Floating objects and their stability

Evelyn Sander
George Mason University, Fairfax, VA USA

January 5, 2025

For almost any floating object, whether a large iceberg, a kayak, or a water bug, there are preferred configurations in which the object floats. Most objects have multiple stable floating configurations. For example, most boats float stably in both an upright position and also in an upside down position.[1] If an object is placed in any other configuration, it will rotate until it has righted itself into one of its stable configurations.

The goal of this chapter is to use 3D printing to study how objects float. You will learn to create a design for a 3D printed object, use theoretical and numerical methods to predict how it will float. You then will float the objects to test your predictions.

In this discussion of floating objects will introduce the physics required for understanding how objects float. To simplify matters, we will consider the case of long objects with a fixed cross section. Such objects will float with their long side horizontal, with the water hitting each cross section and the same point. That is, we have reduced the problem to two dimensions. We can fully model how the objects float by knowing the two-dimensional shape of the cross section and the density of the object.

There are exceptional objects, such as a long cylinder with circular cross section, which float equally well in every configuration – this is the basis of log rolling competitions, see Fig. 1. While it is of mathematical interest, known as Ulam's floating body problem, to find and classify other objects with no preferred floating direction, we are more interested in the typical case, where there are a finite number of distinct stable floating configurations.

We proceed as follows: Sec. 1 describes how to design and print an object to float. Sec. 2 introduces the physics of floating objects, in particular Archimedes' Principle, and the potential energy of an object floating in any configuration. The local minima of this potential energy function correspond to stable floating configurations. The computation of the potential energy can be quite complicated for all but the simplest

[1] During the Apollo missions, NASA had to contend with these two stable floating positions during splashdown of the command module. They could not know whether it would settle in the water rightside up, termed "Stable 1" or upsidedown, termed "Stable 2".

Figure 1: Log rolling competition at the Minnesota State Fair August 2024. The difficulty with walking on a floating log is that due to its circular cross section, there is no preferred floating direction. Photo by D.M. Anderson.

shapes, but there is already existing Matlab code for computing it. Sec. 3 explains how to use this code to predict how your print will float.

The code described in this chapter is part of the ongoing software package IcebergProject, as it was inspired by trying to understand how icebergs float. It was created by students, lead by GMU faculty D.M. Anderson and E. Sander. It is written in Matlab and is available from the following link `https://github.com/danielmanderson/IcebergProject`.

1 Designing floating objects

This section shows you how to create your own customized printable STL file for a long thin object with fixed cross section, where the cross section is given by an image that you get to choose. You will also save the data points that create this print so that you can test the data to predict how the print should float.

Here is a step-by-step guide.

Step 1. **Download code.** Designing the floating object is done using code from the folder DesignFloat of the IcebergProject `https://github.com/danielmanderson/IcebergProject`.

