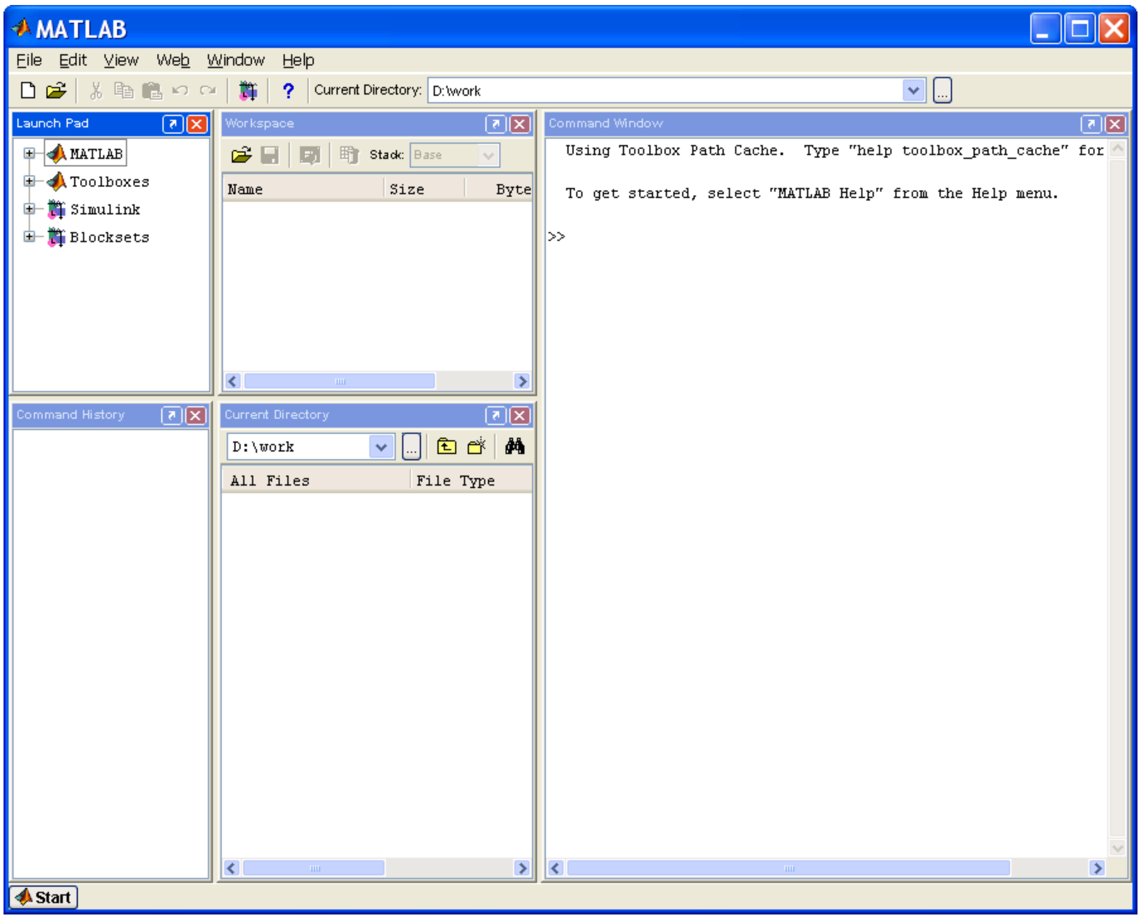
CURRENT SUBJECTS IN COMPUTER SCIENCE

PRACTICE: Introduction to Matlab

1. MATLAB's environment

The next figure shows Matlab's main screen:



```
 Using Toolbox Path Cache.  Type "help toolbox_path_cache" for

 To get started, select "MATLAB Help" from the Help menu.

>>
```

There exists five basic elements in Matlab:

- ***Command window***: Works as a command interface. Basic instructions are typed in the command window.
- ***Workspace***: It shows the variables that are being used by Matlab.
- ***Command history***: It is a list of commands that have been introduced by the user.
- ***Current directory***: It shows the contents of the current directory.
- ***Launch pad***: It permits the user to launch applications in Matlab's environment.

From the menu view it is possible to configurate Matlab's appearance. The user may choose which windows can be activated or hidden.

2. How to find help in MATLAB

You can find help by different ways in Matlab's environment:

- **Command line help**
  You can access help from the command line. The basic syntax is as follows: *help nameoffunction*. Help appears in text format. For example, we can access the help for the function inv that calculates the inverse of a Matrix.

```
>> help inv
```

- **Online help:** www.mathworks.com

- **Examples and demos:** You can access different examples and demos by typing the command demo into the Command window.

