THE UNIVERSITY OF RHODE ISLAND

EGR 106
Foundations of Engineering II

Lecture 2
Scripts and Arrays

THINK BIG WE DO℠

This Week's Topics

Scripts

- Script concept and useful script commands
- Matlab file editor
- Example: Week 1 Assignment

Arrays

- Arrays & array size
- Creating Arrays in Matlab
- Concatenation, the "Colon" Operator and the `linspace` command

Scripts – Simple Programs

Last week, commands have been typed in the command window:

- Executed by pressing “enter”
- Edited using the arrow keys or the history window

This is a very tedious method to use MATLAB

Much better to save the commands in a script file

Script Concept

A file (called ‘m-file’) containing MATLAB commands

- Can be re-executed
- Is easily changed/modified
- Is savable and can be e-mailed to someone else

Commands are executed one by one, sequentially

- File is executed by typing its name (without .m)
- Results appear in the command window
  - (or use ; to suppress output to the command window)

Can be created using any text editor

- .m extension
- Listed in Current Directory window

Useful Script Commands

clc – clears the command window

clear – clears variables from workspace

close all – closes all figure windows

Good to use at beginning of code

% - creates comment, everything to the right of % is ignored

pause – stops operation and waits for a key press

pause(n) – stops operation and waits for n seconds

disp – a simple way to display text or arrays in the Command Window

format compact – removes spaces between lines in the command window output

Example: Week 1 Assignment

1. $3 * 8 - \frac{25}{3+2}$ = _________________________ . Think about which operation happens first (so called *precedence of operators*); MATLAB follows PEMDAS.

2. 3^{π} = _________________________ . A superscript means raise to a power, computed in MATLAB using the hat operator ^. In MATLAB, the constant π is entered using the command *pi*.

3. $\sqrt{3.1415}$ = _________________________ . Use lookfor and help to find out how to compute square roots. Note that lookfor takes only a single word to search with.

4. $5.15 + 9.3\, e^{3.14}$ = _________________________. *e* is the mathematical constant 2.71828… so *e* with a superscript is that constant raised to a power. While the hat ^ operator would work, for the special case of *e* there is a built in function in MATLAB called exp. Type help exp in the command window to learn how to use this. Note the need for parentheses to identify the exponent! Also, note that the 9.3 is multiplying the exponential – be sure to use a multiplication sign!

Example: Week 1 Assignment (cont.)

5. For this problem you will generate a simple graph using MATLAB. Specifically, for t between 0 and 14 sec you will plot the x-y trajectory of a projectile released with initial velocity v_0=100 m/sec at an angle A=35 degrees. The corresponding equations are:

$$x = (v_0 \cos A)t, \qquad y = (v_0 \sin A)t - \frac{1}{2} g t^2$$

You should use the following commands to generate the plot – type them exactly as shown (including the periods), the particular syntax used will make more sense over the next few weeks. Note that MATLAB uses angles in radians.

```
v=100; A=35*pi/180;
t=0:0.01:14;
x=v*cos(A)*t;
y=v*sin(A)*t-0.5*9.81*t.^2;
plot(x,y)
```

Then use the commands xlabel, ylabel, and title to annotate the plot (use help to find out how to use these); make up some appropriate labels. Finally, to personalize it, use the command text to include your name and today's date on the plot. Once you have the plot visible on your screen, call either the TA or myself over to check your result. S/he will initial the box below if you have <u>all</u> of the desired components.

