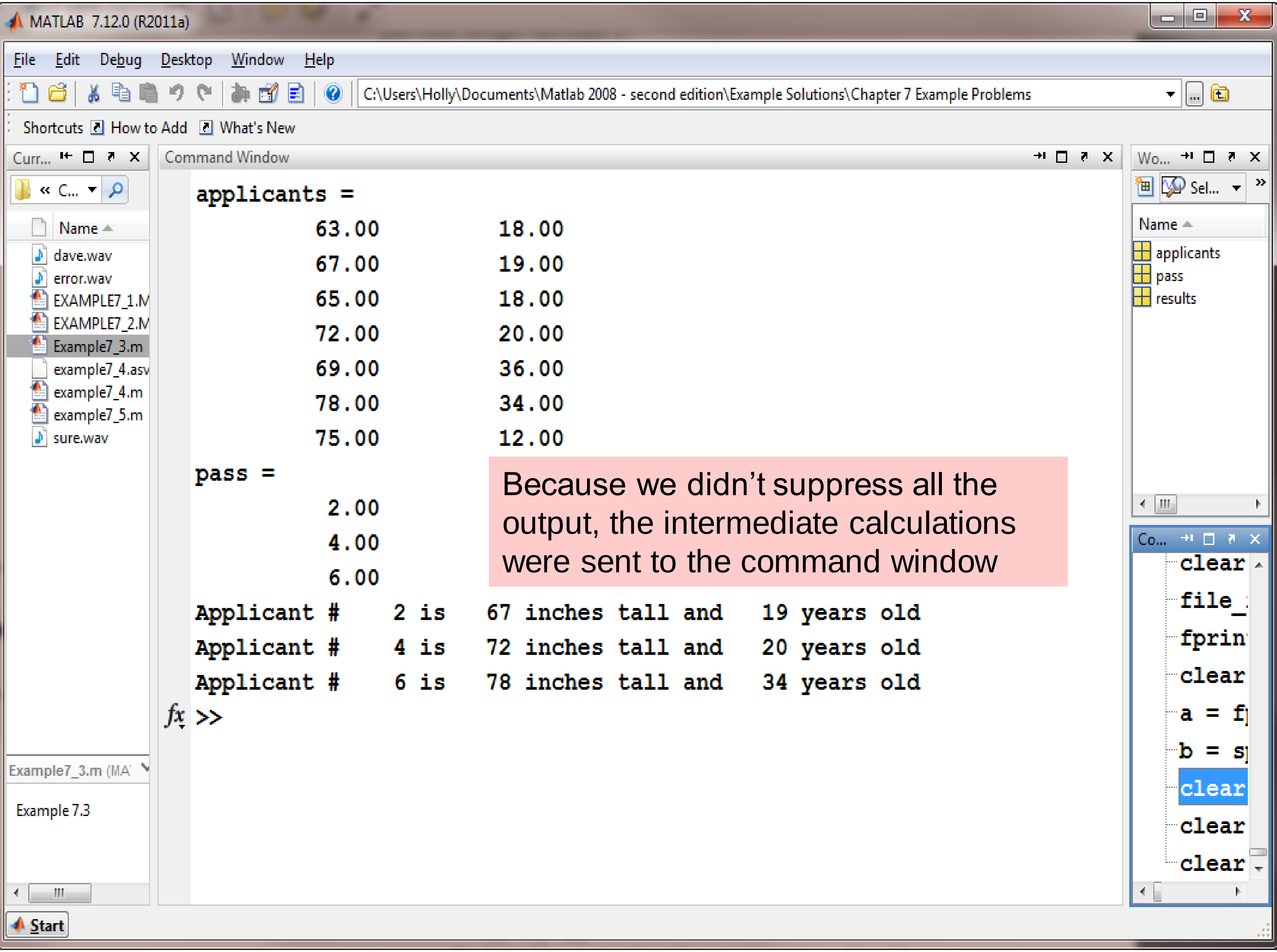
Let’s use Pseudo-code to plan this program

- Create a 7x2 matrix of applicant height and age information
- Use the find command to determine which applicants are eligible
- Use fprintf to create a table of results



```
    % Create a 7x2 matrix of applicant height and age information
    applicants = [63,18; 67,19; 65,18; 72,20; 69,36; 78,34; 75,12]
    % Use the find command to determine which applicants are eligible
    pass = find(applicants(:,1)>=66 & applicants(:,2)>=18 & applicants(:,2)<35)
    % Use fprintf to create a table of results
    % First group the results into a table
    results = [pass, applicants(pass,1),applicants(pass,2)]';
    fprintf('Applicant # %4.0f is %4.0f inches tall and %4.0f years old\n',results)
```

This is the M-file program to determine who is eligible



```
applicants =
          63.00          18.00
          67.00          19.00
          65.00          18.00
          72.00          20.00
          69.00          36.00
          78.00          34.00
          75.00          12.00
pass =
           2.00
           4.00
           6.00
Applicant #    2 is   67 inches tall and   19 years old
Applicant #    4 is   72 inches tall and   20 years old
Applicant #    6 is   78 inches tall and   34 years old
>>
```

Because we didn’t suppress all the output, the intermediate calculations were sent to the command window

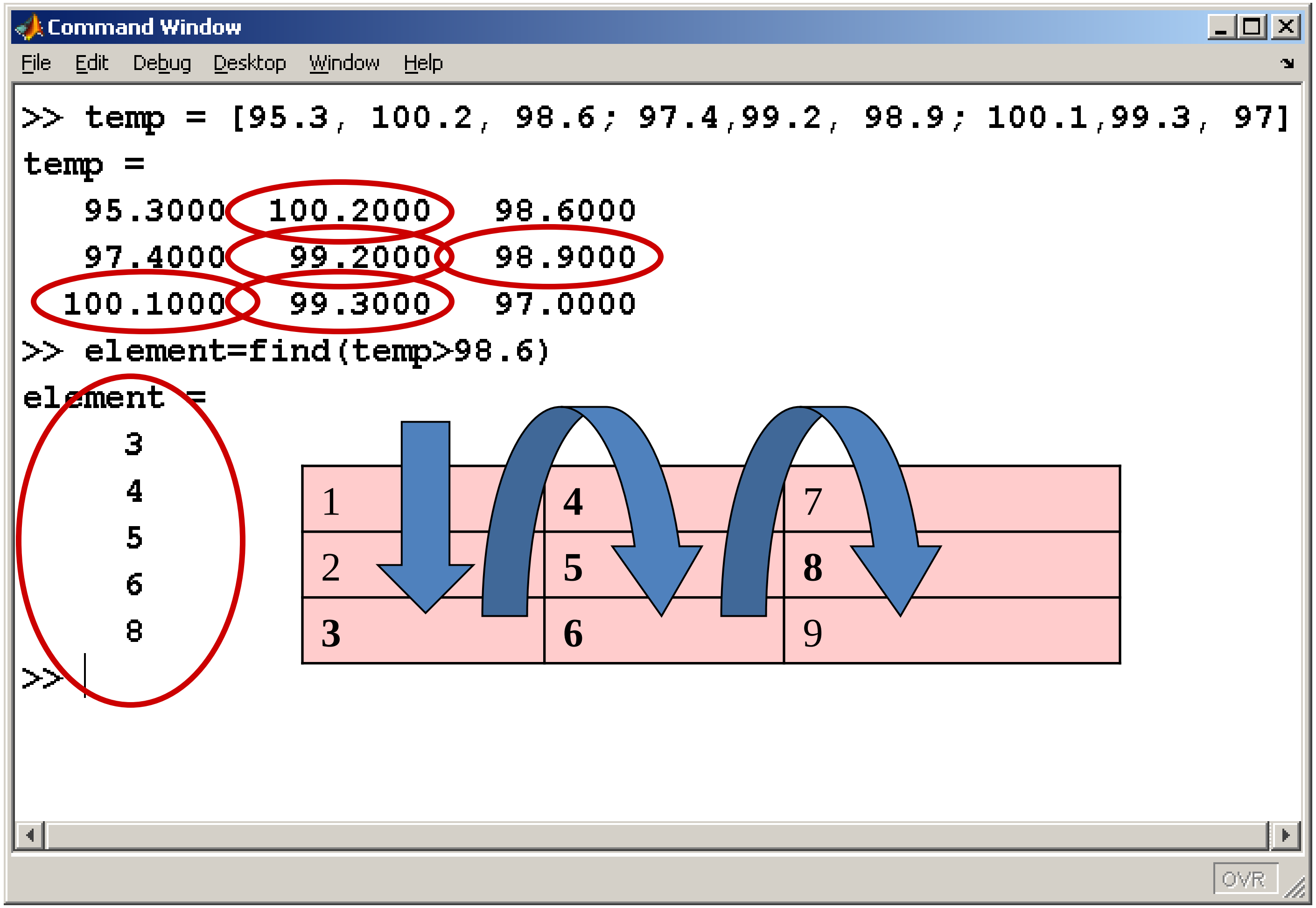
The find command can return either…

- A single index number identifying an element in a matrix
- A matrix of the row numbers and the column numbers identifying an element in a matrix
  - You need to specify two results if you want the row and column designation
  - **[row, col] = find(*criteria)***

Imagine you have a matrix of patient temperature values measured in a clinic

Station 1	Station 2	Station 3
95.3	100.2	98.6
97.4	99.2	98.9
100.1	99.3	97

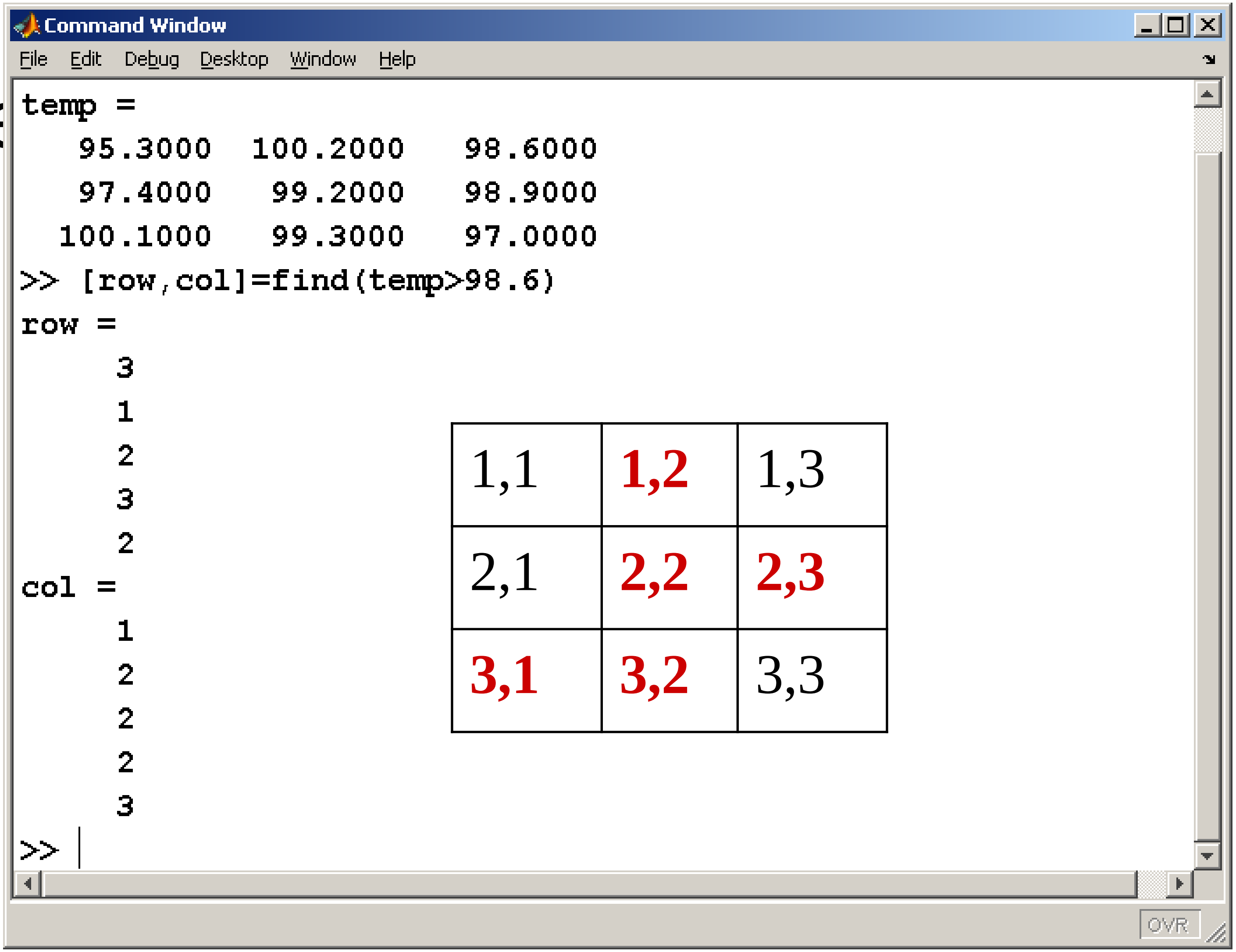
Use the find command to determine which patients have elevated temperatures



```
>> temp = [95.3, 100.2, 98.6; 97.4,99.2, 98.9; 100.1,99.3, 97]
temp =
   95.3000  100.2000   98.6000
   97.4000   99.2000   98.9000
  100.1000   99.3000   97.0000
>> element=find(temp>98.6)
element =
     3
     4
     5
     6
     8
>>
```

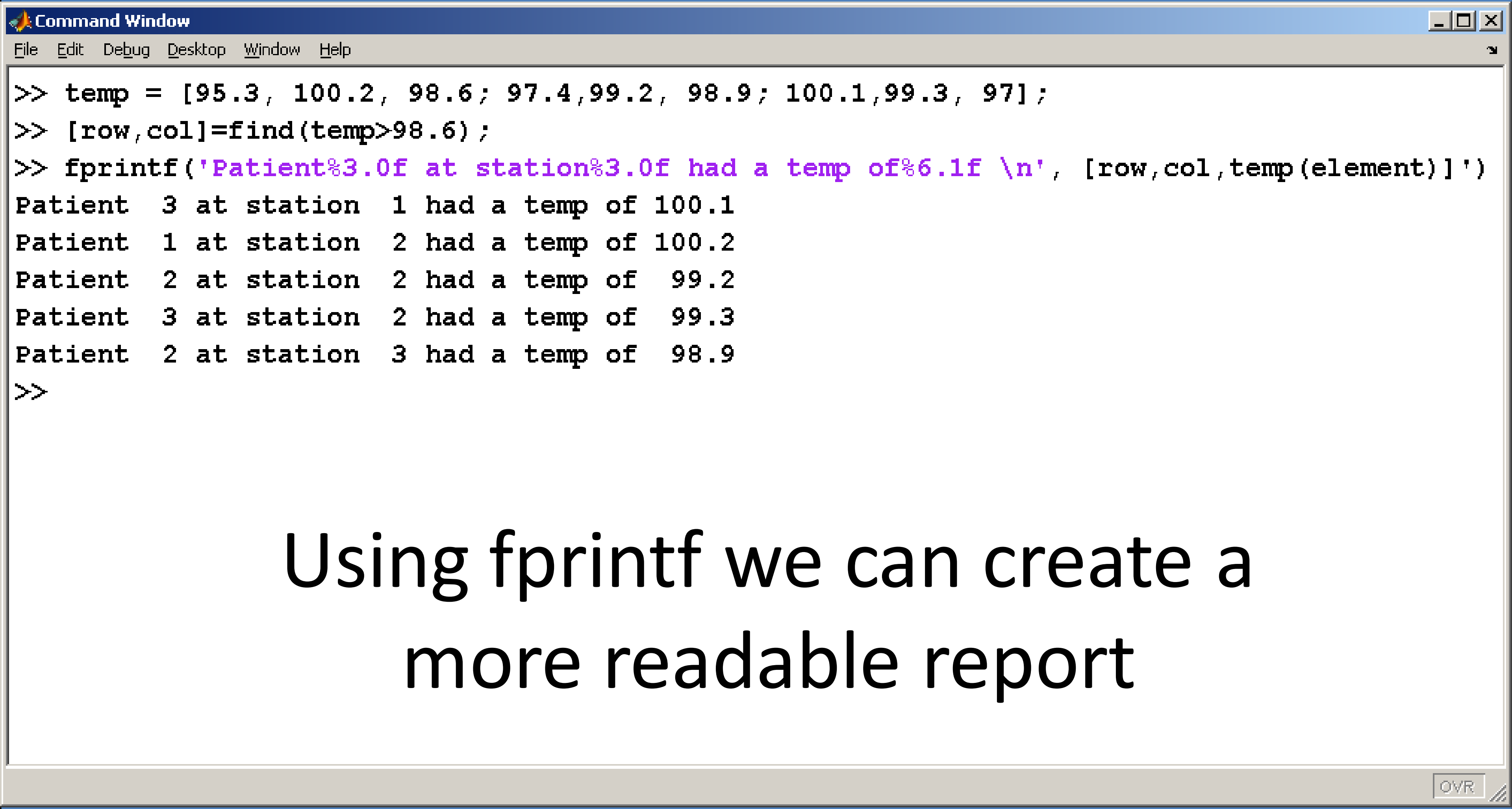
1	**4**	7
2	**5**	**8**
3	**6**	9

If we



```
temp =
   95.3000  100.2000   98.6000
   97.4000   99.2000   98.9000
  100.1000   99.3000   97.0000
>> [row,col]=find(temp>98.6)
row =
     3
     1
     2
     3
     2
col =
     1
     2
     2
     2
     3
>> 
```

1,1	**1,2**	1,3
2,1	**2,2**	**2,3**
3,1	**3,2**	3,3



```
>> temp = [95.3, 100.2, 98.6; 97.4,99.2, 98.9; 100.1,99.3, 97];
>> [row,col]=find(temp>98.6);
>> fprintf('Patient%3.0f at station%3.0f had a temp of%6.1f \n', [row,col,temp(element)]')
Patient  3 at station  1 had a temp of 100.1
Patient  1 at station  2 had a temp of 100.2
Patient  2 at station  2 had a temp of  99.2
Patient  3 at station  2 had a temp of  99.3
Patient  2 at station  3 had a temp of  98.9
>>
```

Using fprintf we can create a more readable report

7.4 Selection Structures

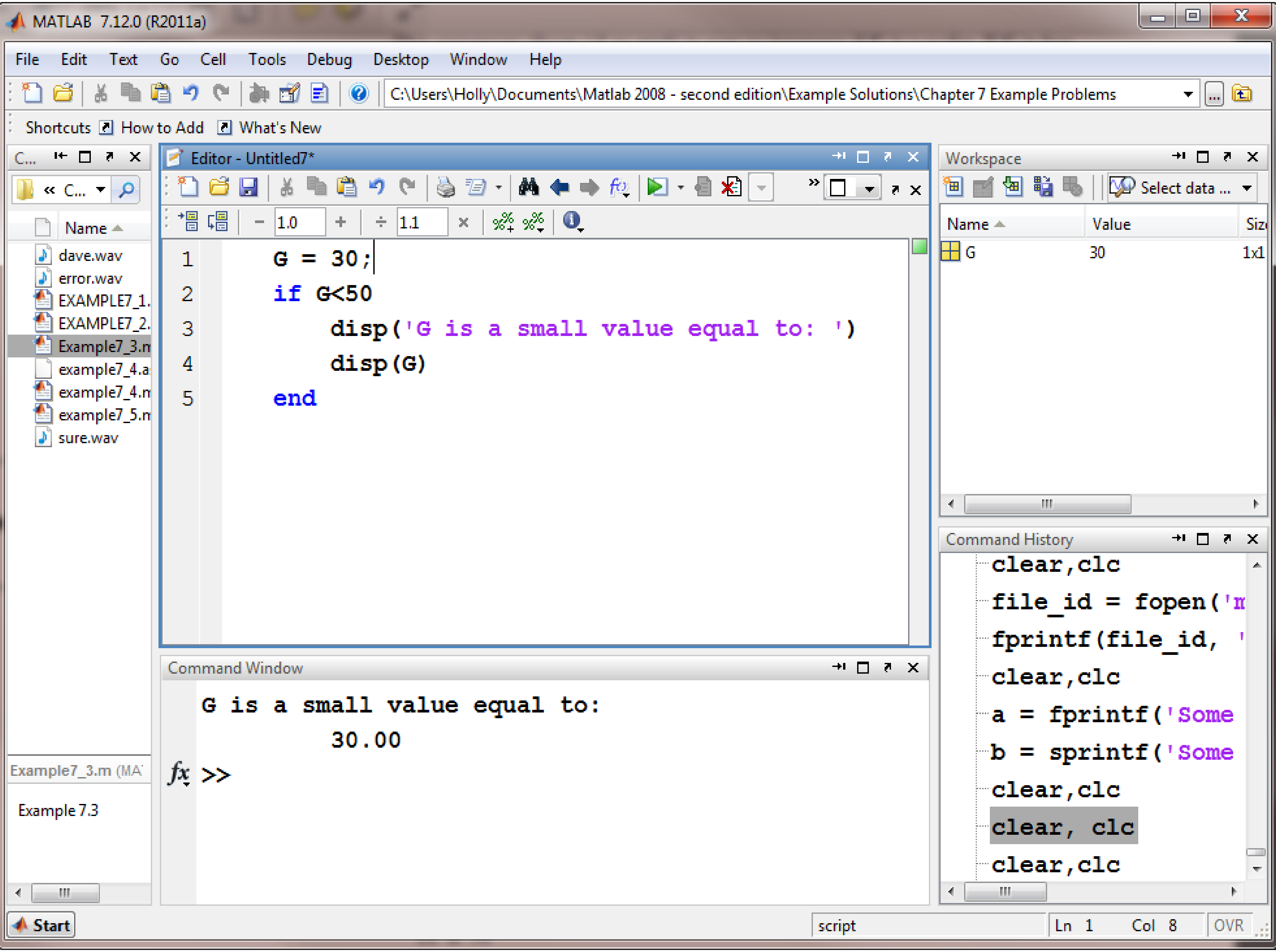
- Most of the time the **find** function should be used instead of an **if**
- However, there are certain situations where **if** is the appropriate process to use

Simple if

```
if comparison
      statements
end
```

For example….

```
if G<50
   disp(‘G is a small value equal to:’)
   disp(G);
end
```



```
G = 30;
if G<50
    disp('G is a small value equal to: ')
    disp(G)
end
```

Name	Value	Siz
G	30	1x1

```
G is a small value equal to:
        30.00

>>
```

```
clear,clc
file_id = fopen('m
fprintf(file_id, '
clear,clc
a = fprintf('Some
b = sprintf('Some
clear,clc
clear, clc
clear,clc
```

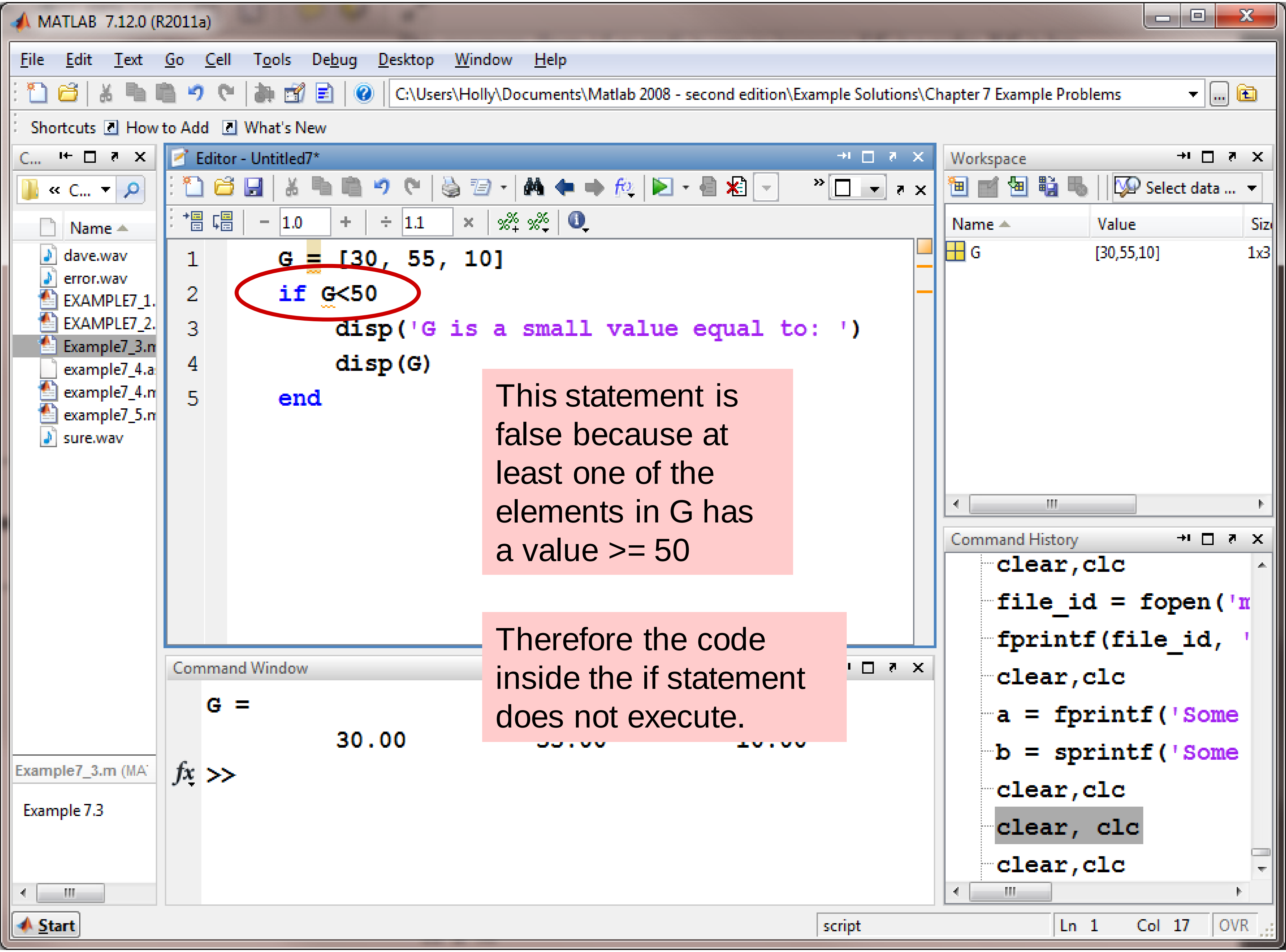
If statements

- Easy to interpret for scalars
- What does an **if** statement mean if the comparison includes a matrix?
  - The comparison is only true if it is true for **every** member of the array

