Homework F4

Tuesday, December 15, 2020 11:25 PM

document
(2)

Chapter 37

Homework 4: Center of Mass and Center of Buoyancy of Boats

Contents

In this homework assignment you will be exploring the concept of center of mass and center of buoyancy and how they relate to the motion of a solid submerged in water. These exercises will provide you with a conceptual and computational foundation for our explorations of boat stability next semester.

37.1 Exploration of Buoyancy

For this part of the assignment you will be working with a 3D physics simulation of a solid (let's call it a boat) submerged in water. To start the simulator open MATLAB, go to the your QEASimulators folder (you should have this in your MATLAB drive as a part of the work you did for the last assignment), and run the following two commands in MATLAB's command window.

```
>> addpath('Boats');
>> qeasim start fourteen_boats
```

If all went well, a web browser will pop up with this visualization of, you guessed it, 14 boats!!

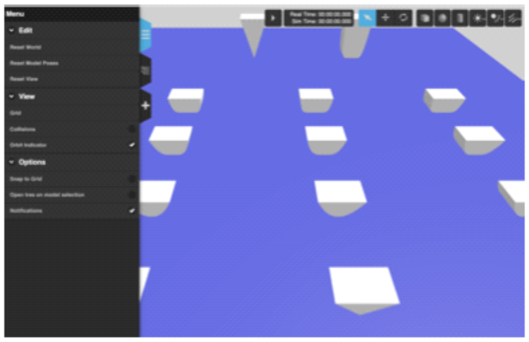


Figure 37.1: A visualization of a simulation of fourteen boats.

It will be useful to refer to each of these boats with a number (both when we run MATLAB code later, but also to refer to them in your answers). Here is a visualization with each of the boats labeled.

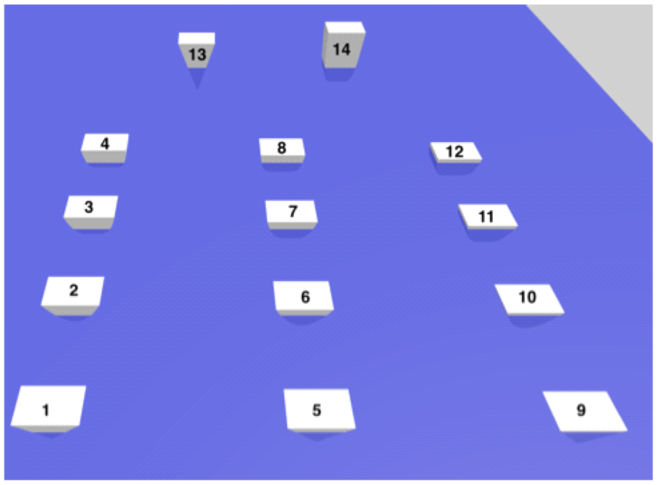


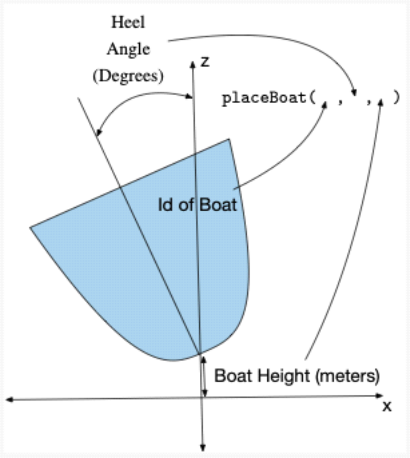
Figure 37.2: Each of the fourteen boats labeled with its number.

Boats 1–4, 13, and 14 are low density (their density is $\frac{1}{4}$ that of water). Boats 5–8 are medium density (their density is $\frac{1}{2}$ that of water). Boats 9–12 are high density (their density is $\frac{3}{4}$ that of water).

Exercise 37.1

Perform some experiments in the simulator and record your observations. We'll walk you through the most important actions you will need to perform in the simulator, but if you want a more complete rundown you can refer to the Gzweb user guide to learn how to use the interface.

