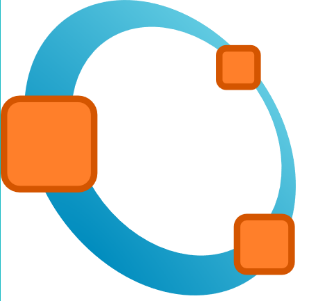
**Octave Tutorial - Ch 14**

*Markup and Numerical Integration*

**The Idea of Markup:**

So far, you have seen 2 ways of executing Octave code:

1. Command Window: Write and execute one line at a time. This is the first way that you learned to execute commands. It is most useful for small tasks that you do not need to be saved or repeated.
2. Scripts: Write many commands and execute them (or a portion of them) all at once. This is done in the Editor by typing plain-text files that are stored locally on your hard drive with a “.m” extension. It is very useful for writing long files, but it can still only be enjoyed by someone who has Octave/MATLAB installed. Also, the user will have to look in multiple places: To see how your code is laid out, they must read the script. To see the calculated answers, they must scroll through the Command Window. To see all the figures, they will have to click on all the individual windows that pop up.

But there is a third way of executing Octave code:

1. Publishing Markup: This is useful for presenting all of your code, graphs, and computational output in a clear, professional, logically flowing document. Your reader can then read the document without needing to have Octave/MATLAB installed.

**First Markup Example:**

%% My First Markup

%

% This is my \*first\* markup document!

%% In Markup, I will learn how to:

%

% \* Write plain text

% \* Insert graphs

% Now, here's some Octave code

x = 0 : 0.1 : 2\*pi;

y = sin(x);

plot(x,y)

1. In the editor, open a new file. Save it in your regular scripts folder as first\_publish.m.
2. In the file, paste the code shown here:  
   Save this.
3. Go to the Command Window and type: publish('first\_publish.m')
4. In your computer’s file system, open your scripts folder. You should find a new folder called “html.” Open the folder. You should be able to find a file called first\_publish.html. HTML is a file type that your browser can read. If you have never opened a plain HTML file before, you might need to do something like:   
    Right-click 🡪 open with 🡪 Firefox/Chrome/Safari/Edge   
   (Word and some PDF readers can open HTML, but some equations and graphics might not come out right.)

**Ways to Publish:**

One benefit of Markup is that your code will be published to a format that others can read without needing to install Octave. But what format will it be published into? There are 3 main options:

* ‘html’ – This is the easiest and is the default for publishing.
* ‘latex’ – LaTeX (pronounced “Lay-Teck”) is a math markup language, used for formatting professional papers.
* ‘pdf’ – This is pretty nice, but it requires you to have a LaTeX interpreter installed on your machine. If you really want to try it without a LaTeX interpreter, octave-online can publish to PDF for you.
* The documentation also mentions ‘doc,’ ‘ppt,’ and ‘xml’ formats, but it says that they are not implemented yet.

**Documentation:**

Check out the documentation for Publishing Markup, either within your Octave program or by visiting <https://octave.org/doc/v6.1.0/Publishing-Markup.html>. Look through some examples. You are not expected to memorize anything and will NOT be tested on it. Rather, this will just give you an idea of some tools that are available.

Along with the documentation, Mr. Malan will provide an example of what a final version might look like. That example will contain some tips.

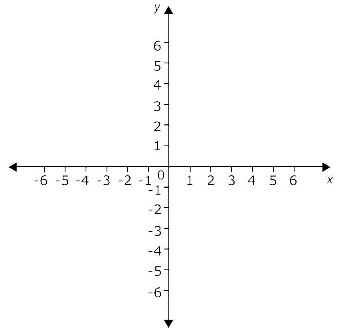
**What’s Coming:**The Octave portion of Exam 14b will not be done on the test day. Rather, you will pick a topic from chapter 13 or 14 and write a .m file concerning this topic which can be published into a beautiful HTML file. Full instructions will be given later, but for now, the basic idea is:

* It should look good enough that you could submit it as part of a university portfolio, simultaneously showing your Octave skills, your Multivariable Calculus skills, and your ability to write and explain.
* It should work through the topic, flowing freely between headings, written content, Octave code, and graphs. If “flowing freely” doesn’t make sense, pick up any math textbook, open to a random lesson, and look at how easy it is to read the words, equations, and graphs.