Consider this bit of code

```
 G=[30,55,10]
 if G<50
   disp(‘G is a small value equal to:’)
   disp(G);
end
```

The code inside the if statement is not executed, because the comparison is not true!!



```
G = [30, 55, 10]
if G<50
    disp('G is a small value equal to: ')
    disp(G)
end
```

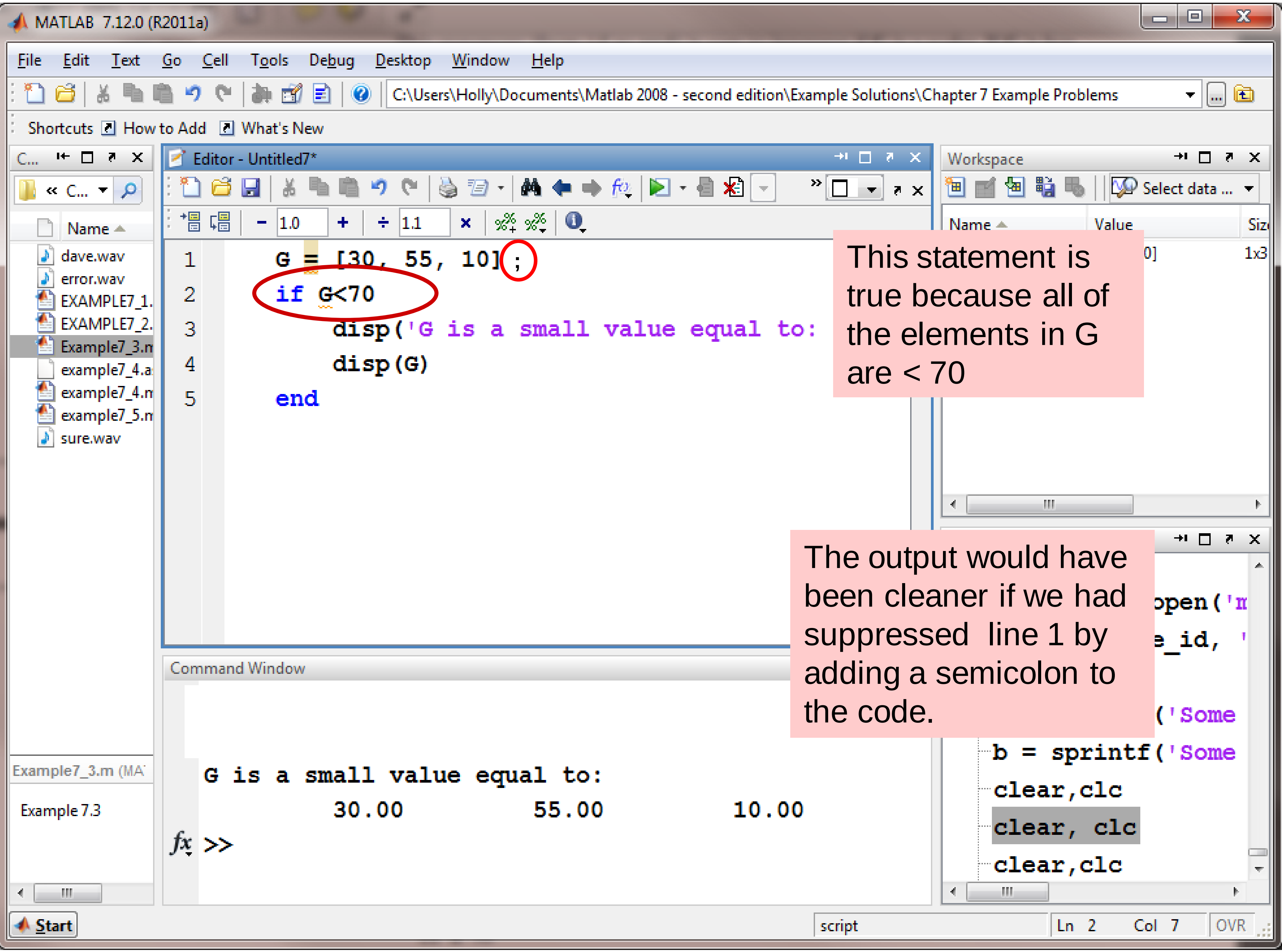
This statement is false because at least one of the elements in G has a value >= 50

Therefore the code inside the if statement does not execute.

Name	Value	Size
G	[30,55,10]	1x3

```
G =

          30.00        55.00        10.00

>>
```

```
clear,clc
file_id = fopen('m
fprintf(file_id, '
clear,clc
a = fprintf('Some
b = sprintf('Some
clear,clc
clear, clc
clear,clc
```



```
1    G = [30, 55, 10];
2    if G<70
3        disp('G is a small value equal to:
4        disp(G)
5    end
```

This statement is true because all of the elements in G are < 70

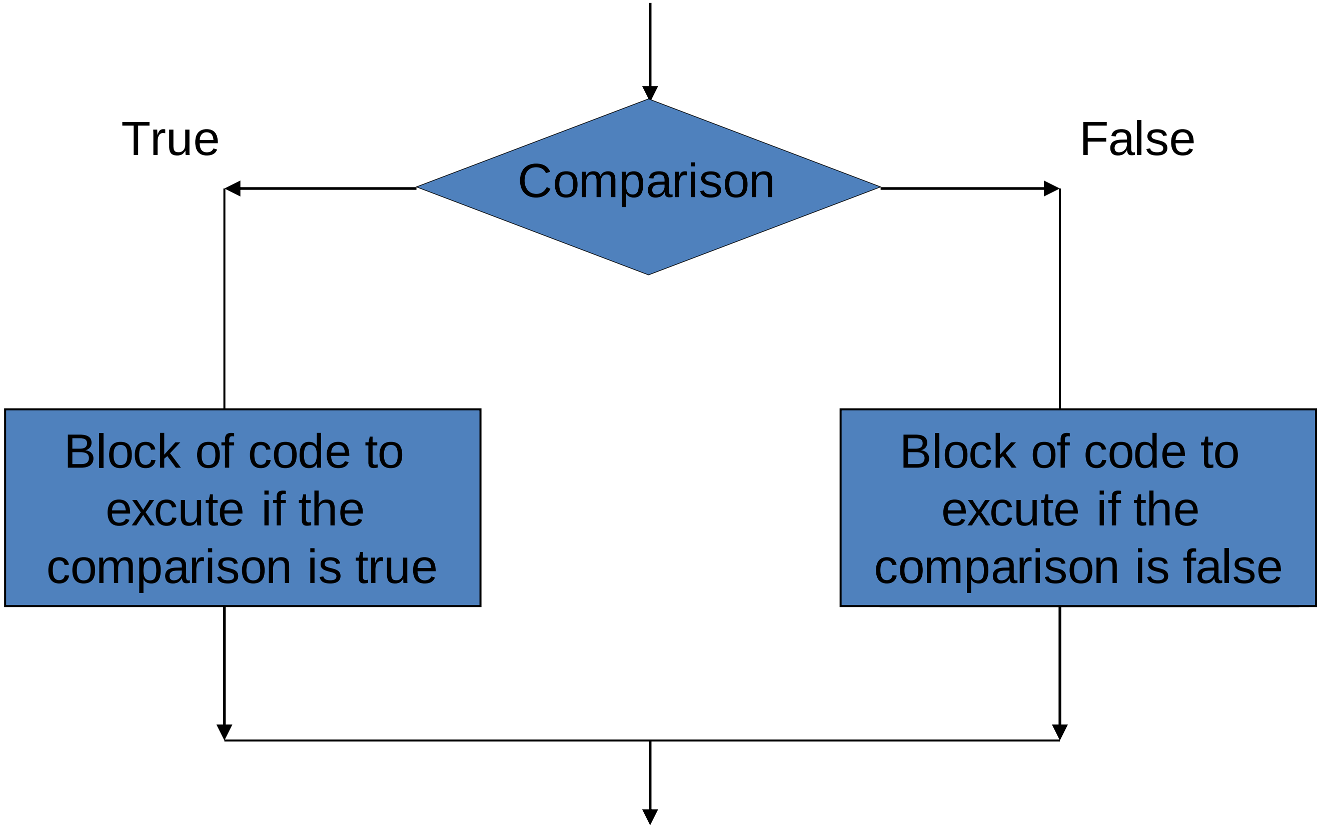
```
G is a small value equal to:
        30.00          55.00          10.00
fx >>
```

The output would have been cleaner if we had suppressed line 1 by adding a semicolon to the code.

The if/else structure

- The simple if triggers the execution of a block of code if a condition is true
- If it is false that block of code is skipped, and the program continues without doing anything
- What if instead you want to execute an alternate set of code if the condition is false?

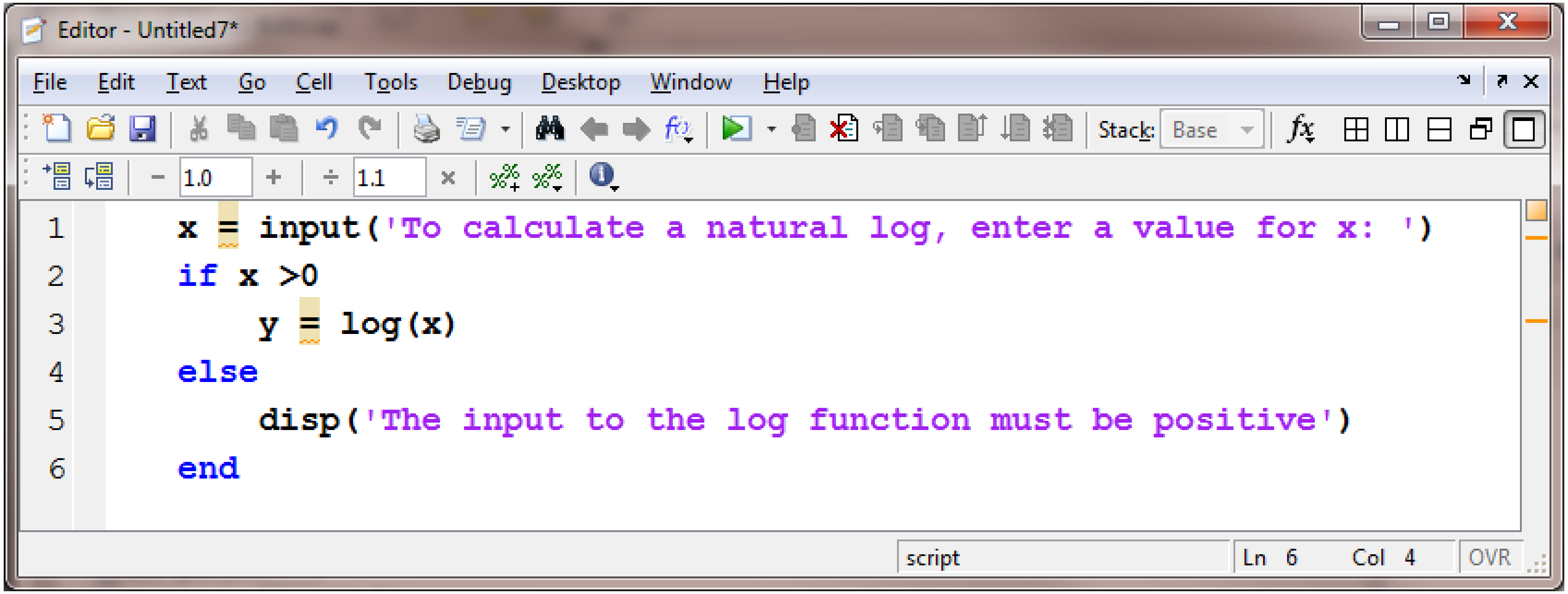
Flow chart of an if/else structure


True
Comparison
False
Block of code to excute if the comparison is true
Block of code to excute if the comparison is false

Use an if structure to calculate a natural log

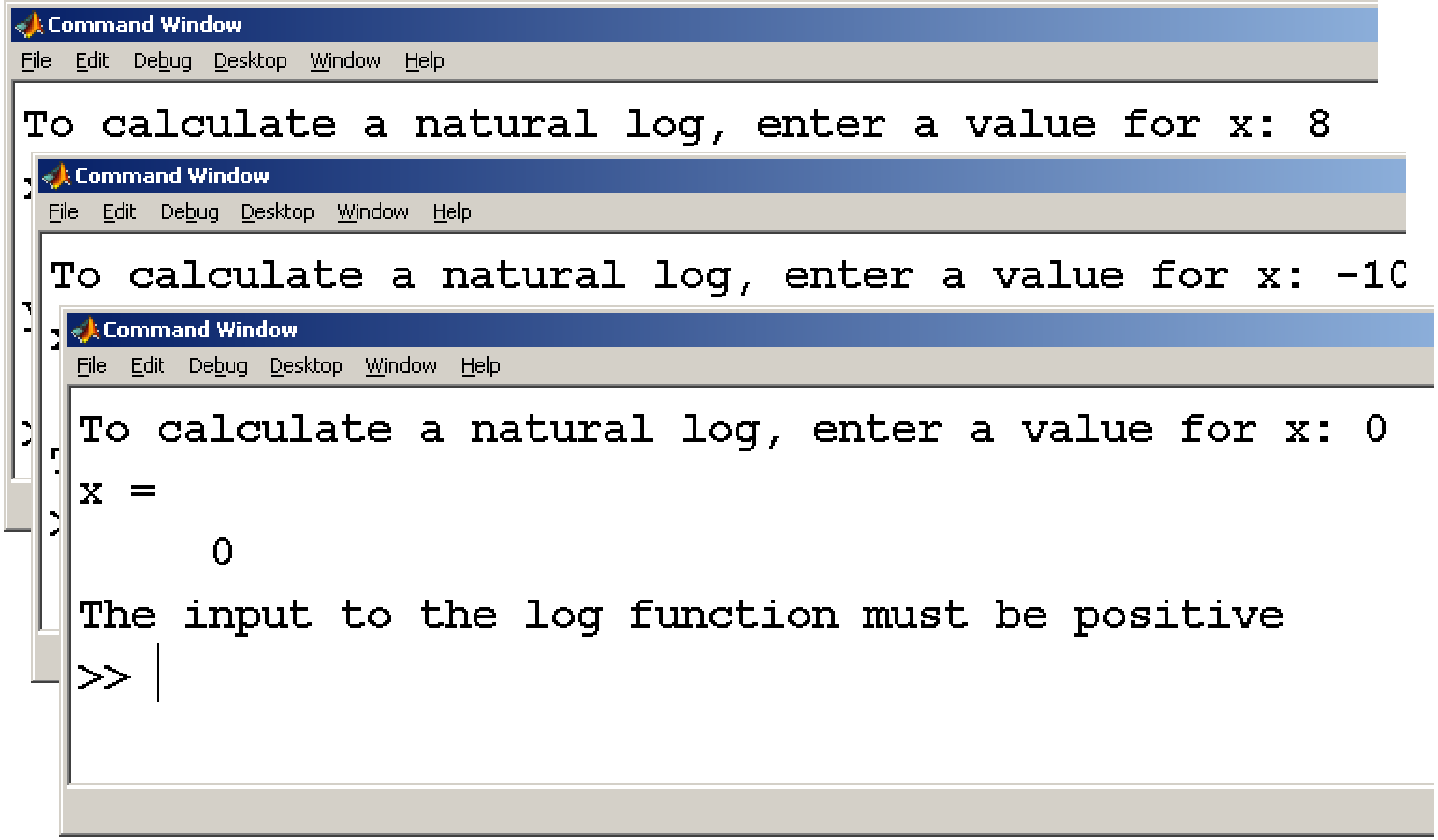
- Check to see if the input is positive
  - If it is, calculate the natural log
  - If it isn’t, send an error message to the screen

M-file Program



```
    x = input('To calculate a natural log, enter a value for x: ')
    if x >0
        y = log(x)
    else
        disp('The input to the log function must be positive')
    end
```

Interactions in the Command Window


Command Window
File Edit Debug Desktop Window Help

```
To calculate a natural log, enter a value for x: 8
```

Command Window
File Edit Debug Desktop Window Help

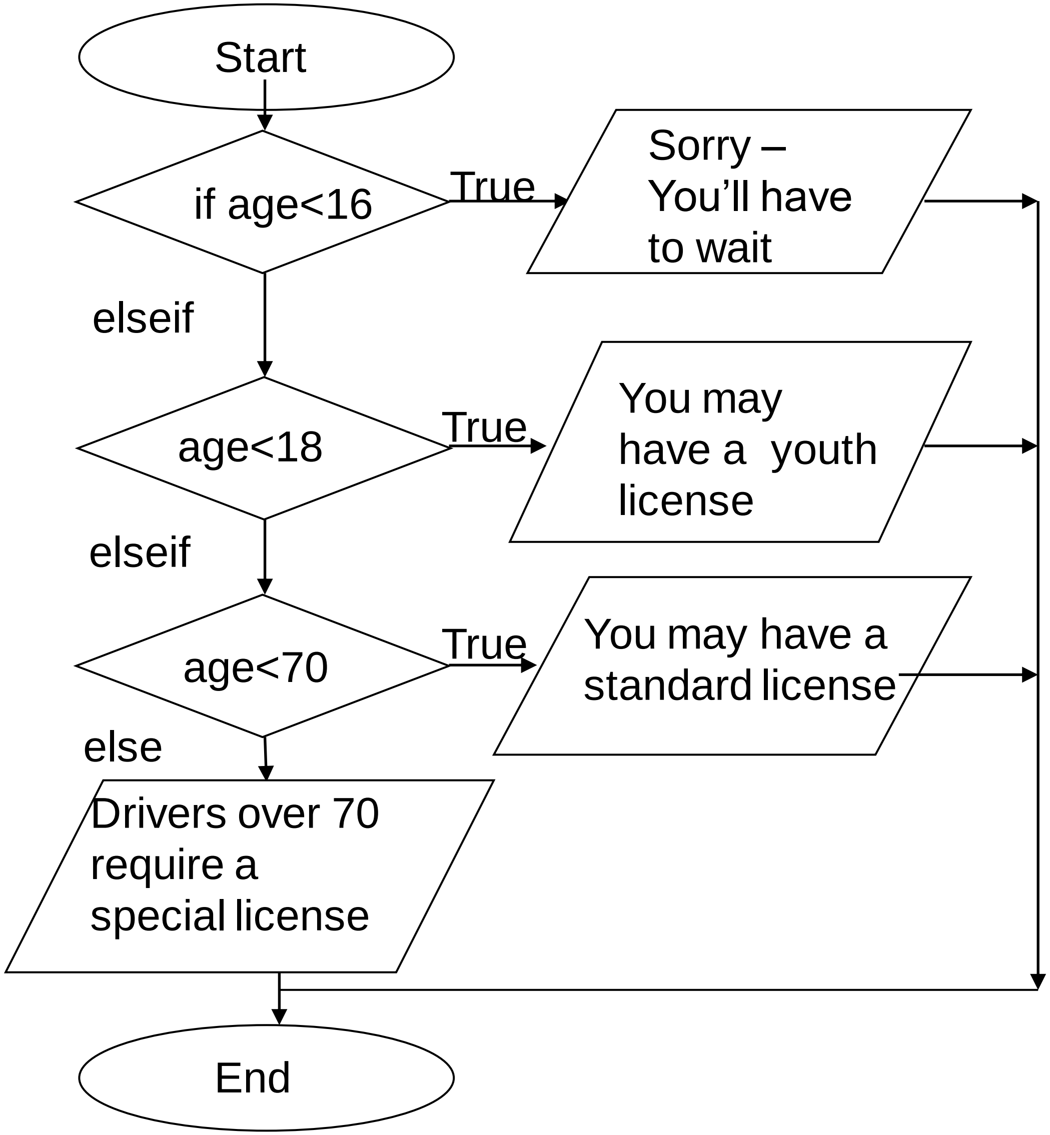
```
To calculate a natural log, enter a value for x: -10
```

Command Window
File Edit Debug Desktop Window Help

```
To calculate a natural log, enter a value for x: 0
x =

     0

The input to the log function must be positive
>> 
```

The if/else/elseif structure

- Use the elseif for multiple selection criteria
- For example
  - Write a program to determine if an applicant is eligible to drive



