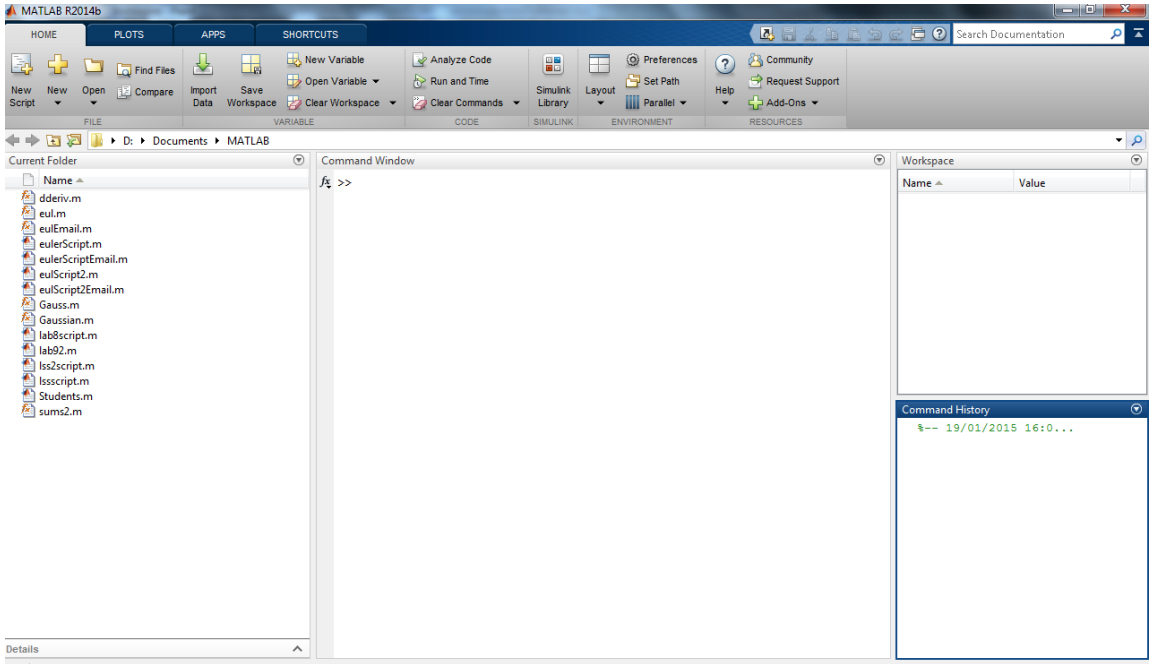
MATH 340 - Lab Instructor: Valeria Barra