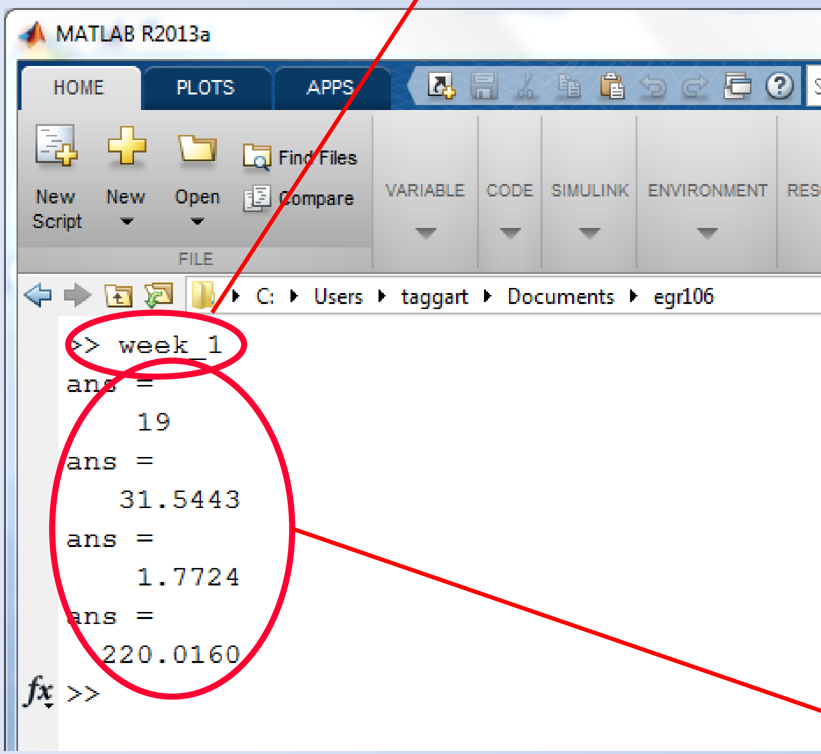
Example: Week 1 Assignment (cont.)

Matlab script (week_1.m):

```
% EGR 106 - Assignment 1 - solution
clc; clear; format compact
% 1
disp('Problem 1:')
3*8-25/(3+2)
pause
% 2
disp('Problem 2:')
3^pi
pause
% 3
disp('Problem 3:')
sqrt(3.1415)
pause
% 4
disp('Problem 4:')
5.15+9.3*exp(3.14)
pause
% 5
v=100; A=35*pi/180;
t=0:0.01:14;
x=v*cos(A)*t;
y=v*sin(A)*t-0.5*9.81*t.^2;
plot(x,y)
xlabel('x')
ylabel('y')
title('Problem 5')
text(200,-100,'John Doe, January 29, 2019')
```

