MATLAB 101

Contents

Getting Started

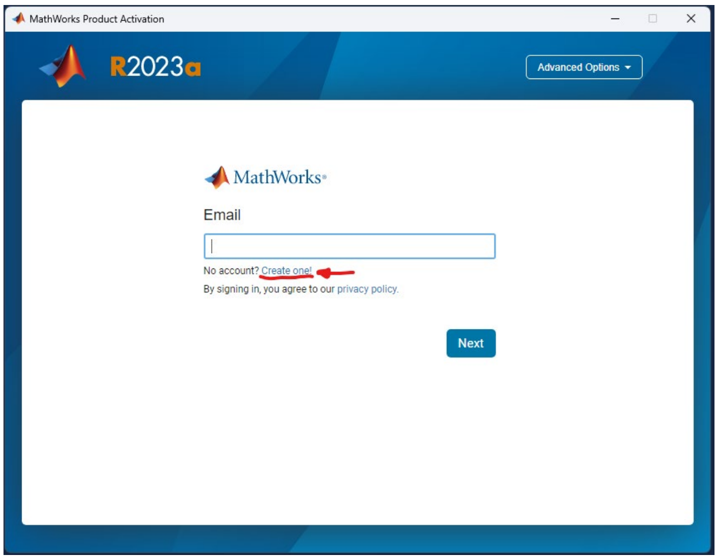
In this section

Step 1:

Boot up MATLAB by clicking the icon

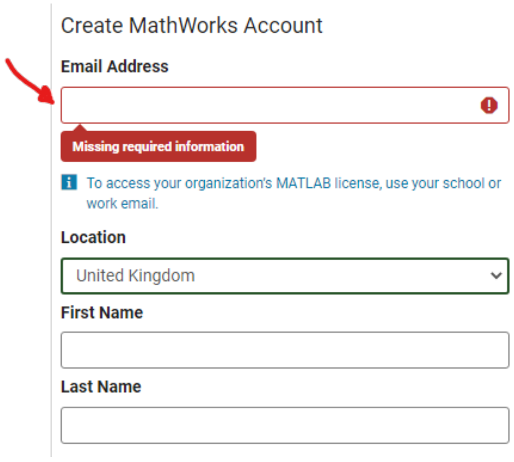
Step 2:

Click on create a new account unless you already have used MATLAB


MathWorks Product Activation
R2023a
Advanced Options
MathWorks
Email
No account? Create one!
By signing in, you agree to our privacy policy.
Next

Step 3:

Sign up with your college email


Create MathWorks Account

Email Address

Missing required information

To access your organization's MATLAB license, use your school or work email.