Start

if age<16

True

Sorry – You’ll have to wait

elseif

age<18

True

You may have a youth license

elseif

age<70

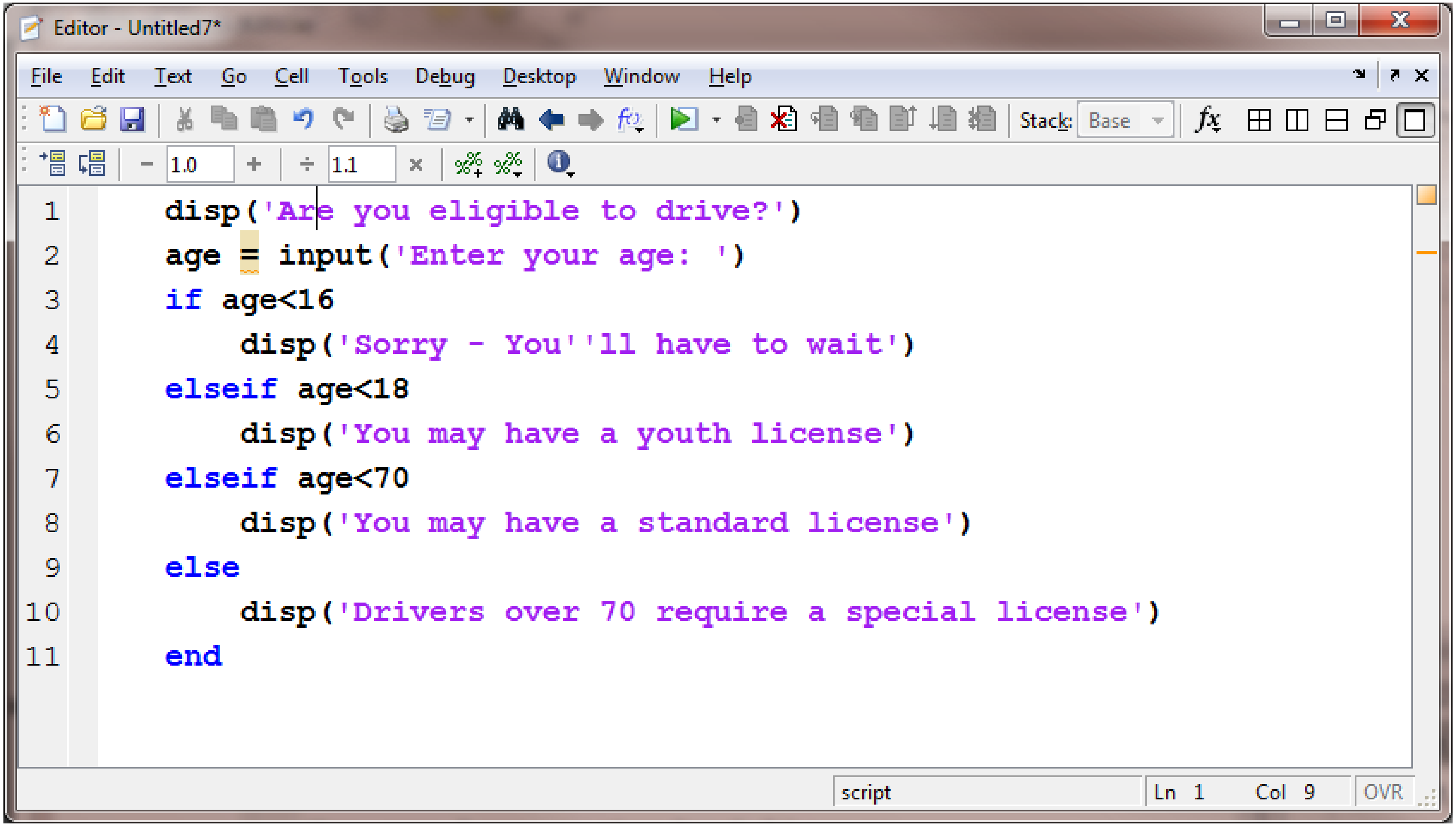
True

You may have a standard license

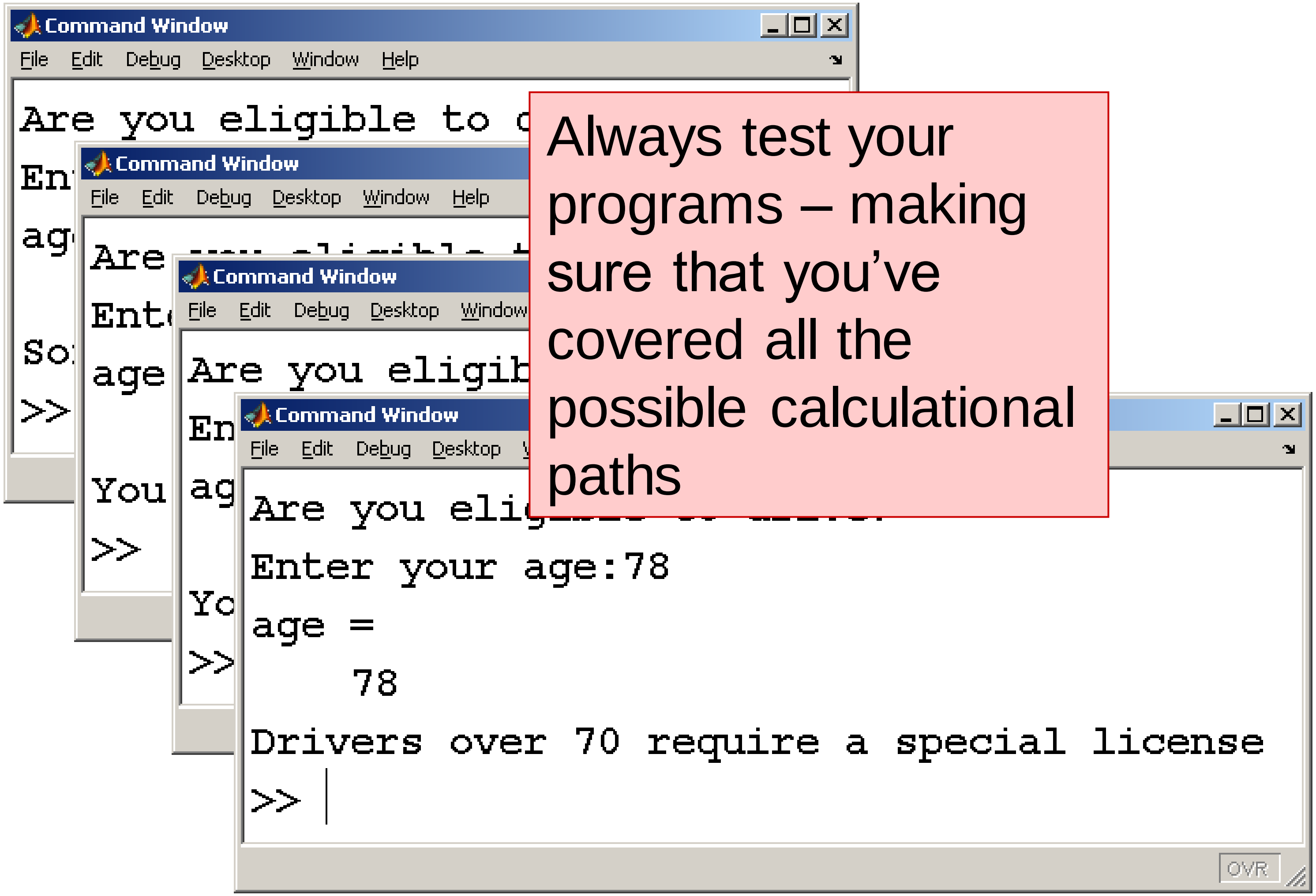
else

Drivers over 70 require a special license

End



```
disp('Are you eligible to drive?')
age = input('Enter your age: ')
if age<16
    disp('Sorry - You''ll have to wait')
elseif age<18
    disp('You may have a youth license')
elseif age<70
    disp('You may have a standard license')
else
    disp('Drivers over 70 require a special license')
end
```



```
Are you eligible to drive?
Enter your age:78
age =
    78
Drivers over 70 require a special license
>>
```

Always test your programs – making sure that you've covered all the possible calculational paths

As a general rule…

- If structures work well for scalars
- For vectors or arrays use a find function or..
- Combine if structures with a repetition structure
- Repetition structures are introduced in the next chapter

switch/case

- This structure is an alternative to the if/else/elseif structure
- The code is generally easier to read
- This structure allows you to choose between multiple outcomes, based on some criterion, which must be exactly true

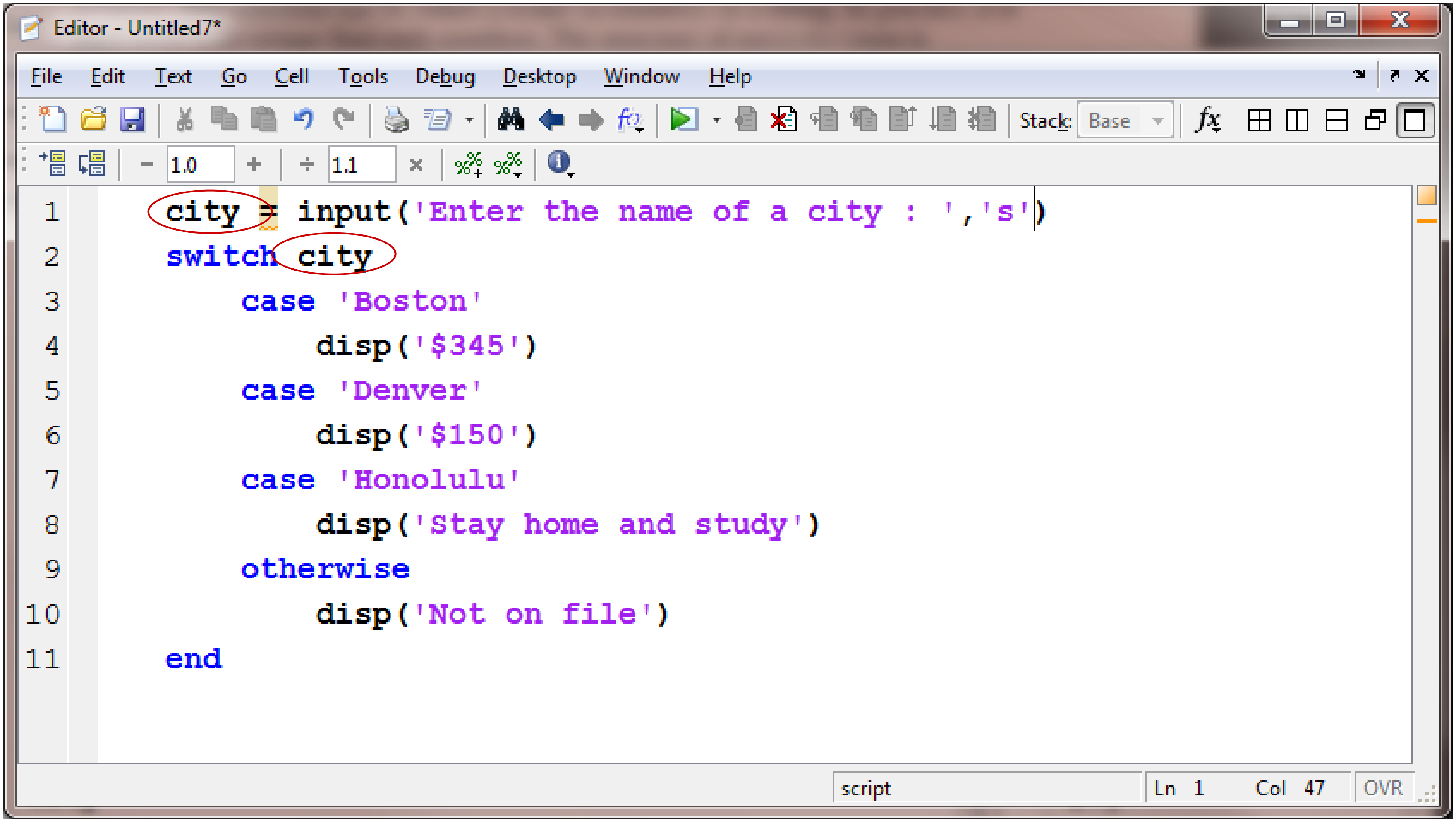
When to use switch/case

- The criterion can be either a scalar (a number) or a string.
- In practice, it is used more with strings than with numbers.

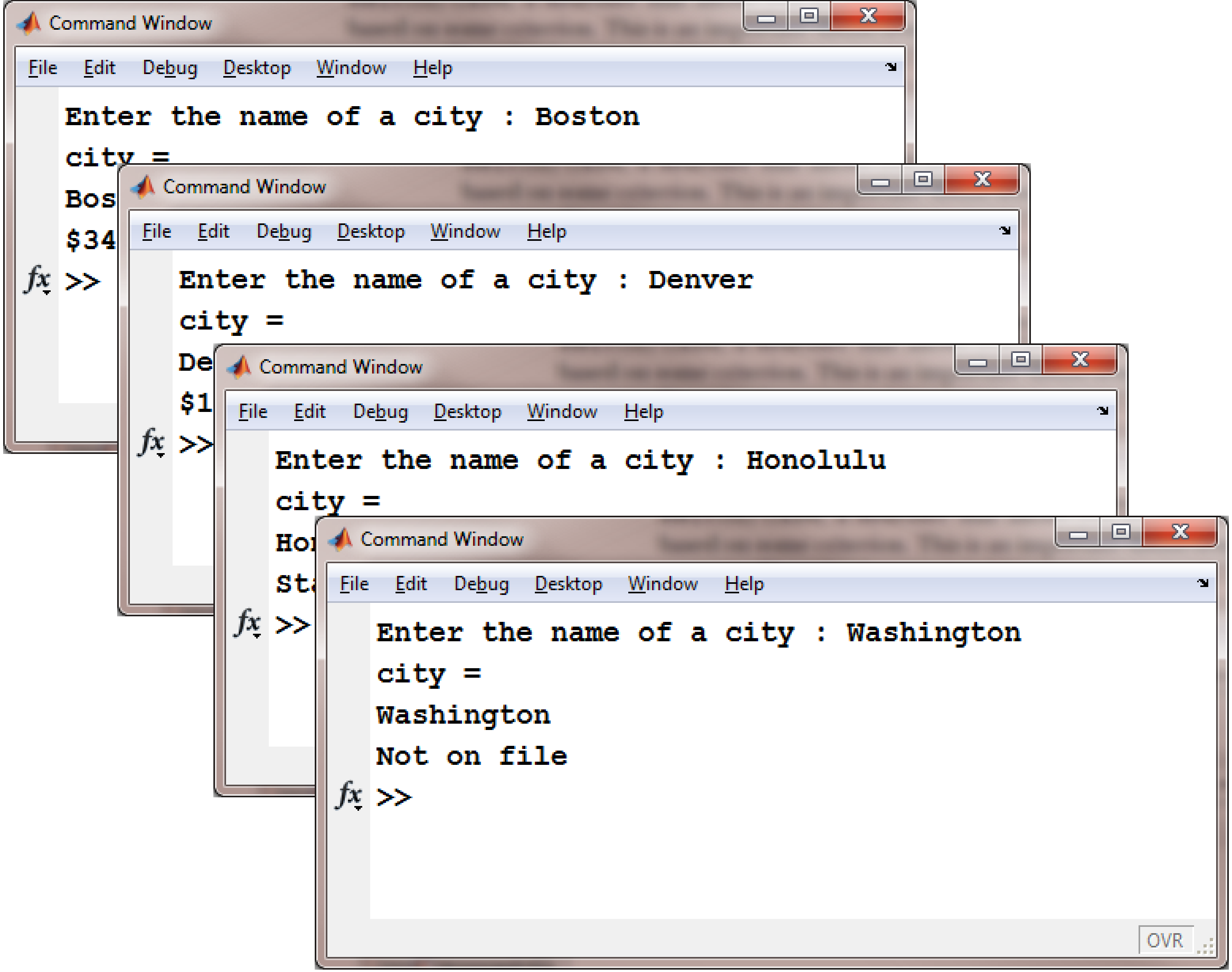
The structure of switch/case

```
switch variable
case option1
        code to be executed if variable is exactly
        equal to option 1
case option2
        code to be executed if variable is exactly
        equal to option 2
        …
case option_n
        code to be executed if variable is exactly
        equal to option n
otherwise
        code to be executed if variable is not
        equal to any of the options
end
```

Suppose you want to determine what the airfare is to one of three cities



```
city = input('Enter the name of a city : ','s')
switch city
    case 'Boston'
        disp('$345')
    case 'Denver'
        disp('$150')
    case 'Honolulu'
        disp('Stay home and study')
    otherwise
        disp('Not on file')
end
```



```
Enter the name of a city : Boston
city =
Bos
$34
>>
```

```
Enter the name of a city : Denver
city =
De
$1
>>
```

```
Enter the name of a city : Honolulu
city =
Ho
St
>>
```

```
Enter the name of a city : Washington
city =
Washington
Not on file
>>
```

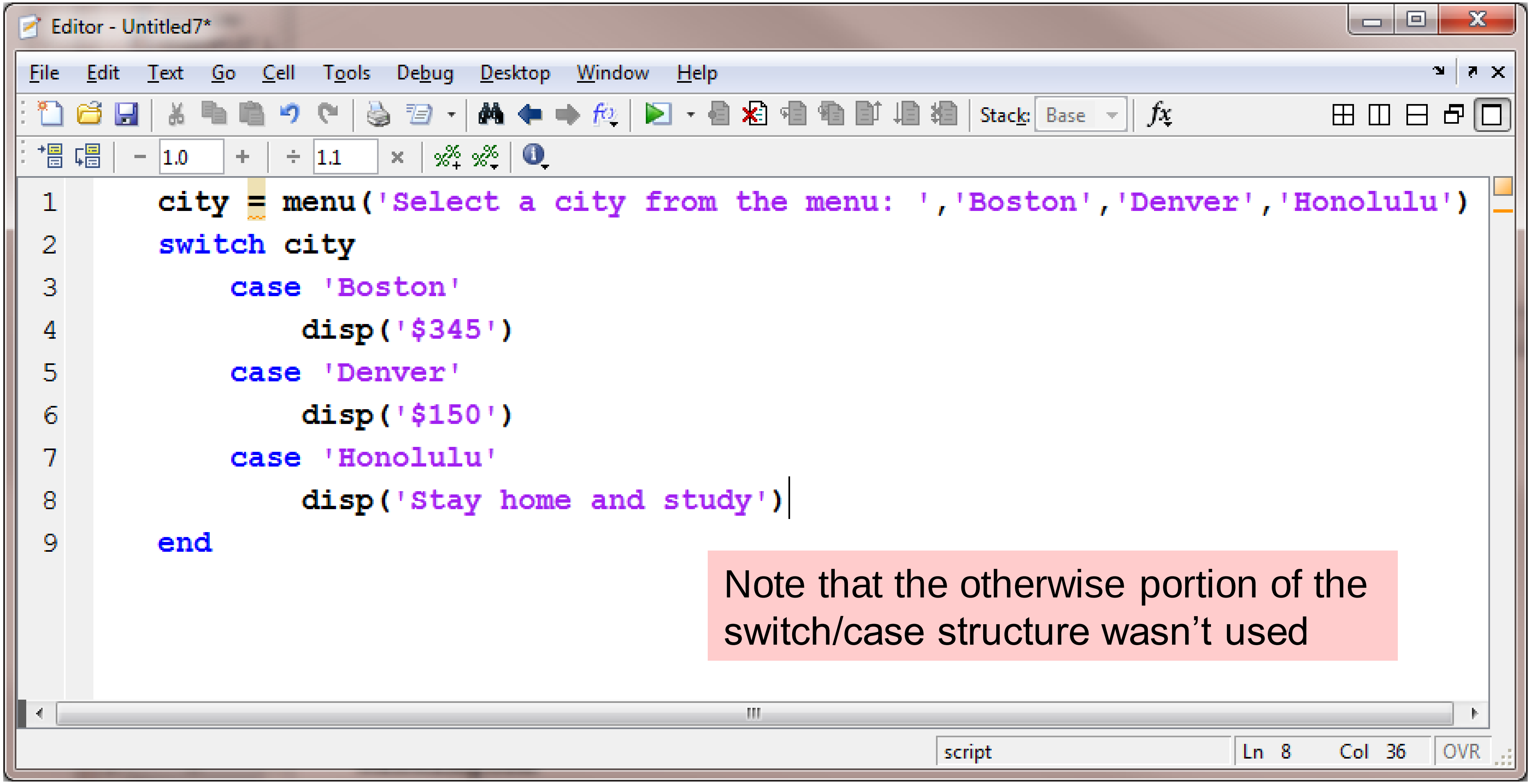
Remember…You tell the **input** command to expect a string by adding ‘s’ in the second field.

Menu

- The menu function is often used in conjunction with a **switch/case** structure.
- This function causes a menu box to appear on the screen with a series of buttons defined by the programmer.

```
input = menu(‘Message to the user’,’text for button 1’,’text for button 2’, etc)
```

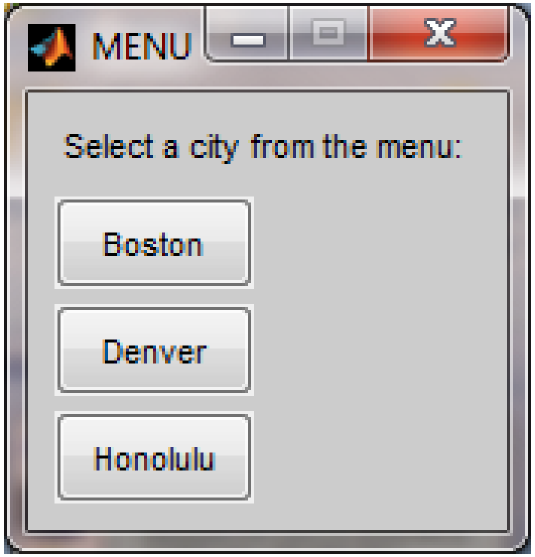
- Because the input is controlled by a menu box, the user can’t accidentally enter a bad choice
- This means you don’t need the otherwise portion of the switch/case structure



```
city = menu('Select a city from the menu: ','Boston','Denver','Honolulu')
switch city
    case 'Boston'
        disp('$345')
    case 'Denver'
        disp('$150')
    case 'Honolulu'
        disp('Stay home and study')
end
```

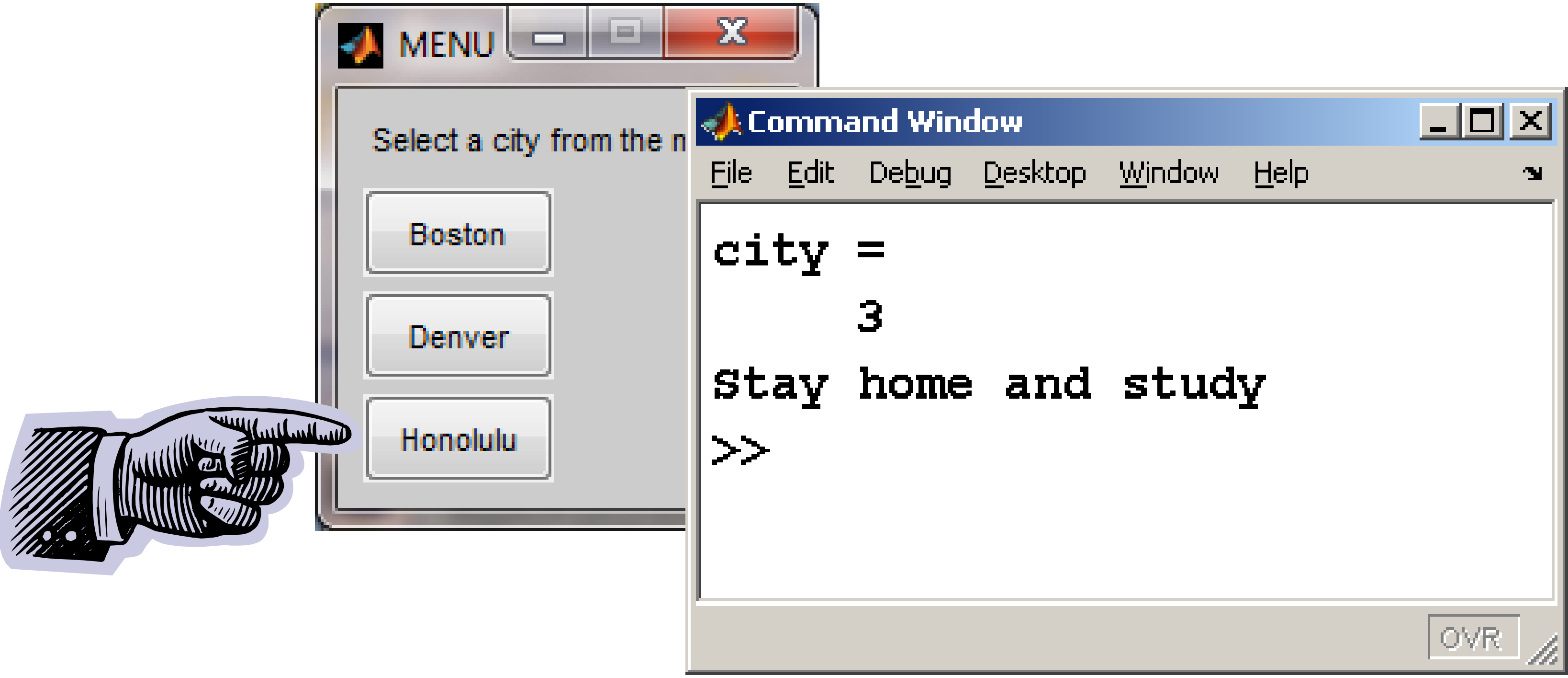
Note that the otherwise portion of the switch/case structure wasn't used

When you run this code a menu box appears



Instead of entering your choice from the command window, you select one of the buttons from the menu

If I select Honolulu…



```
city =

     3

Stay home and study
>>
```

Summary

- Sections of computer code can be categorized as
  - sequences
  - selection structures
  - repetition structures

Summary – Sequence

- Sequences are lists of instructions that are executed in order

Summary – Selection Structure

- Selection structures allow the programmer to define criteria (conditional statements) which the program uses to choose execution paths

Summary – Repetition Structures

- Repetition structures define loops where a sequence of instructions is repeated until some criterion is met (also defined by conditional statements).

Summary – Relational Operators

- MATLAB uses the standard mathematical relational operators
  - <
  - <=
  - >
  - >=
  - ==
  - ~=

Recall that = is the assignment operator, and can not be used for comparisons

Summary – Logical Operators

- MATLAB uses the standard logical operators
  - && and
  - || or
  - ~ not
  - xor exclusive or

Summary – Logical Functions

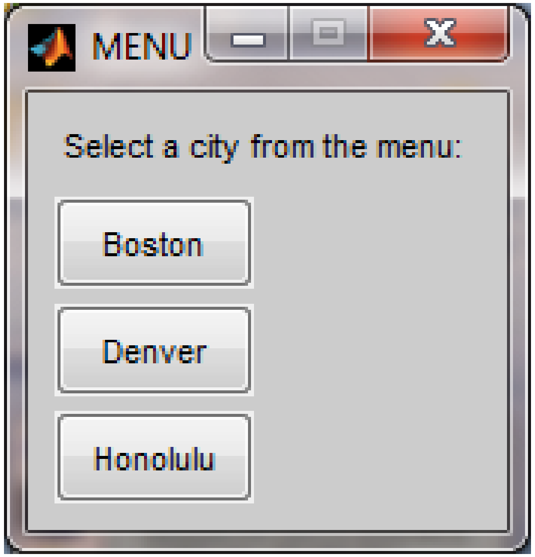
- The **find** command is unique to MATLAB, and should be the primary logical function used in your programming
- It allows the user to specify a condition using both logical and relational operators, which is then used to identify elements of a matrix that meet the condition.