**View Command:**

When your graphs are published in markup, the user will not be able to click and rotate them. To get around this, you can specify a viewing angle using the “view” command. Try these two commands

peaks

 view(0, 30)

Both of these arguments are angles measured in degrees. The indicates the orientation   
from which we normally view axes when the paper is on the table and we are looking   
down on it: The positive -axis goes the right. Now try:

view(10, 30)

view(20, 30)

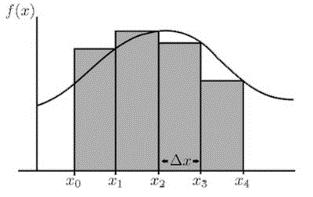
view(30, 30)

view(180, 30)

view(-360000010, 30)

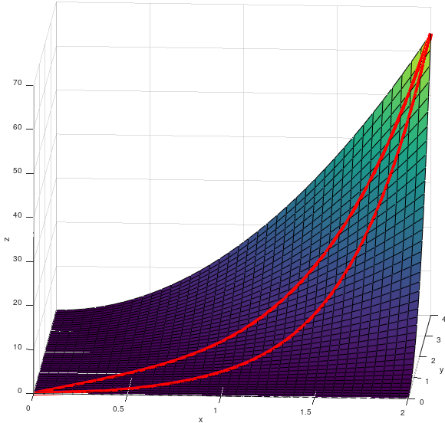
Once you have a feel for the first argument, try the second argument. You’ll figure it out.   
Or just read the documentation.

There are plenty of other (fun!) commands to control how a figure is displayed (rotate, zoom, pan, campos, camroll, camtarget…), but “view” can probably do most of what you need.

**Types of Double/Triple Integrals:**

At its heart, Octave is a numerical program, meaning that it is really good with numbers but not variables. So if it wants to evaluate a single integral, such as , it will NOT start by finding an antiderivative. Rather, it will use a “numerical” approach, such as partitioning the area into small intervals and then using the height of the function to find the areas of many rectangles. The [actual methods](https://octave.org/doc/v6.1.0/Numerical-Integration.html#Numerical-Integration) underlying the commands are quite advanced, but they are still numerical.

The main integral commands that I recommend are [integral2](https://octave.org/doc/v6.1.0/Functions-of-Multiple-Variables.html#XREFintegral2) and [integral3](https://octave.org/doc/v6.1.0/Functions-of-Multiple-Variables.html#XREFintegral3). They can do most of the double/triple scalar integrals in our class, including integrals with infinite (“**Inf**”) limits of integration. Just like in class, outer limits of integration must not involve any variable that has already been dealt with in an inner integral. To read some benefits of integral2 over a similar function, check out [this MATLAB forum page](https://www.mathworks.com/matlabcentral/answers/63555-difference-between-integral-and-quad-functions).

Here is 14.2 #10. Visual shown on right.

(To rotate the visual, copy and paste this footnote[[1]](#footnote-1) into the Command Window.)

This can be set up and evaluated over many lines…

f = @(y,x) x.^2.\*y.^2; 🡨 follow the “Left to Right” order

lower\_x = @(y) y/2; from above.

upper\_x = @(y) sqrt(y);

**integral2**(f, 0, 4, lower\_x, upper\_x) 🡨 Notice the order in which the limits are written.

Or it could all be evaluated in one long command containing the same inputs:

integral2(@(y,x)x.^2.\*y.^2, 0, 4, @(y)y/2, @(y)sqrt(y))

As long as any anonymous functions are defined with parameters in the correct order, you should be able to have inner limits of integration that are factors of any outer variable of integration.

Polar: Most polar double integrals are evaluated in a “” order of integration, such as

But this means that the integrand function should originally be defined as f=@(t, r) rather than f=@(r,t)

Now try integral3:

14.6 #12

Any anonymous function you wish to use must be   
initiated (such as @(\_\_, \_\_, \_\_)) with variables in this order. (Or “opposite of this order”)

So the triple integral above could be started like this:

f = @(y,z,x) \_\_\_\_\_\_\_\_\_; 🡨 notice the “yzx” order matches with the blue, green, and yellow above.

z\_upper = @(y) \_\_\_\_\_;

x\_upper = @(y,z) \_\_\_\_\_; 🡨 notice the “yz” order matches with the blue and green above.

integral3(f, 0, 3, ……) 🡨 Can you finish it? The answer is approximately 2.1184.