Location

United Kingdom

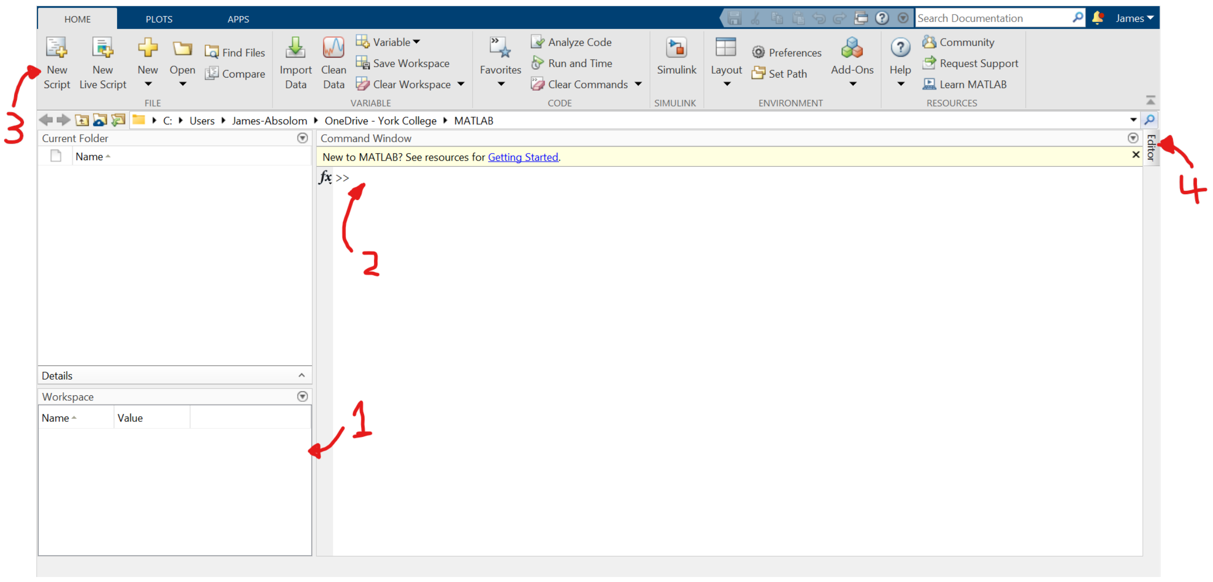
First Name

Last Name

Step 4:

You should already have licsenses but if you don't please ask me and we will look at what we can do

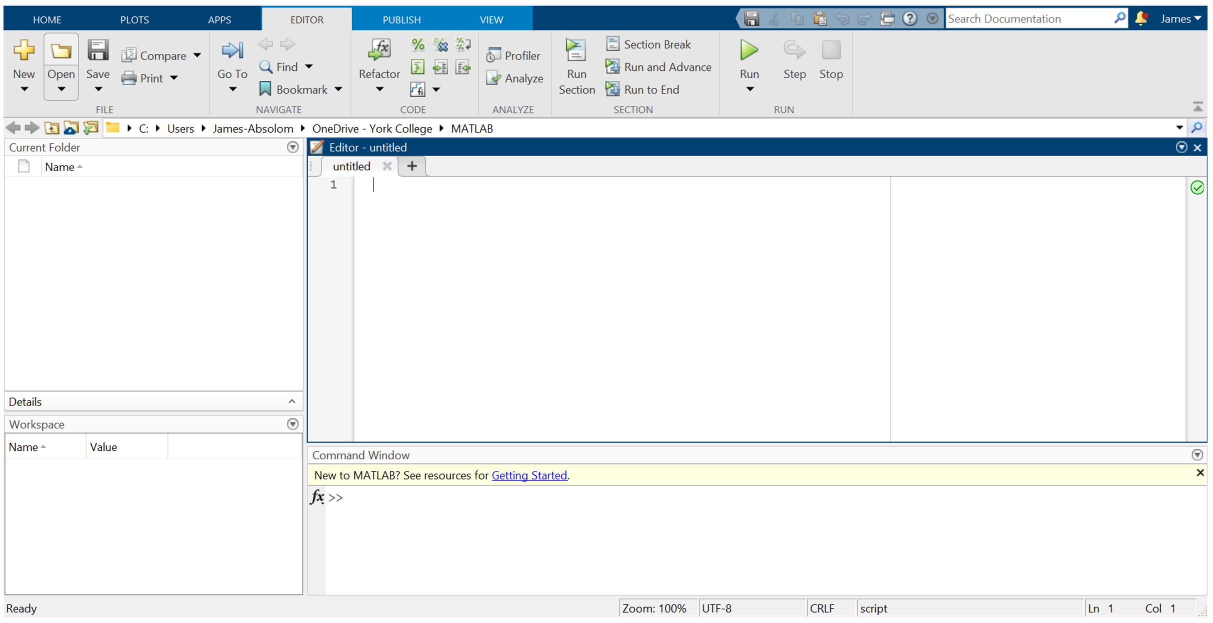
Understanding the MATLAB UI



When you start up MATLAB you should be met with this screen.

