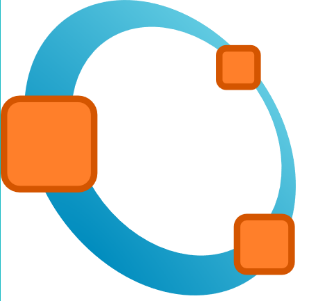
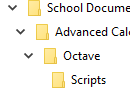
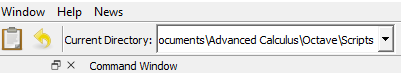
**Octave Tutorial - Ch 12**

*Scripts, Functions, and Vector-Valued Functions*

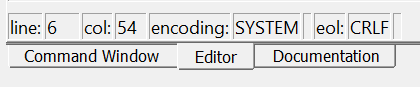
**Scripts (Intro):**

Sometimes a certain process might involve a few inputs but many steps. Instead of re-typing all   
the commands every time, we can write a script that will run all the commands together. Simply put,  
a script is just a text file full of commands. It is stored on your hard drive, and when you want Octave to do those commands, you tell it to “run” the script.

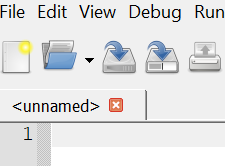
Before writing and saving a script or program, we need to make sure that Octave is saving and looking for files in the right place. Create a folder (or choose an existing folder) where all your small Octave files will be saved. I have a folder called “Octave” in my Advanced Calculus folder. Within that, have a subfolder called “Scripts.”   
  
  
You ***could*** go to the top of your console and set your “Current Directory” to be this folder…  
   
…but that would not be enough to make it stick. Octave would just switch it back to the default every time you open the program. (Mine goes back to C:\Users\Steven Malan every time.)   
  
To set it permanently:

* On PC: Edit 🡪 Preferences 🡪 General   
  On Mac: octave-gui 🡪 Preferences 🡪 General
* Under “Octave Startup,” unselect “Restore working directory of previous session”
* Click on “Browse” to set the default directory.
* Select “Apply” and then “OK”

If you do not see the new setting, close Octave and reopen it.

**Writing a Script:**

To write a script, you will need to work in the editor. The tab can be found at the bottom of your window:

Once you have set the correct directory, and entered the Editor, you should see that the current script is unnamed. Hit save (Ctrl+S or Command+S or “File 🡪 Save” or first icon with a blue arrow) and title it Hello\_World.m (The .m extension will happen automatically.)

In the text body, enter some commands that you would like Octave to process all at once. Try this script:

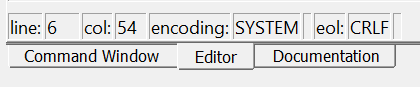
x = 3

y = 7

z = x \* y

% Now I will print out a message

fprintf('Hello World!!! \nThis is my first script!\n')

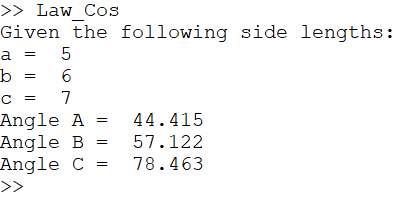


Save and then go back to the Command Window.

The Command Window is where you have typed all your   
individual commands up to this point.

In the Command Window, type Hello\_World and see what happens.

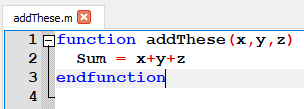
Now go back to the editor and create a new script called Law\_Cos.

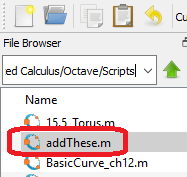
1. This script will require that you enter an inverse cosine function. Figure out how to make that work. I (Malan) went to the Command Window and guessed commands until I got one that would give me 3.1416 when I put in -1.   
   Alternatively, you could click on Documentation, go to the Index tab and start typing in key words.   
   Or Google it.
2. In the Editor, first create a place to enter triangle side lengths:  
    a = 5  
    b = 6  
    c = 7
3. With pencil and paper, start with the law of cosines: and solve for .
4. In the Editor, add the next line:  
    C = Here you will write a function that uses inverse cosine to find the appropriate angle.  
    Hint: you must always show multiplication with \*
5. Hit save. Go back to the editor and type Law\_Cos
6. If you did it right, you should get an angle in radians. But suppose we wanted degrees. Change your function, hit save, and try it again. (If you use a=5, b=6, c=7, then the answer should be .)
7. *****Challenge*: Try to improve the script so that it gives a more robust and readable output:
   1. Use fprintf() to insert text
   2. Given all side angles