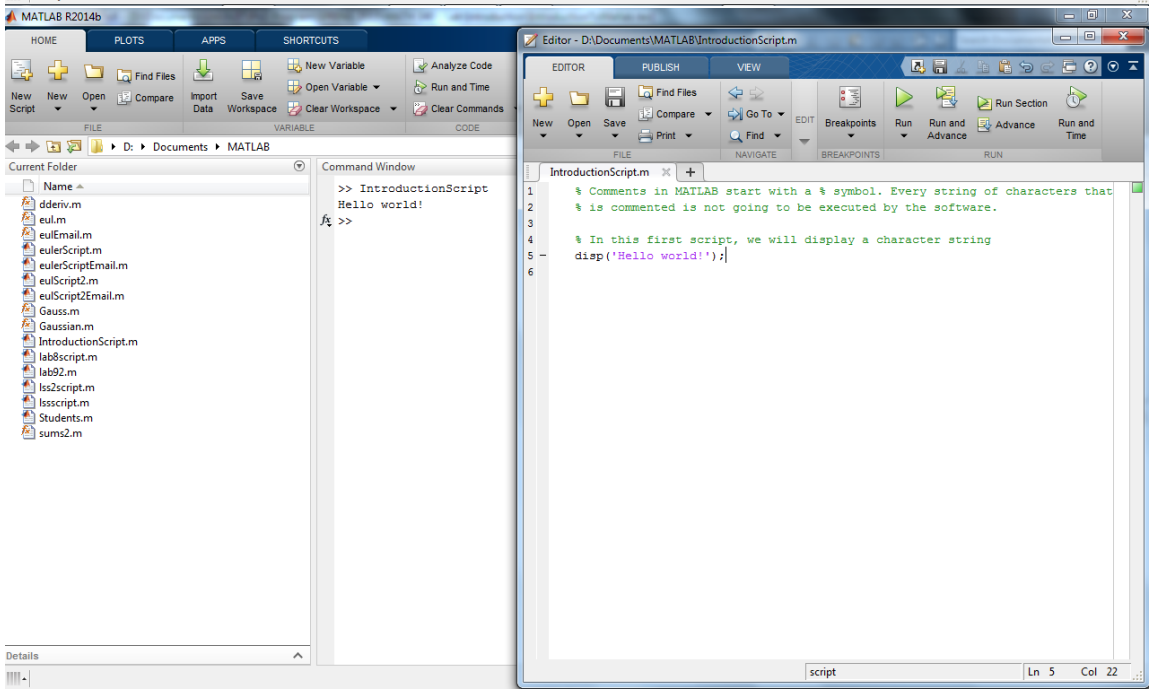
Introduction to MATLAB programming

How to get started

When you open MATLAB, you can have different views, but one of the default ones is usually as in the following figure. Usually on the left you see the Current Folder, where you need to have the files that you want to run. The Command Window is the environment where you can type and execute commands, or in any moment ask for the documentation of any command by typing `help name_of_the_command`. The Workspace is a very useful environment, especially to debug your code, because it lists all the variables stored in that moment, their data types and their sizes. If you want to see the content of any of those variables, just double click on their name and you will see another tab with the contents of your Variables. The Command History summarizes all the commands or executions of the scripts that you have run; by double clicking on any of those commands in the list you can retrieve the same command. We will open a new Script file by clicking the button on the top left corner and we will write our program in there rather than in the Command Window, so that our file will be saved and not lost when the software is closed. When you open a new Script or a new Function, MATLAB shows them in the Editor environment. This can be docked or undocked from the main page of the software. The outputs and the Error messages will be shown by MATLAB in the Command Window.

The result of the execution of any line that is not terminated by a `;` (semicolon symbol) is going to be shown in the Command Window. BUT we will not use this gimmick to display our outputs, rather we will learn how to use the commands `disp`, `sprintf` and `fprintf`. Please see the file "Printing and Formatting your outputs" at my webpage https://web.njit.edu/~vb82/teaching.html.


MATLAB R2014b
HOME
PLOTS
APPS
SHORTCUTS
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
D: ▸ Documents ▸ MATLAB
Current Folder
Name
dderiv.m
eul.m
eulEmail.m
eulerScript.m
eulerScriptEmail.m
eulScript2.m
eulScript2Email.m
Gauss.m
Gaussian.m
lab8script.m
lab92.m
lss2script.m
lssscript.m
Students.m
sums2.m
Command Window
>>
Workspace
Name
Value
Command History
%-- 19/01/2015 16:0...
Details
Ready


MATLAB R2014b
HOME
PLOTS
APPS
SHORTCUTS
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
FILE
VARIABLE
CODE
D: ▸ Documents ▸ MATLAB
Current Folder
Name
dderiv.m
eul.m
eulEmail.m
eulerScript.m
eulerScriptEmail.m
eulScript2.m
eulScript2Email.m
Gauss.m
Gaussian.m
IntroductionScript.m
lab8script.m
lab92.m
lss2script.m
lssscript.m
Students.m
sums2.m
Command Window
>> IntroductionScript
Hello world!
>>
Details
Editor - D:\Documents\MATLAB\IntroductionScript.m
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
Go To
Find
EDIT
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
BREAKPOINTS
RUN
IntroductionScript.m
1 % Comments in MATLAB start with a % symbol. Every string of characters that
2 % is commented is not going to be executed by the software.
3
4 % In this first script, we will display a character string
5 disp('Hello world!');
6
script
Ln 5
Col 22

Data Types

In MATLAB there are several types of variables, but by default MATLAB stores all numeric variables as double-precision floating-point values. The collection of numerical variables in MATLAB via *arrays* is very handy. In MATLAB every array is indexed starting with the entry number 1, unlike other programming languages, such as C, C++. It is not needed to declare the data type of the variable you are creating, but you can simply declare it and initialize it at the same type. ALL assignments in MATLAB, and programming in general, require that you specify the value that you want to assign on the right hand side of an equal sign and the variable you want to store it into on the left hand side of your assignment, as in the examples:

```
A = [1 2 3]; % a row-array that stores the numbers 1, 2, 3 (the entries in each row can also be separated by commas). Spaces before and after the equal sign "=" are not considered, nor necessary
B = [4; 5; 6]; % a column-array that stores the numbers 4, 5, 6
```

To reference one or more indexed entries of an array, MATLAB uses the round parentheses:

```
C = A(2); % it assigns the second entry of the array A to the variable C
D = [1 2 3; 4 5 6]; % a 2 × 3 matrix called D
E(1,2) = D(2,3); % it references to the second row and third column entry of the matrix D and assigns it to the first row and second column of the matrix E
```