Summary – if family

- The family of if structures allows the programmer to identify alternate computing paths dependent upon the results of conditional statements.
  - if
  - else
  - elseif

Summary switch/case

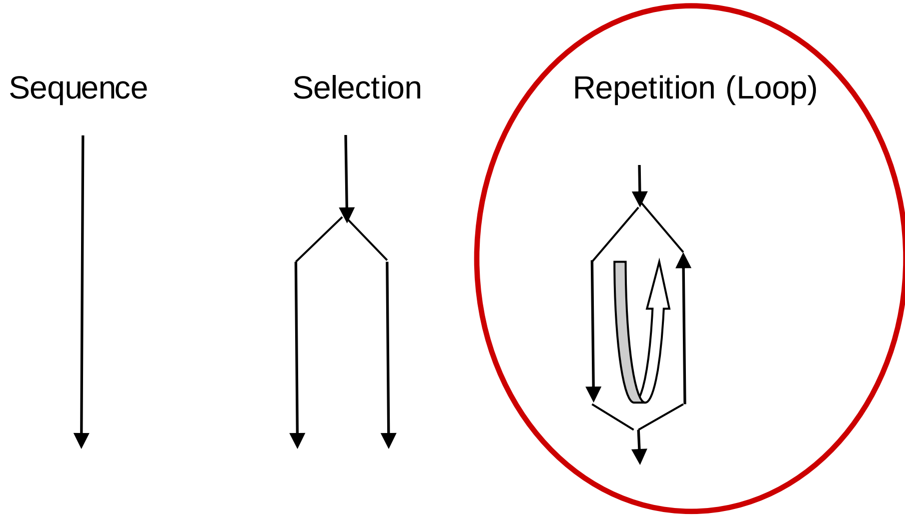
- Similar to the if/elseif/else structure
- Commonly used with menu



Select a city from the menu:

Structures

- Sequence
- Selection
- Repetition



Types of Loops

- Loops are used when you need to repeat a set of instructions multiple times
- MATLAB supports two types of loops
  - for
  - while

When to use loops

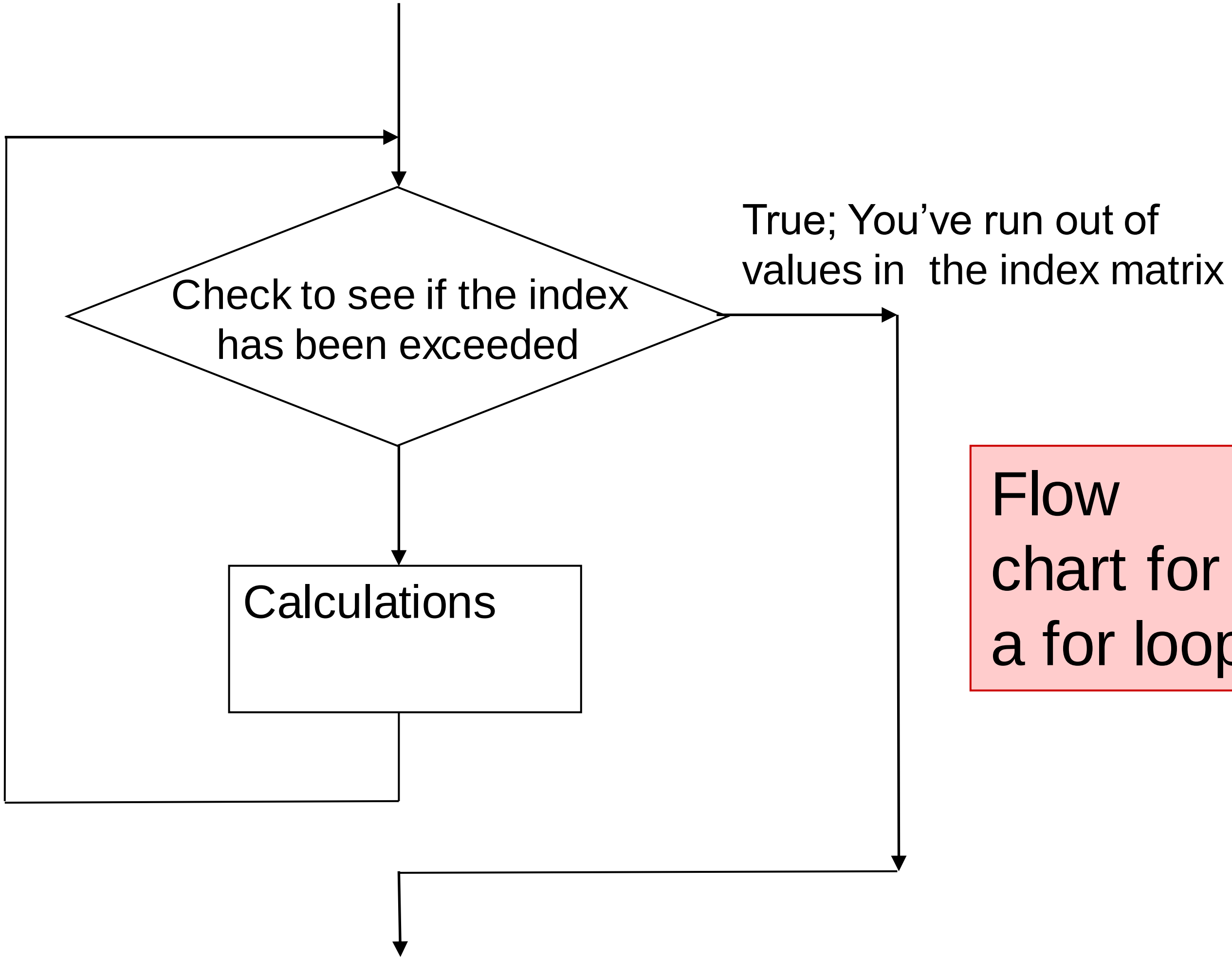
- In general loops are best used with scalars, or with the values stored in a matrix used one at a time
- Many of the problems you may want to attempt with loops can be better solved by vectorizing your code or with MATLAB’s logical functions such as

  find

7.5 For Loops

```
for  index = [matrix]
      commands to be executed
end
```

The loop is executed once for each element of the index matrix identified in the first line



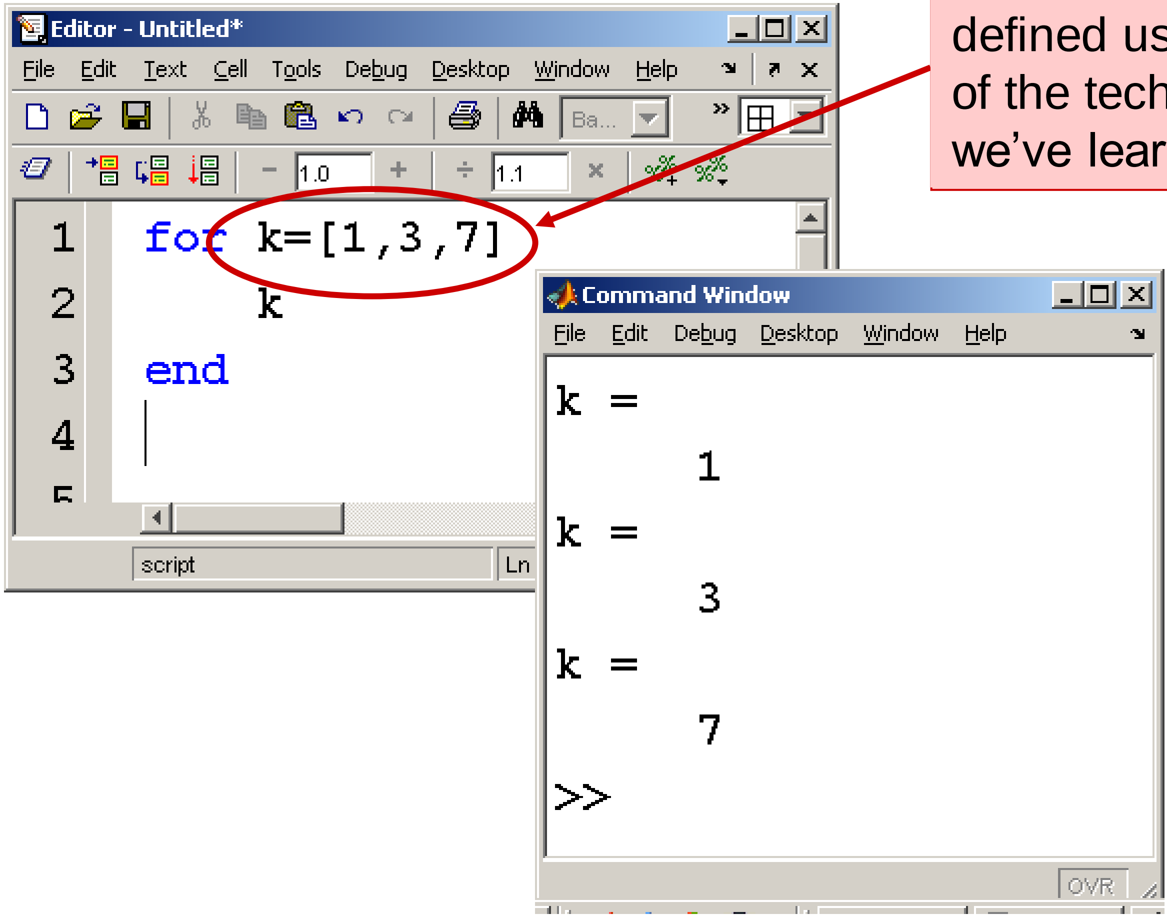
Check to see if the index has been exceeded

True; You’ve run out of values in the index matrix

Calculations

Flow chart for a for loop

Here’s a simple example

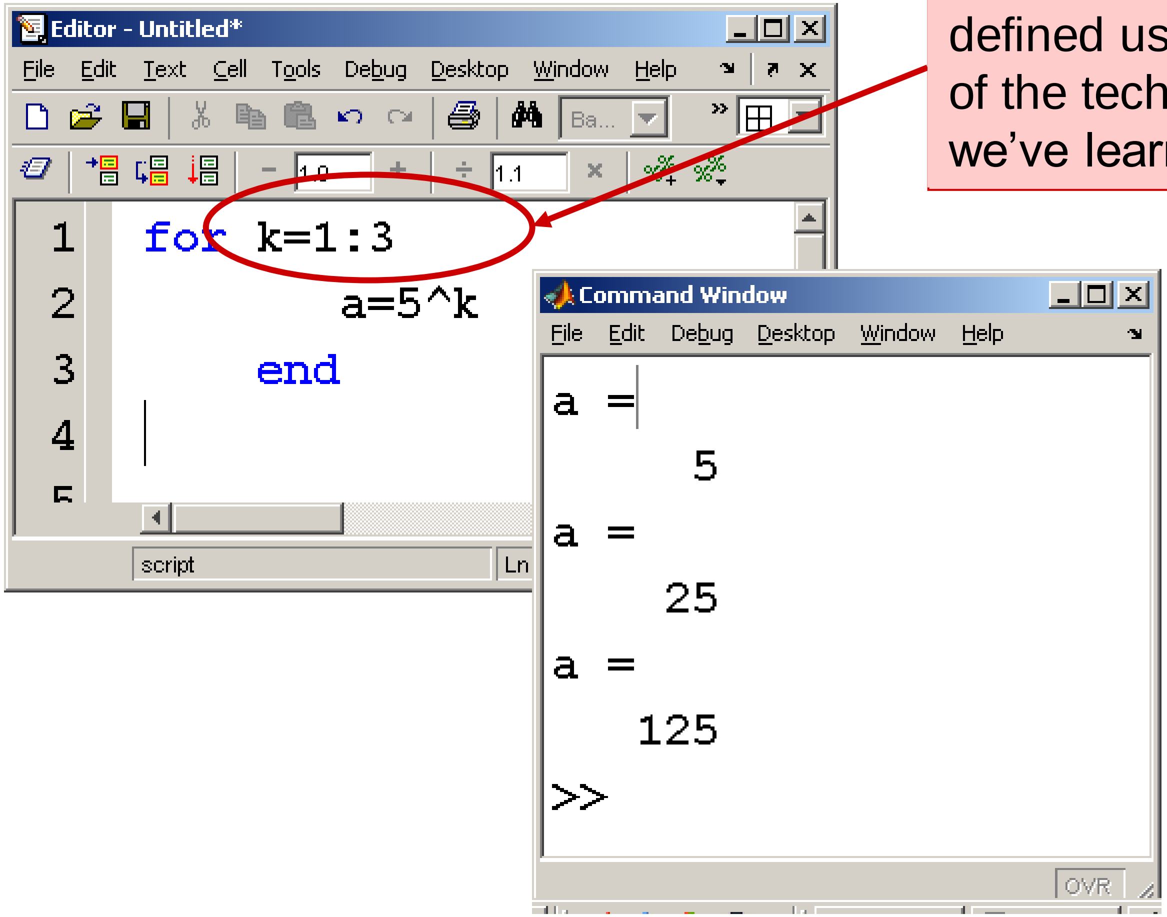

Editor - Untitled*
File Edit Text Cell Tools Debug Desktop Window Help
1.0
1.1
script
Ln
Command Window
File Edit Debug Desktop Window Help
OVR

```
for k=[1,3,7]
      k
end
```

```
k =

      1

k =

      3

k =

      7

>>
```

the index can be defined using any of the techniques we’ve learned

Here’s a simple example



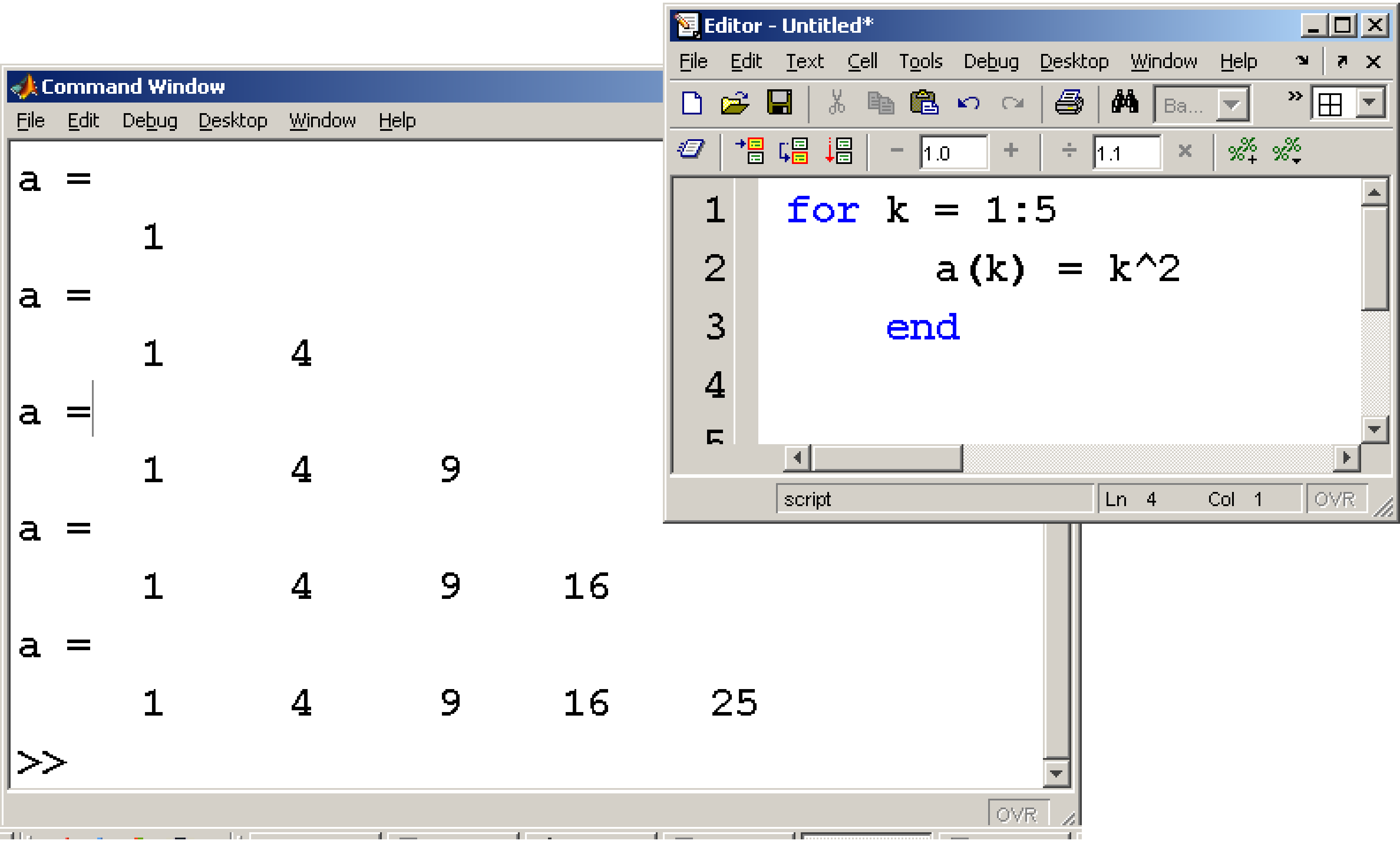
Editor - Untitled*
File Edit Text Cell Tools Debug Desktop Window Help
1.0
1.1
script
Ln
Command Window
File Edit Debug Desktop Window Help
OVR

```
for k=1:3
        a=5^k
    end
```

```
a =

      5

a =

     25

a =

    125

>>
```

the index can be defined using any of the techniques we’ve learned

One of the most common ways to use a loop is to define a matrix



Editor - Untitled*
File Edit Text Cell Tools Debug Desktop Window Help
Ba...
1.0
1.1
script
Ln 4 Col 1
OVR
Command Window
File Edit Debug Desktop Window Help
OVR

```
for k = 1:5
      a(k) = k^2
   end
```

```
a =

     1

a =

     1     4

a =

     1     4     9

a =

     1     4     9    16

a =

     1     4     9    16    25

>>
```

Hint

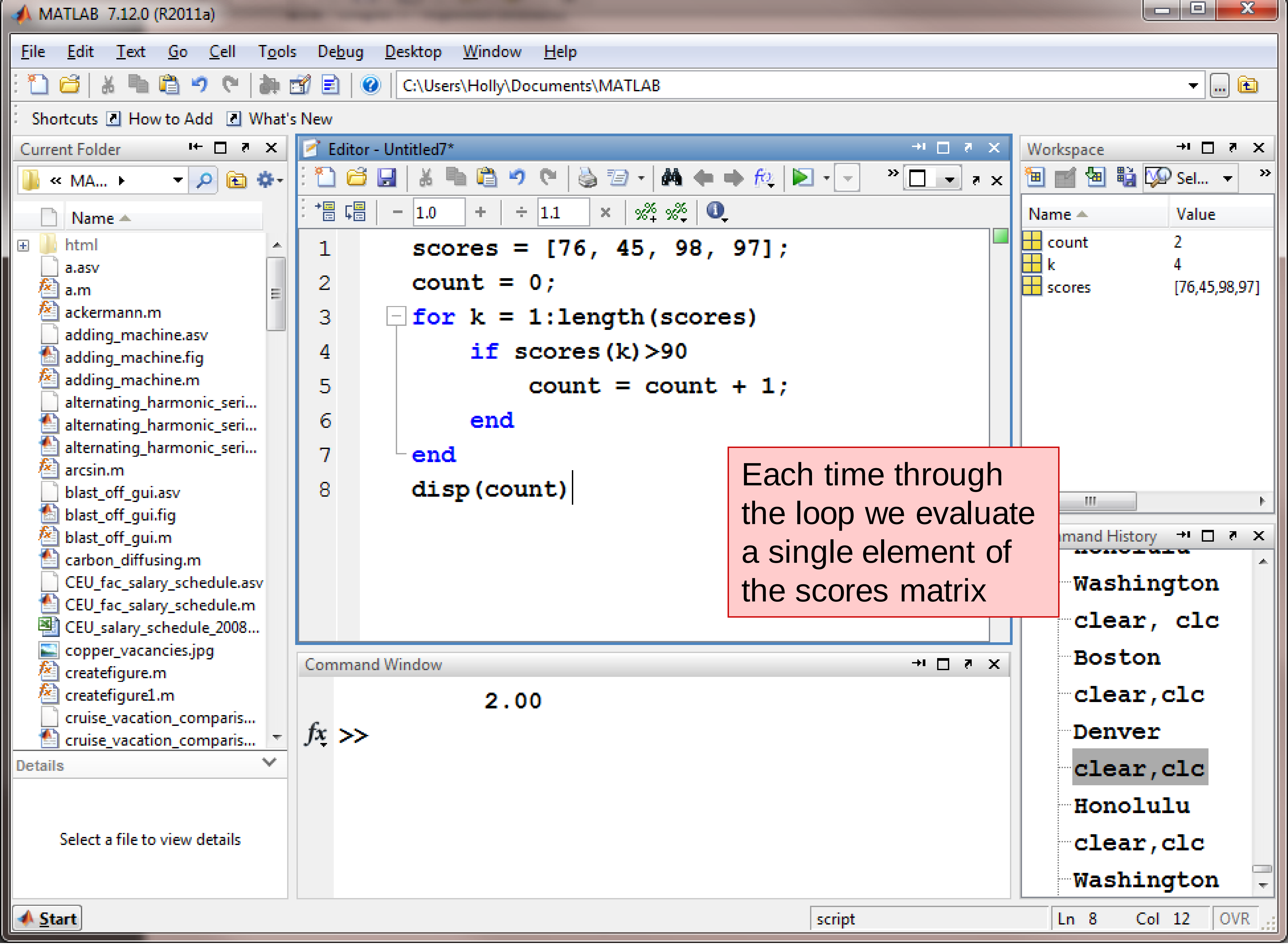
Most computer programs do not have MATLAB’s ability to handle matrices so easily, and therefore rely on loops similar to the one on the previous slide to define arrays. It would be easier to create the vector **a** in MATLAB with the following code

```
        k=1:5
        a = k.^2
```

which returns

```
k =
   1    2    3    4    5
a =
   1    4    9   16   25
```

This is an example of ***vectorizing*** the code.



```
scores = [76, 45, 98, 97];
count = 0;
for k = 1:length(scores)
    if scores(k)>90
        count = count + 1;
    end
end
disp(count)
```

Each time through the loop we evaluate a single element of the scores matrix

Name	Value
count	2
k	4
scores	[76,45,98,97]

```
        2.00
>>
```

```
Washington
clear, clc
Boston
clear,clc
Denver
clear,clc
Honolulu
clear,clc
Washington
```

Summary of the for loop structure

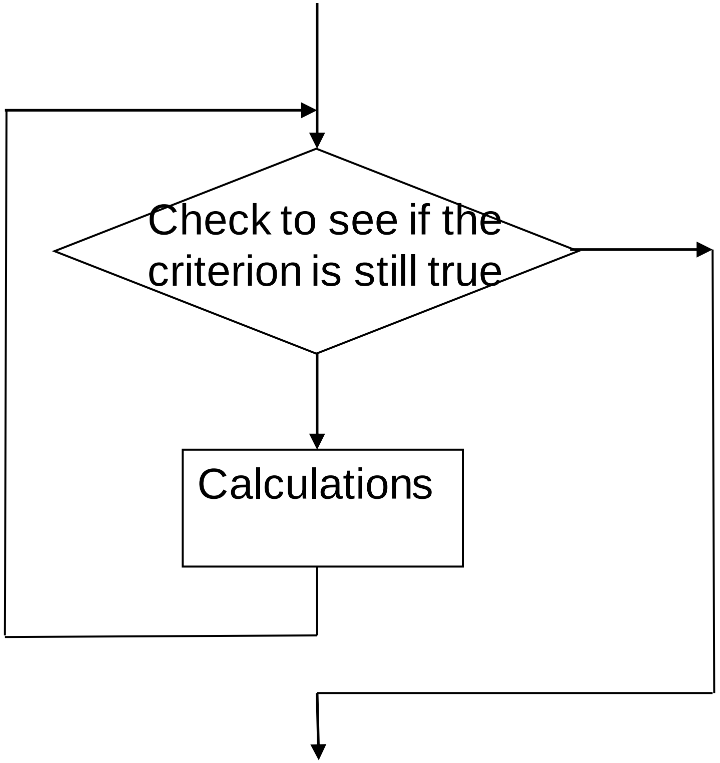
- The loop starts with a **for** statement, and ends with the word **end**.
- The first line in the loop defines the number of times the loops will repeat, using an index number.
- The index of a **for** loop must be a variable. (The index is the number that changes each time through the loop.) Although **k** is often used as the symbol for the index, any variable name can be used. The use of **k** is a matter of style.

