Automating Crochet Patterns for Surfaces of Revolution

Megan Martinez[1] and Amanda Taylor Lipnicki[2]

[1]Mathematics Department, Ithaca College, Ithaca, NY, USA; mmartinez@ithaca.edu
[2]Math and CS Division, Alfred University, Alfred, NY, USA; tayloral@alfred.edu

Abstract

A surface of revolution is created by taking a curve in the xy-plane and rotating it about some axis. We develop a program which automatically generates crochet patterns for surfaces by revolution when they are obtained by rotating about the x-axis. In order to accomplish this, we invoke the arclength integral to determine where to take measurements for each row. In addition, a distance measure is created to optimally space increases and decreases. The result is a program that will take a function, x-bounds, crochet gauge, and a scale in order to produce a polished crochet pattern.

Introduction

Physical models of mathematical objects allow the theoretical to become tangible. This project began as a way to help students visualize the quadric surfaces by creating crochet models. As we tinkered with patterns and calculations, we realized our methods were developing an algorithm to produce crochet patterns for surfaces by revolution. We fully developed our methods into a CoCalc worksheet that produces crochet patterns for surfaces of revolution.

Surfaces of revolution in fiber arts are not new. The question of creating knitting patterns to construct surfaces of revolution was introduced by Amy F. Szczepański [2], who developed a method for giving the number of stitches that should appear in each row of a pattern. Our work is different from hers in a few ways: we use different methods for determining the number of stitches in a row and go further to produce a polished crochet pattern. Our output looks like something you would find on Ravelry.com (a popular repository of knitting and crochet patterns).

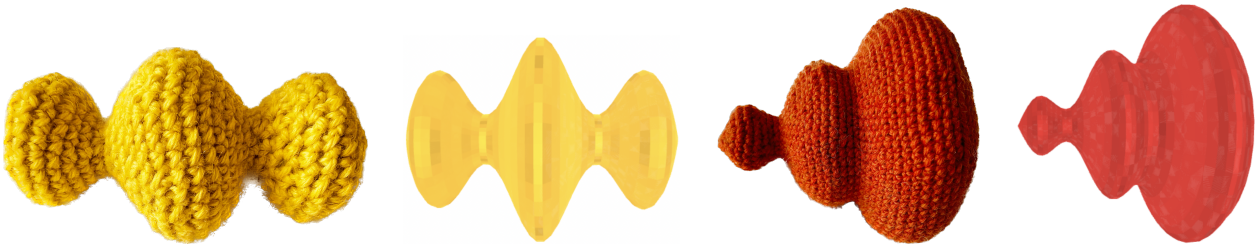
We endeavored to produce crochet instructions that would create a physical model as true to the theoretical one as possible. This includes planning where to measure the circumference of a surface to emphasize local minima and maxima, spacing increases and decreases to minimize distortion but keep instructions easy to follow, and identifying problematic behaviors for our program. While our pattern specifically creates a crochet pattern, it would be relatively straightforward to tinker with the output to read like a knitting pattern. Practically, there is not enough different between the mechanics of the two crafts to greatly affect the work we have done.

Accessing the Code

Our code is written in a Sage worksheet and freely accessed using CoCalc. The code is published in a GitHub repository [1] and can be brought into CoCalc using the second link in the provided citation. The "README.md" file has basic instructions for operating the code. If you select "Crocheting Surfaces of Revolution (v13).sagews" you will see the code, but you must click "edit" to evaluate the code and work on your own patterns. You can do this without a CoCalc account, though creating one is free.

