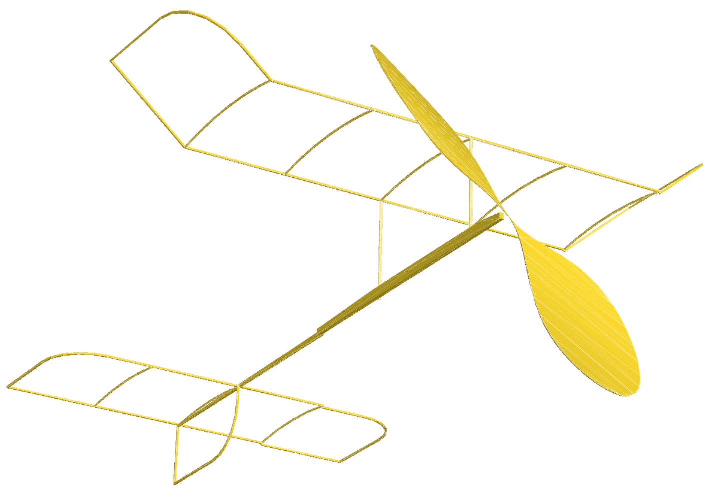
Designing an Indoor Model using OpenSCAD

Roie R. Black

January 14, 2021



Building the Wing

Let's start off this design by building the wing.

Design Constraints

The class rules specify two constraints on the design of the wing. The projected span is limited to 18 inches, nd the maximum chord is limited to five inches. Based on a survey of indoor models, we will use a flat center section with wing tips angled upward, providing the needed dihedral for stability. Here is our starting geometry:

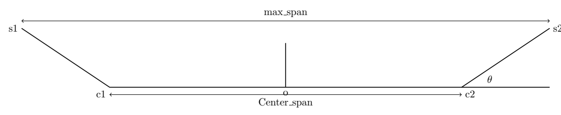


Figure 1: Basic Wing Layout

The constraint in this figure is **max_span**. We are free to choose the other dimensions as long as we respect this limit.

Circular Arc Airfoils

Many indoor model airplanes use a simple circular arc airfoil.

The rules specify limits on the chord for both the wing and stabilizer. have some specified chord for either the wing or stabilizer, and the airfoil is specified as an arc with a maximum height that is some percentage of that chord. So, we are given the chord, and thickness as a percentage value. The challenge is to figure out the radius of the arc that satisfies these values.

Arc Geometry

Since we start off with the chord and thickness specified as a percentage, we need to get rid of the percentage:

Given:

- **c** - the chord of the wing
- **T** - the camber as a percentage of **c**

$$t = Tc/100 \tag{1}$$

Here is a diagram showing what we are dealing with:

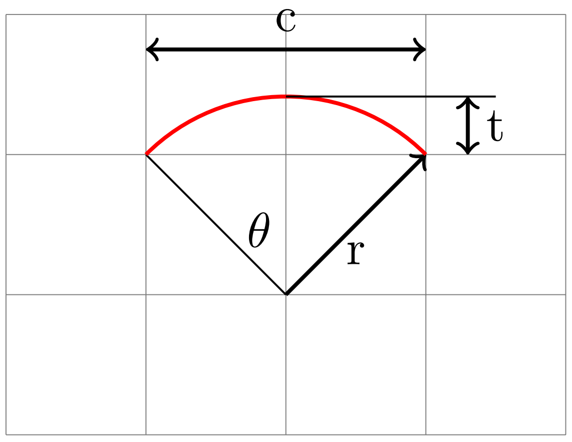


Figure 2: Circular arc Geometry

From this figure, we can write two equations:

$$r\sin(\theta) = \frac{c}{2} \tag{2}$$

$$r - r\cos(\theta) = t \tag{3}$$

The unknown variables are now:

- **r** - the radius of the circular arc
- θ - one half of the total angle swept by the arc

While we could drag out our old math books from high school, I would rather let my computer do the hard work. Time for SymPy [1]!