Example: Week 1 Assignment

Type script name in Command Window



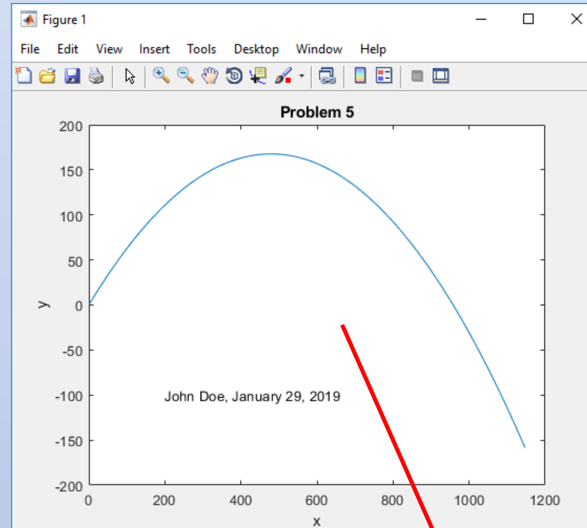


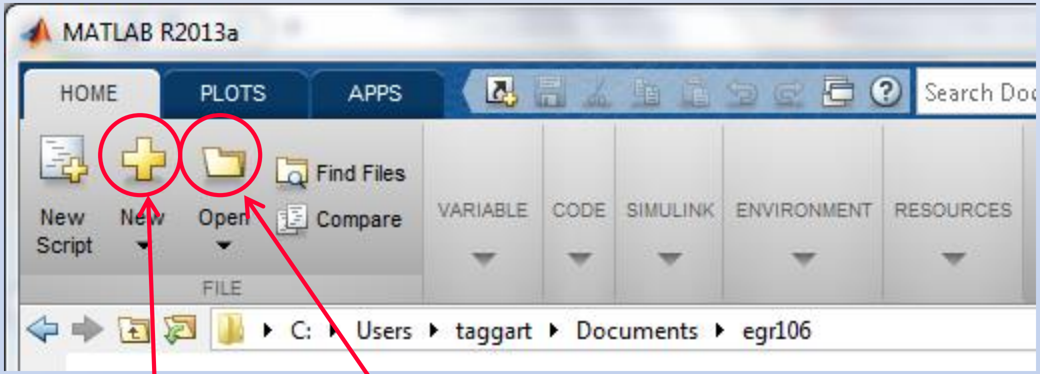
Figure window created by plot command

Script output displayed in Command Window

Matlab's Built-in, Color Editor:

Can create a new file or open an existing M-file (icons or click on file name)

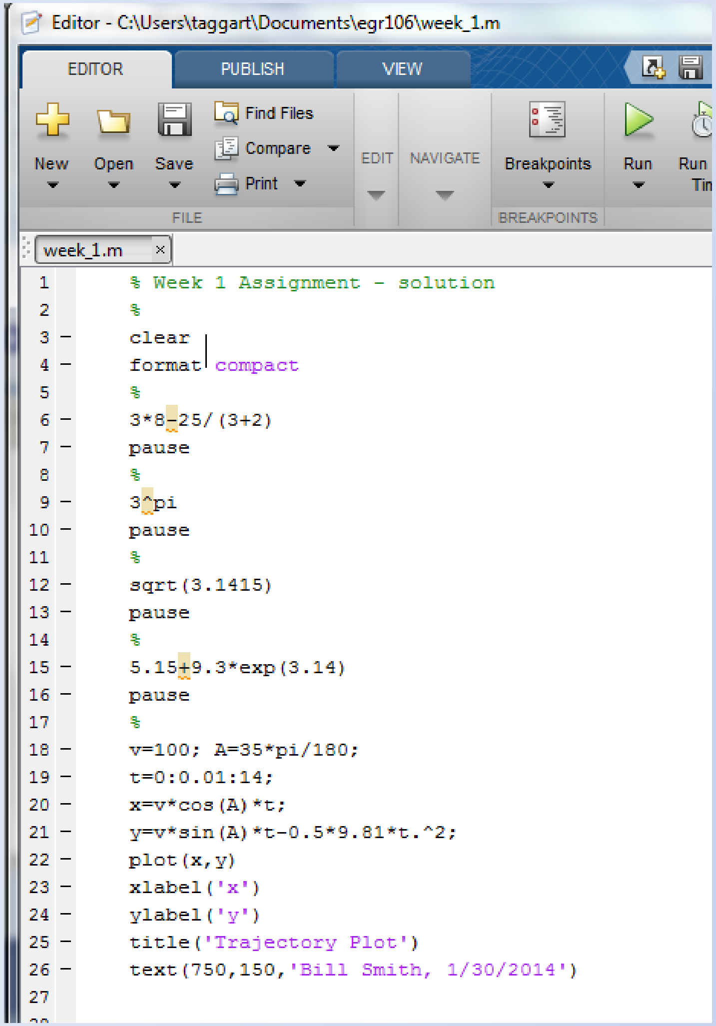
Color used to aid in file creation (command types, typos, etc.)