Inputs & Crochet Terminology

Our program takes the following inputs:



(a) (b) (c) (d)

Figure 1: *Figure (a) shows a crocheted model with dimensions* $3.25" \times 2.5" \times 2.5"$ *where* $f(x) = \cos(2x) - \frac{x^2}{10} + 2$, $a = -4$, $b = 4$, $S = 20$, $R = 24$, $scale = 0.4$ *and Figure (c) shows a crocheted model with dimensions* $6" \times 5.25" \times 5.25"$ *where* $f(x) = \frac{sin(3x)}{2} + \frac{2x}{3}$, $a = 0$, $b = 5.5$, $S = 20$, $R = 24$, $scale = 0.7$. *Figures (b) and (d) show the computer models of these surfaces.*

- $f(x)$ is the function that will be rotated about the x-axis to create the surface. The function $f(x)$ should be positive on (a, b) and $f'(x)$ should be defined on $[a, b]$ in order for our code to function.
- a is the x-value determining the start of the surface
- b is the x-value determining the end of the surface.
- S is the stitch gauge; that is, the number of stitches that fit in 4"
- R is the row gauge; that is, the number of rows that fit in 4"
- $scale$ is the measure of one unit in inches (this allows one to decide how large the model will be)

We have provided two examples of crocheted outputs from our program in Figure 1.

Our goal is to output a crochet pattern, so it is helpful to measure distances using the units of "rows" or "stitches." This means our program makes use of the crocheter's individual gauge (the number of stitches and rows that fit in a 4" crocheted square). In order to achieve an accurate surface, the crocheter should make a gauge swatch using their yarn and crochet hook of choice. We have worked to minimize the techniques a crocheter needs, so all of our shapes are crocheted in-the-round using the spiral method. The experienced crocheter is encouraged to alter this by using a joining method, if desired. We use the following crochet techniques and symbols:

- Sc: Single crochet. Sc5 would mean "single crochet 5 stitches."
- Inc: Make two crochet stitches in one stitch; turns one stitch into two.
- Dec: Invisible decrease; turns two stitches into one.

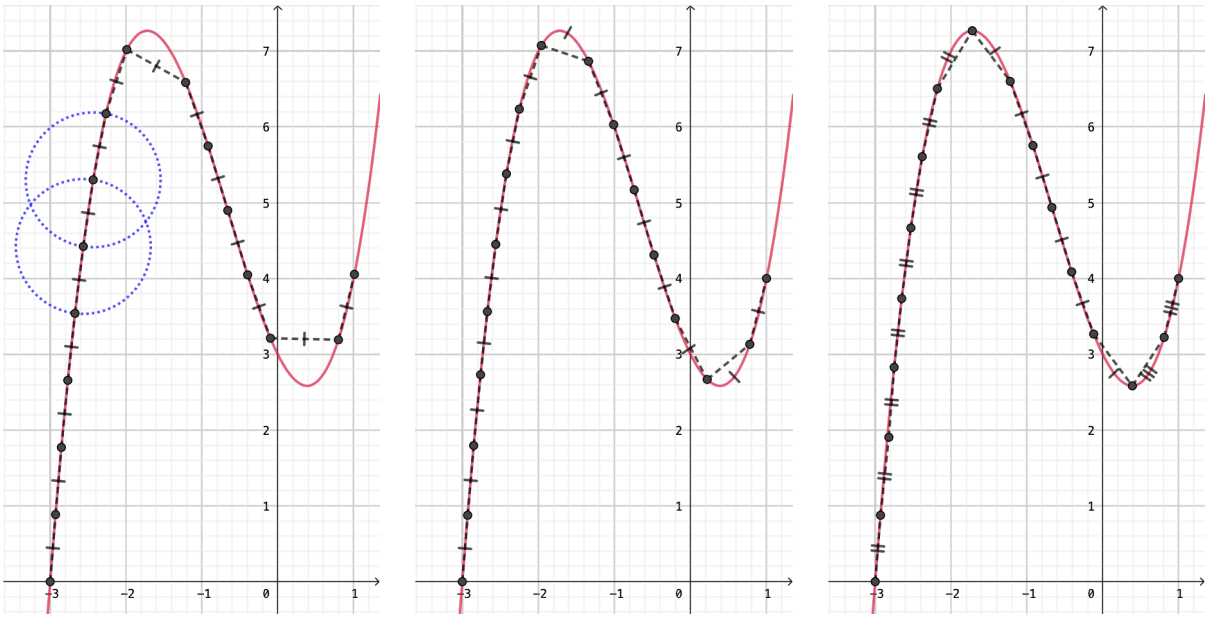
We recommend that crocheters use a locking stitch marker to mark the beginning of each row. This is helpful in knowing when you will move on to the next row of instructions.

