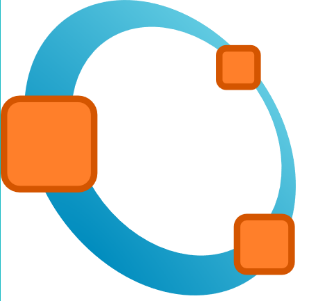
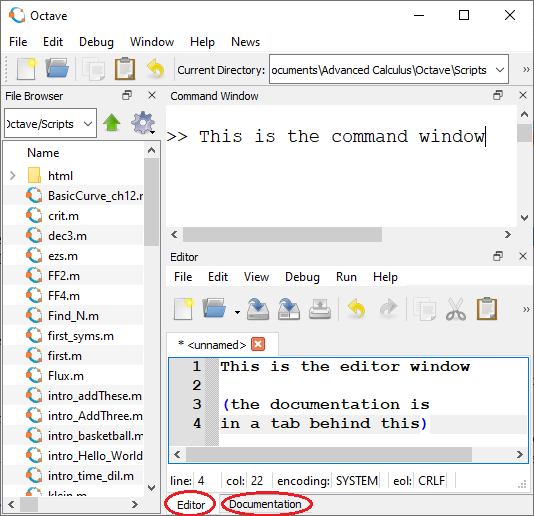
**Octave Tutorial - Ch 13**

*Surfaces, Tangent Lines/Planes, and the Symbolic Package*

**Visual Rearrangement:**

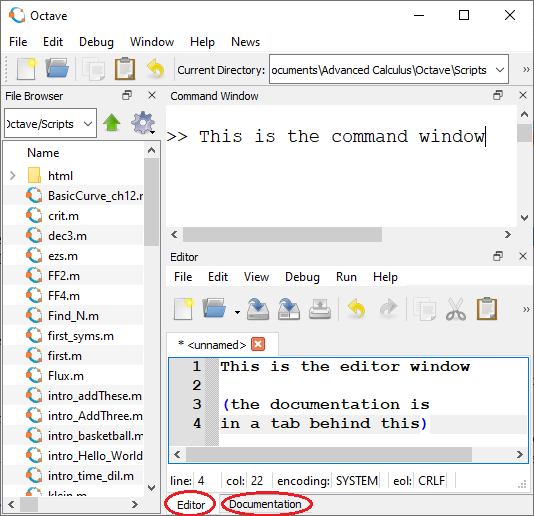
Now that you are comfortable with the different sections of the Octave window, you should adjust it to fit your needs. I personally only feel the need to see the File Browser, Command Window, Editor, and Documentation.

Here is a (squashed, ugly) view of approximately how I like my window.

There are two main ways to change the view:

1. (at top) Window 🡪 select/unselect the “Show” options.
2. Click and drag the section headings to move them around.

You can also hide the toolbar…

  
  
… by right-clicking on it and unselecting “Toolbar.”

**Plot Commands:**

When you plot a graphic, a new window pops open automatically. Now it is time to learn how to open/close/ manipulate plotting windows by executing commands. We will start in the command window, executing commands one-at-a-time. But all of these skills could be used in a single script. For graphing we will use some very basic pre-built graphing functions (“sombrero” and “peaks”).

First adjust your windows, so that you can see these instructions, the command line, and any   
  
plots all simultaneously. It might help to pop the command window out for the time being.

Then follow this progression of commands, one at a time, to see what each one does. You will often have to click back on the command window. You are expected to master all these commands and be able to write a script to do similar commands in any order.

>> figure

>> figure

>> figure 1

>> sombrero

>> figure 2

>> peaks

>> title("Mr. Peak", 'fontsize', 30, ‘color’, ‘blue’)

>> rotate3d (then try rotating them with your mouse)

>> rotate3d

>> figure 3

>> ezplot(@(x) sin(x))

>> hold

>> ezplot(@(x) cos(x))

>> hold 🡨 what does “hold” do without specifying on/off?

>> ezplot(@(x) sin(2\*x))

>> figure 2

>> clf

>> figure 1

>> close

>> figure 2

>> close

>> figure 3

>> close

At this point, all of your plotting windows should be closed. So now we’ll try something new:

>> figure

>> subplot(2,4,1)

>> peaks

>> subplot(2,4,2)

>> peaks(20)

>> peaks(10)

>> title('much less refined')

>> subplot(2,4,3)

>> sombrero

>> rotate3d (and then try rotating them)

>> subplot(2,4,1)

>> peaks

>> xlabel('x!!')