The symbol : (colon) in MATLAB is very useful (it reads "from ... to ..."), and it allows to reference group of consecutive entries, as in the examples:

```
F = 1:10; % it creates an array of integers equally spaced (the default spacing is 1) and assigns it to the variable F
G(1,2:11) = F; % it assigns the array F to the first row of G, from the second to the eleventh column
H = 0:0.1:1; % it creates an array that has as first entry a 0, that it is uniformly spaced with spacing 0.1, and it has as last entry 1. Q: How many entries does this
```

array have? A: If you are not sure, you can retrieve the length of any array by simply using the command `length`

Operations with arrays

If you multiply two arrays or matrices `A` and `B`, by simply typing `A*B`, `MATLAB` will perform a row-by-column matrix multiplication. Any operation between arrays or between an array and a scalar that you want to perform componentwise in `MATLAB` is very easily performed by the " . " (dot operator). Examples:

```
I = A.*B'; % the component-wise product of the arrays A and B transposed (note the transposition of the vector B by the ' operator) is stored inside of an array I
   >> I = 4   10   18
L = A*B; % the row-by-column matrix multiplication between the 1 × 3 array A and the 3 × 1 array B produces a scalar, as a result of the scalar product or "dot product" between vectors
   >> L = 32
M = B.^2; % it elevates to the power 2 each entry of the array B
```

Note that `MATLAB`, that stands for "Matrix Laboratory", is really very handy to work with arrays and matrices. The examples just shown above with the shortcuts ":" or "." would be needed to be calculated through the use of loop control sequences in any other programming language.

Loops

With loop control statements, you can repeatedly execute a block of code.

`for` loop:

`for` statements loop a specific number of times, and keep track of each iteration with an incrementing index variable, that is AUTOMATICALLY incremented by `MATLAB`

(the default increment is +1). Examples:

```
for i=2:9 % if no spacing is specified, the default one is +1
   A(i) = 2*i; % every time it multiplies 2 * i and assigns it to the i^th entry of a variable A. (hence, it will allocate from the second to the ninth entry of A)
end

for j=0:2:8 % the index j starts from 0 and with spacing 2 goes up to 8
   B(j+1) = 2^j; % it stores the even powers of 2 from 2^0 to 2^8 inside of the 1^st, 5^th, 7^th, 9^th entries of B
end
```

`while` loop:

`while` statements loop as long as a condition remains true. For example:

```
i = 10; % the scalar i is initialized at 10
while i >= 0
% until i is greater or equal to 0
   i = i-2; % every time it subtracts 2 to i
end
```

Q: What will be the final value of the variable `i`?

MATLAB functions

To implement a task, it is useful and more organized to create a `function` that will perform the task separately from the Script file where you initialize your parameters and format or display your output. Not only this saves you time, it allows for generality of your programs and you can easily and quickly retrieve the same block of code and implement it with different numerical values. For example, if I ask you to write a program that computes the summation of some integer numbers, it is useful to write a `function` that performs the summation in a separate file from your script and simply call your function by typing its name and the eventual input arguments from your Script file. The syntax in `MATLAB` to declare a new function is:

```
function [ output_args ] = Name_of_your_function( input_args ) % the signature of the function has the key-word function, the output arguments in brackets, an equal sign, the name of your function, and the input arguments in parentheses
```

Example of a function that performs the summation of the first n integers:

```
function [ S ] = MySum( n )
% the body of the function here
S = 0;
for i=1:n
   S = S + i; % Note that every time the variable S is overwritten
end
```

Then I can call this `function` from my Script file with different input arguments as many times as I want. For instance:

```
S1 = MySum(5); % It assigns the output of the call of the function MySum with the argument 5 to the variable S1
```

If you need to execute the same task and therefore call the same `function` with several different parameters, it really saves you a lot of times if you call that `function` in a loop, for instance:

```
for i=1:10
   S(i) = MySum(i); % It assigns the output of the call of the function MySum with the argument i that varies at every loop, to the i^th entry of the variable S that will be an array of increasing size at every loop iteration
end
```