Identifying Row Landmarks

The first order of business for our program is to identify x-values (which we call landmarks) where we will measure the circumference of our surface. These correspond to each row of the crochet pattern.

Szczepański does this by approximating the curve defining the edges of a shape by using line segments of a fixed length, h, with one endpoint at (x_i, y_i) and the other where the curve intersects $(x-x_i)^2+(y-y_i)^2 = h^2$ (an example of this is shown in Figure 2(a)). We take a different approach by using arclength calculations to place the rows. We will use this to position rows at the local extrema of a function, as we will see in the next section.

Throughout this paper, we will illustrate our program mechanisms by returning to an example with the following set of inputs: $f(x) = x^3 + 2x^2 - 2x + 4$, $a = -3$, $b = 1$, $S = 22$, $R = 25$, $scale = 0.18$.



(a) (b) (c)

Figure 2: *The function $x^3 + 2x^2 - 2x + 4$ plotted with x-landmarks that (a) use Szczepański's method of approximating the curve using circles to find equal-length secants, (b) segment the curve into pieces with equal arclength, and (c) are placed at local extrama first and then placed to segment the curve into pieces with equal arclength between the local extrema.*

We begin by finding the arclength of our curve in rows:

$$L = \frac{scale \cdot R}{4} \int_a^b \sqrt{1 + (f'(x))^2}\, dx = \frac{(0.15)(25)}{4} \int_{-3}^{1} \sqrt{1 + (3x^2 + 4x - 2)^2}\, dx \approx 16.162 \text{ rows.} \quad (1)$$

Notice $scale$ is in inches/unit, $\frac{R}{4}$ is in rows/inch and our integral is in units. We round to 16 rows and minimize the effect of that rounding by evenly placing the rows along the curve. We place an initial chain row at $x_0 = -3$, then include 16 more rows of instruction. The next row landmark, x_1, is a distance of $\frac{16.162}{16} \approx 1.010$ down the curve. In general

$$\frac{scale \cdot R}{4} \int_{x_0}^{x_i} \sqrt{1 + (f'(x))^2}\, dx = i \cdot \frac{L}{[L]},$$

where $[x]$ denotes x rounded to the nearest integer. This isn't a very "nice" equation to solve algebraically, so we use a "guess and check" protocol to find each x_i accurate to two decimal places. To mitigate the effects of rounding error, we always measure our arclength from $x_0 = a$.

In our current example, we have rows at the following x-landmarks: $x_0 = a = -3, x_1 = -2.93, x_2 = -2.84, x_3 = -2.76, x_4 = -2.67, x_5 = -2.56, x_6 = -2.42, x_7 = -2.25, x_8 = -1.96, x_9 = -1.34, x_{10} = -1.01, x_{11} = -0.74, x_{12} = -0.48, x_{13} = -0.20, x_{14} = 0.22, x_{15} = 0.78, x_{16} = b = 1.00$. Each of these values have been plotted in Figure 2(b). Notice that there are local extrema at $x = \frac{-2 \pm \sqrt{10}}{3}$, between our rows. This means that when knit or crocheted, our model will have slightly less accentuated peaks and valleys. As we increase the value of $scale$ in our program, we will work in more rows and this effect will be minimized. As we decrease $scale$, the problem worsens.