7.6 While Loops

- While loops are very similar to for loops.
- The big difference is the way MATLAB decides how many times to repeat the loop.
- While loop
  met.

```
while  criterion
       commands to be executed
end
```

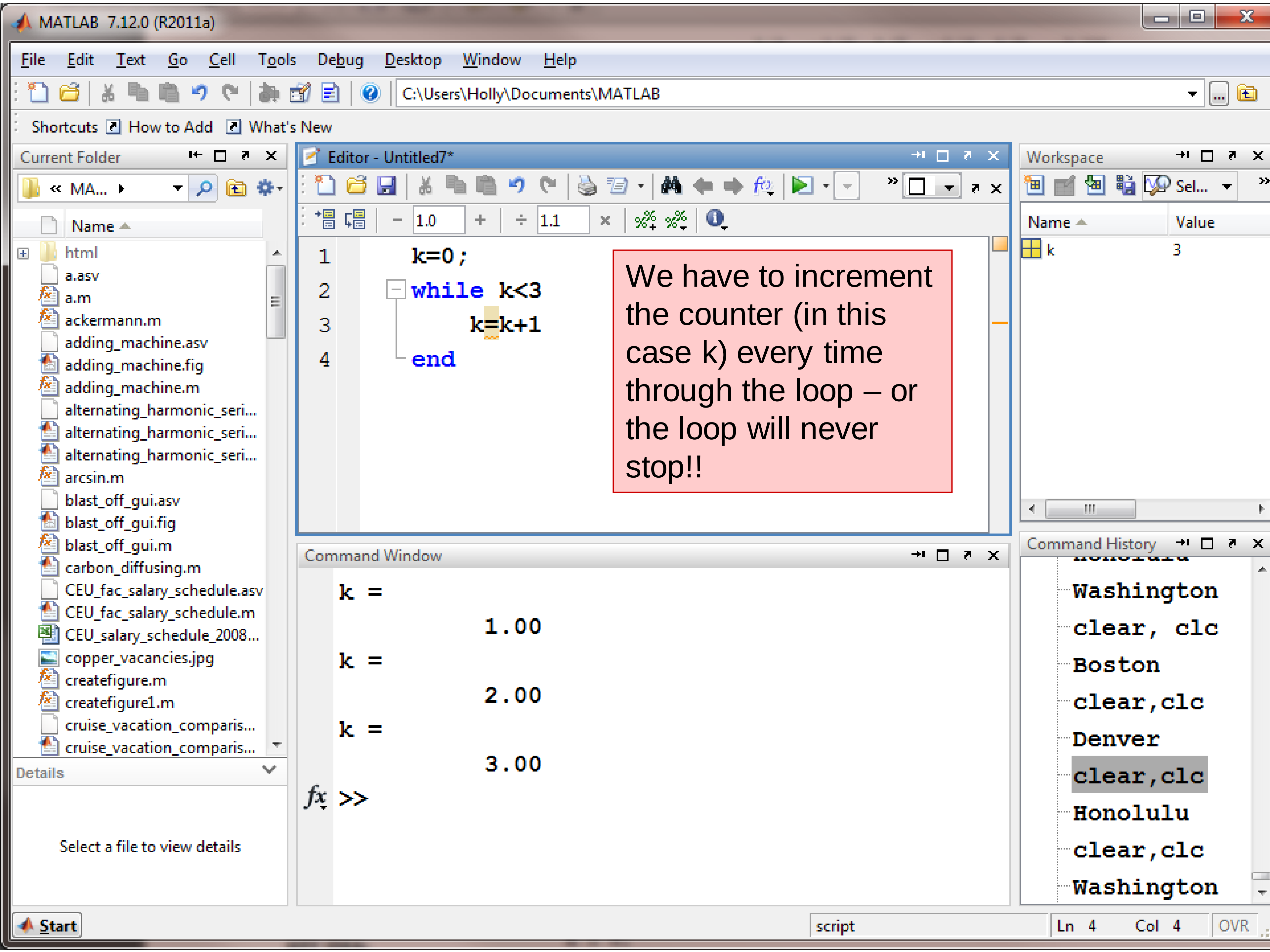
Flow Chart for a while loop



Check to see if the criterion is still true

Calculations

The criterion is no longer true and the program exits the loop


MATLAB 7.12.0 (R2011a)
File Edit Text Go Cell Tools Debug Desktop Window Help
C:\Users\Holly\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
« MA... ▸
Name
html
a.asv
a.m
ackermann.m
adding_machine.asv
adding_machine.fig
adding_machine.m
alternating_harmonic_seri...
alternating_harmonic_seri...
alternating_harmonic_seri...
arcsin.m
blast_off_gui.asv
blast_off_gui.fig
blast_off_gui.m
carbon_diffusing.m
CEU_fac_salary_schedule.asv
CEU_fac_salary_schedule.m
CEU_salary_schedule_2008...
copper_vacancies.jpg
createfigure.m
createfigure1.m
cruise_vacation_comparis...
cruise_vacation_comparis...
Details
Select a file to view details
Editor - Untitled7*
1.0
1.1
Workspace
Sel...
Command Window
Command History
Start
script
Ln 4 Col 4
OVR

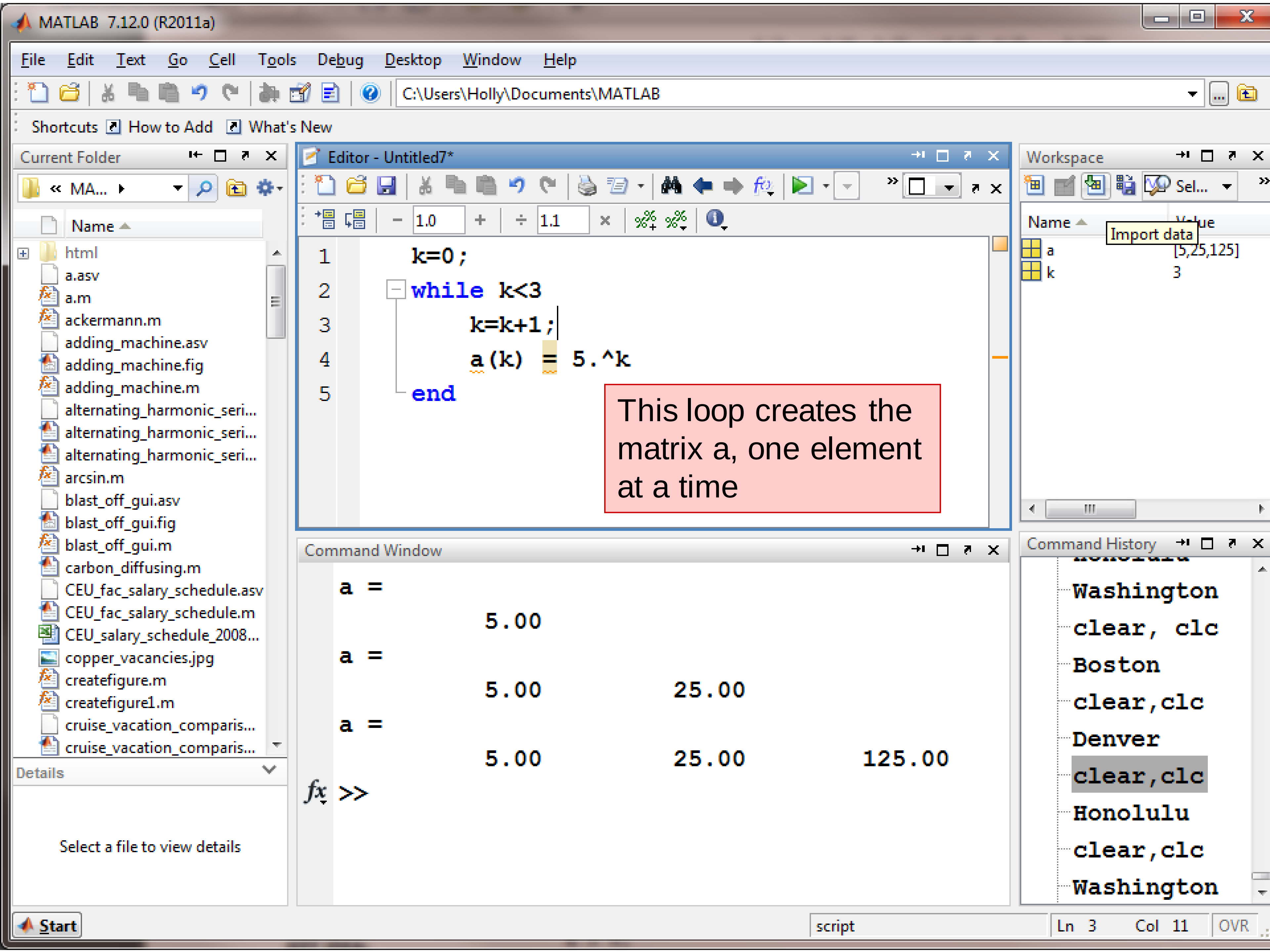
```
k=0;
while k<3
      k=k+1
end
```

We have to increment the counter (in this case k) every time through the loop – or the loop will never stop!!

```
k =

          1.00

k =

          2.00

k =

          3.00

>>
```

Name	Value
k	3

```
Honolulu
Washington
clear, clc
Boston
clear,clc
Denver
clear,clc
Honolulu
clear,clc
Washington
```



```
k=0;
while k<3
    k=k+1;
    a(k) = 5.^k
end
```

This loop creates the matrix a, one element at a time

Name	Value
a	[5,25,125]
k	3

```
a =

        5.00

a =

        5.00         25.00

a =

        5.00         25.00         125.00

>>
```

Hint

If you accidentally create a loop that just keeps running you should

1. Confirm that the computer is actually still calculating something by checking the lower left hand corner of the MATLAB window for the “busy indicator”
2. Make sure the active window is the command window and exit the calculation manually with **ctrl c**

Summary

- Sections of computer code can be categorized as
  - sequences
  - selection structures
  - repetition structures

Summary – Sequence

- Sequences are lists of instructions that are executed in order

Summary – Selection Structure

- Selection structures allow the programmer to define criteria (conditional statements) which the program uses to choose execution paths

Summary – Repetition Structures

- Repetition structures define loops where a sequence of instructions is repeated until some criterion is met (also defined by conditional statements).

Summary – Relational Operators

- MATLAB uses the standard mathematical relational operators
  - <
  - <=
  - >
  - >=
  - ==
  - ~=

Recall that = is the assignment operator, and can not be used for comparisons

Summary – Logical Operators

- MATLAB uses the standard logical operators
  - && and
  - || or
  - ~ not
  - xor exclusive or

Summary - Loops

- MATLAB supports both
  - for loops
  - while loops
- **For** loops are primarily used when the programmer knows how many times a sequence of commands should be executed.
- **While** loops are used when the commands should be executed until a condition is met.
- Most problems can be structured so that either **for** or **while** loops are appropriate.

College of Electronics Engineering

Systems & Control Engineering Department

MATLAB Programming
SCE2304

Lecture 8
(Simulink – A Brief Introduction)

Zeyad T. Shareef

Objectives

After reading this lecture, you should be able to:

- Understand how Simulink uses blocks to represent common mathematical processes
- Create and run a simple Simulink model
- Import Simulink results into MATLAB

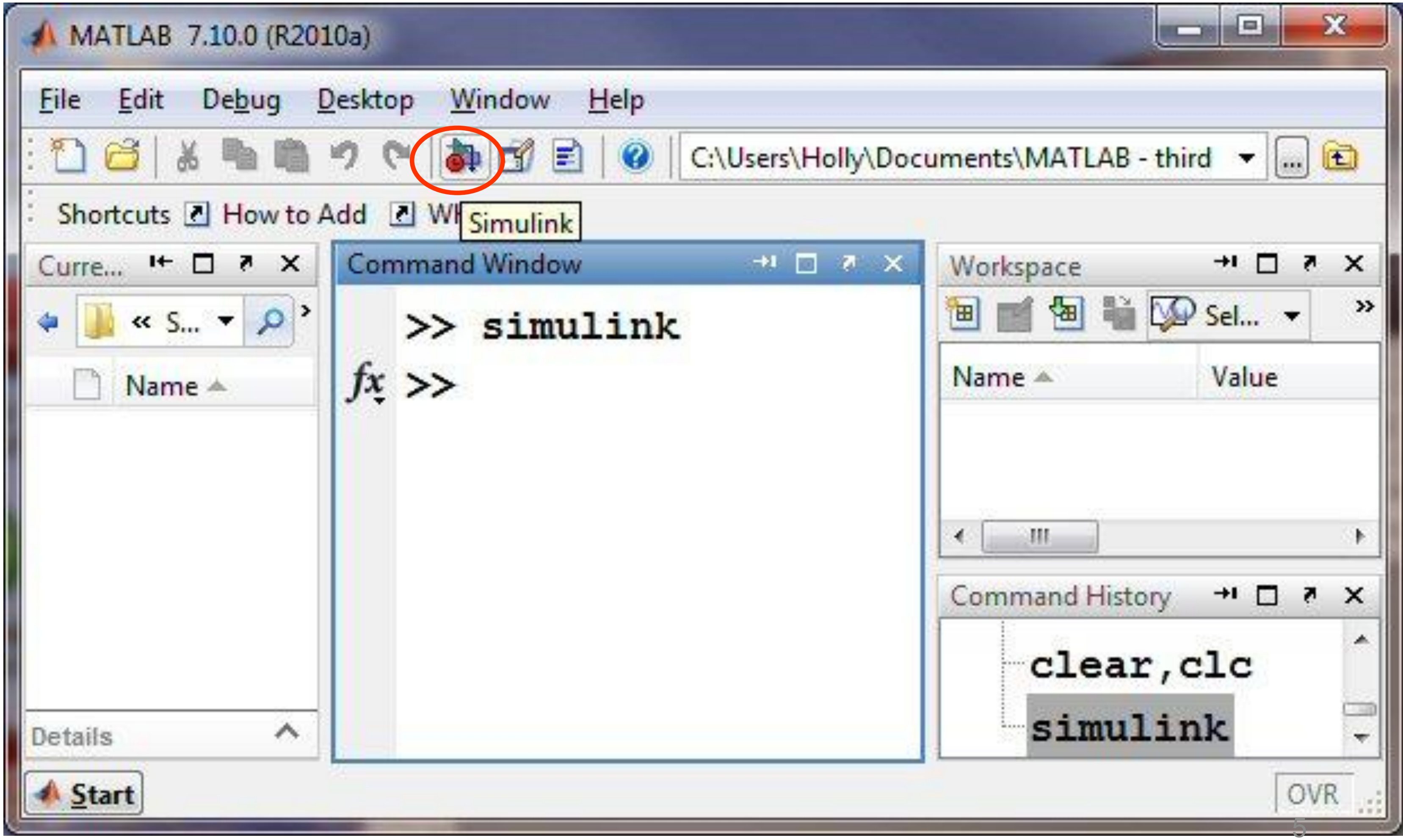
Simulink

- An interactive, graphics-based program that allows you to solve problems by creating models using built-in blocks
- Requires MATLAB to run
- Included with the Student Version – but is an add-on to the Professional version of MATLAB

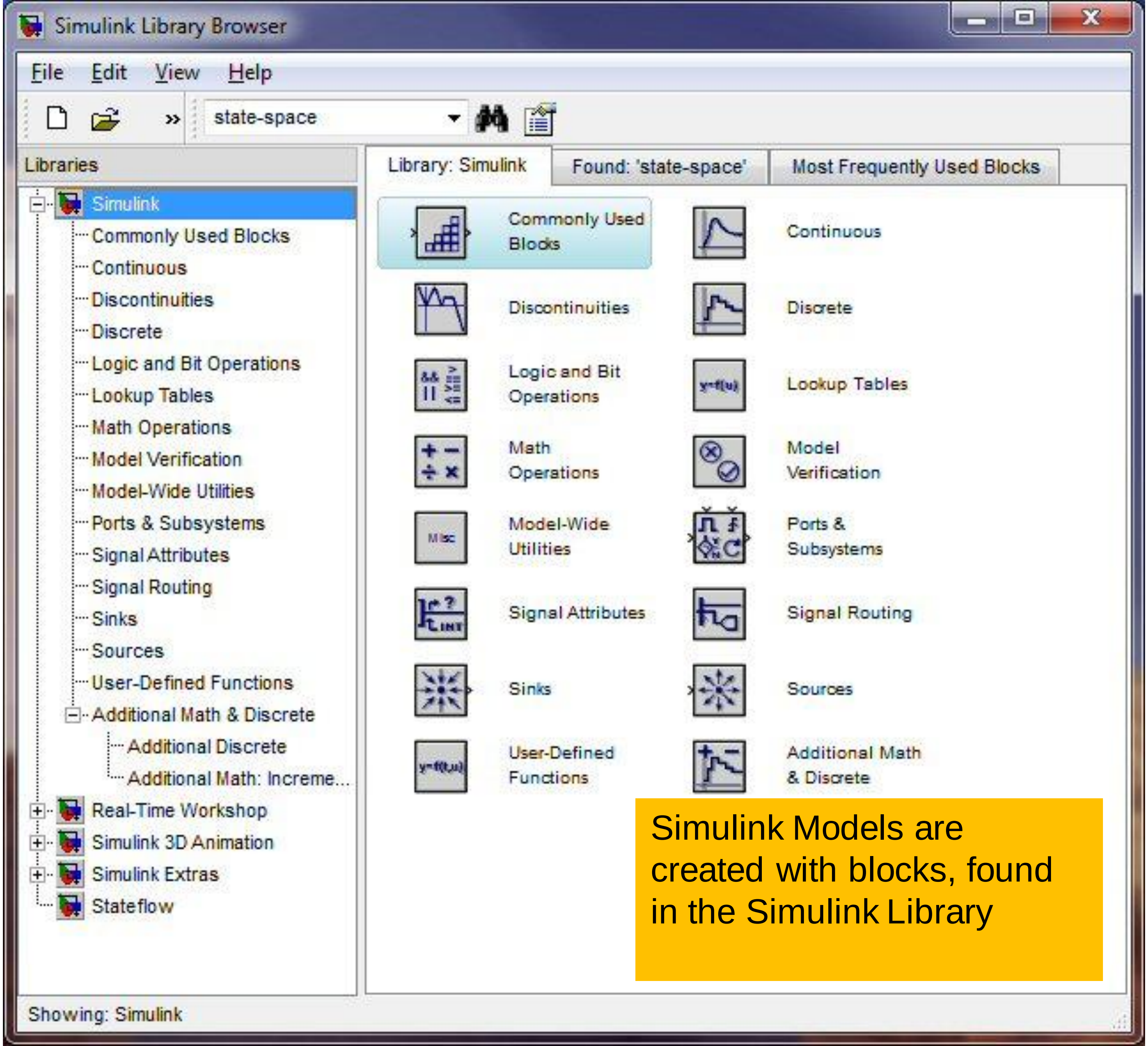
9.1 Applications

- Convenient for analyzing dynamic systems
- Commonly used in signal processing
- Similar to the approach used with analog computers
- Terminology is related to electrical components