1. Unpause the simulation by clicking the play button. Watch the boats come to equilibrium (meaning they stop moving). Examine the waterlines of each boat (measured from the bottom of the boat to the water). How do these waterlines compare across differently shaped boats? How do these waterlines compare across boats of different density?
2. Observe the frequency of oscillations of each boat. To observe the boats dropping into the water again, you can reset the simulator state by clicking the "Reset Model Poses" button under the "Edit" menu. How do the frequencies of oscillation compare across differently shaped boats? How do the frequencies of oscillation compare across boats of different densities?
3. Next we'll be manipulating the state of the boats using MATLAB. We have provided a function called `placeBoat`. The first input to `placeBoat` is the number of the boat you want to manipulate (see the figure above for the mapping of numbers to boats), the second argument is the heel angle of the boat in degrees, and the third angle is the height of the bottom-center of the boat relative to the water. For instance `placeBoat(1, 0, 1)` would drop boat 1 from a height of 1m (make to unpause the simulation to see what the boat does when placed in this configuration). Here is a diagram that shows each argument to `placeBoat`.



Different heights and shapes have very little impact on the oscillations. Mostly, the first wave amplitude is bigger. More density means the amplitude is smaller and it hits equilibrium sooner, and the speed is slower. Above/below the water seems to be symmetrical. Shape doesn't seem to make a difference.

For now we want you to focus on explore the boat's linear motion (i.e., with the boat held flat). Here are some experiments you might want to try.

- Drop a boat from some height above the water and observe what happens (e.g., run `placeBoat(1,0,2)` to drop the first boat from a height of 2 meters above the water). Try this for a few different boats and record some observations.
- Place a boat below the water and observe what happens (e.g., run `placeBoat(1,0,-0.5)` to place the first boat 0.5 meters below the water). Try this for a few different boats and record some observation.
- Determine the waterline of a boat by placing it at different depths. How do you know when you've found the waterline? Find the waterlines of a few of the boats and record your measurements.

4. We have also provided a function called `measureBoat` that will tell you a particular boat's current position and heel angle. The input to `measureBoat` is the number of the boat you'd like to measure. The function returns two outputs. The first output is a 3D column vector that holds the x, y, and z location of the boat in meters. The second output is the heel angle in degrees of the boat. For example, if you run `[pos, heel] = measureBoat(1)` then you should get back the current position and heel angle of boat 1.

   Using `measureBoat` determine the waterlines of a few of the boats.

Experimental:
Waterline for 1 = -0.25 (density?), 4 = -.15, 9 = -.45, 12 = -.4
Measured: 1 = -.2501, 4 = -.1459, 9 = -.4330, 12 = -.3872

Exercise 37.2

Next, we'll examine the rotational motion of the boats.

1. Using the function `placeBoat` rotate some of the boats by a small angle (try 10 degrees or less) (set the boat's depth using the waterline you found for the boat when it was floating flat). What happens to each boat when you rotate them a small amount (e.g., do they do they rotate

They rotate back. Heavier boats rotate faster, and rounder boats rotate smoother.

13 and 14 fall over

after being given a small heel angle? Do they bob up and down?) Write down any interesting patterns you see across the different boats. What happens when you rotate boats 13 and 14?

2. Using the function `placeBoat` rotate some of the boats by a medium angle (maybe 45 degrees or so). Try to set the boat's depth so they don't bob too much (you may find that the waterline has changed significantly from when the boat was floating flat). Write down any interesting patterns you see across the different boats.

3. Using the function `placeBoat` find the angle of vanishing stability (AVS) of a few of the boats (you can decide exactly how many to do and how you want to record the results). Describe the procedure you use to find the AVS of each boat. Remember that the AVS is the heel angle at which the boat stops righting itself and instead flips over. When determining the AVS, you'll want to make sure the boat is floating at its natural waterline for that particular angle (i.e., initially the boat shouldn't bob). Note any interesting patterns you see in the AVS values of different boats (e.g., how do the AVS values change with density?).

Heavier boats rotate slower but oscillate less. Rounder boats oscillate more.

1: 100deg
4: 95deg
9: 78deg
12: 87deg

Density decreases AVS.
Impact of shape is unclear.

37.2 Computational Approaches to Calculating Center of Mass and Center of Buoyancy

Now that you've performed some experiments to explore the behavior of the boats, you're going to develop a computational approach to compute two quantities that will be of primary importance in predicting the motion of the boat under various conditions: the center of mass and the center of buoyancy.