Opens an Existing File in Editor Window to Edit & Save

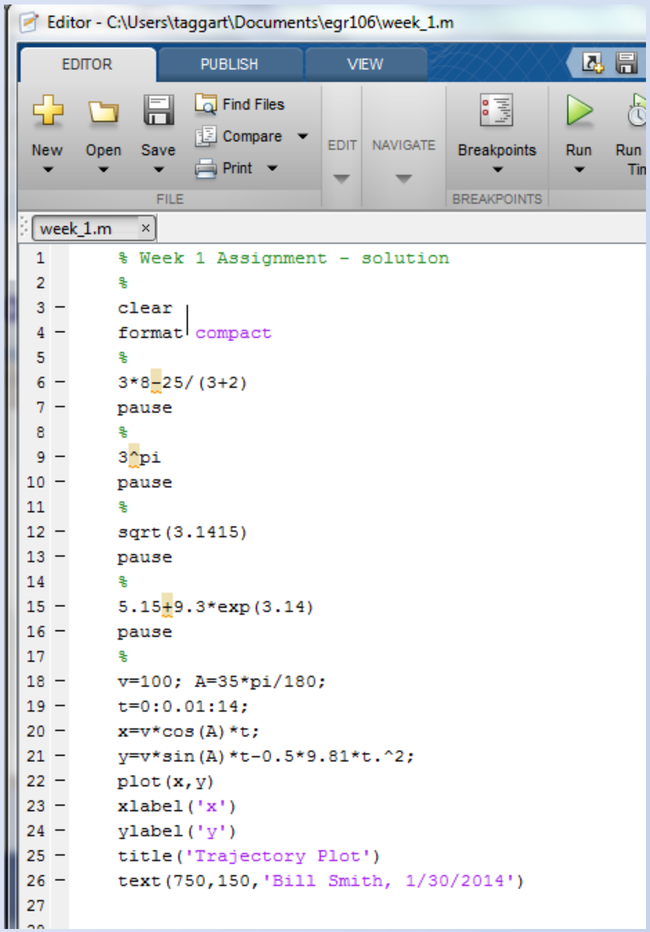
Opens Editor Window to Create New M-File

Editor Window



In this window:
- type & edit commands
- run program
- save work when finished

Features of MATLAB Editor



- Familiar Windows menus
- Line numbers
- “Run” button or F5
- Comment lines (begin with ‘%’)
- Note use of semicolons to suppress output
- Note use of colors

Array Concept

Arrays are the fundamental data units in MATLAB

Rectangular collection of data

<u>All</u> variables are considered to be arrays

$$\text{yield} = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$

Data values organized into rows and columns

Array Size

Size or dimension of an array:

number of rows and columns

written as R by C or R x C

where R = number of rows

C = number of columns

e.g.

$$\text{yield} = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$

yield is 3 by 4

$$\text{test} = \begin{bmatrix} 4 & 5 & 3 & 5 & 0 \end{bmatrix}$$

test is 1 by 5

Special Size Arrays

scalar: 1 x 1 array 4 or [4]

row vector: 1 x C array

$[\ 9\ \ 7\ \ 5\ \ 4\ \ 2\ ]$ is a 1 x 5 row vector

column vector: R x 1 array

$$\begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$$ is a 3 x 1 column vector

Special Size Arrays (cont.)

If $R = C$ => *square* matrix

$$\begin{bmatrix} 4 & 5 & 3 \\ 10 & 4 & 66 \\ 18 & -3 & 2 \end{bmatrix}$$

If $R = C = 0$ => *null* matrix [ ] (a pair of empty brackets)

Creating Arrays

Direct specification:

- Name followed by an equal sign (=), just like variables
- List values within a pair of brackets ([ ])
- Enter data one row at a time
  - left to right, top to bottom order
  - space or comma between the values
  - rows separated by semicolons or the *enter* key

Creating Arrays - Examples

To get

$$b = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$

type

b = [4,5,3,9; 10,4,66,20; 18,-3,2,0]