Plotting

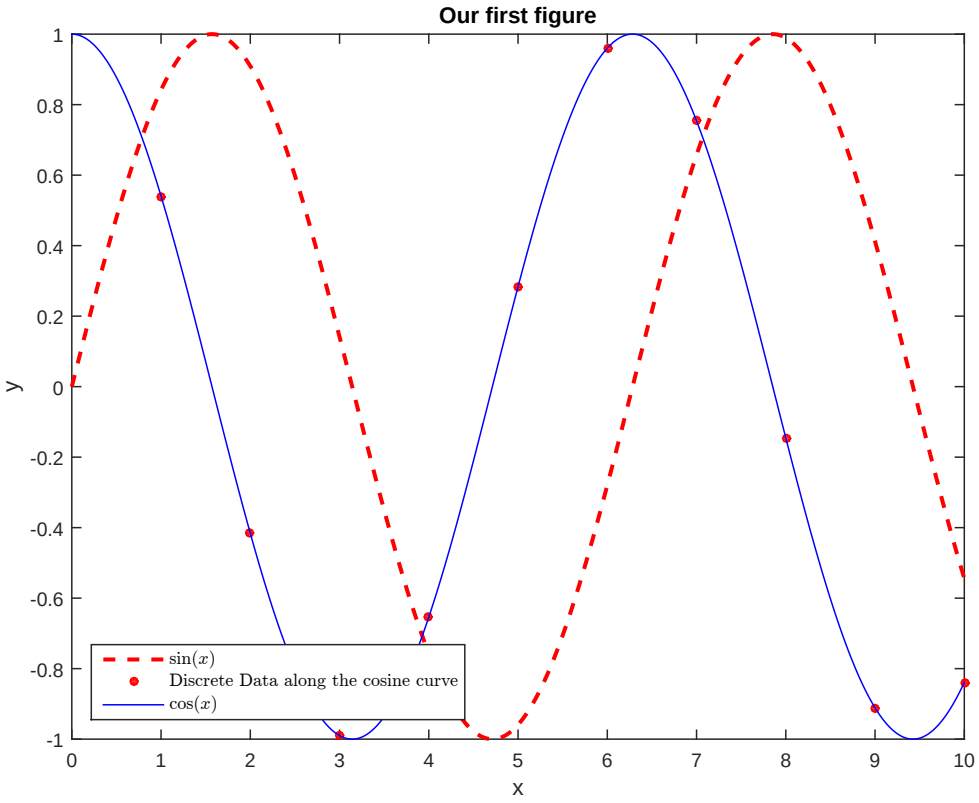
Visual data are very useful to understand your results. It is very important that if I ask you to plot something, you remember to also include axes labels, plot title and the legend, if more than one curve is shown in the same figure. `MATLAB` has several types of line styles, markers and colors, to distinguish different curves in your

graphs. To see a list of them and examples, type `help plot` in your Command Window. Example of two curves superimposed in the same figure through the command `hold on`:

```
x = linspace(0,10,1000); % it creates an array x with 1000 linearly equispaced points between 0 and 10 that will be used as domain of our curves
plot(x,sin(x),'--r','LineWidth',2); % plot of sin(x) with a dashed red line of width 2 (the default value for LineWidth is 1)
hold on % this is the command to plot more curves in the same figure
plot([1:10],cos(1:10),'or','MarkerFaceColor','r','MarkerSize',4); % plot of ten discrete red dots along a cosine curve
hold on
plot(x,cos(x),'b');% plot of a cosine blue solid curve
xlabel('x'); % sets the x-axis label
ylabel('y'); % sets the y-axis label
title('Our first figure'); % sets the title as a character string
legend({'$\sin(x)$','Discrete Data along the cosine curve','$\cos(x)$'},...
'interpreter','latex','location','SouthWest'); % the different legend entries need to be grouped inside of curly parentheses. The legend, as other commands, has some options. One of these that I have used in this example says that we are going to use LaTeXsyntax and the other sets the location of the legend. You can let MATLAB decide this by putting 'best' as argument of the option 'location'. Note the three ... that allow multiline commands in MATLAB
```

How to Debug your code

The first time that you try to run your code, it will probably not work because of some syntax mistake or bug that you have. Any Error message is displayed by `MATLAB` in red in the Command Window. You can also see the link at the line at which the error occurs and go there by simply clicking on it. To Debug your code `MATLAB` has very useful tools like *breakpoints*. Breakpoints serve to stop the execution


Our first figure
y
x
sin(x)
Discrete Data along the cosine curve
cos(x)
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
8
9
10

of your code. It is therefore useful to set a breakpoint before the line at which your error occurs and check the data types or the temporary values of your variables. To set a breakpoint just double click on the little dash next to a line number and `MATLAB` will mark that line with a red dot. Then, when you run the program, it will stop at the set line and by simply hovering your mouse pointer over a variable, `MATLAB` shows you a pop-up window with the summary of the variable data type and content. If it is too long to display in the pop-up window, you can always access the variable's content in the Workspace environment.

How to easily display your code and results

Whenever you have to submit a Lab assignment, you have to SHOW ALL YOUR WORK and RESULTS. To do so in `MATLAB` without the need of copying the text of your programs or the images produced to put them in a separate file to print out, there is the Publish tab in the Editor window that helps you saving a lot of time. The Publish tab in fact prints out all the code that you have in a Script file and the produced output (figures, tables, etc). You can set the extension (html file, Word file, pdf file) of the published document.