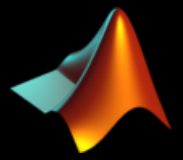
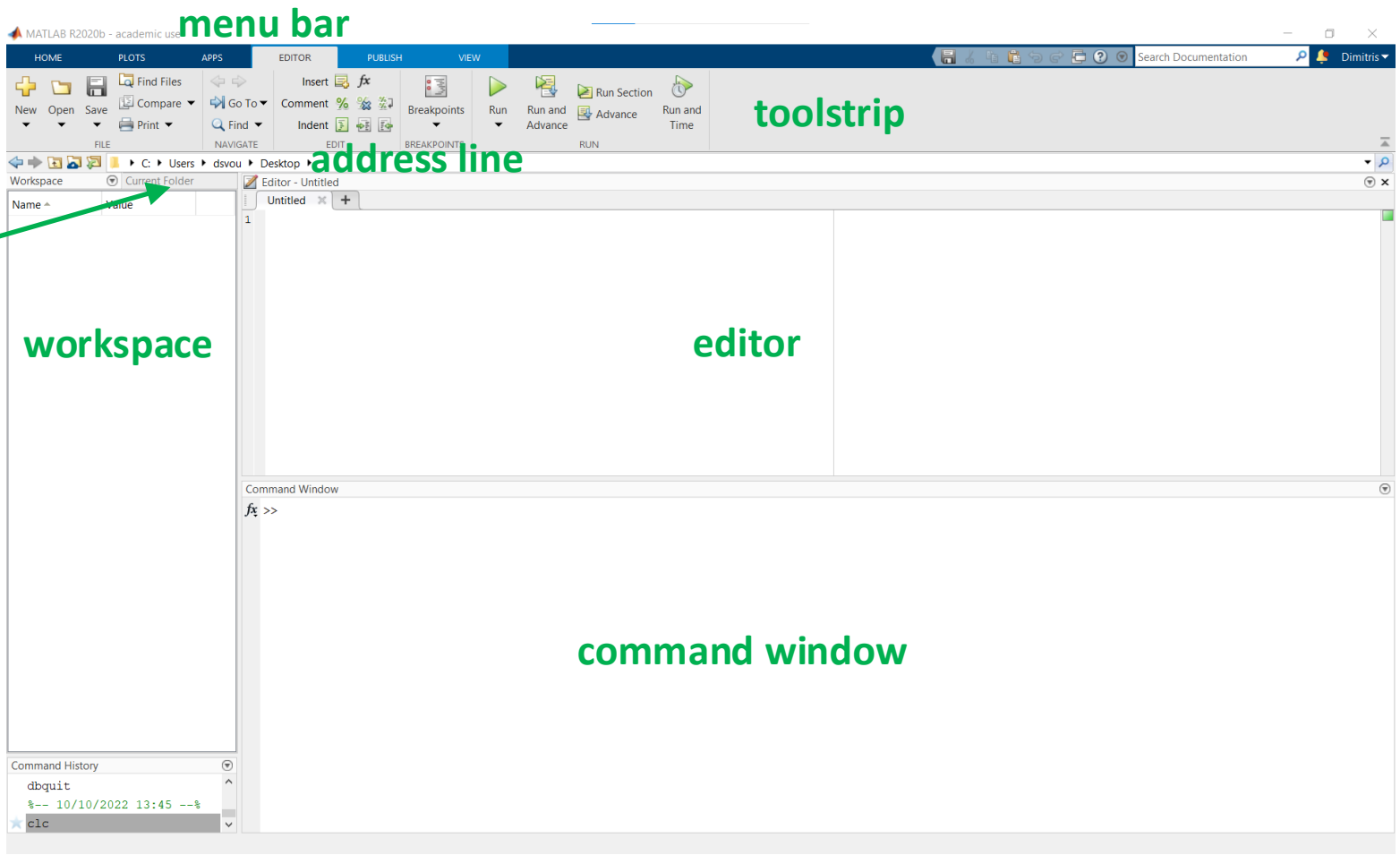


Programming with MATLAB

Basic operations – scalars/vectors

Dimitris Voudouris, PhD
d.s.voudouris@gmail.com

The MATLAB environment



Matlab as a calculator

Perform basic operations by using the following symbols:

+	addition	same class
-	subtraction	
*	multiplication	same class
/	division	
^	exponent	

MATLAB as a calculator

The execution of the operations follows the order:

1. First execute whatever is inside the parentheses
2. Then execute the exponents
3. Then perform the multiplications and divisions
4. In the end, perform the additions and subtractions

For operations of the same class, execute from left to right

e.g.
5 * 2 / 4 * 2 is not expressed as 10 (5*2) divided by 8 (4*2), but as…
10 divided by 4 and the outcome is then multiplied by 2
So the correct answer is **5**

MATLAB as a calculator

Perform on paper the three following series of operations:

((6+9)/3)^2+(4^3/(10/5)-21/(9/3))

(6+9/3)^2+(4^3/(10/5)-21/(9/3))

(6+9)/3^2+(4^3/(10/5)-21/(9/3))