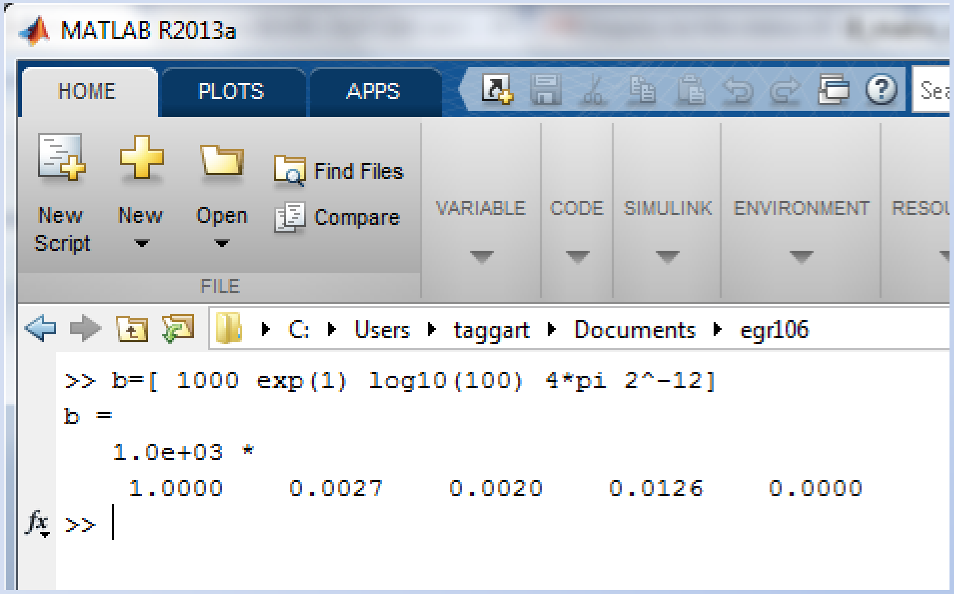
or

```
b = [  4,  5,  3,  9
      10, 4,  66,  20
      18,  -3,  2, 0  ]
```

Press Enter to start new lines

Creating Arrays (cont.)

Can use simple math operations as well as numbers as the entries:



Note the common format of all entries in the response

(exp(1) = e = 2.71828, $\log_{10}(100) = 2$, $2^{-12} = 0.00024414$)

MATLAB scales the exponent to the largest entry !!

Creating Arrays (cont.)

Scaling is sometimes deceptive:

```
Command Window
>> a = [ 3000*pi 3 .03 ]

a =

   1.0e+003 *

     9.4248     0.0030     0.0000

>> b = [ 3000*pi 3 0 ]

b =

   1.0e+003 *

     9.4248     0.0030          0
```

Not really zero

Really zero

Concatenation

Concatenation – “gluing arrays together”

if a = [1 2 3] & b = [4 5 6]

Attaching left to right – use a comma

[a, b] $\longrightarrow$ $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$

Attaching top to bottom – use a semicolon

[a; b] $\longrightarrow$ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Concatenation (cont.)

Note that sizes must match for this to work:

if $a = [\,1\;\;2\;\;3\,]$ $\quad b = \begin{bmatrix} 4 & 5 \\ 10 & 4 \end{bmatrix}$

then

$[\,a, b\,] = ??$ $\qquad [\,a; b\,] = ??$

Size needs for concatenation:

# of rows the same for side by side (comma)

# of columns the same for top to bottom (semicolon)

The Colon Operator

The *colon* operator

first : increment : maximum

yields a row vector of equally spaced values

Examples:	0 : 2 : 10	=>	[0 2 4 6 8 10]
	1 : 5	=>	[1 2 3 4 5]
	7 : -2 : -3	=>	[7 5 3 1 -1 -3]
	1 : 2 : 8	=>	[1 3 5 7]

Note – does not hit 8!!

Note: default for increment is 1

The `linspace` Command

`linspace` – like the colon operator, but definitely gets the last number on the list

linspace (start, last, number of values)

Examples:

```
linspace(0,10,6)  =>   [ 0  2  4  6  8  10 ]
linspace(0,1,4)   =>   [ 0  0.333  0.667  1 ]
```

Default for number of values is 100

```
linspace(0,10)  => [ 0  0.101 0.202 … 10  ]
```

100 points, 99 intervals