The table below describes the geometry and density of each boat. Note that we use a coordinate system where x goes across the boat, z is up and down, and y goes along the boat (i.e., in the direction perpendicular to the boat's cross section). We use meters for units of length.

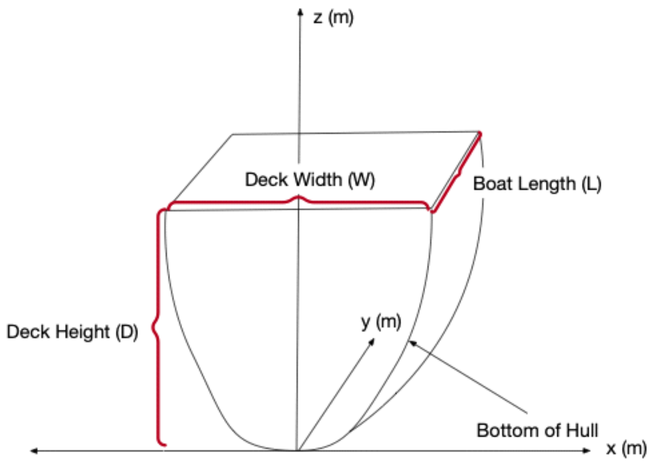


Figure 37.3: Our coordinate system and other conventions for describing the shapes of the boats.

Boat ID	Height (D)	Width (W)	Bottom of Hull	Length (L)	Density
1	0.5m	1m	$D\frac{\lvert 2x\rvert}{W}$	0.6m	$250\ kg/m^3$
2	0.5m	1m	$D\left(\frac{2x}{W}\right)^2$	0.6m	$250\ kg/m^3$
3	0.5m	1m	$D\left(\frac{2x}{W}\right)^4$	0.6m	$250\ kg/m^3$
4	0.5m	1m	$D\left(\frac{2x}{W}\right)^8$	0.6m	$250\ kg/m^3$
5	0.5m	1m	$D\frac{\lvert 2x\rvert}{W}$	0.6m	$500\ kg/m^3$
6	0.5m	1m	$D\left(\frac{2x}{W}\right)^2$	0.6m	$500\ kg/m^3$
7	0.5m	1m	$D\left(\frac{2x}{W}\right)^4$	0.6m	$500\ kg/m^3$
8	0.5m	1m	$D\left(\frac{2x}{W}\right)^8$	0.6m	$500\ kg/m^3$
9	0.5m	1m	$D\frac{\lvert 2x\rvert}{W}$	0.6m	$750\ kg/m^3$
10	0.5m	1m	$D\left(\frac{2x}{W}\right)^2$	0.6m	$750\ kg/m^3$
11	0.5m	1m	$D\left(\frac{2x}{W}\right)^4$	0.6m	$750\ kg/m^3$
12	0.5m	1m	$D\left(\frac{2x}{W}\right)^8$	0.6m	$750\ kg/m^3$
13	1.5m	1m	$D\frac{\lvert 2x\rvert}{W}$	0.6m	$250\ kg/m^3$
14	1.5m	1m	$D\left(\frac{2x}{W}\right)^8$	0.6m	$250\ kg/m^3$

Exercise 37.3

First let's compute the center of mass of one of our boats. In Week 4b we saw two different approaches for solving this problem. The first approach we saw was to chop up the boat into a bunch of little rectangles and treat each of those rectangles as a discrete mass. The second approach we saw was to use a double integral to compute the center of mass. In this exercise we're going to use the first approach (chopping up the boat in rectangles) for reasons that will become clear as you move through the problem. Since each of these boats is made from a simple 2D shape that has been extruded along its y-axis, we'll analyze the 2D cross section of the boat and use that to compute the center of mass. Next semester you'll see how to extend this analysis to a 3D boat hull (or perhaps by the end of this you may already be able to see yourself how you might do this).

1. Choose one of the boats. In MATLAB using the function `meshgrid` define a 2D grid that encompasses the entire cross section of the boat's hull (it's okay if the meshgrid extends beyond the boat hull, but make sure you it fully covers the hull). Take the resulting matrices for your x and z points and reshape them into a matrix that is $2 \times N$ where N is the total number of points in your mesh grid (i.e., each point in your meshgrid should be represented as 2×1 column vector in this matrix).