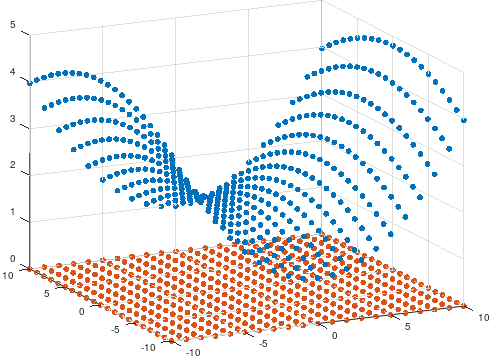
>> subplot(2,4,8)

>> ezplot(@(x) sin(x))

>> subplot(2,4,4) (have you figured out what the “2,4” represents?)

>> peaks(4)

>> subplot(2,4,1)



**Meshgrid (needed for 13.6 vector fields):**

When graphing , a human might want to think of a few points, calculate the value for each and then, plot all the values, until they form a nice surface. See example here.[[1]](#footnote-1)

But ***how does a machine*** pick the points?

😐 Old Way: “Surf” 🡨 user must create a “meshgrid” to specify exactly which values.

☺ New Way: “ezsurf” 🡨 computer assumes you want and , (unless otherwise specified) with 60 divisions in each dimension, (unless otherwise specified).

Similar commands include:

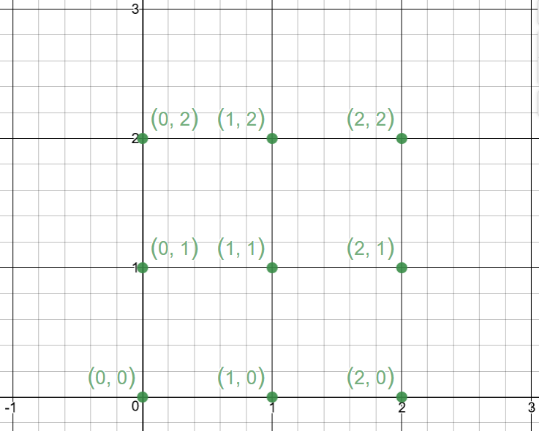
😐 “Plot” 🡨 user must lay out a list of values, and a list of values. Octave will do a dot-to-dot.

☺ “Ezplot”🡨 User creates a function (f = @(t) \_\_\_\_\_) and tells Octave to just “figure out” some good values.

Also see “Plot3” (😐) vs “Ezplot3” (☺ - good for making roller coasters)

Bad news: Currently (December 2020), Octave does not have an “ezvectorfield” command. So in section 13.6 and in all of chapter 15, we must work the old fashioned way to plot vector fields:

1. Define a grid of values.
2. Use the “quiver” command to draw an arrow at each point.

So how do we create a bunch of points (such as here)?

Super Basic (slow):

Create an list X = 0:2 (if you want lower case later,

Create a list Y = 0:2 then don’t user lower case here.)

Tell Octave to put them together [x,y] = meshgrid(X,Y)

These lower-case and variables will be the same variables that you use in your functions.

In your first few tries, don’t use semicolons. Can you see how Octave is thinking of the grid? (You might think that the values are listed upside down. Don’t worry about this.) For bigger examples, ***remember to use a semicolon!***

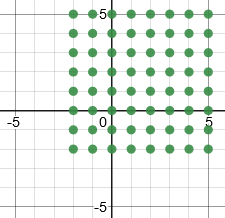
Check your skills: Make a grid where:

with increments of 0.5  
 with increments of 2

Medium Speed (all in one step):

If you are comfortable with a longer line of code, you can define your and points ***within*** the meshgrid command.

Try this: [x,y] = meshgrid(0:0.5:2, 0:2)  
In this case we do not need to make temporary capital X or Y lists.

Super Fast (but less specific):

If and are to be defined the same way, then you can just put one argument into meshgrid. The grid shown here could be programmed quickly by:  
 [x,y]=meshgrid(-2:5)

To try it out step-by step, try this:

[x,y]=meshgrid(-3, 0.5, 3)

f = x.^2 + y.^2 🡨 Don’t use an anonymous function. [[2]](#footnote-2)

surf(f) (or “mesh(f)” or “surfc(f)” or “meshz(f)”)

**Symbolic Package:**

At its heart, Octave works numerically, rather than symbolically. So, Octave can find , but not by using calculus. Rather, it may use lots of rectangles to approximate the area.