Spherical integrals are often evaluated in the order . In that case, the integrand function would need to be defined as:

f = @(th,ph,rh) (Note: your variables do not have to just be single letters. You could even define the variables as  
f = @(theta, phi, rho). Just be consistent.)

**Chapter 14 Skills**

14.2 – Basic Double Integrals

* All “integral2” instructions given above
* Use ezsurf to sketch a surface (review)
* Challenge: After sketching a surface, find vector value functions to draw “cuts” along the surface. Then use hold on and ezplot3 to add these to your figure

14.3 – Polar Double Integrals

* If an integral has a “round” region, originally defined the and , and difficult to evaluate by hand, try this:
  + Use integral2 to evaluate the integral in its given “” form
  + By hand, convert to a polar double integral
  + Use integral2 to evaluate the new double integral and see if it matches your previous answer.
* If surface is originally given in terms of and , use ezsurf to graph it. Maybe try ezsurf(f, 'circ')

14.4 – Triple Integrals

* Examine integral3. Read the documentation and try an integral that is “nicely” given with order of integration.
* Find a different integral that is not given with order of integration. Mindlessly re-write variables to get order.

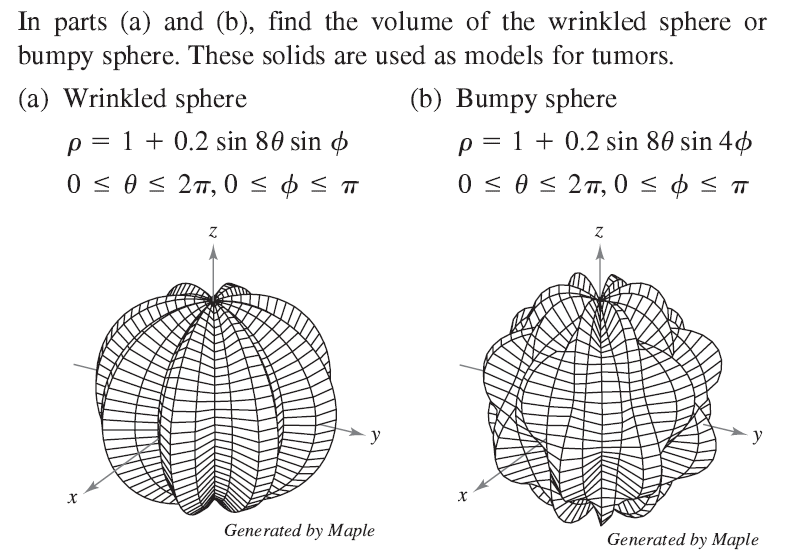
14.5 – Surface Area

* No new applications. Just set up a double integral, and then use integral2, as before.

14.6 – Triple Integrals

* Evaluate integrals that might otherwise be considered too ugly.

14.7 – Cylindrical and Spherical Triple Integrals

* Evaluate triple integrals that might otherwise be considered too messy to do by hand. For practice, find the volumes of the bumpy spheres at the bottom of page 1041 (not in PDF textbook)  
   

14.8 – Jacobians

* If the original function was vertically simple (but with an ugly integrand), use Octave to numerically evaluate the given integral. Then perform the “ substitution,” get an answer, and compare it to the original.
* If the original region R was ugly then integral2 cannot be easily implemented.

1. f = @(x,y) x.^2.\*y.^2;

   ezsurf(f, [0,2,0,4], 40); shading interp; hold on;

   curve\_10a = ezplot3(@(t)sqrt(t), @(t)t, @(t)t.^3, [0, 4], 30); set(curve\_10a, 'linewidth', 3, 'color', 'r');

   curve\_10b = ezplot3(@(t)t, @(t)2\*t, @(t)4\*t.^4, [0 2], 20); set(curve\_10b, 'linewidth', 3, 'color', 'r');

   rotate3d on; [↑](#footnote-ref-1)