Your output might look something like this:

**Function:**

Similar to a script, a function is a set of instructions that one saves in a text document. But whereas a script is a set of commands that runs all by itself, a ***function*** requires input(s).

1. Go to the editor and save a new file as: addThese.m
2. In the first line write:  
   **function addThese(x,y,z)** hit Enter  
   Octave should recognize that you are creating a function and should indent accordingly.  
   To avoid error messages make sure the file and the function have the same name.
3. Inside the function, write Sum = x+y+z (no semicolon)
4. Save
5. Go to the Command Window and type addThese(10, 20, 17) What happens?

You now have a function that will add any three numbers. Octave has some built-in functions (sin(), log(), dot()) which take input(s) and give an answer. But now you have a new function that you can always use, as long as you keep it saved in your working directory. Even when you close octave and re-open it, the function will still be there for you to use. You’re welcome ☺.

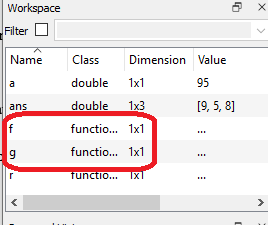
Writing functions is a great skill, but it will ***not*** be a huge focus on for us. Much more often, we will define simple functions that we can use temporarily, until we close the program. These are called…

**Anonymous Functions:**

Real official functions always involve saving a new text file onto your hard drive. Such functions will always be available, as long as you still have the text file saved in the right place.

But sometimes we just need to define a less-formal function, called an anonymous function. Try these on the command line:

* f = @(x) x.^2 🡨 this means To avoid errors, use a period to   
   emphasize component-wise operations  
  Type f(35)
* g = @(w, x, y, z) sin(w\*z) + y – x 🡨 What would this function look like in typical notation?  
  Type g(pi/3, 5, 8, 2)

You should notice that these functions are now saved in your workspace.   
(Your workspace is probably on the left side of your screen. If you don’t see it click Window 🡪 Show Workspace.)   
Just like any constant you define. In the screenshot here, I have “a” defined as 95.

Similar to constants, these workspace elements will disappear when you

* Restart Octave
* Type: clear

**Compatibility with MATLAB[[1]](#footnote-1):**

Octave tries to be compatible with MATLAB, but in some cases, it offers more freedom. This can be a problem if you are ever going to share a “.m” file with a someone else (such as in a college application portfolio). So when Octave gives you freedom, it might be better to use the more restrictive MATLAB style.

|  |  |  |
| --- | --- | --- |
| **Task** | **How it Works on Octave** | **How it works on MATLAB** |
| Taking a transpose | [2, 4, 5, 6]’ or [2, 4, 5, 6] ’  You can have a space before the transpose marker. Or not. | [2, 4, 5, 6]’  You cannot have a space |
| Quotes | " " and ' ' are treated exactly the same. | In older versions, only ' ' was allowed. In newer versions, " " is allowed, but sometimes it means something else. When in doubt, just use ' '. |
| Printing Text | printf(‘this is some text.\n’) or  fprintf(‘this is some text.\n’) | fprintf(‘this is some text.\n’) |
| Checking if two values are not equal | 5 != 6 or 5 ~= 6  (Answer is “1” which indicates “true”) | 5 != 6 |
| Commenting | % or # | Just % |
|  |  |  |
|  |  |  |

**Chapter 12 Skills**

12.1 – Graphing Vector-Valued Functions

* Drawing a VVF   
  First define three anonymous functions:  
   x = @(t) cos(t); 🡨Note: These *could* all be typed on the same line  
   y = @(t) sin(t); as long as you separate them with semicolons.  
   z = @(t) sin(4\*t)+0.3\*cos(22\*t);   
  Call the ezplot3 command  
   basic: **ezplot3(x,y,z)**  
    
   to specify: **ezplot3(x,y,z, [0, 3\*pi], 80, “animate”)**   
    
   domain of t parameter # of points include this to see animation  
   (default is ) (default is 500)
* In order to change the style of the line, first define the anonymous functions as before. Then execute the ezplot3 command, but name it. This name is called a “handle.”  
    