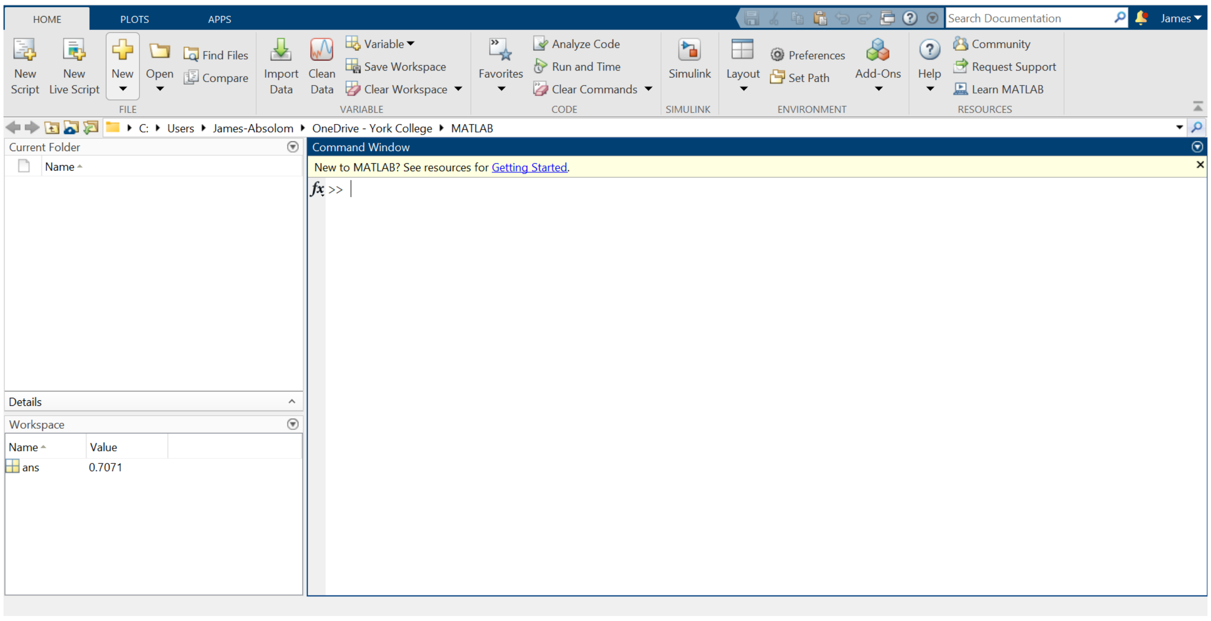
- Area 1 is the workspace which shows all the variables you are using, this is important when you are programming
- Area 2 is the command area, in this area you can type basic commands which MATLAB runs, these can be anything from basic maths to installing packages
- Button 3 is the new script button; this allows you to create a new script, these scripts allow you to run long lines of commands
- Button 4 leads to the programming area (you may not have this button but if you click new script then it will come up alongside some other windows)

When you click the new script button the editor should come up like this:



Using MATLAB as a calculator

If you click the X arrow at the top of the editor panel then you should get back to a screen like this:



New to MATLAB? See resources for Getting Started.

```
fx >> |
```

Name	Value
ans	0.7071

You can type in the command area to run basic commands

Give it a go by typing out these commands then also write out the answers MATLAB gives you:

- 5 + 3 =
- 10 / 2 =
- sqrt(25) =
- sin(pi/4) =

Can you explain why sin(pi/4) gives the answer it does?

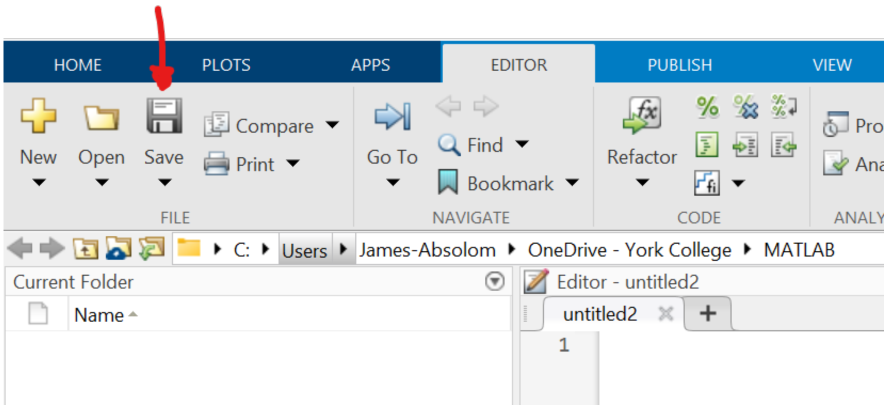
Making your first script

Step 1:

Click on the new script button again and get the editor window open.