0.0.1 SymPy

SymPy is a Python program that knows how to do a lot of math. It manipulates *symbols*, not numbers, and that is a lot of what you should have learned in a trigonometry classes.

Here is how we get SymPy to solve our equations:

Listing 1: python/circular-arc.py

```
import sympy

r,c,t,theta = sympy.symbols('r c t theta')

eq1 = 2 * r * sympy.cos(theta) - c
eq2 = r -r * sympy.sin(theta) -t

sol = sympy.solve([eq1,eq2],[r,theta])
print(sympy.latex(sol))
```

Running this code, we find our solution:

```
r,c,t,theta = \
  sympy.symbols('r c t theta')

eq1 = 2 * r * sympy.cos(theta) - c
eq2 = r - r * sympy.sin(theta) - t
sol = sympy.solve([eq1,eq2],[r,theta])
```

Here are the solutions:

$$\left[\left(c^2/8t + t/2,\ 2\operatorname{atan}\left(\frac{c-2t}{c+2t}\right)\right)\right] \tag{4}$$

Here, I asked SymPy to solve the two equations for the two *symbols* **r** and θ. The two solutions are shown. The first one is the **radius** we need to our code. We do not really need to worry about the angle part.

We will use these result to set up an OpenSCAD module that will build our ribs.

Warning! If there are any parents reading this, do not let your kids know about SymPy. On the other hand, if you cannot solve your kids math homework, you might try SymPy on the sly! YMMV!

0.1 Ribs

The ribs for this model will be simple circular arcs. We need to specify the chord, and the camber thickness as a percentage of the chord, we also need to specify how thick the rib will be and how deep the rib will be. We will create a simple module to let the user generate as many ribs as needed, adjusting each with the needed parameters:

```
rib(
   chord=5,
   camber=6,
   thickness=1/32,
   height=1/16
);
```

When I defined the rib module, I set up default values as shown above, so to activate a standard rib, I just need to write **rib()**.

The code needed to generate a rib is a bit involved, but basically involves creating a short cylinder, cutting out the center of that cylinder leaving a tube, then slicing off the part of the ring that leaves just the rib. We then rotate it and align it for use later.

Here is an example rib, ready for use:

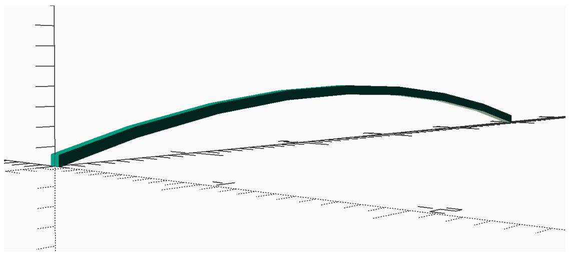


Figure 3: Basic rib

0.2 Wing Center Section

The leading and trailing edges of this model will be simple 1/16" balsa sticks. These are easy to model in OpenSCAD as really skinny cubes. The center section of the wing holds five equally spaced ribs. This code is pretty simple, so I packaged it in a simple module:

```
wing_center_section(
   chord=5, nribs= 5, span=12
);
```

Here is this code:

This code uses a loop to place the ribs. Details on all of these language constructs can be found in the *Users Manual* **??**

Figure 4Wing Center Section.

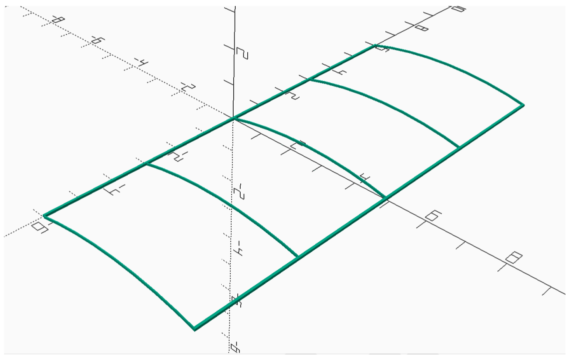


Figure 4: Wing Center Section

There is one detail in this image worth examining closer. Figure 6 shows the joint at the outer rib. This allows the tip section to mate at this joint.