Open from the command line – or use the icon

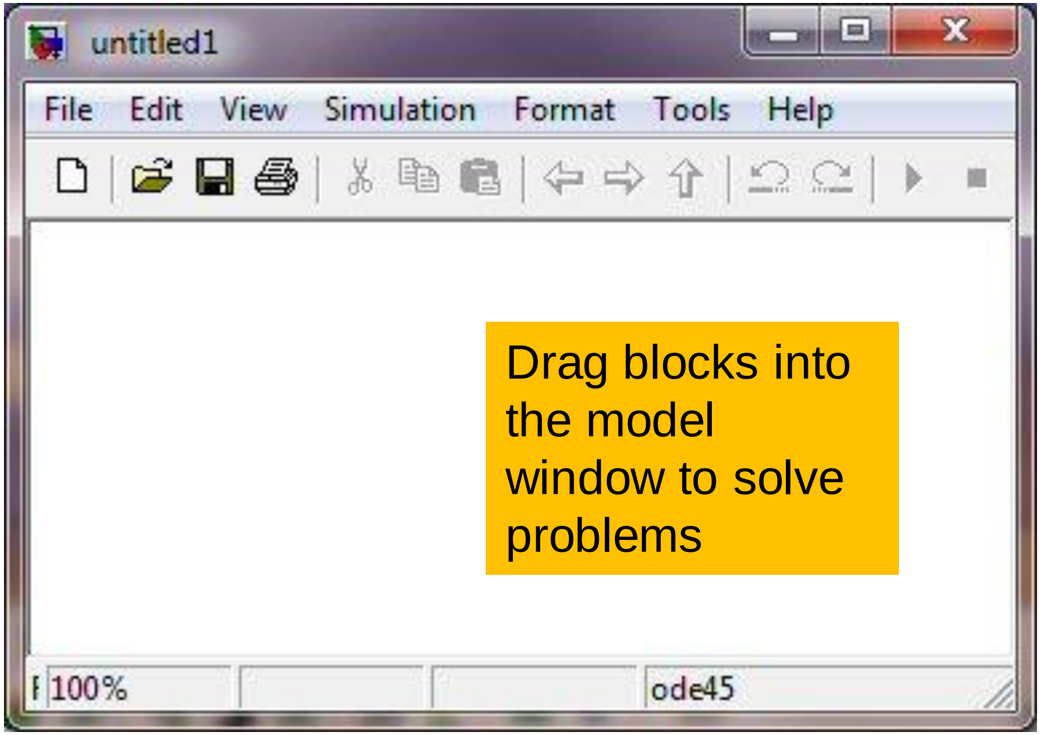


```
>> simulink
>>
```

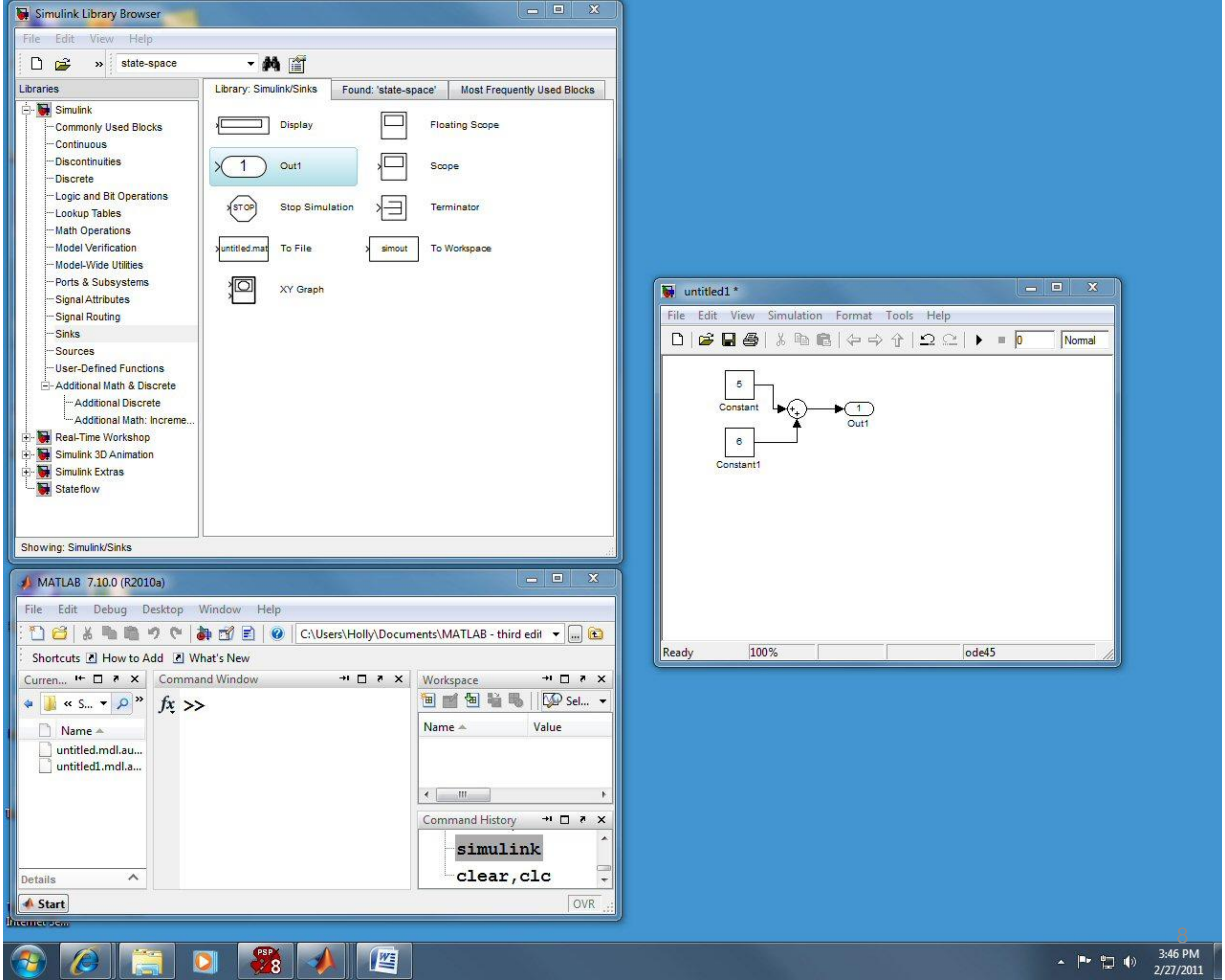


Simulink Models are created with blocks, found in the Simulink Library

The model window is where Simulink models are created and executed

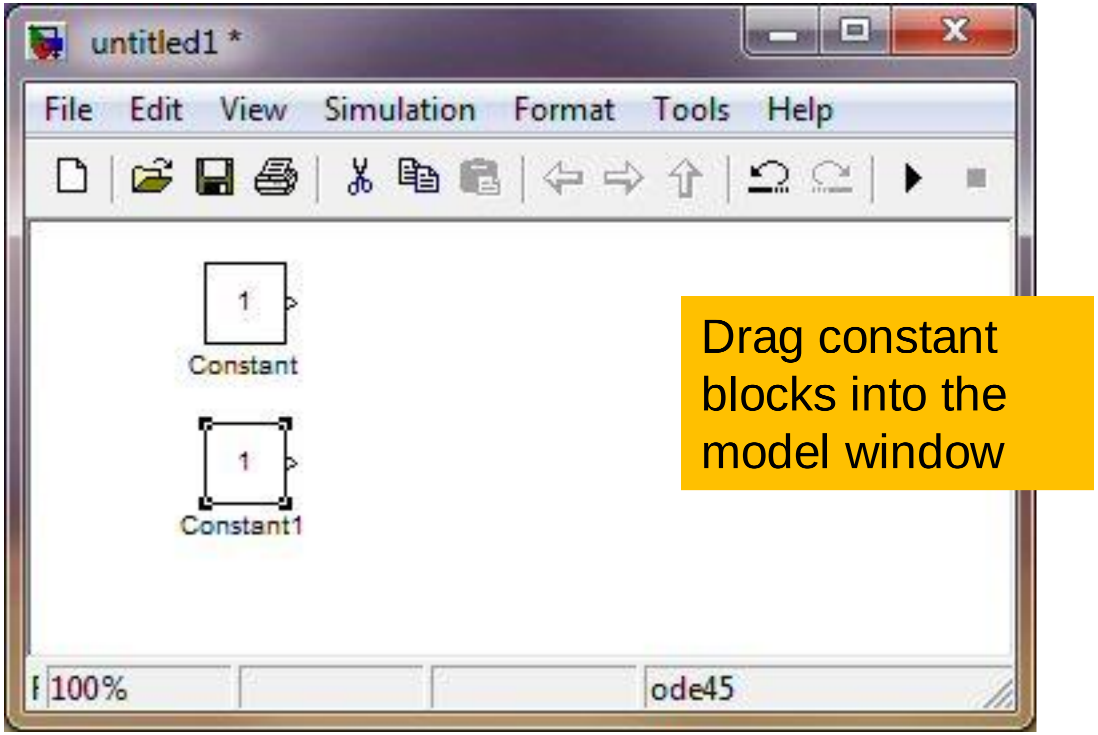


Drag blocks into the model window to solve problems



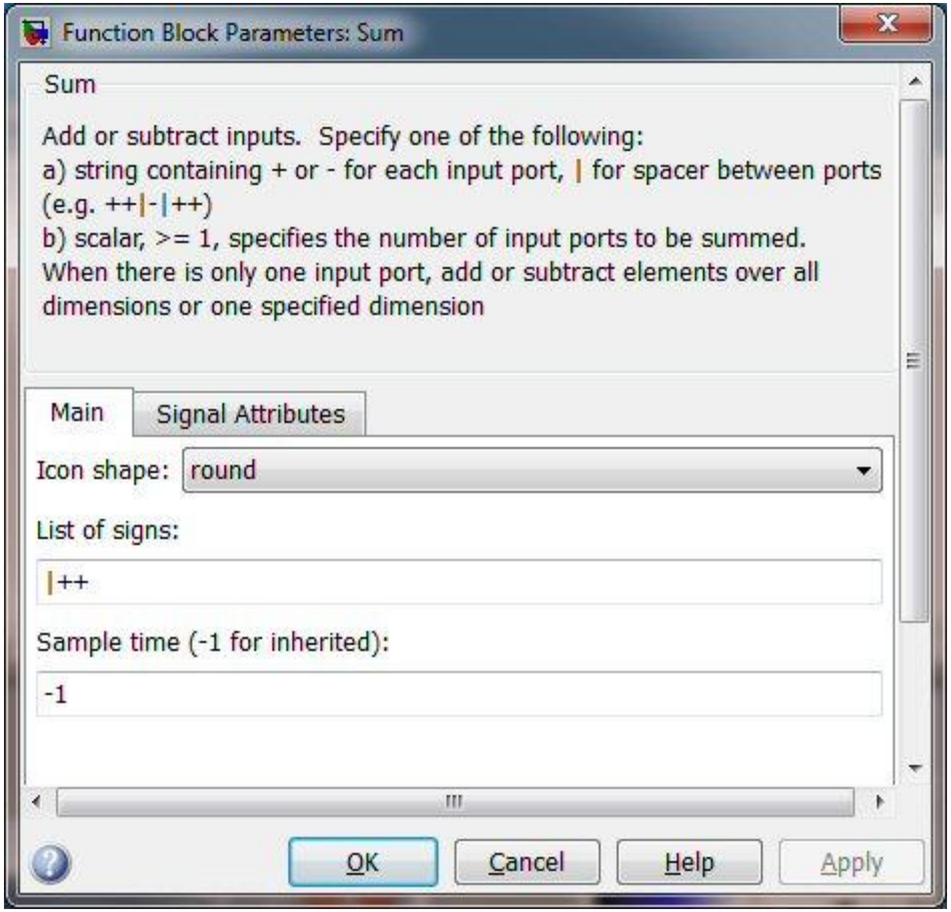
```
simulink
clear,clc
```

Create a simple model to add two numbers together



Drag constant blocks into the model window

Sum Block



Sum

Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

Icon shape: round

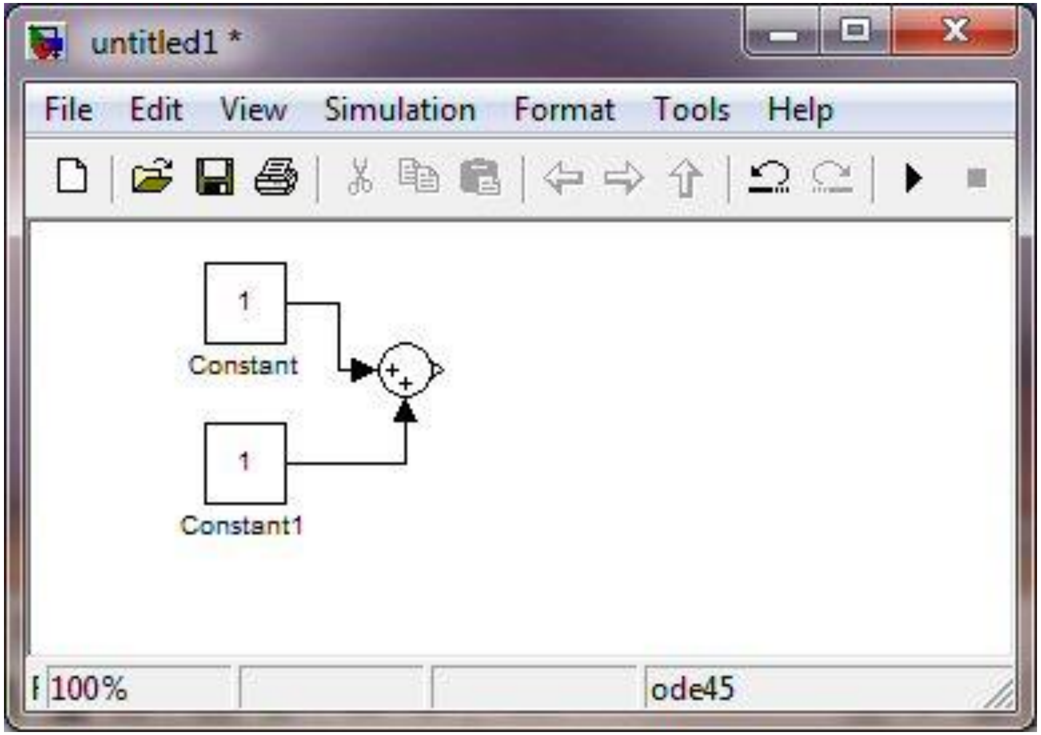
List of signs:

|++

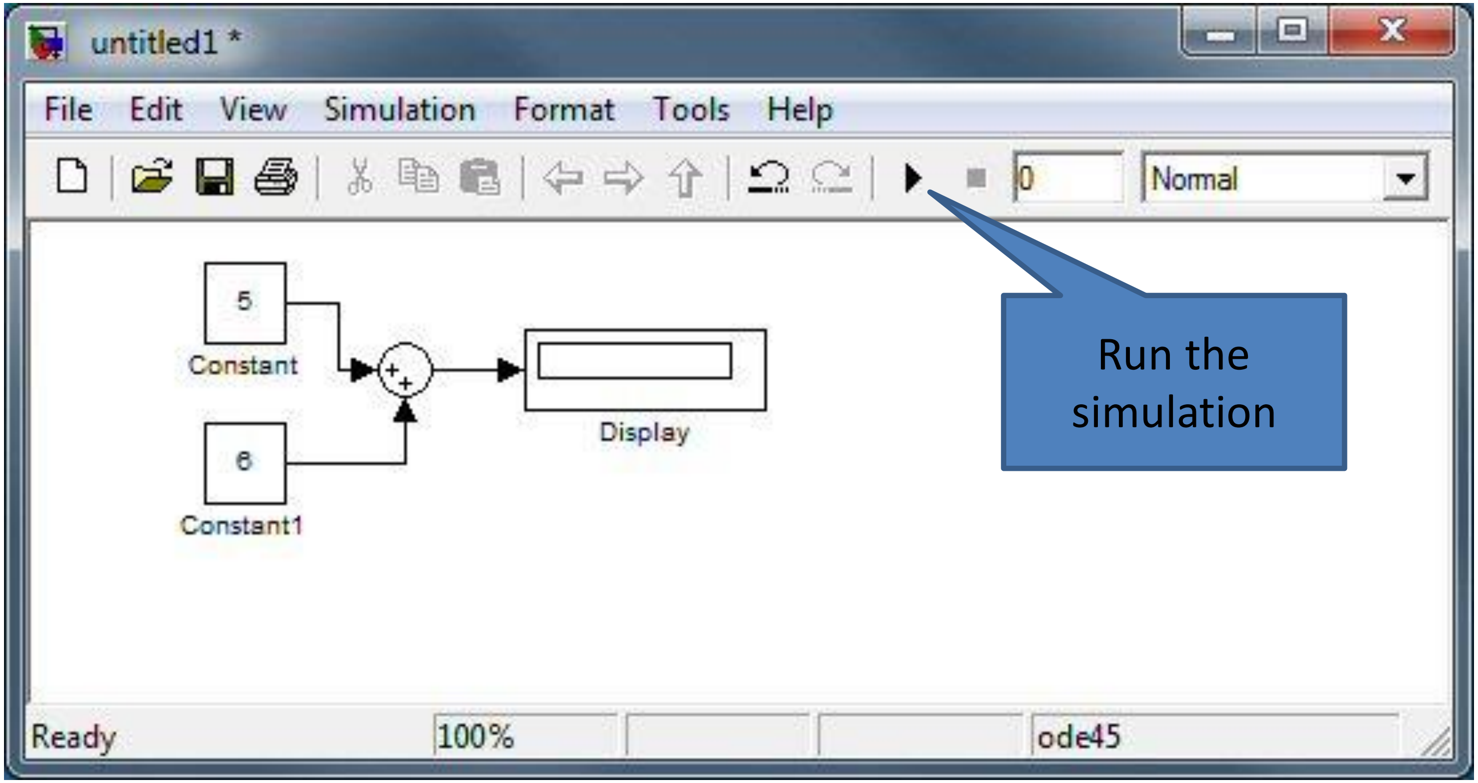
Sample time (-1 for inherited):

-1

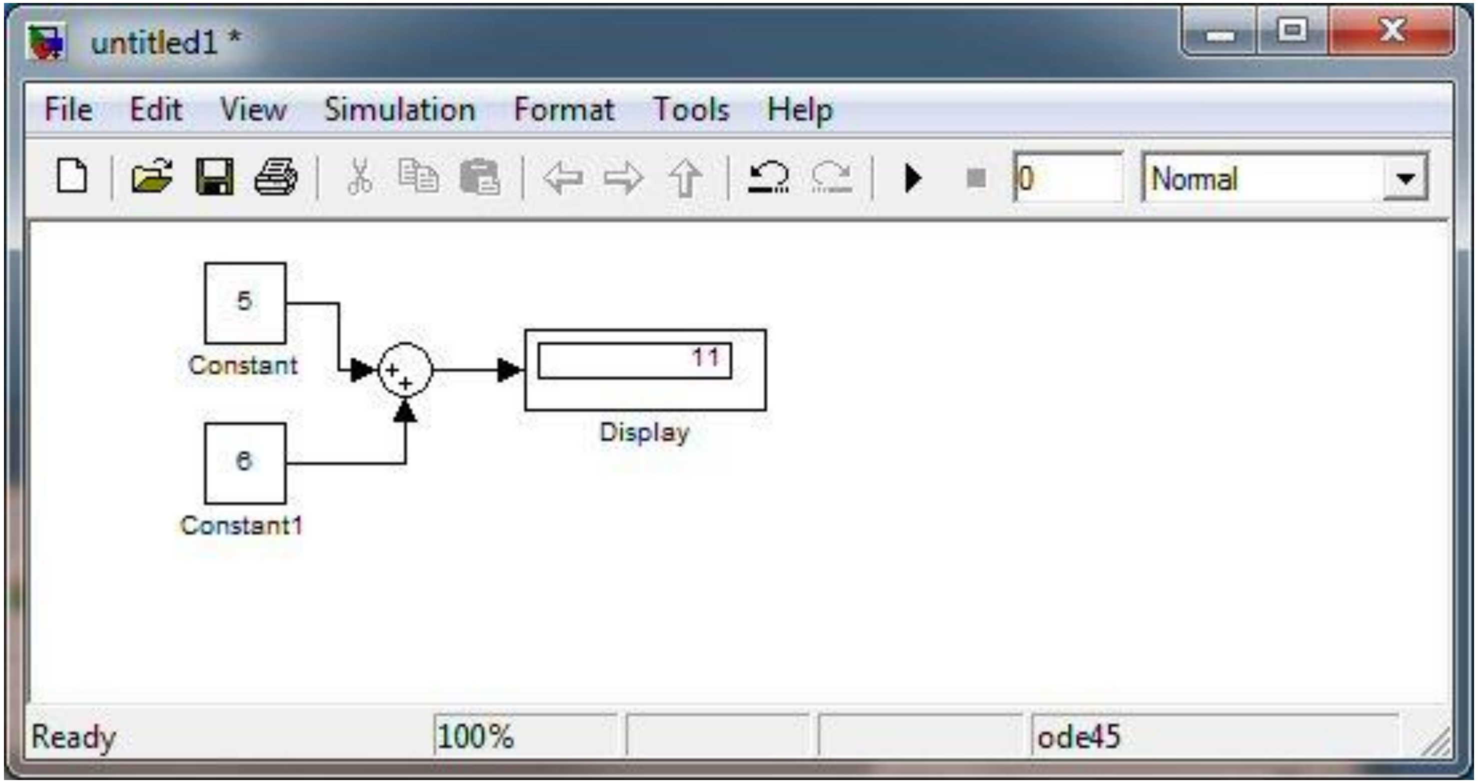
The constants are connected to a sum block. Change the values in the constant blocks by double clicking and modifying the constant value field

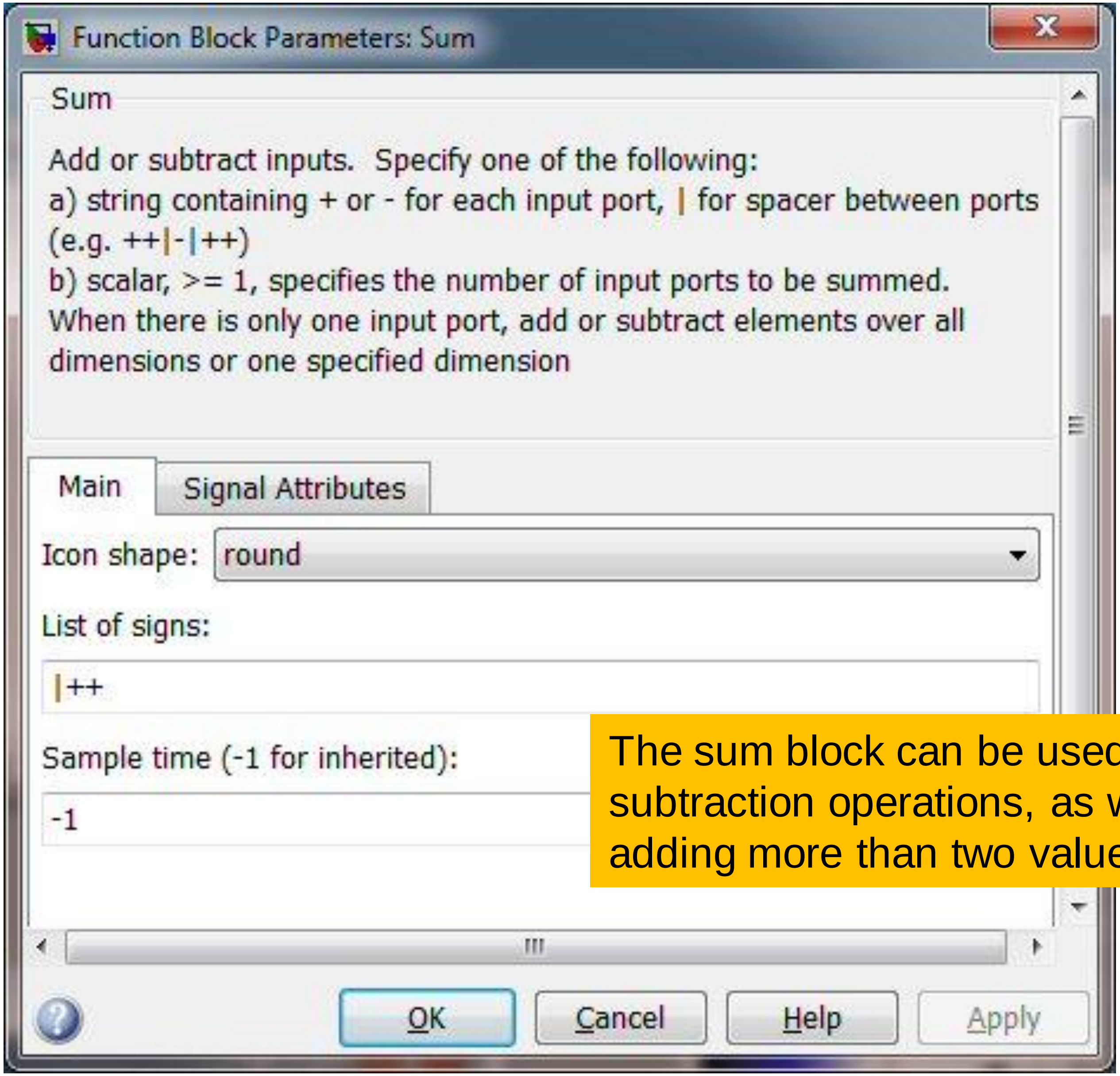


Completed Model



Run the simulation


untitled1 *
File
Edit
View
Simulation
Format
Tools
Help
0
Normal
5
Constant
6
Constant1
+
+
11
Display
Ready
100%
ode45



Function Block Parameters: Sum

Sum

Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

Icon shape: round

List of signs:

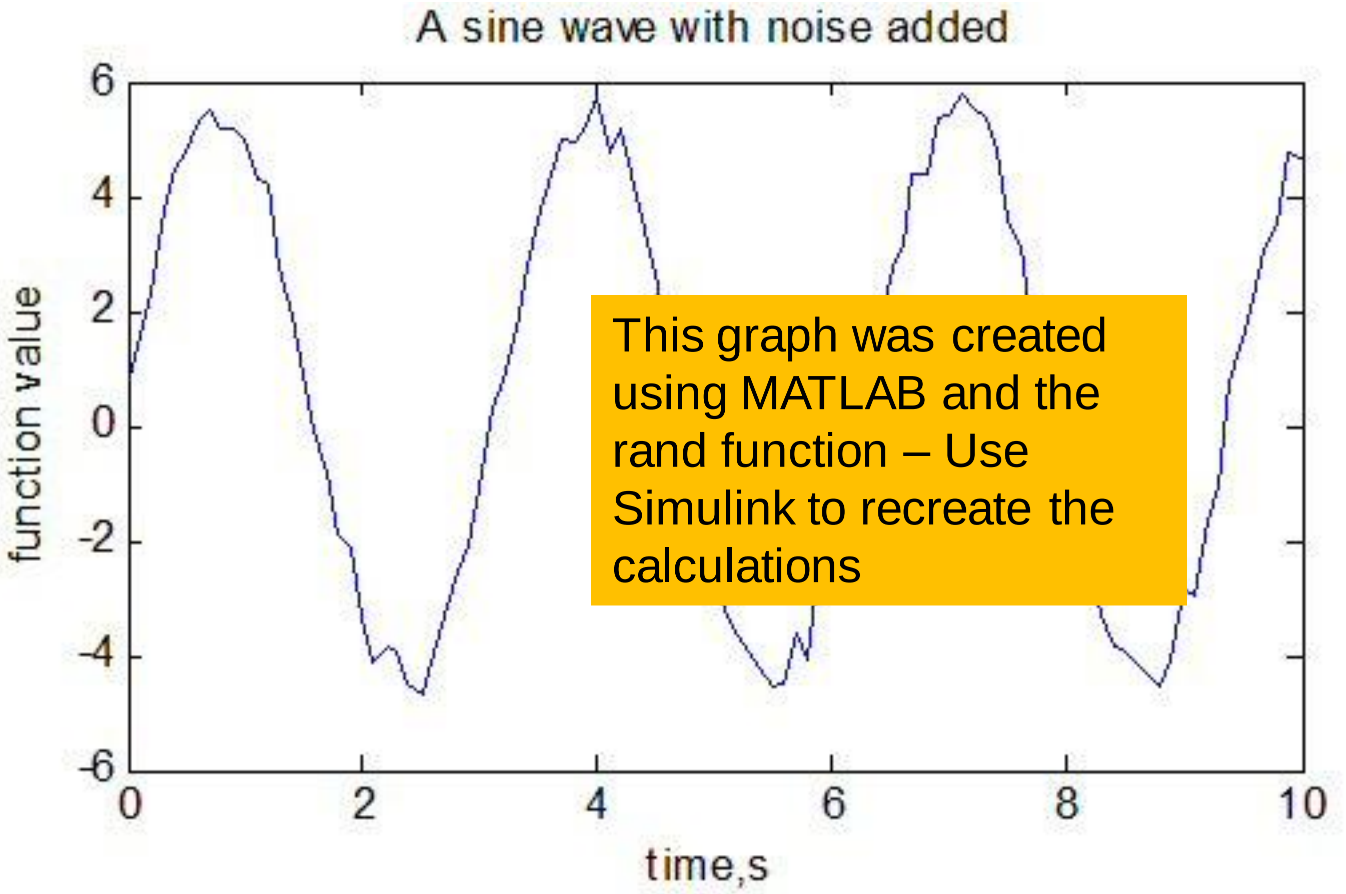
|++

Sample time (-1 for inherited):