   **CoolGraph = ezplot3(x,y,z, [0, 3\*pi], 80);** 🡨 skip animation this time.  
  Then use the “set” command to sketch the graph with a specified color and style.  
    
   **set(CoolGraph, ‘color’, ‘blue’, ‘LineStyle’, ‘:’, ‘LineWidth’, 3);**  
    
  This will go ahead and print your ezplot3 function, but according to the specifications.

Note: When I type quotes in Microsoft Word, they come out in a nice curly form:

“ ” or ‘ ’

But Octave’s/MATLAB’s quotes are actually a different character:

" " or ' '

So whenever you see quotes in these PDF instructions, you should not copy/paste that code into Octave. Just type it out by hand.

I could just hit Ctrl+z every time Word gives me curly quotes. But I choose not to. 😁

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Color** | **What to Type** |  | **LineStyle** | **What to Type** |
| White | ‘w’ or ‘white’ |  | Solid (default) | ‘-’ (or nothing) |
| Magenta | ‘m’ or ‘magenta’ |  | Dotted | ‘:’ |
| Cyan | ‘c’ or ‘cyan’ |  | Dash and Dot | ‘-.’ |
| Red | ‘r’ or ‘red’ |  | Dashed | ‘--’ |
| Green | ‘g’ or ‘green’ |  |  |  |
| Blue | ‘b’ or ‘blue’ |  |  |  |
| Yellow | ‘y’ or ‘yellow’ |  |  |  |
| Black | ‘k’ or ‘black’ |  |  |  |
| (Want millions of different color? Scroll down to pp. 5-7.) | | | | |

* Want more colors? Scroll to the end of this PDF.
* To be able to click-drag to rotate image, you have a few options:
  + Click on the rotate button in the figure window
  + Type **rotate3d on** in the Command Window or script.
  + If you simply type **rotate3d**, this will toggle the rotate button on/off.

12.2 – Differentiation and Integration of Vector-Valued Functions

Many of the skills in our chapter would require a computer algebra system, which Octave is not. (The “Symbolic” package is very powerful and turns Octave into a CAS, but it is complicated to install.) For numerical differentiation, we can use the “optim” package. For numerical integration, Octave does not need any special package. (If the following does not work for Mac users, please complete it at <https://octave-online.net/>.)

* Intro to numerical differentiation  
  **pkg load optim** 🡨 this loads the “Optim” package. It will stay loaded until you close Octave.  
  **f = @(x) x.^2;  
  deriv(f, 3)** 🡨 takes a derivative and evaluates it at x=3.
* If , find   
  **pkg load optim;  
  x=@(t)t.^2; y=@(t) log(t); z=@(t) 6-t;  
  a = [deriv(x,3), deriv(y,3), deriv(z,3)]**  
  “Wow, that’s a lot to type. Maybe I should just write a script…”
* Warmup with numerical integration. (does not need the optim package)  
  f = @(x) sin(x).^2  
  integral(f, 3, 5)
* Now suppose you have defined 3 anonymous functions as part of a VVF  
  x =@(t) \_\_\_\_\_\_\_ ; y=@(t) \_\_\_\_\_\_\_\_\_\_ ; z=@(t) \_\_\_\_\_\_\_\_\_\_\_\_\_ ;  
  lower\_bd =   
  upper\_bd =   
  def\_int = [integral(x, lower\_bd, upper\_bd), integral(y, lower\_bd, upper\_bd), integral(z…… )]