Being able to zoom in on details like this is handy for making sure your design is clean.

Tip Design

The tip is another rectangular section, except I decided to round off the leading edge corner. There are no ribs in this section, except for a flat rib at the tip. Here is the geometry I decided on:

The tip module uses a simple supporting module that generates the curved section. This will be a balsa stick formed over a template. By writing the tip as

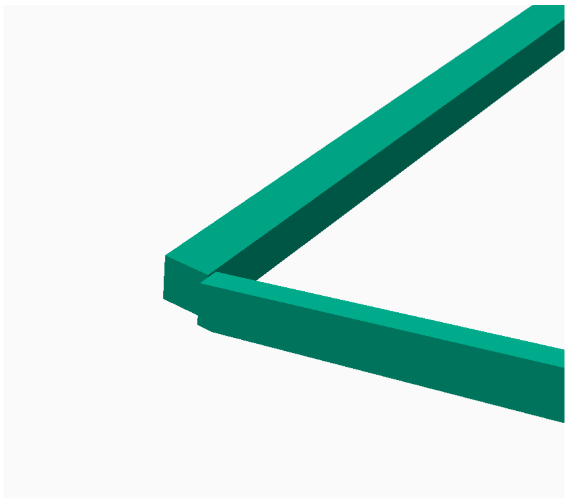


Figure 5: Tip Joint Detail

a module, I will be able to reuse this design on the stabilizer and the fin!

```
tip_section(span=3, chord=5, radius=2);
```

There is one interesting issue in the tip design. Many builders taper the leading and trailing edges so the tip is lighter. In this design, I decided to taper just the leading and trailing edge tip spars and make the circular arc and the tip thinner square stock. The puzzle is how to do this in OpenSCAD.

It turns out to be easy. Once again, we use the *difference* operation. We generate a square strip the size we want, then set up a block longer than the strip and place it at a slight angle so we end up chopping off the unwanted material leaving a tapered strip. It is easy enough to do this twice to get the result we are after. This is sanding with no mess!

Figure 7 shows the basic idea.

Figure 8 shows the final tip section.

0.3 Tip Templates

I kept this design simple. The only templates needed will be used for forming the curved segments at the

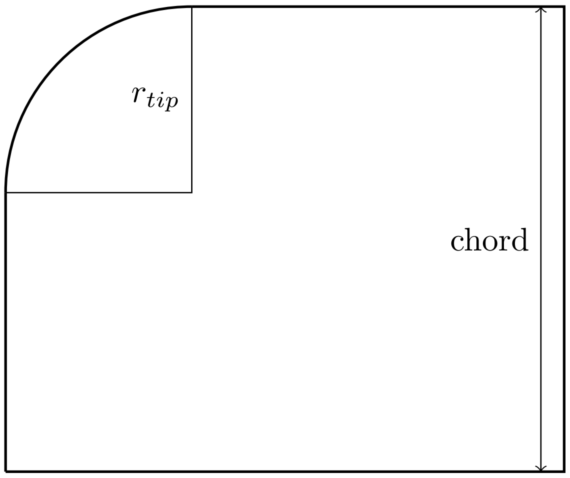


Figure 6: Tip Design

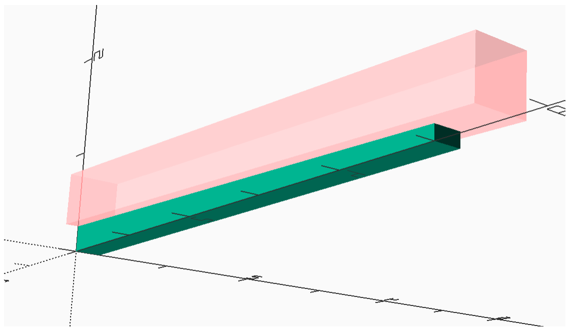


Figure 7: Trimming Block

tips. Fortunately, OpenSCAD can help generate printed templates we can use! Here is the code used.