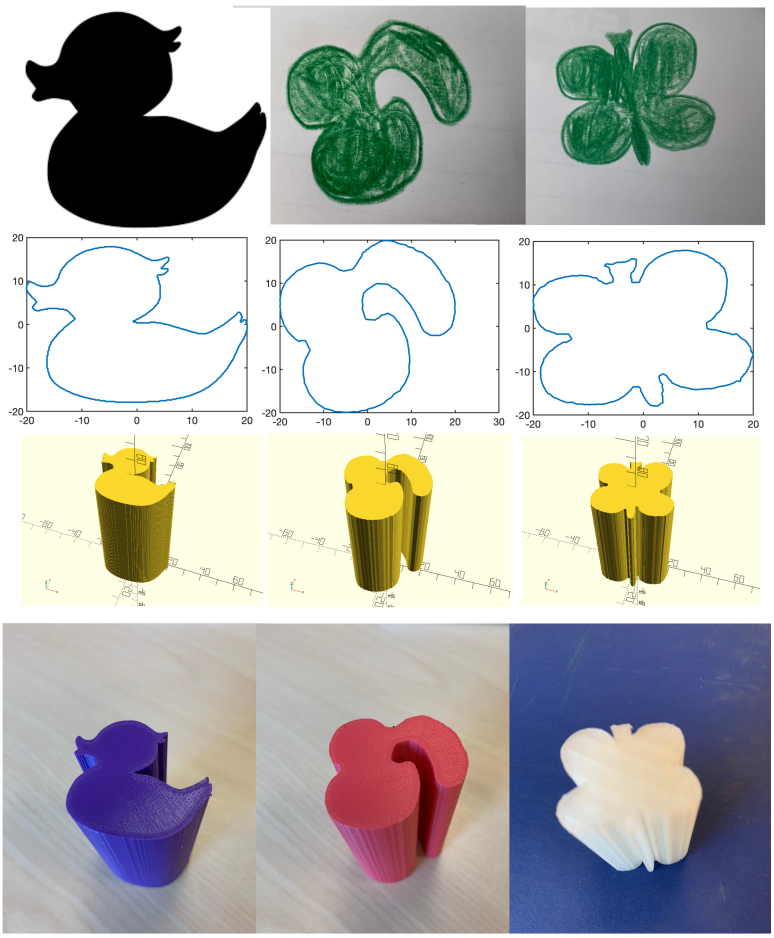

Figure 2: Using the codes `figure2points.m` and `makefloat.scad`. Top row: Three starting image files, which we call the shapes "duck," "monster," and "butterfly." Middle row: The corresponding polygonal shape created by `figure2points.m` using the cutoff values of $0, 0.21$, and 0.33 respectively (found by trial and error). Third row: The resulting OpenSCAD 3D object created by `makefloat.scad`. This can now be saved as an STL file and 3D printed. Bottom row: Photos of the corresponding 3D printed objects.

Step 2. **Find the shape of the cross section.** You will need to find or create a file with an image in the shape of the cross section of your object and put it in the folder containing the downloaded code. The shape must be simply connected, meaning that it has a single connected boundary (with no holes inside of it). You can either find an image online, or drawing a filled in shape on a piece of paper, and take a photo of the picture.

Fig. 2 top row shows three different examples of image files. The first is a down-

loaded file, and the second and third are hand drawn. (For best results, fully color in your shape with a thick black felt tip marker.)

If you are using a photo of a hand drawn object, or if your image is multicolored, then you will have better results if you preprocess the image as follows before going on to the next step. The goal is to make sure that the distinction between shape and boundary is clear so there are no erroneous points added in when the object is digitized.

- Take a photo with a clean background without shadows or specks.
- Crop the photo leaving only a small amount of background. However, be careful, as the edge of the shape cannot touch the boundary.
- Increase the contrast and exposure to make the distinction between the shape and the background as great as possible. (If you still have issues, you can also try changing the sharpness or blur.)
- Downsize the image file. It should be less than 1000×1000 pixels. Often an image of size 400×400 pixels suffices.

Step 3. **Digitize your cross section and convert to polygon.** In Matlab, navigate to the folder containing the downloaded code (Step 1), as well as your image file. In Matlab, `figures2points.m` that you downloaded. To use this code, you type `figure2points(input,output,cutoff)`, where input is the input filename (with the suffix), output is the output filename (without a suffix), and cutoff is the cutoff intensity (between 0 and 1) defining the outside of the shape.

You will need to experiment with the value of cutoff. Here are guidelines for values to try first, modifying as needed:

- For a photo of a hand drawn object, first try a cutoff value of 0.001.
- For a downloaded black and white image, first try a cutoff value of 0.

For example, to use the file `duck.png` and create a Matlab data file `duck.mat`, with cutoff intensity 0, you type

```
[xvalues,yvalues] = figure2points('duck.png','duck',0);
```

Fig. 2 middle row shows the corresponding polygon for each of our example images. It was necessary to set a different value of cutoff for each of these images. This code takes an image and converts it into a counterclockwise oriented polygon. It does this in three ways:

a. It plots a picture of the polygon created. The polygon is scaled to have a maximum length of 40mm. If the picture does not look right, vary the cutoff value and retry.

b. It creates the data file `duck.mat`.

c. It creates a data file formatted for OpenSCAD titled `poly.scad`.

d. It gives the cross sectional area for the polygon created. Since 3D prints are measured in millimeters, the area is measured in square millimeters.

Step 4. **Create an STL file.** After completing Step 3, open the OpenSCAD file `makefloat.scad` and create an STL file, as in the bottom row of Fig. 2.

a. By default, the height of the print is 70mm. Change the variable hei on line 1 to change this value.

b. Render (Design menu).

c. Export as STL (File in the Export Menu).

You should now be able to 3D print a floatable object. Alternatively you might want to wait until you have read the theoretical sections and completed a prediction on how the object will float. We will soon see that the *density* of the object has significant impact the stable floating configurations of the object. The density is varied by changing the percentage infill you choose when printing.

When you print, make sure to set your infill between 10% and 85% (the upper end applies to PLA filament). To explain the lower limit, in all calculations, we assume that the printed object has constant density. For any infill lower than 10%, we have found that the objects are no longer close to constant density. For the upper limit, PLA is heavier than water, and we find that above around 85% infill the object will sink in water. Fig. 3 shows experimental data for infill versus density for PLA.