12.3 – Velocity and Acceleration

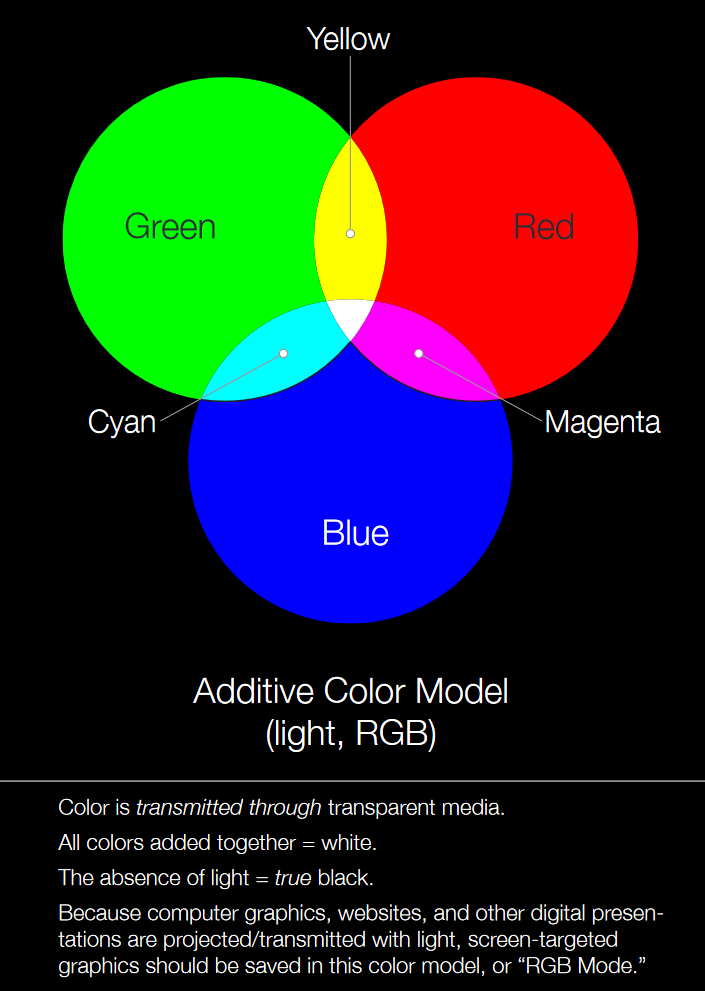
* Find speed by taking the magnitude of .  
  **pkg load optim;  
  x=@(t)t.^2; y=@(t) log(t); z=@(t) 6-t;  
  a = [deriv(x,3), deriv(y,3), deriv(z,3)];  
  norm(a)**

12.4 and 12.5 – No Octave skills tested.

**More Colors:**

*This section is not necessary for your quiz, but will be necessary for your roller coaster project.*

When selecting a color for a line, there are 8 basic options. But with a little curiosity and time, you can specify any[[2]](#footnote-2) color that you want.

Computers display color through light. The primary colors *of light* are Red, Green, and Blue. Mixing them can create light of any color.

Try the following commands[[3]](#footnote-3):

x = 0 : 0.5 : 2\*pi; y = sin(x);

plot(x, y, ‘color’, [1, 0, 0], ‘LineWidth’, 20)

Then repeat the command trying the following. What colors do they   
produce? How do they relate to the picture shown here and below?

[0, 1, 0]

[0, 0, 1]

[1, 1, 0]

[1, 0, 1]

[0, 1, 1]

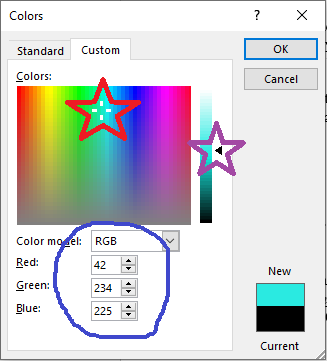
[0, 0, 0]

[1, 1, 1] 🡨 This one is weird. Can you figure out what happened?

In total, there are 8 combinations of zeros and ones. But for each entry [ \_\_, \_\_, **\_\_**], you can actually specify any value ***between zero and one***. Try this:

plot(x, y, ‘color’, [0.6, 0, 0.5], ‘LineWidth’, 20)

Want to specify any color? You just need to find its **RGB** values and then divide them each by 255. See below for reason.[[4]](#footnote-4)



To find **RGB** values, first find an interactive color palette. There are some online options[[5]](#footnote-5), but the easiest one might be in Microsoft Word.