3. Variables and matrices in MATLAB

Variable names can be as long as 19 characters. Matlab differentiates between upper and lower case letters. The type of a variable is assigned automatically. Variables can be:

- Integer
- Real
- Complex
- Character

The following sentence creates a variable named `pepe` and assigns the value of 7:

```
>> pepe = 7

pepe =
      7
```

Matlab shows on the screen the result of the operation. We can write a semicolon at the end of the sentence to avoid it. For example:

```
>> pepe = 7;
```

All the variables in Matlab are considered as matrices. For example:

- **n x m Matrix**: Bidimensional matrix
- **n x 1** or **1 x n Matrix**: Vector (it can be accessed in the same way as a matrix)
- **1 x 1 Matrix**: Scalar.

The following line creates a matrix in Matlab and names it A.

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =
      1     2     3
      4     5     6
      7     8     9
```

As you can see, different rows are separated by a semicolon. Columns are separated by spaces. The following points sum up the main operations related to matrices:

- You can test the value of a particular variable, by typing its name in the Command window:

```
>> A

A =
      1     2     3
      4     5     6
      7     8     9
```

- List existing variables in Matlab's environment:

```
>> who

Your variables are:

A         pepe
```

- Delete a particular variable in memory:

```
» clear pepe
» who

Your variables are:

A
```

Now, the variable **pepe** has been deleted from Matlb's workspace.

4. Basic Matrix Computation

Matlab offers a lot of operations related to matrices. For example, if we create the matrix A.

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =
      1     2     3
      4     5     6
      7     8     9
```

- Then, we can access any of its components by its row and column:

```
» A (1,3)

ans =
     3
```

Notice that `ans` *is the default variable where Matlab stores the result of any operation. On the other hand, we can store the result of any operation by asigning it to other variable name, for example:*

```
» k = A(1,3)

k =
     3
```

- We can also acces to a whole row or column of a matrix by using the **:** operator.

For example, we can access all the elements in row 2:

```
» A(2,:)

ans =
     4     5     6
```

Or in column 3:

```
» A (:,3)

ans =
     3
     6
     9
```

- Rows and columns can be also accessed by groups. For instance, you can access rows 1 and 2 and columns 2 and 3:

```
» A(1:2,2:3)

ans =

     2     3
     5     6
```

- Any element in a matrix can be modified and assigned a different value. For example:

```
» A(1,1) = 9

A =

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

- On the other hand, you can also add new elements to a existing matrix:

```
» A(4,4) = 1

A =

     9     2     3     0
     4     5     6     0
     7     8     9     0
     0     0     0     1
```

Matlab completes with zeros the elements which are not specified.

5. Main arithmetic operations

Matlab permits the following operations with matrices (scalars).

• Sum:	+
• Difference:	-
• Product:	*
• Division:	/

- Traspose: ‘
- Power: ^

The next example multiplies two matrices A and B.

```
» A = [1 2;3 4]

A =
     1     2
     3     4

» B = [2 4; 6 8]

B =
     2     4
     6     8

» C = A*B         % matrix product

C =
    14    20
    30    44
```

The following functions are of great importance when operating with matrices in Matlab:

- Obtaining the inverse of a matrix: `inv` function.

```
» A = [1 2;3 4]

A =
     1     2
     3     4

» B = inv(A)

B =
   -2.0000    1.0000
    1.5000   -0.5000
```

- Creating a matrix with zeros or ones.

```
» A = zeros(1,4)

A =
     0     0     0     0

» B = ones(2,3)

B =
     1     1     1
     1     1     1
```

- Creating a vector with consecutive elements:

```
» a = [0:1:5]     % start 0, end 5, step 1

a =
     0     1     2     3     4     5
```

```
» a = [5:-1:0]    % start 5, end 0, step -1

a =
     5     4     3     2     1     0

» a = [0:.2:1]    % start 0, end 1, step .2

a =
         0    0.2000    0.4000    0.6000    0.8000    1.0000
```

6. Working in MATLAB

There exists two different possibilities to work in Matlab:

- **By directly typing commands in the Command window.**

This enables to make operations step by step.

- **By programming in Matlab’s environment (*.m)**

Matlab permits to execute commands that are previously stored in a file. This acts as a batch file, and commands are executed one by one. The file must have a ‘.m’ extension. The file can be executed from the command window by typing the name of the file.

7. Basic Matlab programming

Compared to other programming languages, Matlab offers some advantages to the user, for example:

- Automatic selection of variable type.
- Easy computation of complex numbers.
- Interpreted language.
- Integrated debugger.

8. Control sentences in MATLAB

We show now the main control sentences in Matlab:

Loops:

```
for variable = expresion
      sentences
end


while expresion
      sentences
end
```

Conditional if/else/elseif:

```
if expresion
      sentences
```

```
        elseif expresion
              sentences

        elseif expresion
              sentences

        else
              sentences
        end
```

9. Functions in MATLAB

We would like to create a function in Matlab that calculates the length of the hypotenuse given both sides of the triangle.

Step 1: Creating a file named hypotenuse.m.

Choose the menu option: File->New->M-file. This would open an editor window. Next, write down the following commands.