   Hint: you may find it useful to use the syntax `X(:)` to convert the matrix X into a column vector by unrolling it columnwise (e.g., if `X = [1, 2; 3, 4]` then `X(:)` will give the column vector `[1; 3; 2; 4]`).

2. Create an $N \times 1$ column vector that contains the mass of each of the N areas that comprise your mesh grid. Since some points in your mesh grid might lie outside the boat's hull, you'll want to make sure that you give these areas a mass of 0.

   Hint: You can use Boolean expressions in MATLAB to test whether or not a point is in your hull. For example, if you have your meshgrid points in the $2 \times N$ matrix P, the bottom of your boat's hull defined by the curve $z = x^2$, and the top defined by the curve $z = 1$, you could use the following code to create a Boolean vector with a 1 when the corresponding column of P represents a point in the hull and 0 when the point is outside the hull.

   ```
   insideBoat = P(2,:) >= P(1,:).^2 & P(2,:) <= 1;
   ```

Hint 2: To make sure you did things correctly you can visualize your function `insideBoat` using the `scatter` function.

```
scatter(P(1,insideBoat), P(2,insideBoat));
```

3. Compute the 2D center of mass of the boat using your $2 \times N$ matrix of meshgrid locations along with your $N \times 1$ column vector of masses. If you want to use loops at first, go ahead, but perhaps try to use the matrix multiplication tools we learned about in module 1 to speed up and simplify your code (we did this same problem in Week 4b).

Exercise 37.4

Using one of the boat shapes (you can use the same one you did for the previous problem or choose a new one), compute the center of buoyancy of the boat assuming the water is at at the level $z = d$ (if you are working towards improving your MATLAB programming chops, this would be a great opportunity to create a function that takes d as an input and returns the center of buoyancy of the boat as its output). The center of buoyancy of the boat is defined by the following formulas where a_i is the area of the ith region.

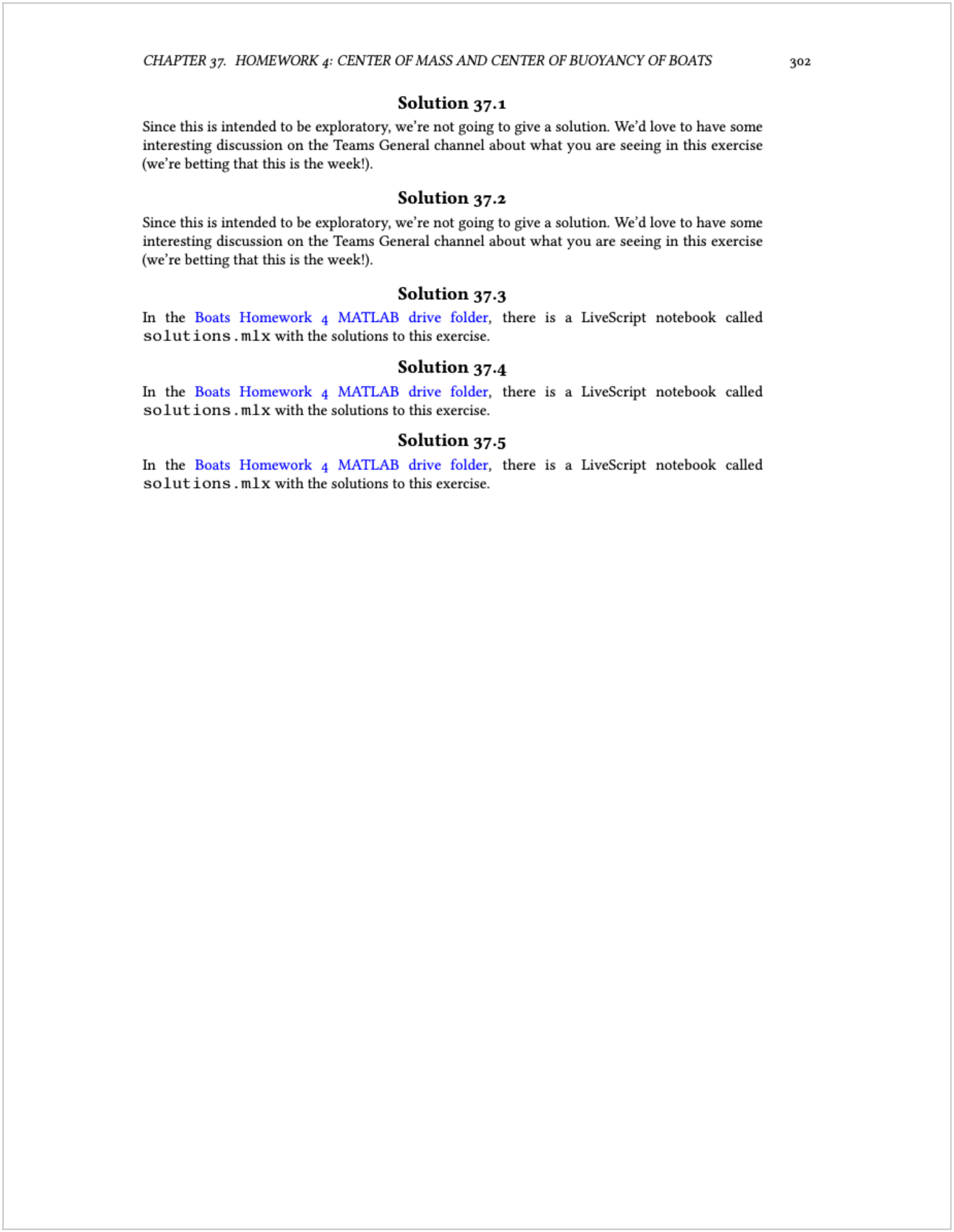
$$x_{COB} = \frac{1}{\sum_i a_i} \sum_i a_i x_i \tag{37.1}$$