Step 2:

Save your script using the save button in the top left corner of the screen



Name the script "myFirstScript.m"

Step 3:

Let's make our first script, put this into the editor:

```
clc;
clear;
close all;

disp('Hello, MATLAB!');
```

Lets discuss what all these commands do:

- **clc;** – removes everything from the command window making it easier to see your program run
- **clear;** – removes any variables from the workspace window
- **close all;** – closes any open figures (graphs and tables)
- **disp('Hello, MATLAB!');** - This displays a message to the command window, you can put text or variables into this area.

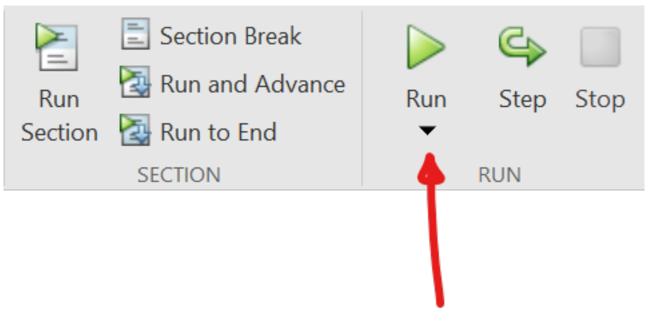
Looking at the code we can concur multiple things:

- Every line ends with a ; which is really important as otherwise it wouldn't work
- The **clc, clear** and **close all** combo is a really good practice to include at the start of all of your scripts

Step 4:

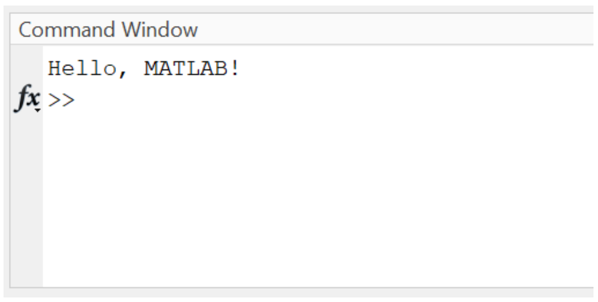
We can then save this script using the save button again

We can also run this script by pressing **F5** or by pressing the **Run Button** located at the top of the window



Step 5:

We should get an output in our command window saying Hello, MATLAB!



```
Command Window
Hello, MATLAB!
fx >>
```

Step 6:

We can change the script to use variables allowing us to quickly change values, change your code to be like this:

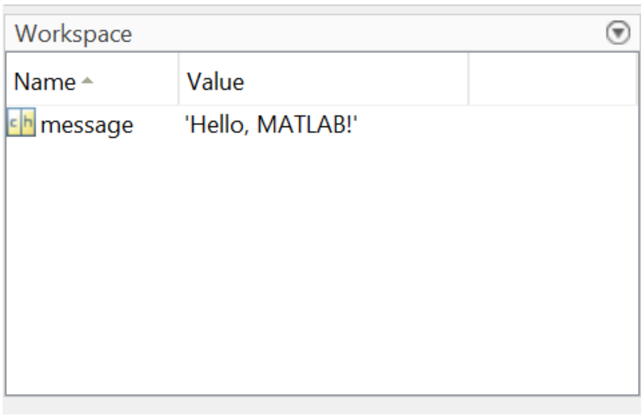
```
clc;
clear;
close all;

message = 'Hello, MATLAB!';

disp(message);
```

As we can see we are assigning the text 'Hello, MATLAB!' to the variable **message**. You can name your variable whatever you want if you change it both times its mentioned.

Variables are important as they allow us to quickly change values without having to look throughout our code.

Step 7:

Run the code again using the run button. You should see a variable appear in the workspace section with the name **“message”** and the value **“Hello, MATLAB!”**, like so:



```
message    'Hello, MATLAB!'
```

Step 8:

Can you change the text to say “Wow, Maths is cool!” and then write the code in the box below:

Writing a maths script

Let’s write out a simple maths script to add two numbers.

