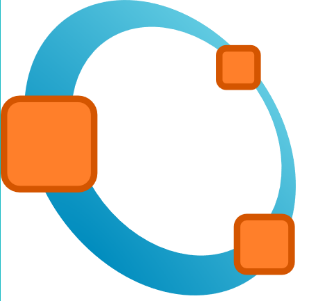
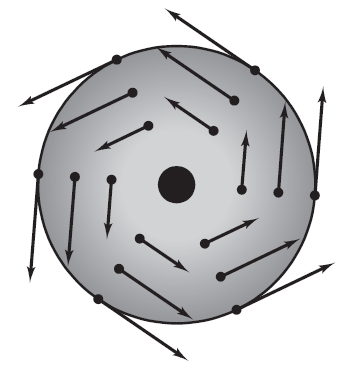
**Octave Tutorial - Ch 15**

*Visualizing Vector Fields and Approximating Important Integrals*

**Pursuits:**

As in previous chapters, we will focus on visuals and calculations. Some pursuits:

* Sketching a vector field in 2d or 3d
* Visualizing work by adding a path to our vector field
* Visualizing flow by seeing how different “streams” progress.
* Sketching parametric surfaces
* Using **integral**, **integral2** and **integral3** to finish calculations for work, circulation, and flux.



**Sketching a Vector Field in 2d:**

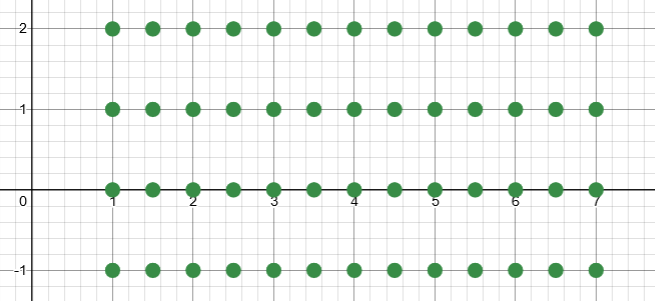
Whether drawing a vector field by hand or programming a computer the draw one, the three main steps are similar:

1. Define a set of points, such as the black dots in the image at right (from page 1055 of our textbook). This is done on Octave via the Meshgrid function.
2. At each coordinate, calculate the components of a vector.
3. Draw all the vectors.

Closely study the instructions for drawing a vector field in 2d. Then you should be able to extend the ideas to draw one in 3d.

1. Meshgrid – Octave uses the Meshgrid command to define a grid of coordinates.

The grid of coordinates shown here could be laid out in Octave using the following command:

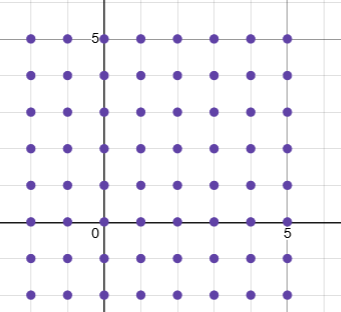
 **[x,y] = meshgrid(1:0.5:7, -1:2);**

Can you see/remember how this command works?

See Chapter 13 Octave Lesson for more details.

To actually generate a visual of the coordinates in Octave, paste the following line of code after defining the meshgrid.

**w = 0\*x; scatter3(x,y,w, 'filled'); rotate3d on;**



If you can be satisfied with a simple layout where the and dimensions match, such as the grid shown at right, then the ***absolute fastest*** way to define such a meshgrid would be

**[x, y] = meshgrid(-2 : 5)**

2. Now at each point in your grid, you must calculate the and components of a vector. Let’s follow Exercise 17 from 15.1 of our textbook:

**[x,y] = meshgrid(-5:5);**

**M = 0.25\*x.\*y;**

**N = 0.125\*y.^2;**

In the first step, we gave the computer a grid to think about. Something like this.

Then we create a matrix “**M**” of values. Something like this.

These are all the components.

The “**N**” matrix is similar and will contain all the components.

Notice that **M** and **N** are not anonymous functions. They are not ready for a user to input “x=2 and y=13.” Rather, they are (boring) fixed arrays of numbers, calculated based on calculations that were applied to another array (the Meshgrid).”

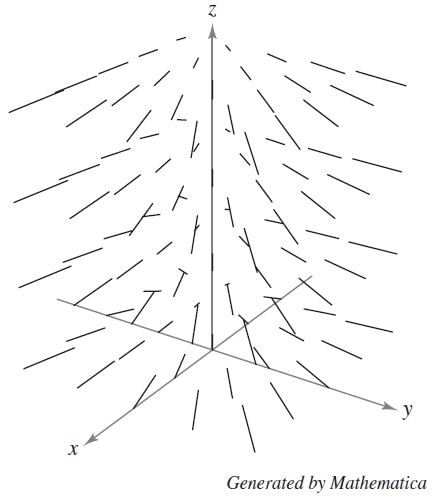
3) The quiver command will put it all together for us.

**quiver(x,y,M,N)**

…or to get fancy with dots and colors…

**quiver(x,y,M,N, 'marker', 'o', 'filled', 'markersize', 3,**

**‘color’, [1 0 0], ‘markerfacecolor’, [0 1 0])**

**Challenge!**

Use **quiver3** to create the vector field shown on page 1071 of our textbook:

It *seems* that values go from -2 to 2, from -2, to 2, and from 0 to 4.