-1

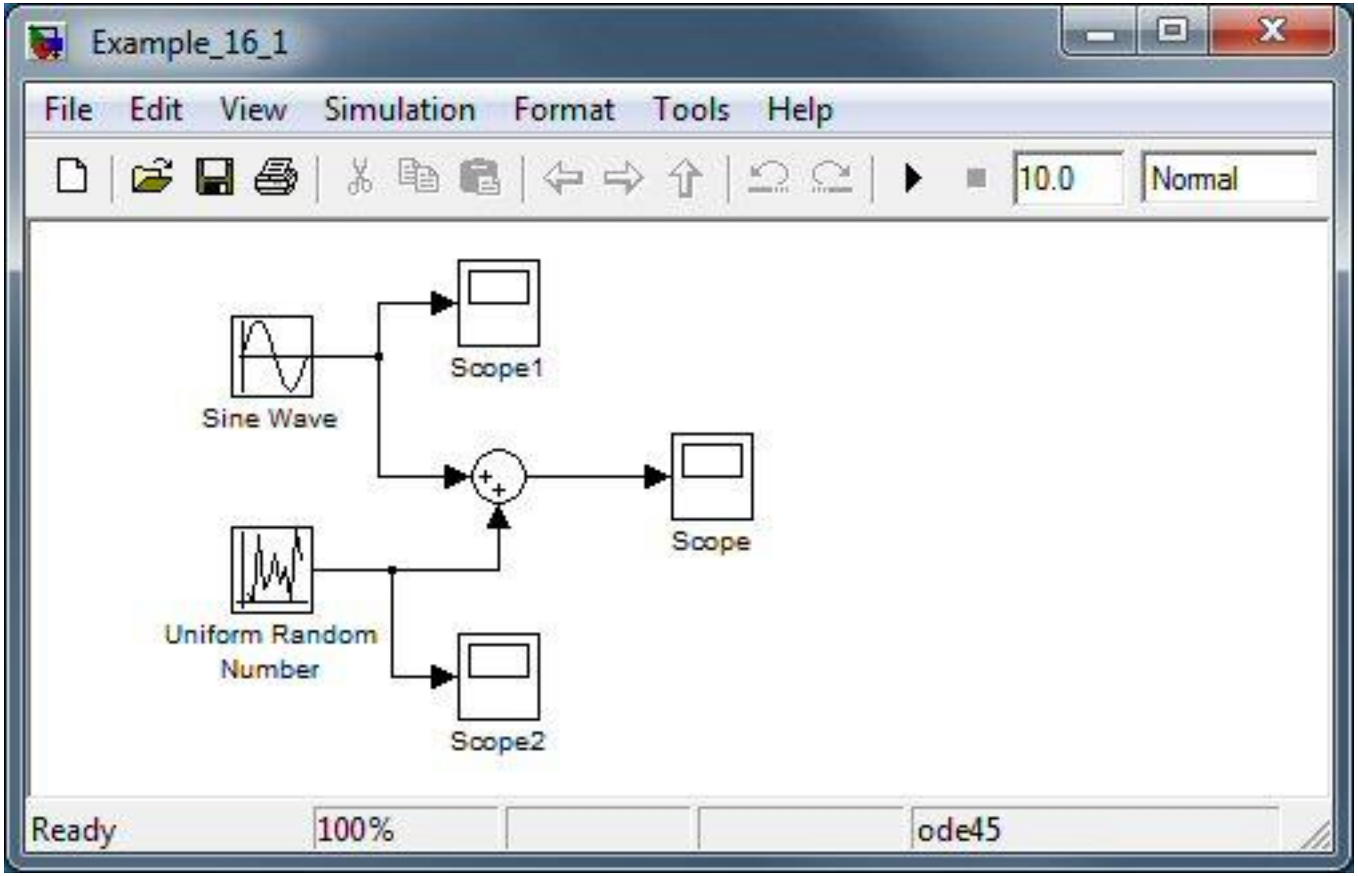
The sum block can be used to perform subtraction operations, as well as for adding more than two values

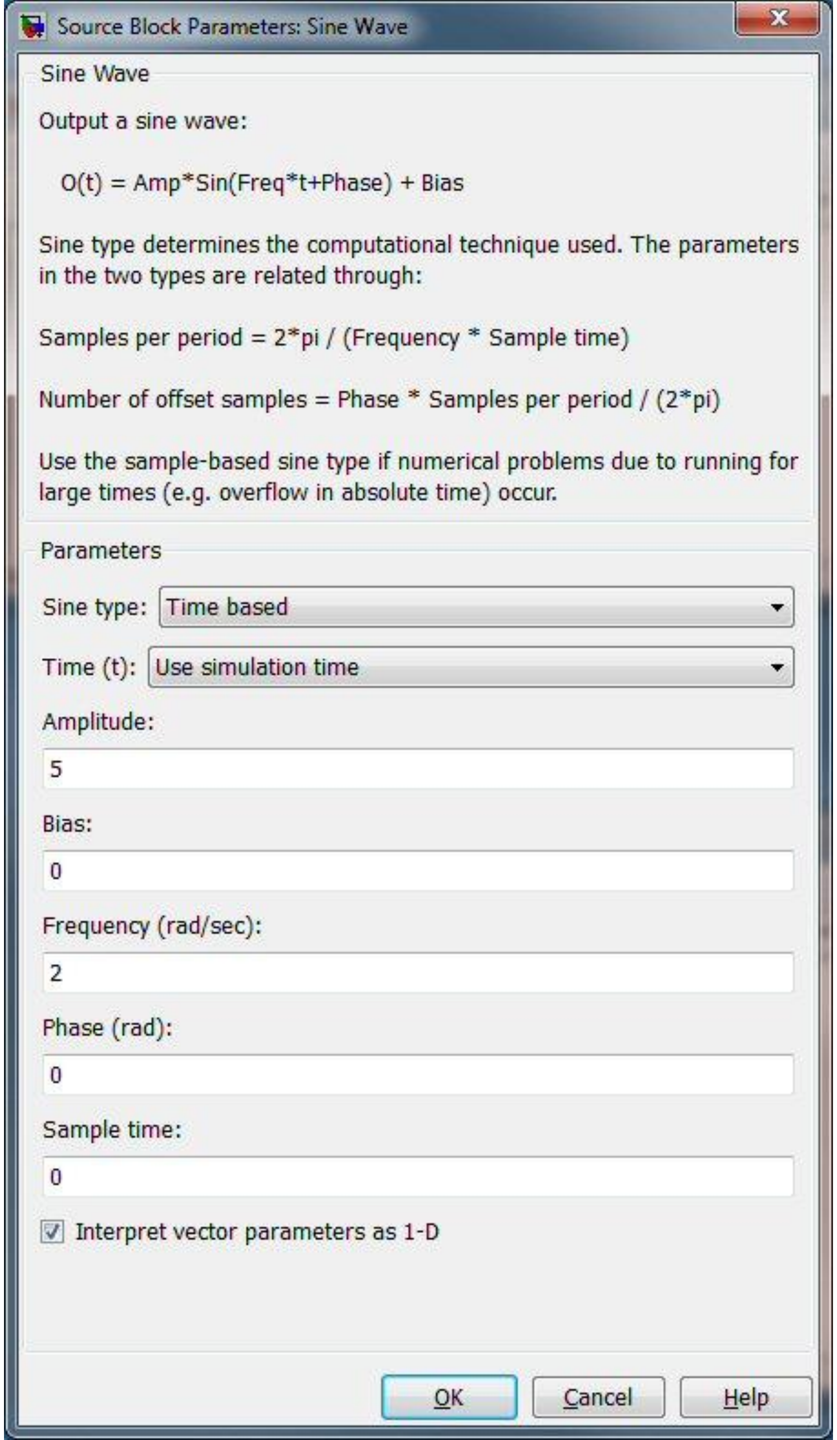
Example 9.1



This graph was created using MATLAB and the rand function – Use Simulink to recreate the calculations

Example 9.1 - Simulink Model





Source 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: 5

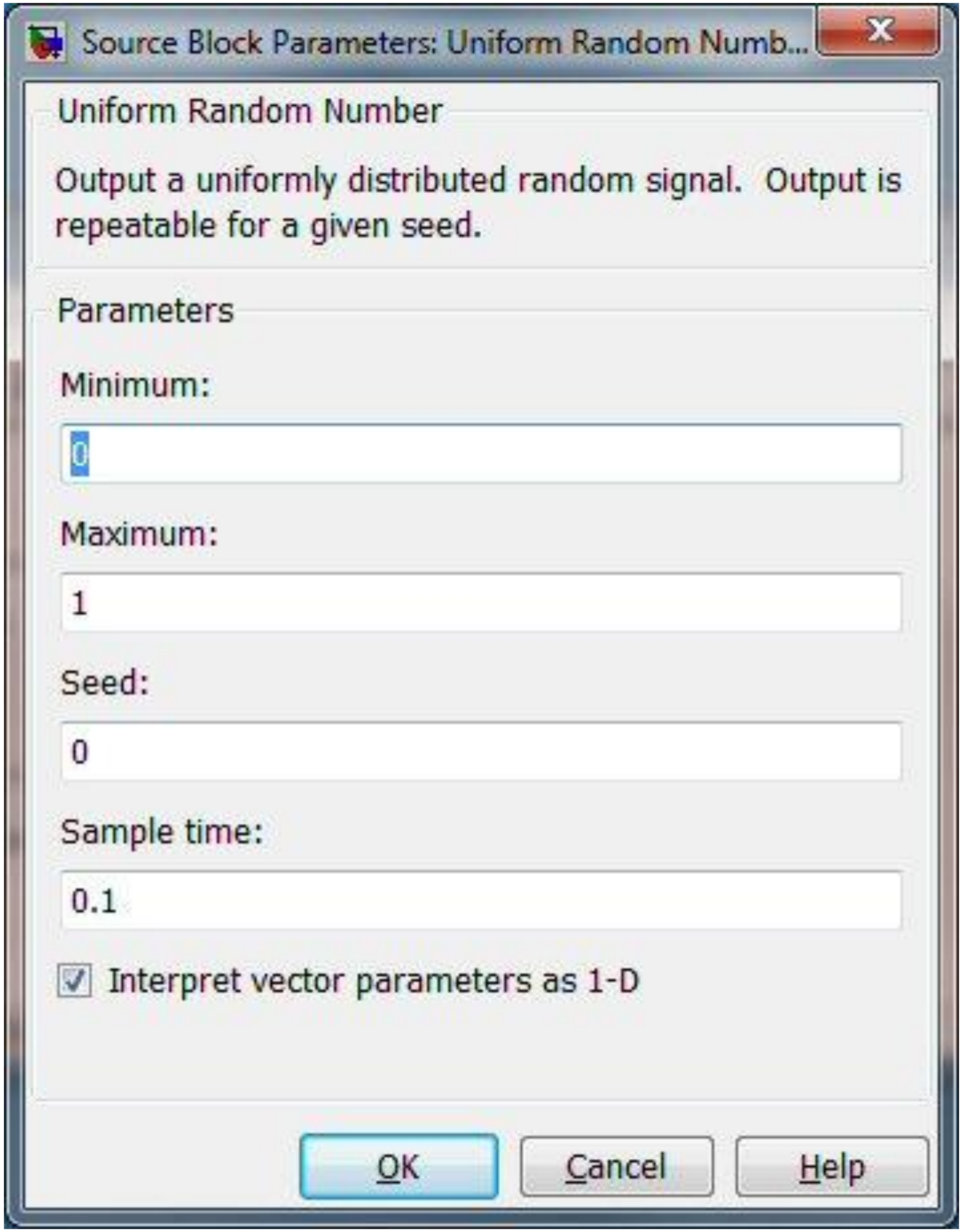
Bias: 0

Frequency (rad/sec): 2

Phase (rad): 0

Sample time: 0

☑ Interpret vector parameters as 1-D



Source Block Parameters: Uniform Random Numb...

Uniform Random Number

Output a uniformly distributed random signal. Output is repeatable for a given seed.

Parameters

Minimum: 0

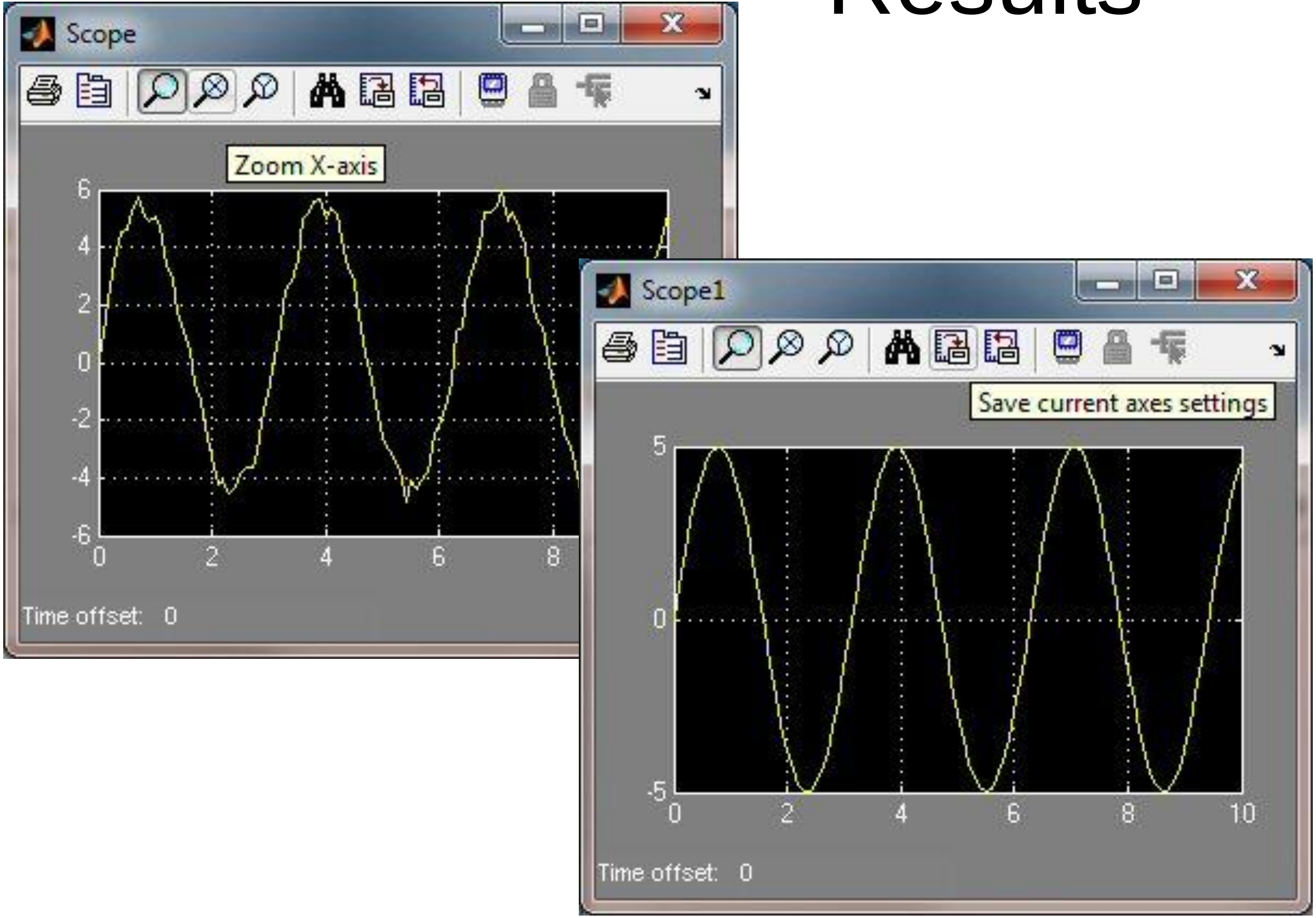
Maximum: 1

Seed: 0

Sample time: 0.1

☑ Interpret vector parameters as 1-D

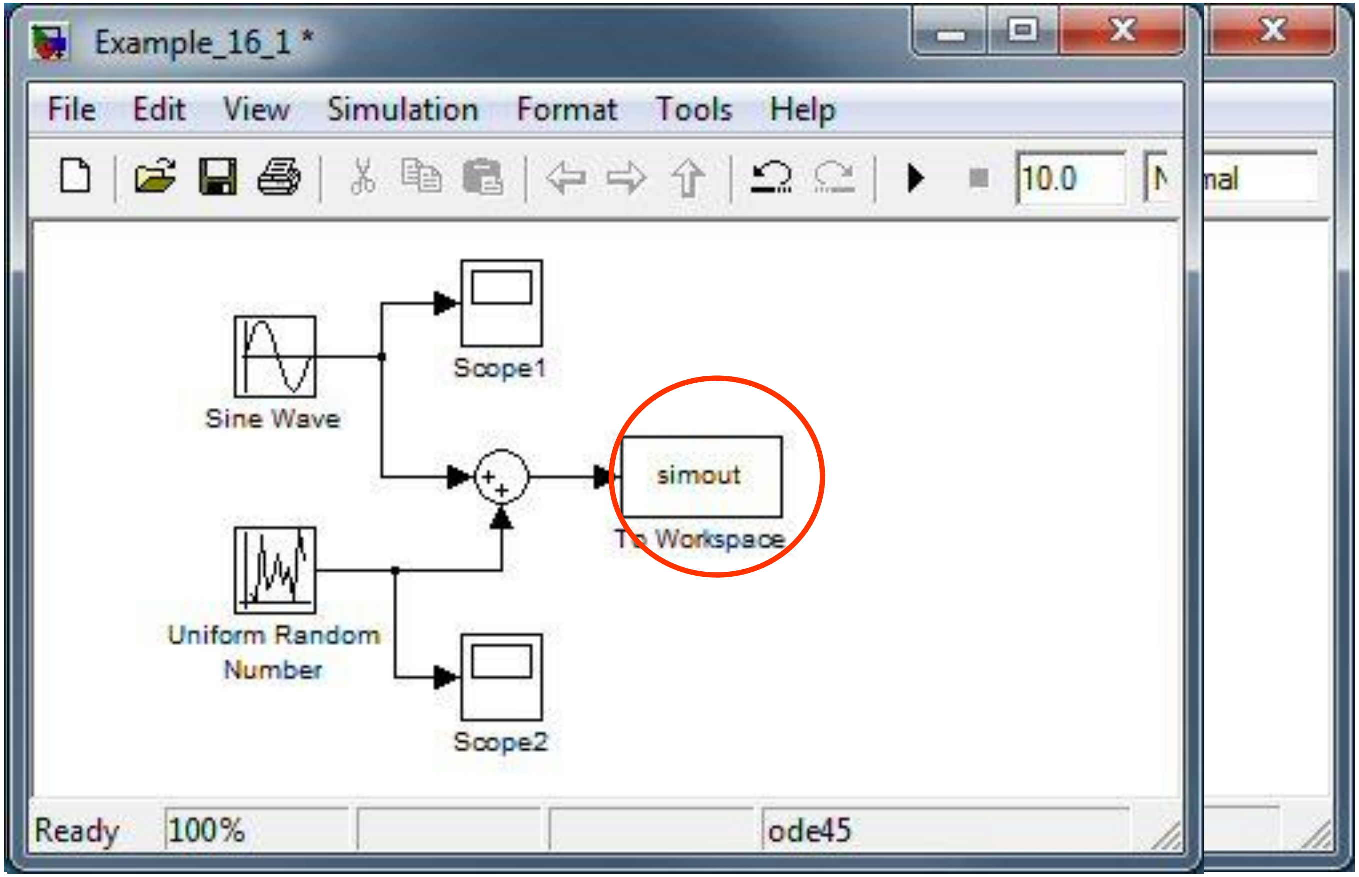
Results



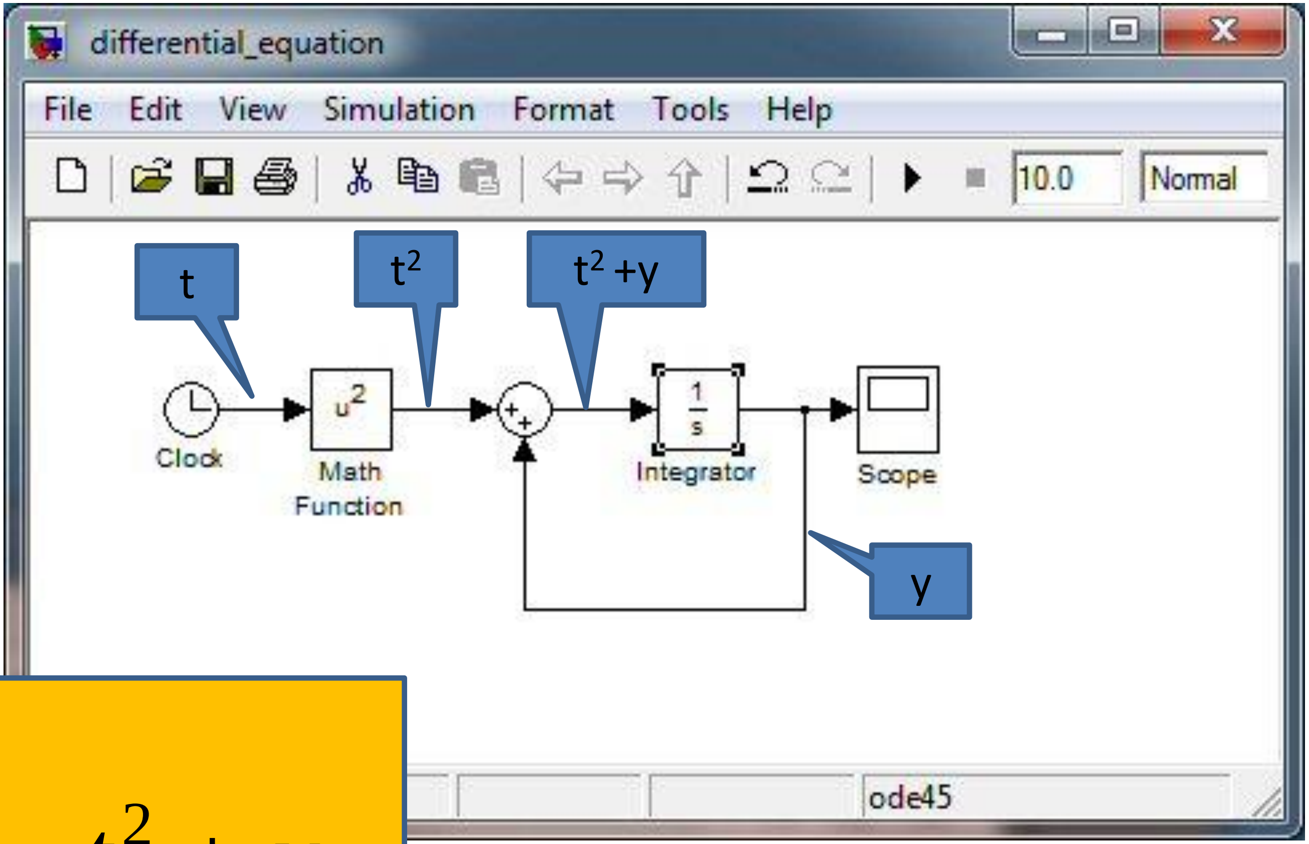
Time offset: 0

Time offset: 0

Instead of sending the results to a ‘Scope’ – we could send them to the MATLAB workspace