* Open a document and go to:

(PC) Font 🡪 Font Color 🡪 More Colors 🡪 Custom

(Mac) Font Color 🡪 More Colors 🡪2nd Tab 🡪 Dropdown menu 🡪 2nd option.

* Play with the two sliders to get the exact color you want.
* In the example here, the RGB values are 42, 234 and 225.

Once you have the RGB values, they must be divided by 255 before using them in Octave. To use the color shown here in Octave, you would enter:

plot(x, y, ‘color’, [**42**/255, **234**/255, **225**/255], ‘LineWidth’, 20)

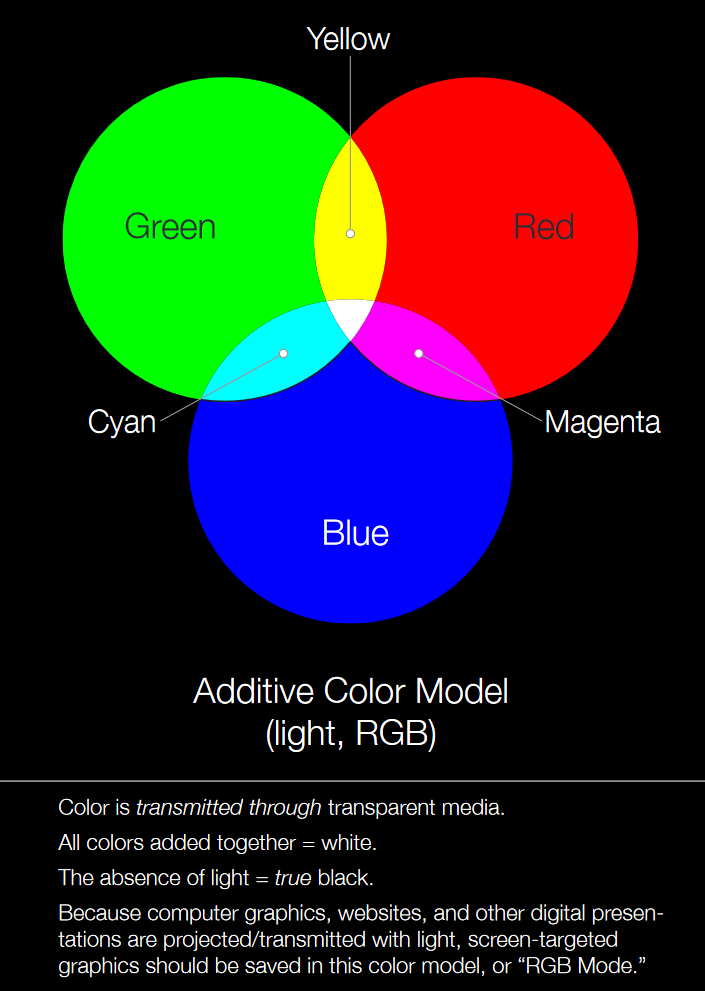


Image: <http://learn.leighcotnoir.com/artspeak/elements-color/primary-colors/>

1. <https://en.wikibooks.org/wiki/MATLAB_Programming/Differences_between_Octave_and_MATLAB> [↑](#footnote-ref-1)
2. To be precise, you have color options. Or if you don’t count plain white. [↑](#footnote-ref-2)
3. Here, I use the old fashioned “plot” command for simplicity’s sake. It simply does a dot-to-dot from one (x,y) coordinate to the next. The “ezplot” and “ezplot3” commands take actual functions and evaluate them at different points (and then do a dot-to-dot with those points). [↑](#footnote-ref-3)
4. Typically, the three colors are recorded as integers from 0 – 255. This gives different values. However, Octave records them on a scale from 0 to 1, so you have to scale them down. [↑](#footnote-ref-4)
5. Such as <https://www.sessions.edu/color-calculator/> or <https://www.google.com/search?client=firefox-b-d&q=rgb+color+picker> [↑](#footnote-ref-5)