2 The physics of floating objects

In this section, we present the physics of floating objects. In some cases we give a general definition, but we will most carefully consider long thin objects with a fixed cross section L, since this is the type of objects designed for 3D printing in the previous section. This guarantees that the object will float along the water on its long axis. We start by presenting some basic definitions. The mass of an object is a scalar measure that should be familiar to all. The center of gravity, or equivalently[2] the center of mass may also be familiar, as it is a point in space that is the balance point of an object, meaning that the principles of motion of the object will not change if the object is

[2]at least they are equivalent on Earth

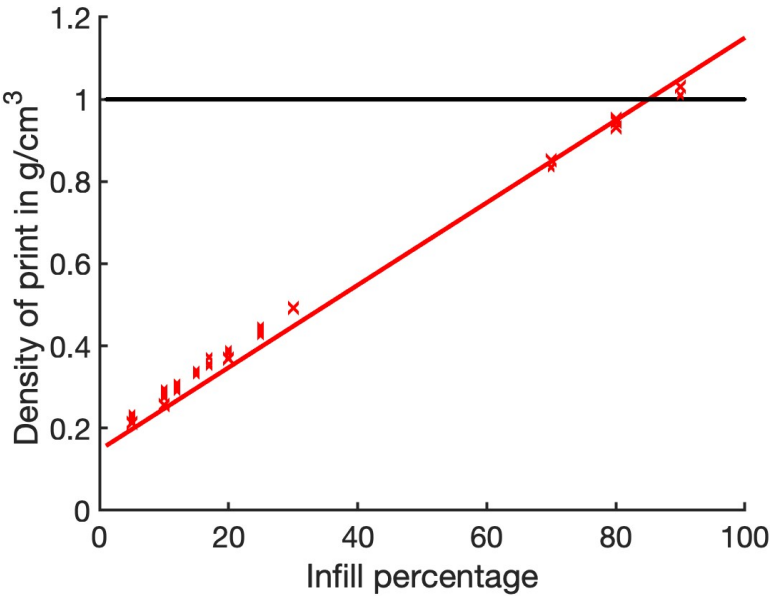

Figure 3: Experimental results for infill percentage versus density for floating objects printed with PLA. The black line shows the density of water: a print with higher density will sink in water. Data is from [?] Anderson et. al.

rotated about this point. The center of buoyancy is perhaps less familiar. It is also a point in space, in this case a balance point of the fluid that has been displaced by an object floating in fluid. The following definitions give the formulas for these three quantities.

Consider a solid object in a domain Ω_3 in three dimensions with nonconstant density $\varrho_{\text{obj}}(x, y, z)$. In the context of 3D printing, we are particularly interested in the special case of a long object with fixed cross section and constant density, like the objects shown in Fig. 2. We additionally assume that the object is floating in a fluid with constant density. We formalize this set of conditions in the following definition.

Definition 2.1 (Fixed cross section object)**.** *We assume an object with constant density ϱ_{obj}, where Ω_3 is a long object with fixed cross section. If the cross section is a two-dimensional domain Ω, then the three-dimensional domain is of the form $\Omega_3 = \Omega \times [0, L]$, where L is sufficiently larger than the largest diameter of Ω. We also assume that the object is floating in a fluid of fixed density ϱ_{fluid}.*

There are three important concepts related to a fully or partially submerged object, which we now define. For each definition, we give the general case of any floating object. Then after each definition, we give the corresponding formulas for this special case defined above in Definition 2.1.

Definition 2.2 (Mass)**.** *The mass of the object is given by*

$$M_{\text{obj}} = \iiint_{\Omega_3} \varrho_{\text{obj}}(x, y, z) dV \quad . \tag{1}$$

Computing the integral of a constant shows that for an object as in Definition 2.1,

$$M_{\text{obj}} = \varrho L A_{\text{obj}}.$$

Since the mass is easily measurable, we know L, and the software has computed A_{obj}, this formula allows us to compute the density ϱ of the object. A note on units: 3D printing is measured in millimeters. The density of water is 1 gram per cubic centimeter. For comparison, make sure to find areas in cm^2, volumes in cm^3, and masses in grams.