```
% This is my practice script:
% First define 2 variables:
a = 3;
b = 4;

% Then let's assume a and b are the lengths of 2 sides % of a right
%triangle and let's calculate the length of the other side (the
hypotenuse) using the
% Pythagorean formula:

c = sqrt(a*a + b*b);
%and, let's see the result:
c
```

Step 2: Saving the file

Choose: File->Save, and save the file to c:\tests as hypotenuse.m.

Step 3: Add the path to Matlab’s path.

This step will tell Matlab where to look for the script file in order to execute it. The easiest way to add the new path is with the menu option File→Set Path. Then click on Add Folder and choose the new path (d:\cscs\matlab).

Step 4: Testing the file

On the command window we can execute the script by typing its name:

```
>> hypotenuse

c =

     5
```

Step 5: Creating a function.

Notice that, on the last example, we assigned the two sides of the triangle at the same script. It would result much more practical to create a function with two parameters that returns the result. We can easily add this feature to Matlab, add the following code to the previously created (see next figure).

```
function [c] = hypotenuse(a, b)

% Then let's assume a and b are the lengths of 2 sides % of a right
%triangle and let's calculate the length of the other side (the
hypotenuse) using the
% Pythagorean formula:

c = sqrt(a*a + b*b);
%and, let's see the result:
c
```

We can now call the function that we have just created from the Command window. Notice that both parameters can be now passed directly from the command window.

```
>> hypotenuse(1,1)

c =

    1.4142
```

10. Graphical plots in MATLAB

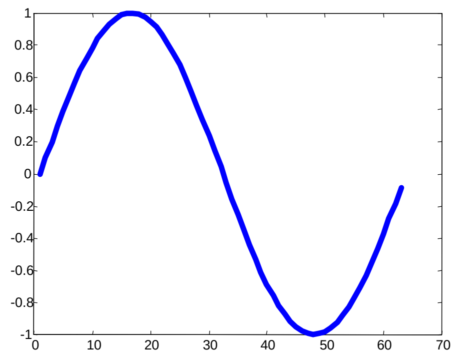
You can easily plot 2D and 3D graphics in Matlab. Most of this functionality is achieved with the function plot. We will show how it works with the following example:

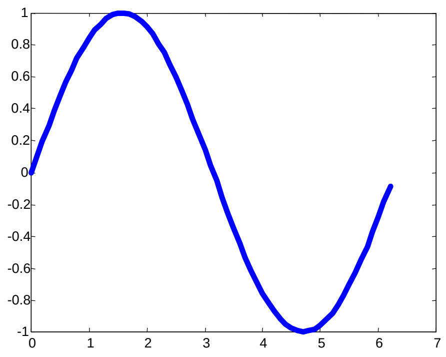
Type the following commands at the Command window.

```
» a = [0:0.1:2*pi];     % a: contains 63 different values from 0 to 2π
» b = sin(a);           % b: results of the sin function at each angle.
» c = cos(a);           % c: results of the cos function at each angle.
```

We can now plot the results in Matlab by means of the plot function:

```
>> plot (b);                        >> plot (a,b);
```





As you may see, at the right figure, **plot** represents the vector **b** against the angles **a** (from 0 to 2π). On the left figure, however, plot represents the results against the its vector index (from 1 to 63). We can represent two graphics at two different windows by using the command '**figure'**. Please, type the following commands:

```
>> plot (a,b);
>> figure;
>> plot (a,c);
```

However, we may be interested in plotting two functions at the same time. This effect can be achieved by means of **'hold on'**. This command tells Matlab to draw the new data without deletting the old one.

```
>> plot (a,b);
>> hold on;
>> plot (a,c);
```

Finally, we can add text to our graphics with the following commands:

- **title:** writes the title of the figure (upper part of the figure).
- **xlabel:** names the x axis of the figure
- **ylabel:** names the x axis of the figure

We can easily find help on these commands by typing:

```
>> help nameoffunction.
```