Table 1: *Calculations adjusting x-landmarks.*

i	$(cR/4)\int_{m_i}^{m_{i+1}} \sqrt{1+(3x^2+4x-2)^2}\,dx$	x-Landmarks
0	8.434	$x_0 = m_0 = -3, x_1 = -2.93, x_2 = -2.84, x_3 = -2.75, x_4 = -2.65,$ $x_5 = -2.53, x_6 = -2.38, x_7 = -2.18, x_8 = m_1 = (-2-\sqrt{10})/3$
1	5.923	$x_0 = m_1 = (-2-\sqrt{10})/3, x_1 = -1.22, x_2 = -0.92, x_3 = -0.67,$ $x_4 = -0.41, x_5 = -0.12, x_6 = m_2 = (-2+\sqrt{10})/3$
2	1.805	$x_0 = m_2 = (-2+\sqrt{10})/3, x_1 = 0.81, x_2 = m_3 = 1$

Table 2: *The number of stitches in each row of our pattern.*

Row	# Stitches	Row	# Stitches	Row	# Stitches
0	6	6	41	12	32
1	12	7	47	13	27
2	18	8	51	14	22
3	24	9	47	15	26
4	29	10	42	16	31
5	35	11	37		

Prioritizing Local Extrema

Our method of using arclengths allows us to make slight adjustments to our x-values to accentuate the local extrema of a shape. We do this by creating a list including the endpoints, a and b, and all the x-values of local extrema for your given function between a and b: $\{m_0 = a, m_1, \ldots, m_n = b\}$. Then between each m_i and m_{i+1}, we find the x-values for where to place rows as in the previous section. The number of rows is determined using the same calculation in Equation 1 with m_i and m_{i+1} as bounds.

Continuing with our example, we construct the list of extrema $\{m_0 = -3, m_1 = \frac{-2-\sqrt{10}}{3}, m_2 = \frac{-2+\sqrt{10}}{3}, m_3 = 1\}$. We then find x-landmarks between each m_i value in Table 1.

Taken together, we get a new list of x-landmarks: $-3, -2.93, -2.84, -2.75, -2.65, -2.53, -2.38, -2.18, \frac{-2-\sqrt{10}}{3}, -1.22, -0.92, -0.67, -0.41, -0.12, \frac{-2+\sqrt{10}}{3}, 0.81, 1$. The result is displayed in Figure 2(c).

Calculating the Number of Stitches in Each Row

Once the x-landmarks have been determined for the placement of each row, it is a simple matter to calculate how many stitches should be in each row. There are $s(i) = [2\pi \cdot scale \cdot \frac{S}{4} f(x_i)]$ stitches in the i-th row. In our running example, the 2nd row has $[2\pi(0.18) \cdot \frac{22}{4} f(-2.84)] = 18$ stitches. The number of stitches in each row of our example has been displayed in Table 2.

Writing the Pattern

While computing the number of stitches in each row of our pattern is a big part of pattern construction, there are many additional considerations for how to output a polished set of instructions.

Increase and Decrease Placement

The most important problem to solve is how to distribute increases and decreases in an effective, yet practical way. When increases or decreases are placed on top of each other in consecutive rows, the smoothness of the finished shape can suffer. If we have the program output instructions that are burdensome to follow, that is impractical as well. We will discuss each of these considerations and how we decided to balance the two.

Suppose we are considering Row 6 from above with $s(6) = 41$ stitches. Row 5 has $s(5) = 35$ stitches, so Row 6 will need to include 6 increase stitches. Following are some options for how those could be distributed. Note that we use the symbol * to block off a section of instructions that will be repeated a designated number of times.

Option 1: Sc35, *Inc* (6 times)

Option 2: *Sc4, Inc* (6 times), Sc5

Option 3: *Sc4, Inc* (2 times), Sc1, *Sc4, Inc* (2 times), Sc2, *Sc4, Inc* (2 times), Sc1

The first set of instructions is the easiest to crochet, but by grouping all the increase stitches together, we are creating a skew that will pull our finished surface to one side. Options 2 and 3 do a much better job of spreading the increases or decreases along the row. Option 2 is computed with a simple division calculation (which we will formalize below)—we call this the *remainder method*. For this option, we have spread out the increases a fair amount, but tacked on additional "remainder" stitches at the end of the row.