If you are wondering how we compute A_{obj}, recall that by the way that we digitized our object, it is a polygon with vertices $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), (x_1, y_1)\}$ in counterclockwise orientation. From this information, a calculation using Green's Theorem shows that the area is given by

$$A_{\text{obj}} = \frac{1}{2} \sum_{1}^{N} (x_k + x_{k+1})(y_{k+1} - y_k).$$

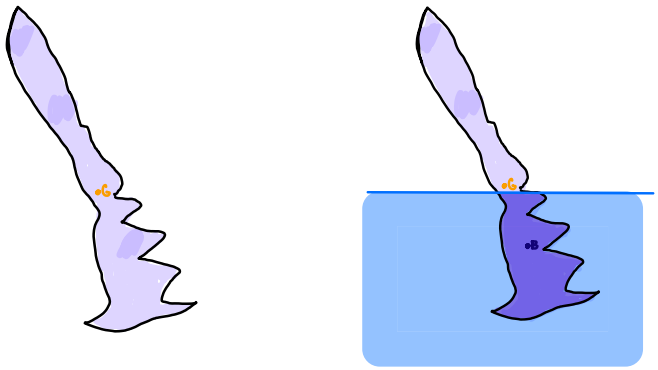

Figure 4: Center of gravity G and center of buoyancy B.

Definition 2.3 (Center of gravity)**.** *The center of gravity is the average location of the weight of an object, or balance point of an object. See Fig. 4. Formally,*

$$\langle G_x, G_y, G_z \rangle = \frac{1}{M_{\text{obj}}} \int_{\Omega} \langle x, y, z \rangle \varrho(x, y, z)) dV \quad . \tag{2}$$

If an object has constant density, the center of gravity is called the centroid.

For objects satisfying Definition 2.1, we expect that the balance point in the long direction will be halfway along the object. Indeed, the following integral calculation confirms this.

$$\begin{aligned} G_z &= \frac{1}{M_{\text{obj}}}\varrho \iiint_{\Omega_3} z \, dz \\ &= \frac{\varrho}{\varrho L A_{\text{obj}}} \iint_{\Omega} \left(\int_0^L z \, dz \right) dA \\ &= \frac{1}{L A_{\text{obj}}} A_{\text{obj}} \left(\frac{L^2}{2} \right) = \frac{L}{2} \, . \end{aligned}$$

A similar integral with cancelations for the other two directions shows that the center of gravity is given by

$$\vec{G} = \langle G_x, G_y \rangle = \frac{1}{A_{\text{obj}}} \iint_{\Omega} \langle x, y \rangle dA \, . \tag{3}$$

How can we compute this value? Just as with the area calculation, since our object is given by a polygon, this center of gravity integral simplifies to a sum of products of x_k and y_k, the coordinates for the vertices.

Definition 2.4 (Center of buoyancy)**.** *If an object is submerged or partially submerged in a fluid, the center of buoyancy is the center of gravity of the fluid that has been displaced.*

If we define M_{sub} to be the mass of the fluid displaced by the object and let $\Omega_{\text{sub},3}$ be the three-dimensional submerged domain, then

$$\langle B_x, B_y, B_z \rangle = \frac{1}{M_{\text{sub}}} \iiint_{\Omega_{\text{sub},3}} \langle x, y, z \rangle \varrho_{\text{fluid}}(x, y, z) dV \tag{4}$$

Assuming that the object satisfies Definition 2.1 and that the fluid has constant density, we can use similar calculations to the previous integrals to show that the center of buoyancy is the centroid of the submerged portion of the object, as in Fig. 4. This is given by

$$\vec{B} = \langle B_x, B_y \rangle = \frac{1}{A_{\text{sub}}} \iint_{\Omega_{\text{sub}}} \langle x, y \rangle dA \, , \text{ and } B_z = \frac{L}{2}.$$

The major physics result that we use in the theory of floating objects is the Archimedes' Principle, which should be thought of as a force balance of the force due to gravity and the force due to buoyancy[3].

[3]In fact the buoyancy comes from the pressure force.

Theorem 2.5 (Archimedes' Principle)**.** *When a floating object is at rest, the upward buoyant force exerted on an object wholly or partially submerged is equal to the weight of the displaced fluid. Written as an equation, this says*

$$M_{\text{object}}g = M_{\text{fluid}}g \ . \tag{5}$$

In the case of a long fixed cross section object defined in Definition 2.1, we can rewrite this and rearranging terms to get that

$$\varrho_{\text{obj}}A_{\text{obj}} = \varrho_{\text{fluid}}A_{\text{sub}} \ .$$

Define R to be the density ratio *(sometimes referred to as the specific gravity) between the object density and the fluid density.*

$$R = \frac{\varrho_{\text{obj}}}{\varrho_{\text{fluid}}}. \tag{6}$$

Note that for floating objects, $0 < R < 1$. The lower inequality comes from the fact that objects have positive density. The upper inequality follows from the fact that if $R \geq 1$, then the object will sink instead of floating.