Observe the reasons for why the outcomes differ

MATLAB as a calculator

Keep an eye on the position of the parentheses

Add spacing between elements that are executed together

For instance, this…
((6+9)/3)^2+(4^3/(10/5)-21/(9/3))

can better be typed like this...
((6+9) / 3) ^ 2 + (4^3 / (10/5) - 21 / (9/3))

Now you have some better visualisation of your syntax

Work with variables

You can assign a value to a variable:

a = 6

b = 2

But a value cannot be the name of a variable:

6 = a ---> error

You can perform usual operations with variables, e.g.:

a + b

a ^ b

c = a + b

d = c – (b*a)

Work with variables

Use a semicolon in the end of the operation to suppress the output:

```
c = a + b;
```

If you use a semicolon in the end of the operation, you can type the next command in the same line:

```
c = a + b; d = c – (b*a)
```

But if you do not use the semicolon, then an error will annoy you:

```
c = a + b    d = c – (b*a)
```

Work with variables

Perform some basic operations with variables and add comments:

```
a = 2;                  % assign a value to variable a
b = 3 * a;              % b is equal to 3 times a
c = a * b ^ 2;          % c = 2 * 6 ^ 2 = 2 * 36 = 72
d = c - (b * a);        % d = 72 - (6 * 2) = 72 - 12 = 60
```

Ask MATLAB to display the outcome:

```
disp('the answer is: ');
disp(d);
```

Helping yourself

Ask MATLAB for some help about the *disp* function:

help disp

Saving your code

It is advisable to save your code in an *.m-file:

Type (or copy/paste) your commands in a new editor and save this file as an .m file under your preferred name.

Saving this file in the folder you are working on is usually the best option.

Change your working directory by using the address line below the toolstrip. You will see the contents of your current directory under the panel *Current Folder*

Some useful syntax

Syntax	Functionality
>>	Matlab is waiting for some input
>> muffins = 2	Assigns *muffins* a value of 2
>> guests = 15	Assigns *guests* a value of 2
>> muffins*members	Multiplies *muffins* by *guests*
>> sdjfwqht = 9	Set variable *sdjfwqht* to 9 (**VERY** bad variable name)
>> whos	Lists all the variables currently in use
>> muffins	Check to see what is the value of *muffins*
>> clc	Clears the command window
>> clear	Clears all variables in the Matlab workspace
>> clear guests	Clears only the variable *guests*
>> %comment	Everything after % is a comment or explanation
>> exit	Exit MATLAB

Creating a vector

Create a vector with all its elements having the same value:

```
v = ones(1, 5)          % vector v has 5 elements, all equal to 1
v = ones(1, 1000)       % vector v has 1000 elements, all equal to 1
v = zeros(1, 20)        % vector v has 20 elements, all equal to 0
```

Create a vector with elements of different values:

```
r = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]     % 10 elements, from 1 to 10
r = [1:10]                              % same as the previous
r = [1:3:10]            % values from 1 to 10 in steps of 3
r = [6:3:27]            % values from 6 to 27 in steps of 3
```

Find out the length of your vector:

```
length(r)
```

Row and column vectors

Create new vectors *r* and *c:*

```
r = [1:3:13]          % 5 elements, from 1 to 13 in steps of 3
c = [1; 4; 7; 10; 13] % same with r but now as a column
```

Transpose the vector:

```
r = r'                % row vector r will now become a column vector
c = c'                % column vector c will now become a row vector
```

Access elements

Choose an element of your vector*:*

```
r(2)          % the value of vector r that is at index position 2
c(5)          % the value of vector r that is at index position 5
c(end)        % the value of vector r that is at the last index position
```

Choose a block of elements of your vector:

```
r(1:3)        % chooses the values at the first 3 indices
r(2:4)        % …the second until the fourth index
r(2:end)      % …the second until the last index
r(2:end-1)    % …the second until the one-before-the-last index
v = v(3:4)    % your original vector v is replaced by a new vector v
              that contains only the elements at the 3rd and 4th index
              positions of your previous vector v
```

Add, subtract, multiply, divide vectors

Do the operation:

```
v1 = ones(1, 5)     % create vector v1
v2 = 1:5            % create vector v2
v3 =  v1 + v2       % add each element of v1 with each element of v2
v4 =  v1 - v2       % subtract each element of v2 from the respective
                            element of v1
v5 =  v2 - 2        % subtract the value 2 from each element of v2
v5 =  v2 * 3        % multiply each value of v2 by the value 3
v5 =  v2 / 2        % divide each value of v2 with the value 2
```

Edit a vector

Replace vector elements

```
v2(3) = 9          % replace the value at index 3 with the value 9
v1(2:4) = 81:83    % replace the elements at indices 2, 3 and 4 with 81,
                   82, and 83, respectively
```

Concatenate two vectors

```
n = [v1, v2]       % n contains v1 and v2 next to each other, in a single row
n = [v1; v2]       % but now v1 is in the first row, and v2 in the second row
% this last operation creates a matrix of 2 rows and 5 columns
```

To concatenated two vectors, they should have the same relevant *size*.

Try:

```
n = [v1 v2']
n = [v1' v2]
n = [v1; v2']
```

For your homework

Things to remember

Add comments throughout each m-file. This will help you avoid mistakes and will help you understand your code when looking at it in the future

Use the function *disp* to display the outcome of each exercise as well as some text accompanying this outcome

Good luck!