Option 3 works to additionally split up that clump of non-increase stitches at the end of the row, by distributing those as well. It is easily argued that something like Option 3 would be the best choice theoretically; however, we must always keep in mind that a real person will be following these instructions and complexity should be minimized. In our program, we opt for the remainder method in Option 2 not because it is the most even distribution of the increases or decreases, but because it makes an attempt at even distribution while maintaining the sanity of the crocheter. Our decision means there is the possibility for a small amount of leaning in our shape, but this effect will depend on how large the remainders are for a particular shape. From our experience, this effect is unnoticeable.

The other benefit to the remainder method is a easy way to minimize overlapping increases and decreases. We can think of our row as a circle (even though we are actually crocheting a spiral) and perform a cyclic shift of the positions of the increases or decreases. For instance, instead of "*Sc4, Inc* (6 times), Sc5," we could choose to make the row instructions "Sc3, *Sc4, Inc* (6 times), Sc2" or even "Sc2, Inc, *Sc4, Inc* (5 times), Sc7"; notice in this last example, we pull from some of the single crochet stitches not in the remainder. To figure out the optimal configuration of the stitches, we create a distance measure.

For each row, we can create a set of ratios that indicate where the increases and decreases fall. The row "*Sc4, Inc* (6 times), Sc5" will have the associated set of ratios: $\{5/35, 10/35, 15/35, 20/35, 25/35, 30/35\}$. The denominator in these ratios is the number of instructions in a row (not the number of stitches). For instance, "*Sc4, Inc* (6 times), Sc5" ends with 41 stitches and is working 35 stitches from the previous row, but "*Sc4, Dec* (6 times), Sc5" ends with 35 stitches and is working 41 stitches from the previous row. Both have the same set of ratios. So we use $\min(s(i-1), s(i))$ for the denominator of the ratios in row i.

Suppose D gives some possible set of positions for the increases or decreases in the i-th row. Define r_i as the function that provides the corresponding set of ratios for row i: $r_i(D) = \{x/\min(s(i-1), s(i)) \mid x \in D\}$. Note that r_i only accepts sets with integer values between 0 and $\min(s(i-1), s(i))$, inclusive. Using these ratios, we create two types of distance measures between two consecutive rows. If D (resp. E) is some set of increase/decrease positions in row $i-1$ (resp. i), then

$$d_1(r_{i-1}(D), r_i(E)) = \min\{\min(|y-x|, 1-|y-x|) \mid x \in r_{i-1}(D), y \in r_i(E)\}$$

Table 3: *An example of the distance calculations to determine the optimal placement of increases or decreases in a row.*

Instructions	$r_6(D'(6))$	$d_1(r_5(D(5)), r_6(D'(6)))$	$d_2(r_5(D(5)), r_6(D'(6)))$
Sc5, *Sc4, Inc* (6 times)	$\{10/35, 15/35, 20/35, 25/35, 30/35, 35/35\}$	0.04433	0.05665
Sc4, *Sc4, Inc* (6 times), Sc1	$\{9/35, 14/35, 19/35, 24/35, 29/35, 34/35\}$	0.01576	0.02808
Sc3, *Sc4, Inc* (6 times), Sc2	$\{8/35, 13/35, 18/35, 23/35, 28/35, 33/35\}$	0.00197	0.00739
Sc2, *Sc4, Inc* (6 times), Sc3	$\{7/35, 12/35, 17/35, 22/35, 27/35, 32/35\}$	0.01675	0.02906
Sc1, *Sc4, Inc* (6 times), Sc4	$\{6/35, 11/35, 16/35, 21/35, 26/35, 31/35\}$	0.04532	0.05764
Sc4, Inc (6 times), Sc5	$\{5/35, 10/35, 15/35, 20/35, 25/35, 30/35\}$	0.04433	0.06158
Sc3, Inc, *Sc4, Inc* (5 times), Sc5	$\{4/35, 9/35, 14/35, 19/35, 24/35, 29/35\}$	0.01576	0.04253
Sc2, Inc, *Sc4, Inc* (5 times), Sc6	$\{3/35, 8/35, 13/35, 18/35, 23/35, 28/35\}$	0.00197	0.03120
Sc1, Inc, *Sc4, Inc* (5 times), Sc7	$\{2/35, 7/35, 12/35, 17/35, 22/35, 27/35\}$	0.02167	0.04729
Inc, *Sc4, Inc* (5 times), Sc8	$\{1/35, 6/35, 11/35, 16/35, 21/35, 26/35\}$	0.05025	0.06634