9.2 Solving Differential Equations



t

t^2

$t^2 + y$

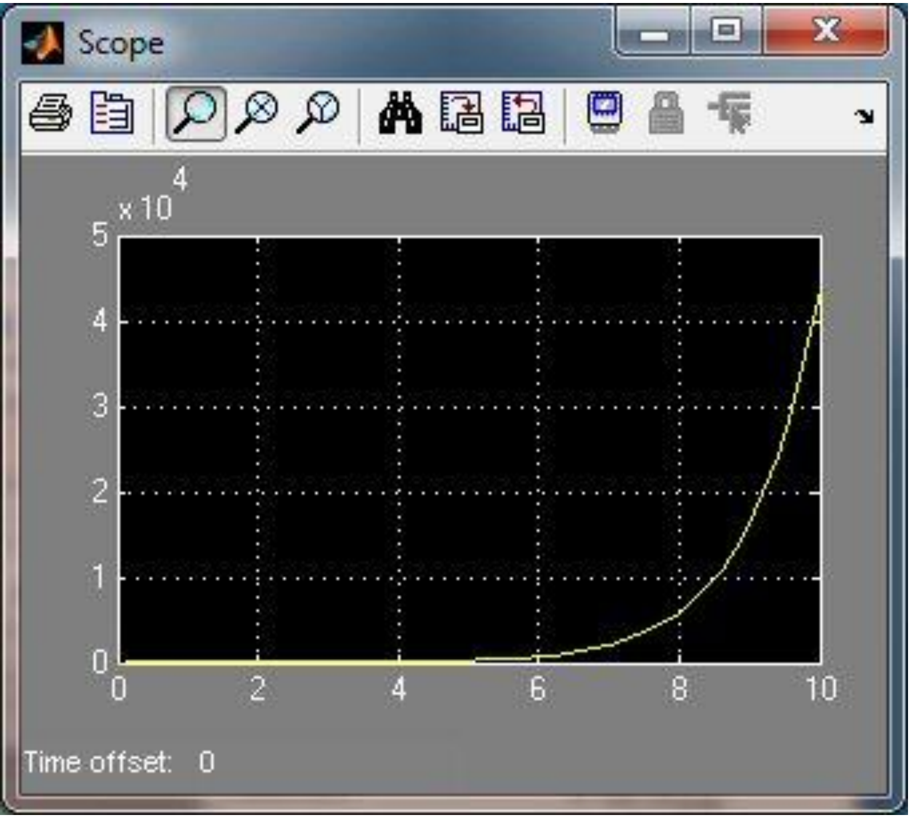
y

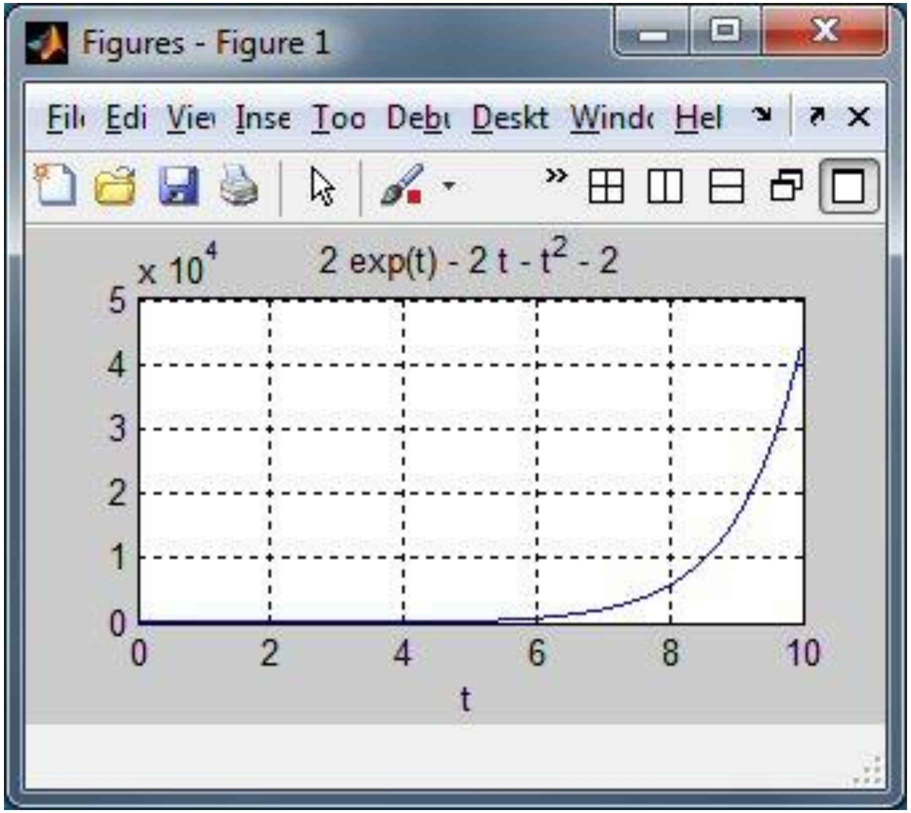
$$\frac{dy}{dt} = t^2 + y$$

The blocks include the following:

- A clock, to generate times (Source library)
- A math function block, modified in the parameter window to square the block input (Math Operations library)
- A sum block (Commonly Used Blocks library)
- An integrator block (Continuous library)
- A scope block (Sink library)

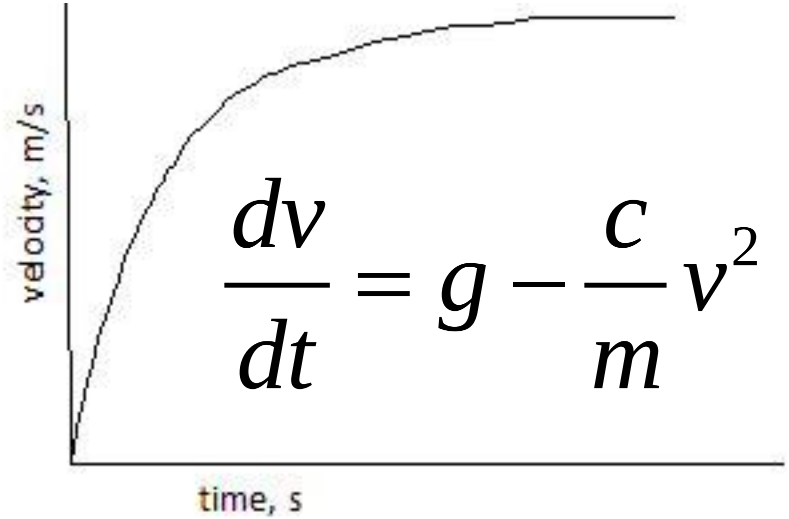
Results from Simulink and from MATLAB’s Symbolic Algebra approach



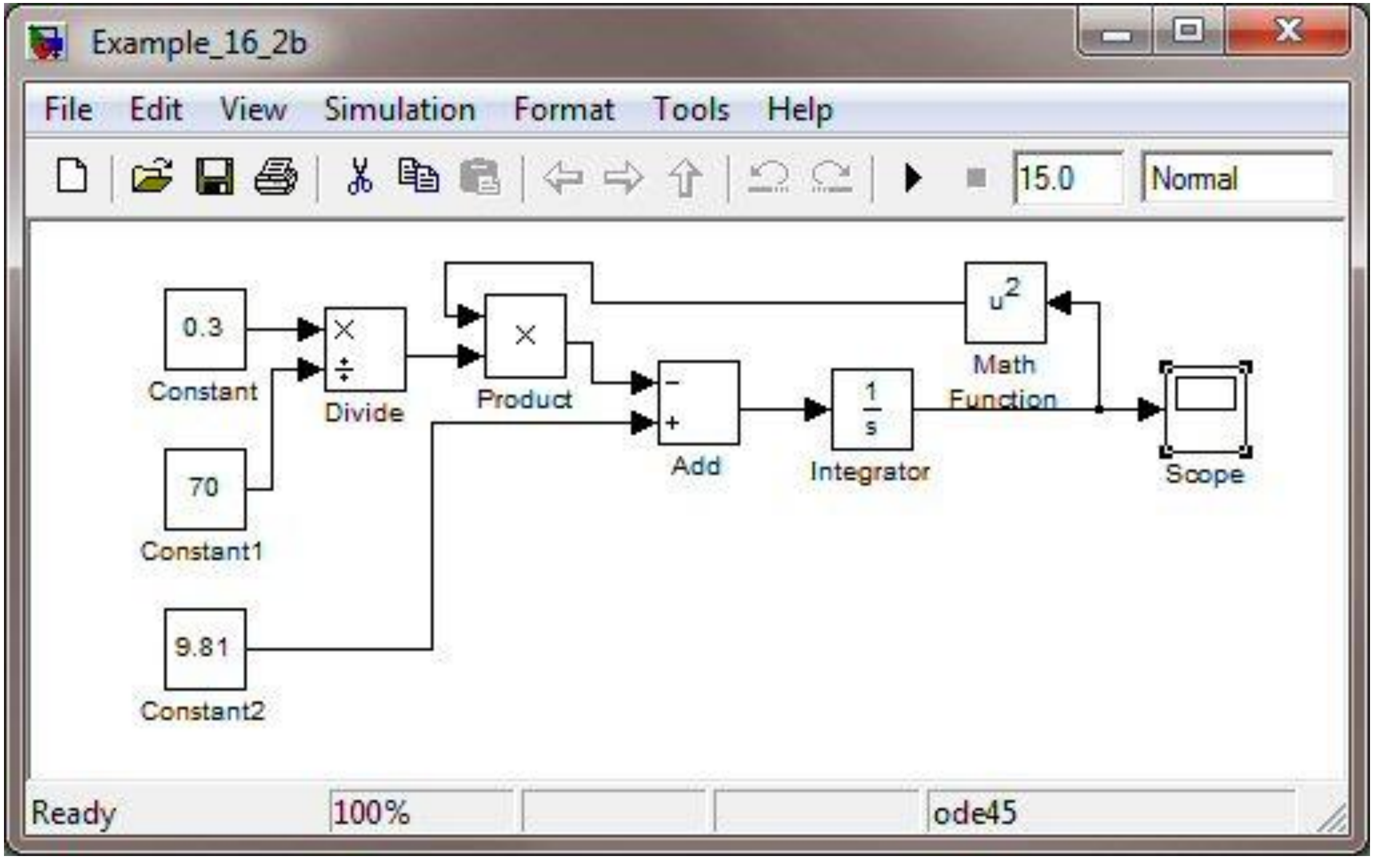


Example 9.3
Behavior of a Falling Object

Predicted behavior when drag is considered

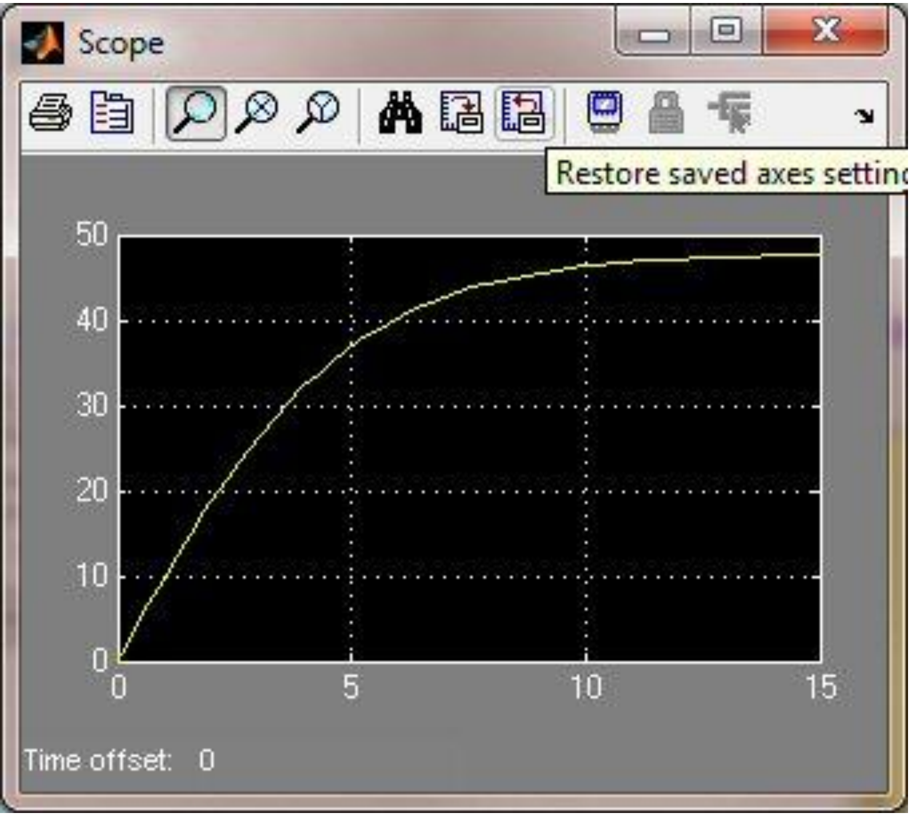


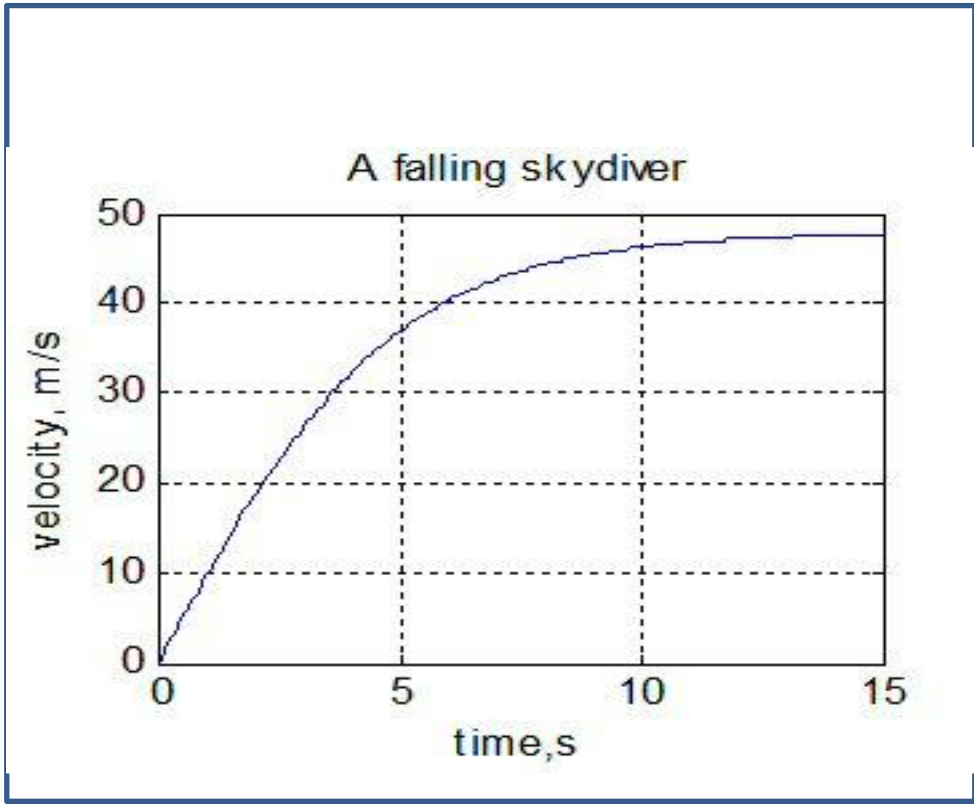
$$\frac{dv}{dt} = g - \frac{c}{m} v^2$$



Example_16_2b
File
Edit
View
Simulation
Format
Tools
Help
15.0
Normal
0.3
Constant
70
Constant1
9.81
Constant2
Divide
Product
Add
Integrator
Math Function
Scope
1/s
u2
Ready
100%
ode45

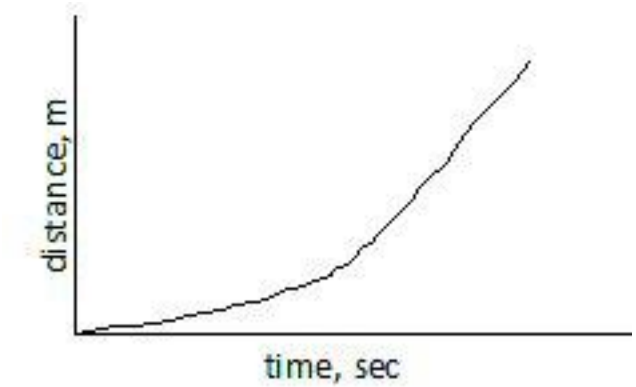
Results from Simulink and from MATLAB's Symbolic Algebra approach





Example 9.4
Position of a Falling Object

Estimate of Position, assuming drag



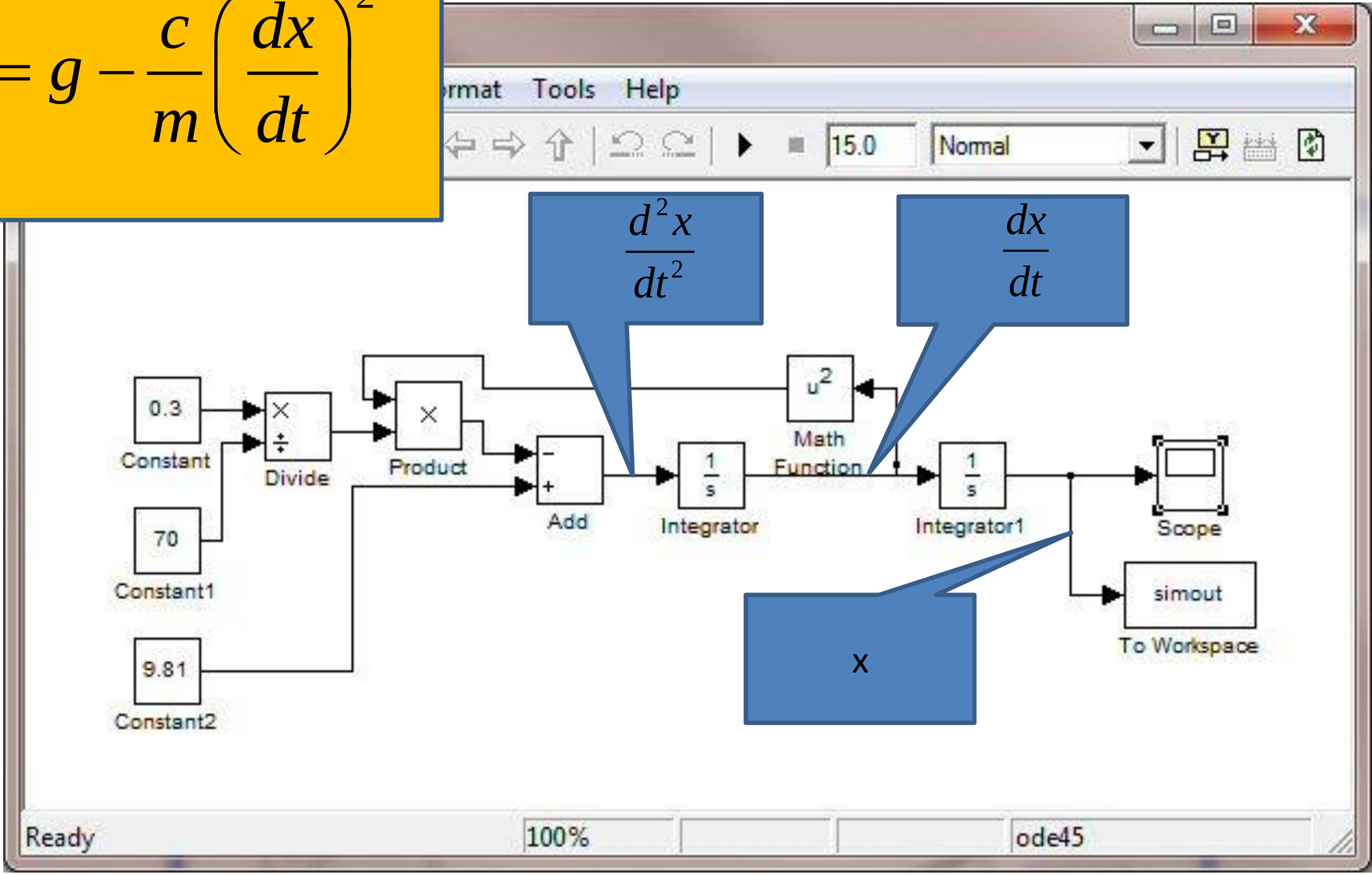
$$\frac{dv}{dt} = g - \frac{c}{m}v^2$$

$$v = \frac{dx}{dt}$$

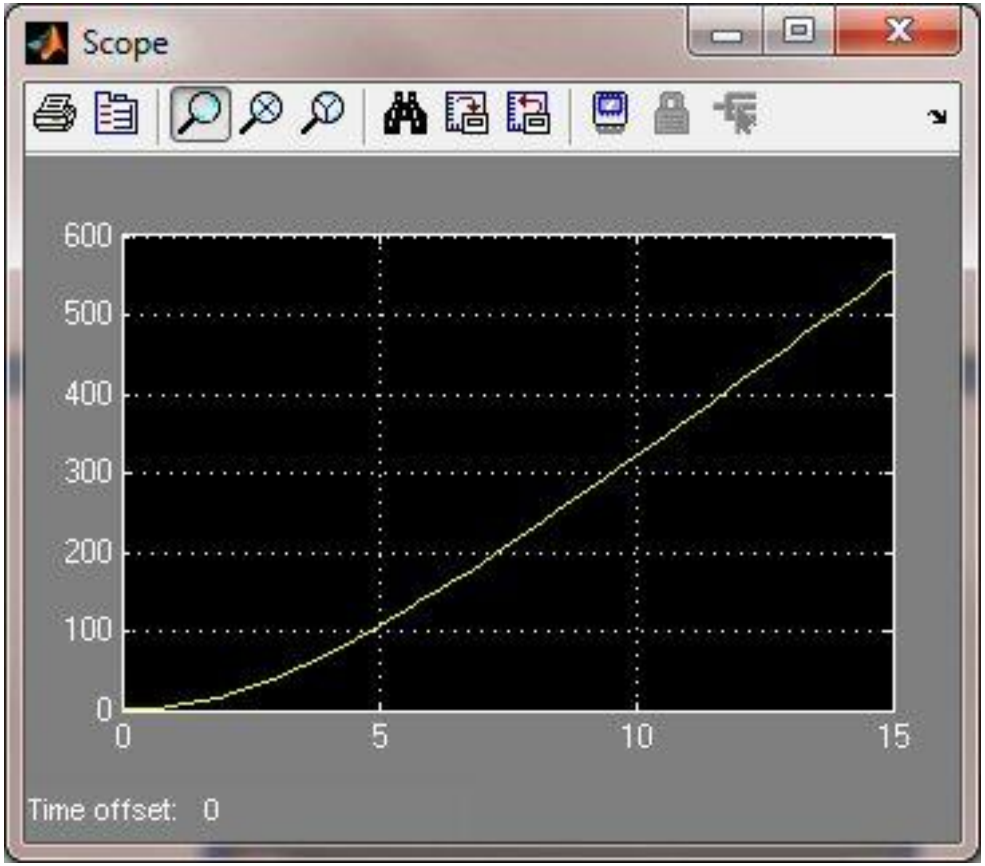
$$\frac{d^2x}{dt^2} = g - \frac{c}{m}\left(\frac{dx}{dt}\right)^2$$

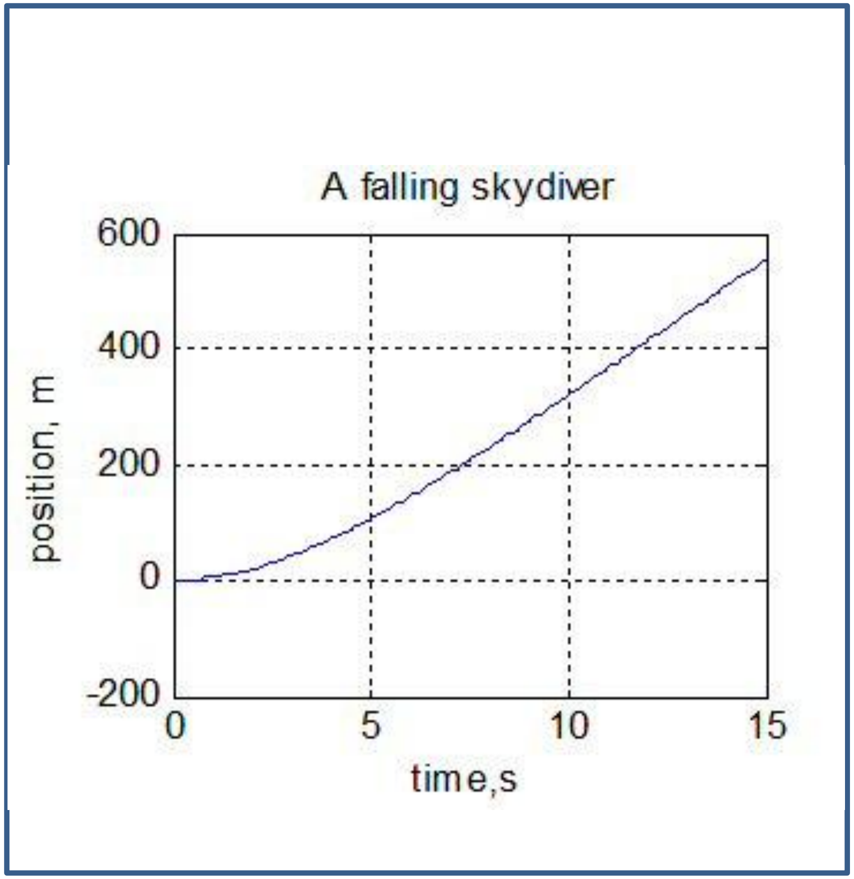
Simulink Model

$$\frac{d^2x}{dt^2} = g - \frac{c}{m}\left(\frac{dx}{dt}\right)^2$$



Results from Simulink and from MATLAB's Symbolic Algebra approach





Summary

- Simulink uses a graphical interface to create models
- It is especially useful with dynamic systems