Listing 2: demo/printer-test.scad

```
use <wing-tip.scad>

left_margin=0.5;
bottom_margin=0.5;
printer_shift = 0.5;

projection(cut=false)
 scale([25.4,25.4,1])
   translate([left_margin,bottom_margin+printer_shi
       wing_tip();
```

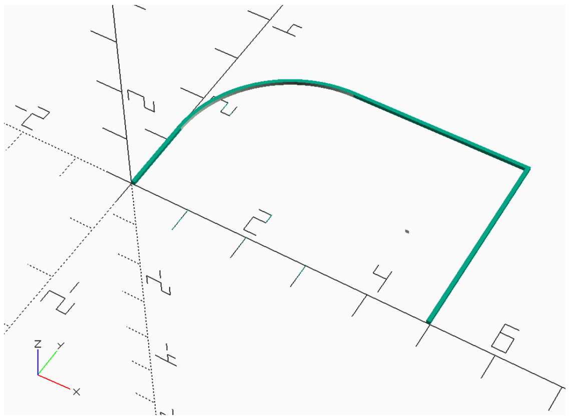


Figure 8: Wing Tip

The *scale* operation is what sized the shape so it prints accurately. Most of the rest of the code just made adjustments so the template printed properly on my printer. These might need adjusting for different printers.

Basically, we set up code that generates a tip section, then flatten it back into a 2D drawing. The operation that does this is called a *projection*. Once that is set up, we will ask OpenSCAD to export this model as an SVG file which can be printed. I use my Chrome browser to do the printing on my home printer. Figure **??** shows what the template looks like in Chrome.

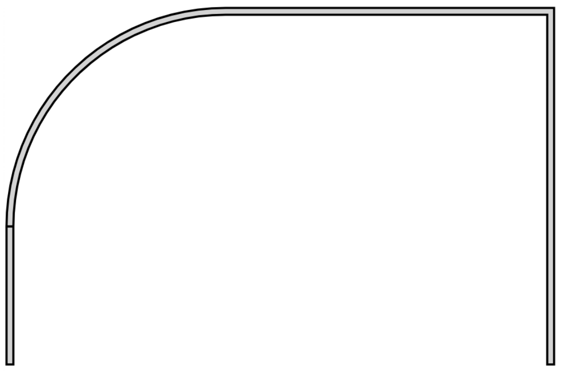


Figure 9: Tip Template

0.4 Wing Assembly

All we need to to to complete the wing is generate the center section, then generate the wing tips and rotate them into place at the proper dihedral angle. The only thing tricky here is generating the two tip sections.

Since the tip is flat, I can generate two of then, and rotate one 180 degrees so it will fit on the opposite side. I do need to do some minor translating to make sure things line up, but by now, this is getting easy!

Figurefig:wing.pngFinal Wing shows the completed wing.

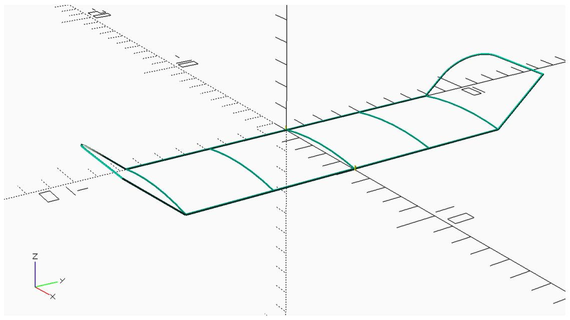


Figure 10: Wing Assembly

References

[1] S. Developers. SymPy, 2021. URL `https://www.sympy.org/en/index.html`.