However, Octave has an extra package called “Symbolic” which can be installed to do exactly these calculations.

☹ Bad news: It’s really hard to install.[[3]](#footnote-3)

☺ Good news: <https://octave-online.net/> has it already installed.

So while you are just learning how to use it, just practice on the web version.   
Try these steps:

pkg load symbolic 🡨 You only have to load the package once per session.

syms x 🡨 Now Octave will think of and treat symbolically.

syms y

f=x^7+cos(y)+x^y

diff(f,x)

diff(f,x,x)

diff(f,x,3)

diff(f,x,y,y,x)

g=x^4 🡨 Can you explain why we do not need the “ . ” ?

int(g)

int(g,2,3)

**Chapter 13 Skills**

13.1 – Sketching a surface

* Sketch a surface and then add a colorbar.   
   f =@(x,y) x.^2 – y.^3  
   ezsurf(f)  
   colorbar  
   colorbar(‘east’)  
   colorbar(‘south’)  
   colorbar(‘southoutside’)  
   colorbar(‘westoutside’, ‘fontsize’, 20)
* Should gridlines be oriented by or by ?  
   g = @(x,y) x.^2 + y.^2   
   ezsurf(g)  
   ezsurf(g, ‘circ’)
* Contour lines   
   f =@(x,y) x.^2 – y.^3  
   ezsurfc(f)  
   ezcontour(f)  
   ezcontourf(f)  
   ezcontourf(f, [-1 3 0 4])  
   ezcontourf(f, [-1 3 0 4])  
   colorbar;

13.3 – Derivatives

* Write a script to sketch a surface and then add tangent lines
* Use Symbolic package to find derivatives

13.4 – Differentials

* none

13.5 – Chain Rule

* none

13.6 – Gradients

* Draw a vector field, using quiver. Here is a super basic example. Can you figure out what each step does?  
   [x,y]=meshgrid(-3:3)  
   fx=2\*x-y; fy=x+2\*y 🡨 (These are not anonymous functions[[4]](#footnote-4))  
   quiver(x,y,fx,fy)  
    
  Be sure to read the “quiver” [[5]](#footnote-5) documentation. Then try the following commands;  
   quiver(x,y,fx,fy,0.7)  
   quiver(x,y,fx,fy,1.2)  
   quiver(x,y,fx,fy,0.1)  
   quiver(x,y,fx,fy,0)
* Use subplot(1,2,\_) to show the surface and gradient field side-by-side   
   (or contour plot and gradient field side-by-side)

13.7 – Tangent Planes and normal lines.

* Write a script to sketch a surface with a tangent plane and/or normal line

13.8 – Relative Extrema

* Write a script where the user inputs , , , , , and , all as anonymous functions. Then the user inputs a critical point and Octave does the second partials test on that point. After calculating D, the final lines should follow the structure:  
   if(d>0 && fxx(x,y)>0)  
   fprintf(‘This is a relative minimum.\n’)  
   elseif( )  
   fprintf(‘ ’)  
   elseif( )  
   fprintf(‘ ’)  
   elseif( )  
   fprintf(‘ ’)  
   endif
* Use the Symbolic package to write a script that does all of the above, but where the user only has to enter the original function and a single point.

13.9 – Applications of Relative Extrema

* none

13.10 – Lagrange Multipliers

* to be determined in January 2021

1. [x,y]=meshgrid(-10:10);

   z = 0.03\*x.^2+1.8\*cos(y/10);

   scatter3(x,y,z, 'filled');

   hold;

   w = 0\*x + 0\*y;

   scatter3(x,y,w, 'filled');

   rotate3d; [↑](#footnote-ref-1)
2. Does this bother you? Think back to the simplest graph you ever made:   
    x = -3:0.5:3  
    y = sin(x)  
    plot(x,y)  
   In this case, y does not need a to be a function. It does not need to be ready for user inputs. It is just a list of values. Leave off the semicolons to see this clearly. [↑](#footnote-ref-2)
3. At least for me. It involves installing a Python math package. But it has to be in exactly the right place on your machine. I (Malan) was able to install it on my home laptop, but not my school laptop. If you are using a PC, and want to try, feel free. I can help. If you are using the app for Mac, it cannot be installed. [↑](#footnote-ref-3)
4. See footnote 2 above. [↑](#footnote-ref-4)
5. Do you know that “quiver” generally means (as a noun) in English? If not, look it up ☺. [↑](#footnote-ref-5)