Let h be the height of the waterline. Then the Archimedes' Principle states that at rest,

$$F_R(h) = \frac{A_{\text{sub}}(h)}{A_{\text{obj}}} - R = 0.$$

The function $F_R(h)$ in the last formula is a continuous function of the height of the waterline h. The last formula in this equation is especially nice, since computers are very good at finding zeros of continuous functions. Also notice that for $h = h_{\text{bottom}}$, the bottommost height of the object, none of the object is submerged, and therefore $A_{\text{sub}}(h_{\text{bottom}}) = 0$. Therefore

$$F_R(h_{\text{bottom}}) = 0 - R < 0.$$

Further, if we let h_{top} be the height of the topmost point of the object then $A_{\text{sub}}(h_{\text{top}}) = A_{\text{obj}}$, implying that

$$F_R(h_{\text{top}}) = 1 - R > 0.$$

Since F_R is a continuous function which changes sign, the Intermediate Value Theorem guarantees that between $h = h_{\text{bottom}}$ and $h = h_{\text{top}}$, there is a point such that $F_R(h) = 0$, as shown in Fig. 5. We have already written a code that finds this point using the numerical bisection method for finding a zero of a continuous function.

All this implies that for any long floating object, as long as we keep it in a fixed orientation, we can determine how far down into the water the object floats. We are now in a position to figure out the orientation that our object floats stably. To do so, we will need a potential energy function.

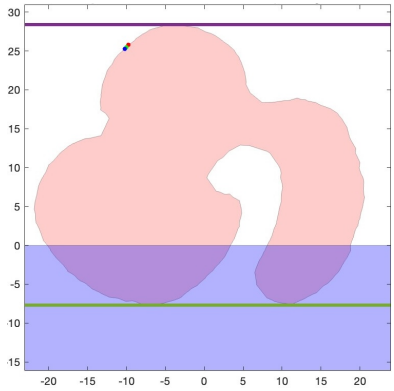

Figure 5: This figure shows a the shape "monster." The green line is at $h = h_{\text{bottom}}$. The purple line is at $h = h_{\text{top}}$. The waterline is somewhere in between, such that for $\varrho = 0.2$ the Archimedes' Principle holds, meaning that $F(h) = 0$.

In order to consider all possible floating configurations of the object, we rotate the object *clockwise* by angle θ, keeping the waterline fixed. For each value of θ, we can then compute the location of h, which depends on both R and θ, and we can compute a center of buoyancy $\vec{B}(\theta, R)$, such that the Archimedes' Principle is satisfied.

Now that we have a value of h corresponding to each value of R and θ, there is a *potential energy* associated with floating at this angle, which we give in the following theorem.[4] The potential energy involves the two forces: the force due to gravity and the force due to buoyancy. We give a brief description of the work done by each of these forces, in the case where θ and R are both fixed. That is, we fix the density of the object, and we do not allow the object to rotate.

The work done against gravity to move an object from a reference point $\vec{D}$ to its current location is mgH_g, where H_g the vertical distance between $\vec{D}$ and the center of gravity $\vec{G}$. If $\hat{n}$ is a unit vector orthogonal to the waterline, then is given by $H_g = \hat{n} \cdot (\vec{G} - \vec{D})$, and the work done against gravity is $mg\hat{n} \cdot (\vec{G} - \vec{D})$.

The buoyancy force corresponds to the work done when the object displaces water. In particular, compared to reference point $\vec{D}$, the work done against buoyancy is $-mgH_B$, where H_B is the distance between the center of buoyancy and $\vec{D}$. The negative sign is due to the fact that the water is displaced from upward to the surface. That is, the work done against gravity is $mg\hat{n} \cdot (\vec{D} - \vec{B})$.

The following theorem states that the potential energy is the total work against these two forces.

Theorem 2.6 (Potential energy)**.** *The potential energy of an object at rest rotated to*

[4]Following Gilbert, *How Things Float*, MAA Monthly (1991).

angle θ with relative density R is given by

$$\vec{U} = M_{\text{obj}} g \hat{n} \cdot \left(\vec{G} - \vec{B}(\theta, R) \right),$$

where $\hat{n}$ is a unit vector normal to the waterline.

The stable floating configurations are the local minimizers with respect to θ of this potential energy at a fixed value of R. We have already described a bisection method for finding $\vec{B}$ at each fixed value of θ and R. For each fixed R this therefore allows us to find $\vec{U}$ and its local minimizers as θ varies. In the next section, we will describe how to use pre-written code to compute this potential energy, as well as how to find the energy minimizers, thereby predicting all stable floating configurations for an object with fixed relative density R.