$$z_{COB} = \frac{1}{\sum_i a_i} \sum_i a_i z_i \tag{37.2}$$

Notice that these are the same formulas we saw for the COM except we sum over each area times its x-coordinate (e.g., for x_{COB}) instead of mass times the x-coordinate (e.g., as we did for x_{COM}). To see the connection to the previous problem clearly, you may want to compute an $N \times 1$ column vector that includes the area of each point in the mesh grid that is below the water and inside the boat hull (assign an area of 0 for any points that don't meet these two criteria).

Exercise 37.5

Plot the center of mass and center of buoyancy of your boat after rotating the boat by some angle about the origin (recall that this is the heel angle of the boat). You can create a function that takes both the depth of the boat and the heel angle and creates the plot or if you are not yet comfortable with that, you can hardcode particular values. You should also visualize the boat hull in some manner (Using `scatter` as we suggested in a previous hint is a good approach. We show how to do this in the solutions notebook). In order to perform this analysis, all you need to do is rotate your boat about the origin by the appropriate angle (time to dust off your 2D rotation matrix from earlier in the semester!).



Solution 37.1

Since this is intended to be exploratory, we're not going to give a solution. We'd love to have some interesting discussion on the Teams General channel about what you are seeing in this exercise (we're betting that this is the week!).

Solution 37.2

Since this is intended to be exploratory, we're not going to give a solution. We'd love to have some interesting discussion on the Teams General channel about what you are seeing in this exercise (we're betting that this is the week!).

Solution 37.3

In the Boats Homework 4 MATLAB drive folder, there is a LiveScript notebook called `solutions.mlx` with the solutions to this exercise.

Solution 37.4

In the Boats Homework 4 MATLAB drive folder, there is a LiveScript notebook called `solutions.mlx` with the solutions to this exercise.

Solution 37.5

In the Boats Homework 4 MATLAB drive folder, there is a LiveScript notebook called `solutions.mlx` with the solutions to this exercise.