Hint for starting:

**[x,y,z] = meshgrid(-2:2, , );**

**What do draw *in* the field:**

Once you have drawn a vector field, try adding a path through it! This will be useful for visualizing work integrals in section 15.2.

**hold on;**

**rx = @(t)...** 🡨 If you already called something else “x” in your Meshgrid, then this cannot be “x”.

**ry = @(t)...**  Alternatively, some people like to use **xx** or **X**.

**rz = @(t)...**

**ezplot3(...)**  🡨 Remember how to draw a rollercoaster? If not, check out the Ch 12 Octave guide.

Another fun thing is to add a “streamline.” This shows where a particle would flow if you if you dropped it in the vector field at a certain position. After creating a vector field, type.

**hold on;**

**streamline(x,y,z,M,N,P, xcoordinate, ycoordinate, zcoordinate)**

this is where I am struggling!!! it keeps throwing an error, saying that it is the wrong number of inputs

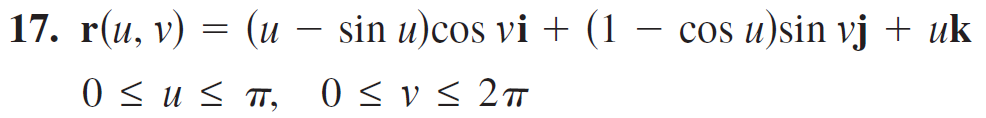
**Parametric Surfaces**

In Chapters 13 and 14, we used ezsurf to sketch surfaces that were defined explicitly as functions of and . Review:

**f = @(x,y) x.^2 + cos(x+y);**

**ezsurf(f, [-3, 3, -4, 4])** 🡨 Notice that only one function is getting plugged in.

But the same command can be used to plot parametric surfaces. Let’s use 15.5 #17 as an example.



This is a very different type of surface. It is ***not*** defined by “for any () pair, calculate .”

Rather , , and are all determined by separate functions.

First, define the component functions.

**x = @(u,v) (u-sin(u)).\*cos(v);** 🡨 Documentation uses **@(s,t)**. You may

**y = @(u,v) ...** use either style. Just be consistent.

**z = @(u,v) ...**

Next, use the **ezsurf** command, and include an array of min/max values for the independent variables u and v.

**ezsurf(x,y,z,[0, pi, 0, 2\*pi])**

And here are some random fun tricks that I recently learned how to do:[[1]](#footnote-1) (Can you see what they do?)

**figure(gcf, 'Name', 'Twisty', 'NumberTitle', 'off')**

**shading interp**

**camlight**

**colormap rainbow**

**colormap bone**

**colormap winter** 🡨 Check out the documentation of “colormap” for all 21 options ☺

**Integrals for Major Theorems:**

The major theorems in Vector Calculus all relate a “boundary calculation” to an “interior calculation” (those are Malan’s terms). They all look something like:

…but with many different kinds of integrals. The ideas are huge and must first be grasped visually and conceptually, before using Octave. But when actually evaluating any of these integrals…

1. The integrand starts out with vector calculations. (not great for base Octave)
2. The integrand can fairly easily be simplified to contain only scalar expressions. Thus, you will be left with only scalar integrals (or or ).



1. In simple textbook examples, the integration might be very clean and easy.   
   But in real life, these integrals **might not have…** 
   1. **…a simple antiderivative ☹  
       or**
   2. **…a nice region of integration. ☹**

***This is precisely where numerical calculations can rescue us!***

Example: Use Octave to verify Green’s theorem for the counter-clockwise path in the vector field

According to GT, 🡨 Do NOT try to type this into Octave.   
 Rather, do a couple more steps by hand.

**(need to finish this)**

**Chapter 15 Skills**

15.1 – Intro to Vector Fields

* Draw 2d VF
* Draw 3d VF
* Add streamlines to a VF

15.2 – Line Integrals

* Set up a scalar line integral by hand. Then use **integral** to evaluate. Not really a new skill. Just a good time to use numerical calculations to get a useful answer.
* Set up a vector line integral by hand. Get it to the point of just having a real valued expression (a function of ) in the integrand. Then use **integral** to evaluate. Not really a new skill. Just a time to use numerical calculations to get a useful answer.

15.3 – Fundamental Theorem of Line Integrals

* None.

15.4 – Green’s Theorem

* Use numerical integration to verify Green’s Theorem. Set up and use **integral**. Set up and use **integral2**.

15.5 – Parametric Surfaces

* Use ezsurf to draw parametric surfaces
* Optional: use “interp” and gouraud shading.

15.6-15.8 – Surface Integral and Flux

* Set up a flux integral. Get it to the point of just having a real valued expression (a function of and s) in the integrand. Then use **integral2** to evaluate.

15.7-15.8 – Major Theorems

* Use Octave to numerically verify that one of the major theorems holds for given problems. This will generally involve setting up complicated integrals by hand, simplifying the integrand until it has only a scalar function of multiple variables, and then using, integral, integral2 or intgral3.

1. In the first command, “gcf” means “get current figure” and is a way of telling Octave to look specifically at the figure you are currently working on. But if you knew it was figure 7, you could replace gcf with 7.

   <https://www.mathworks.com/matlabcentral/answers/471816-change-name-of-figures-in-figures-tab> [↑](#footnote-ref-1)