Note that we can observe without making any calculation that since the gravity acts in a strictly downward direction, in order to be at equilibrium with a balance of the two forces, the buoyancy must act in a strictly upward direction. Therefore, any stable floating configuration has $\vec{G}$ and $\vec{B}(\theta, R)$ stacked vertically. This observation could also be seen using a torque argument. However, we do not use the vertical stacking at any point in the computations described in the above paragraph.

You might wonder why we do not assume that $\hat{n}$ is a fixed upward vector in our equations above. Although that seems most natural, it is often more computationally efficient to allow for the coordinate system to change. In particular, in our calculations, we are keeping the object fixed and moving the waterline counterclockwise by an angle θ. Computationally it is easier to move the waterline, since the object is a complicated polygon, and the waterline is a simple line. This is equivalent to rotating the object clockwise and keeping the waterline fixed.

3 Using potential energy to predict stability

In this section, we describe how to compute predictions for how an object floats in water, based on the theory in the previous section. The code

Step 1. **Download code and data.** Designing the floating object is done using code from the folder PotentialEnergyFloat of the IcebergProject `https://github.com/danielmanderson/IcebergProject`.

If you created your object using the instructions in Sec. 1, then you can load your `.mat` data using the Matlab `load` command. For example moving `monster.mat` to the PotentialEnergyFloat folder, and typing `load monster.mat` defines vertices `xvalues` and `yvalues` of the object in the middle column of Fig. 2

Step 2. **Plot data.** Type `plotshape(xvalues,yvalues,1)`, and make sure that the orientation is counterclockwise – the red, green, and blue points should appear

in counterclockwise order, as seen in Fig. 6. The command also gives the cross sectional area A_{obj} in mm^2. By default, the length of your object is 70 mm. Therefore the volume is

$$V_{\text{obj}} = \frac{A_{\text{obj}} \times 70}{1000} \text{ cm}^3.$$

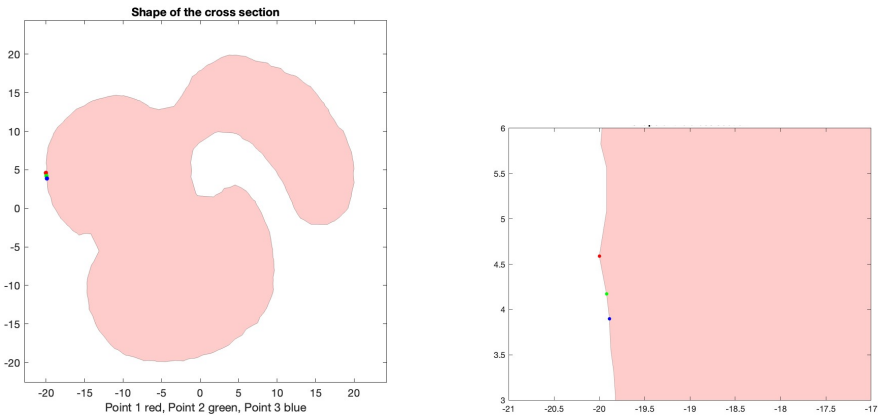

Figure 6: The red, green, and blue points respectively from the command `plotshape` allow you to verify that the object has counterclockwise orientation.

Step 3. **Compute R.** The previous step gave the volume of your object in cm^3. Use a scale to measure the mass of your object in grams, and compute $\varrho_{\text{obj}} = M_{\text{obj}}/V_{\text{obj}}$. If you are floating your object in water $\varrho_{\text{fluid}} = 1$ g/cm^3, and therefore $R = \varrho_{\text{obj}}$.

Step 4. **Compute potential energy minimizers.** To compute the potential energy defined in the previous section, type `PotEnergyCalc(xvalues,yvalues,R)`. The result will look like Fig. 7. The following will computationally find the corresponding local minima, and to see what the object would look like in this stable position, such as in Fig. 8.

```
StableFloatAngles = findPEmin(xvalues,yvalues,R)
```

You can compare these images to what you see. More sophisticated ways of measuring the orientation use careful photos, and digitized angle measurements.