PDF

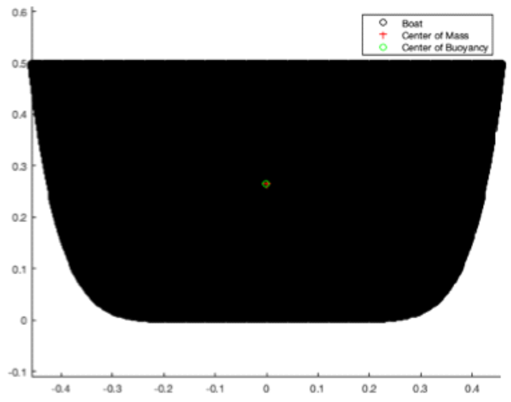
Excercises

Excercises

```
% 37.3, 37.4
[centerOfMass, centerOfBuoyancy, P] = calculate_boat(250, @(P) P(2, :) >= ((2 / 1 * P(1, :)) .^ 8) & P(2, :) <= 0.5)
```

```
centerOfMass = 2×1
    0.0000
    0.2648
centerOfBuoyancy = 2×1
   -0.0000
    0.2648
P = 2×203292
   -0.4575   -0.4575   -0.4575   -0.4575   -0.4575   -0.4575   -0.4575   -0.4575   -0.4575   -0.4555   -0.4555   -0.4555   -0.4555
    0.4915    0.4925    0.4935    0.4945    0.4955    0.4965    0.4975    0.4985    0.4995    0.4745    0.4755    0.4765    0.4775
```

```
clf; scatter(P(1, :), P(2, :), 'k'); axis equal; hold on;
plot(centerOfMass(1), centerOfMass(2), "r+");
plot(centerOfBuoyancy(1), centerOfBuoyancy(2), "go");
legend("Boat", "Center of Mass", "Center of Buoyancy")
```



37.5

```
com_rot = rotation_matrix(45) * centerOfMass,
```

```
com_rot = 2×1
   -0.1873
    0.1873
```

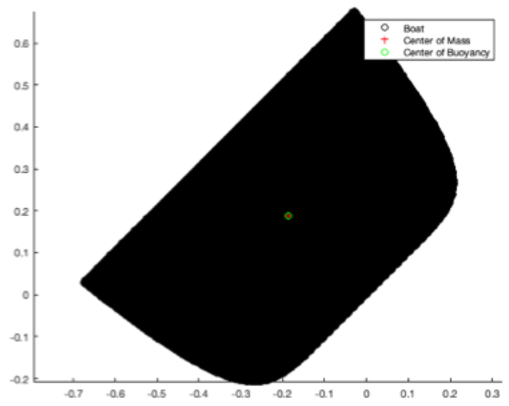
```
cob_rot = rotation_matrix(45) * centerOfBuoyancy
```

```
cob_rot = 2×1
   -0.1873
    0.1873
```

```
P_rot = rotation_matrix(45) * P
```

```
P_rot = 2×203292
   -0.6710   -0.6717   -0.6724   -0.6731   -0.6738   -0.6745   -0.6753   -0.6760   -0.6767   -0.6576   -0.6583   -0.6590   -0.6597
    0.0241    0.0248    0.0255    0.0262    0.0269    0.0276    0.0283    0.0290    0.0297    0.0134    0.0142    0.0149    0.0156
```

```
clf; scatter(P_rot(1, :), P_rot(2, :), "k"); axis equal; hold on;
plot(com_rot(1), com_rot(2), "r+");
plot(cob_rot(1), cob_rot(2), "go");
legend("Boat", "Center of Mass", "Center of Buoyancy")
```



Functions

```
function [centerOfMass, centerOfBuoyancy, pointsInBoat] = calculate_boat(density, predicate)
    xPoints = linspace(-1, 1, 1000);
    zPoints = linspace(0, 1, 1000);

    % I couldn't figure out that I needed to do this without the solutions
    areaPerCell = (xPoints(2) - xPoints(1)) * (zPoints(2) - zPoints(1)); % area represented by each point

    [X, Z] = meshgrid(xPoints, zPoints);
    P = [X(:)'; Z(:)'];

    isInsideBoat = predicate(P);
    pointsInBoat = P(:, isInsideBoat);
    massScalar = (isInsideBoat * density * areaPerCell);
    centerOfMass = (P * massScalar') / sum(massScalar, 2);
    areaScalar = isInsideBoat * areaPerCell;
    centerOfBuoyancy = (P * areaScalar') / sum(areaScalar);
end
```

```
function mat = rotation_matrix(angle)
    mat = [cosd(angle), -sind(angle); sind(angle), cosd(angle)];
end
```