Step 1:

Make a new script and name it “mathsscript.m”

Step 2:

Copy this code into your editor window:

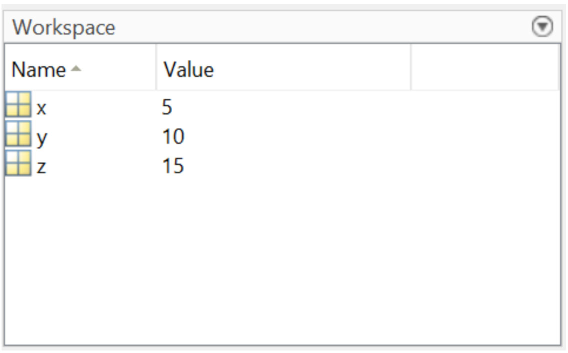
```
clc;
clear;
close all;

x = 5;
y = 10;
z = x + y;
disp(['The value of z is ', num2str(z)]);
```

Lets look through what all the different parts of this code do:

- **x = 5;** - sets the variable x to be equal to 5
- **y = 10;** - sets the variable y to be equal to 10
- **z = x + y;** - sets the variable z to be equal to x + y (5 + 10)
- **disp([‘The value of z is ‘, num2str(z)]);** - This line outputs the result of the addition in a nice format, the square brackets allow us to output multiple things by separating them with a comma. The **num2str(z)** bit converts the number variable (15) into the text variable (“15”)

Step 3:

Save and run the code, you should get the output “The value of z is 15”. You should also have several variables inside your workspace section which look like this:



Workspace

Name	Value
x	5
y	10
z	15

We can see our 3 variables we assigned, “x, y and z” written out with their values.

Step 4:

Can you make a script which adds 3 variables together and then tells you the result? Write your complete script in the box below:

Step 5:

Can you make a script that multiplies 3 variables together and then prints out the maths equation for the sum. An example output would be “4x5x6 = 120”. Can you write out the complete script you wrote:

Vectors and Matrices

Variables can store many different types of data including but not limited to numbers, text and Boolean values. However, they can also be used to store more complex data formats. Some examples of these formats are lists, vectors and matrices.

Step 1:

Can you type out this matrix code into your MATLAB

```
clc;
clear;
close all;

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

Can you write a line, so your program prints out this matrix in the command area? And write the results below in the first box:

In the second box can you write the dimensions of the matrix (how many rows and columns it has) in the format **Rows**x**Columns**

Step 2:

Can you change the code, so you have a matrix with 5 rows and 6 columns. Can you write out the line where you assign the variable below:

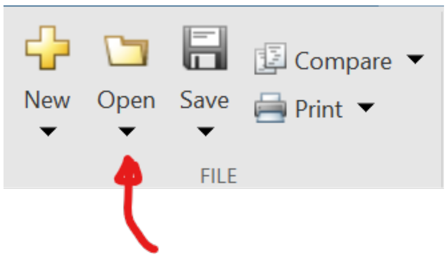
a =

Step 3:

Can you take your matrix and add a different matrix then print the answer (both matrices will require the same dimensions (5x6)). Can you briefly write below what happens when you add the two matrices:

Step 4:

Can you use the **save as** function to save the file as “matricesexample.m” and then open back up “mathsscript.m” using the open button on the top left:



Step 5:

(Make sure you are editing “mathsscript.m”)

Can you write out the following code:

```
clc;
clear;
close all;

a = linspace(0, 10, 5)

disp(a)
```

Can you explain in your own words what the linspace() function does (you may have to put in different numbers and play around with it a bit to understand it):

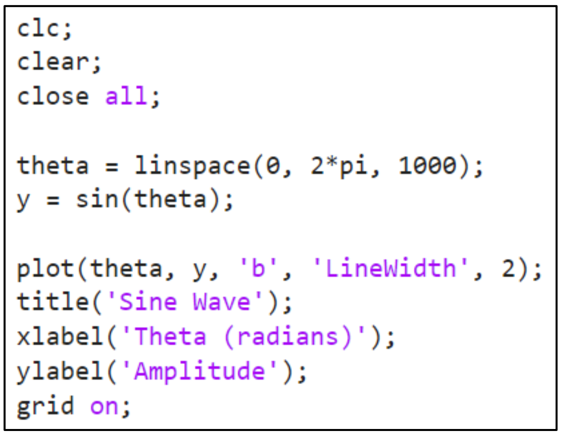
Plotting in MATLAB

Step 1:

Can you make a new file called save it as “mappingexample.m”

Step 2:

Can you copy this code down to plot a simple sine wave:



```
clc;
clear;
close all;

theta = linspace(0, 2*pi, 1000);
y = sin(theta);

plot(theta, y, 'b', 'LineWidth', 2);
title('Sine Wave');
xlabel('Theta (radians)');
ylabel('Amplitude');
grid on;
```

Let’s talk about what’s going on here:

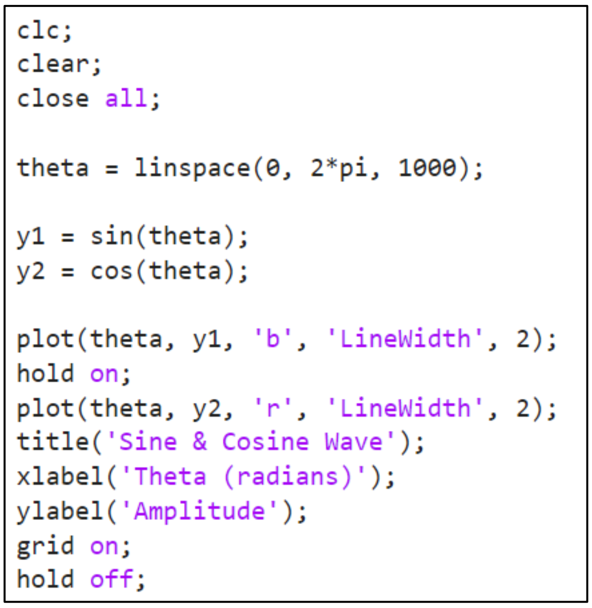
- **theta = linspace(0, 2*pi, 1000);** - This line creates the range of our sine wave for this example we’re creating 1000 points between 0 and 2pi
- **y = sin(theta);** - This line computes the sine function for all the points in the range
- **plot(theta, y, ‘b’, ‘LineWidth’ , 2);** - This plots out the a graph with theta on the x axis and y on the y axis.
    - **‘b’** sets the colour of the wave, there are many different options for colour in MATLAB;
        - `'r'` = Red
        - `'g'` = Green
        - `'b'` = Blue
        - `'k'` = Black
        - `'m'` = Magenta
        - `'c'` = Cyan
        - `'y'` = Yellow
    - **‘Linewidth’, ‘2’** work together to set the width of the line as 2 (by default the thickness is 1 so 2 is double the thickness of normal)

Step 3:

Can you change the value “1000” in the code to something a lot lower like “10”. In the box below can you write what happened to the graph and why it isn’t very helpful:

Step 3:

Let’s plot a cosine wave on the same graph as our sine wave, to do this let’s write this code into our editor area:



```
clc;
clear;
close all;

theta = linspace(0, 2*pi, 1000);

y1 = sin(theta);
y2 = cos(theta);

plot(theta, y1, 'b', 'LineWidth', 2);
hold on;
plot(theta, y2, 'r', 'LineWidth', 2);
title('Sine & Cosine Wave');
xlabel('Theta (radians)');
ylabel('Amplitude');
grid on;
hold off;
```

We already know most of what’s going on here except the line **hold on;** and the line **hold off;.** These lines are put there to stop MATLAB from showing the graph too early before it’s ready. This means that all the different plots can be shown instead of just the first one.

Step 4:

We can plot out the resultant of these two waves by adding y1 and y2 together, assigning this to a new variable and then adding a new line to plot it at after the y2 plot line. Can you do this and write out your resulting code on the next page:

Step 5:

We can write out a legend for each of our waves by putting in this line in our section where we are plotting out the lines:

```
legend('Sine Wave', 'Cosine Wave', 'Resultant');
```

This allows us to easily recognise all the different waves we are viewing.

Step 6:

Make a new script to plot these graphs and their resultant for the range **0<x<100** with the resolution of **1000**.

y1 = 2x

y2 = 4x-10

y3 = -x+20

y4 = the resultant of y1, y2 and y3