Step 5. **Compute θ, R plot.**

To see a summary of all the possible stable positions in an R versus θ plot, as shown in Fig. 9, type

```
makethetaplot(xvalues,yvalues)
```

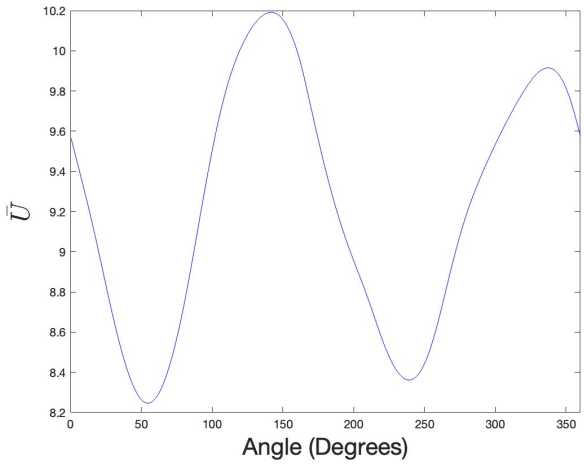

Figure 7: The potential energy for the shape "monster" with $R = 0.4436$, performed using the command `PotEnergyCalc`.

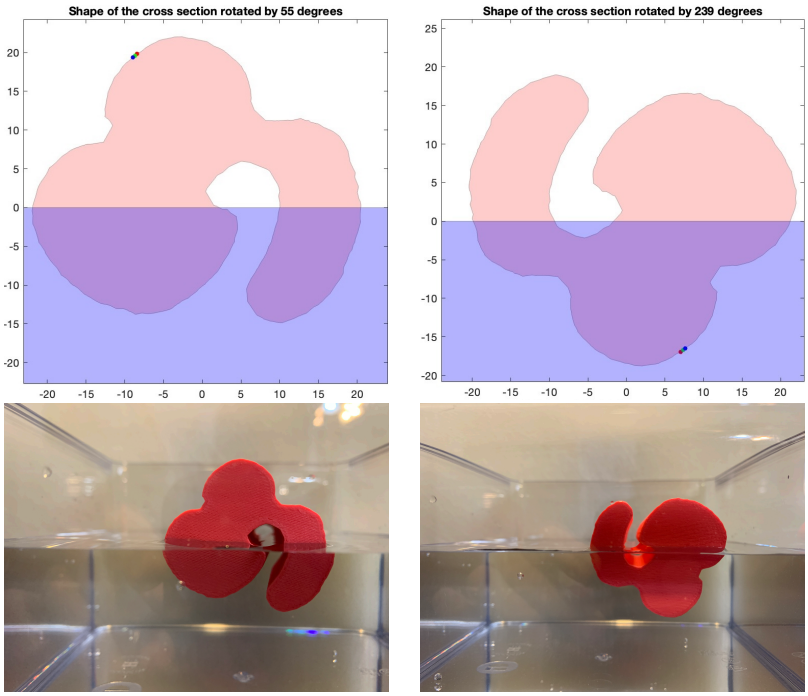

Figure 8: Top: Stable floating configurations for the shape "monster" with $R = 0.4436$ found using command `findPEmin`. Bottom: The actual stable floating configurations for the printed object.

Note: This command may take a long time, especially if your polygonal shape contains many vertices. For a quicker version, type `makethetaplot(xvalues,yvalues,10,10)`, which will give far fewer values of R and θ, but will give a rough sketch of the full picture. Now that you have seen

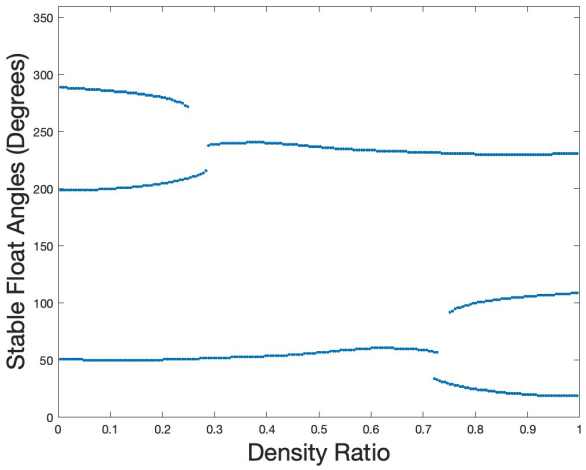

Figure 9: The stable floating configurations for the monster plotted for R versus θ using the command `makethetaplot` .

the diagram, consider reprinting your object at another density to see different, or a different number of, stable positions.

4 Beyond water and two fluid interfaces

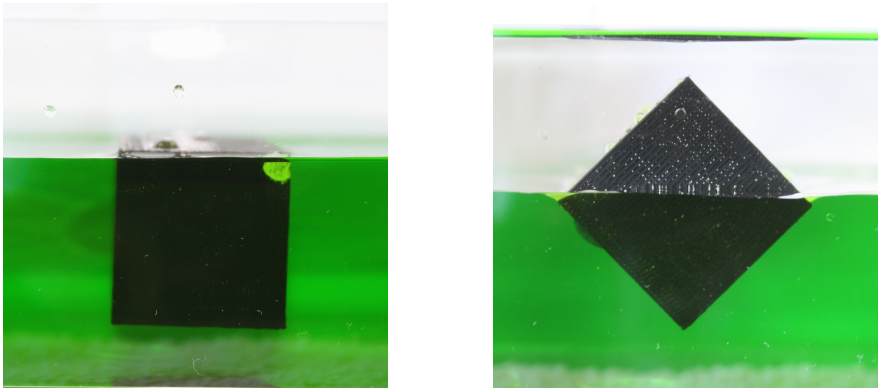

Figure 10: The same object floats in water (left), and in the two fluid interface between water and oil (right). The stable floating configurations are completely different.

The theory above does not change with floating objects in fluids other than water. What does change is the value of ϱ_{fluid}, and correspondingly the value of R. See the table of densities of different fluids and solids. This corresponds to the fact that the same object will have different stable floating configurations in different fluids.

Table 1: Densities of a variety of substances.

Substance	ϱ_{fluid} in g/cm^3
Fresh water	1
Ocean water	1.03 ± 0.03
Mineral oil	0.83 ± 0.01
Corn syrup	1.36 ± 0.03
PLA	1.15 ± 0.02
Sea ice	$0.917\pm$ unknown

As an example of floating in a different fluid, if the monster shape was printed with a density of $0.7g/cm$, then for water, $R_{\text{water}} = 0.7/1 = 0.7$. However, for mineral oil, $R_{\text{oil}} = 0.7/0.83 = 0.84$. Referring to Fig. 9, this implies that when the object floats in water, it will have two stable floating configurations. Using `findPEmin` with $R = 0.7$, we find that these occur at angles 59 and 232 degrees. However, when the object floats in oil, it will have three stable floating configurations, which we compute to occur at 22, 103, and 230 degrees – again these were found using `findPEmin` but this time with $R = 0.84$.

If an object floats in Fluid A and sinks in Fluid B, then this means that it will float in the *two fluid interface* between the two fluids. The way in which an object floats stably in water and how it floats in a two fluid interface is very different, as seen in Fig. 10. However, even in this case, it is possible to find a corresponding value of $R_{2\text{f}}$ which gives the floating configurations, as given by the following formula:

$$R_{2\text{f}} = \frac{\varrho_{\text{obj}} - \varrho_B}{\varrho_A - \varrho_B}$$

As an example, if the monster shape was printed with density $0.85g/cm$, then by Fig. 9, it will have three stable floating configurations in water, at angles 22, 104, and 230 degrees. Since its density is greater than the density of mineral oil, it will sink in oil, and

$$R_{2\text{f}} = \frac{0.9 - 0.85}{1 - 0.83} = 0.12.$$

Therefore, two see how it floats at the two fluid interface, we consult Fig. 9 with the value of 0.41 and see that it will have stable floating configurations at the very different rotations of 50, 201, and 285 degrees.

5 Bibliography

Daniel M. Anderson, Brandon G. Barreto-Rosa, Joshua D. Calvano, Lujain Nsair, and Evelyn Sander (Edited by Maria Trnkova and Andrew Yarmola), Mathematics of Floating 3D Printed Objects, *Proceedings of Symposia in Applied Mathematics (PSAPM)* , 79, American Mathematical Society, 2023. https://arxiv.org/abs/2204.08991

D.M. Anderson, P.R. Bishop, M. Brant, G. Castaneda Guzman, E. Sander, and G. Thomas, Stability of Floating Objects at a Two-Fluid Interface, *European Journal of Physics*, 45: 055001, 2024. https://iopscience.iop.org/article/10.1088/1361-6404/ad5ca8

Alex Dallas, Density of ice, *The Physics Factbook, an encyclopedia of scientific essays.* https://hypertextbook.com/facts/2000/AlexDallas.shtml

Gilbert, How things float, *Am. Math. Monthly* 98 201–16, 1991. https://www.tandfonline.com/doi/abs/10.1080/00029890.1991.11995729

R. Pawlowicz, Key Physical Variables in the Ocean: Temperature, Salinity, and Density. *Nature Education Knowledge* 4(4):13, 2013. https://www.nature.com/scitable/knowledge/library/key-physical-variables-in-the-ocean-temperature-102805293/

Acknowledgments

The research of E.S. was partially supported by the Simons Foundation under Awards 636383.