$$d_2(r_{i-1}(D), r_i(E)) = \frac{1}{|D||E|} \sum_{x \in r_{i-1}(D), y \in r_i(E)} \min(|y-x|, 1-|y-x|)$$

The d_1 measure tells you the smallest pairwise distance that occurs between the ratios of the two rows, while the d_2 measure calculates the average pairwise distance between these ratios. We write $\min(|y-x|, 1-|y-x|)$ to find the distance between two particular ratios; this accounts for the fact that having one near the end of a row and the other near the beginning is physically close, even though it is not numerically. Note that d_1 is not a metric because it fails the positivity condition, but we can show that d_2 is one.

Our pattern creation program uses the following algorithm to find the optimal placement of increases and decreases for any row i where $s(i-1) \neq s(i)$. Define integers q and r such that $0 \leq r < |s(i) - s(i-1)|$ (our remainder) and $\min(s(i-1), s(i)) = q \cdot |s(i) - s(i-1)| + r$. Note that $|s(i) - s(i-1)|$ gives the number of increases or decreases needed in a row, while $\min(s(i-1), s(i))$ gives the number of instructions needed. We then construct $D(i)$, giving the positions of the increases or decreases in the ith row, as follows:

1. Set desired_positions = $\{q \cdot j + 1 \mid 0 \leq j < |s(i) - s(i-1)|\}$ (note that this will group the r remainder stitches at the end of the row). Set prevdistance = 0 and prevmean = 0.
2. For k where $1 \leq k \leq q + r$:
   (a) Set $D'(i) = \{q \cdot j + k \mid 0 \leq j < |s(i) - s(i-1)|\}$, newdistance = $d_1(r_{i-1}(D(i-1)), r_i(D'(i)))$ and newmean = $d_2(r_{i-1}(D(i-1)), r_i(D'(i)))$
   (b) If newdistance > prevdistance or (newdistance = prevdistance and newmean > prevmean), set prevdistance = newdistance, prevmean = newmean, and desired_positions = $D'(i)$
3. Return desired_positions

Our output from this process will be the positions of increases and decreases that maximizes the d_1 distance. In the case of a tie, we choose the option which has the largest d_2 distance. This is especially important in those situations where d_1 is always 0 (meaning it is impossible to avoid an increase or decrease aligning with one in the previous row); in these situations, we just try to get everything else as far apart as possible.

In our running example, the Row 5 instructions are "Sc6, Inc, *Sc3, Inc* (5 times), Sc2." and $r_5(D(5)) = \{7/29, 11/29, 15/29, 19/29, 23/29, 27/29\}$. Our program will then run through all the lines in Table 3 to find the optimal placement of increases in Row 6. The largest d_1 distance is produced with the last set of instructions, so we choose those.

***Roots at** a **or** b*

We don't use functions that have a root between a and b, since this doesn't make for a good physical model. However, we can use functions that have roots *at* a or b. Our program instructs the crocheter to either begin

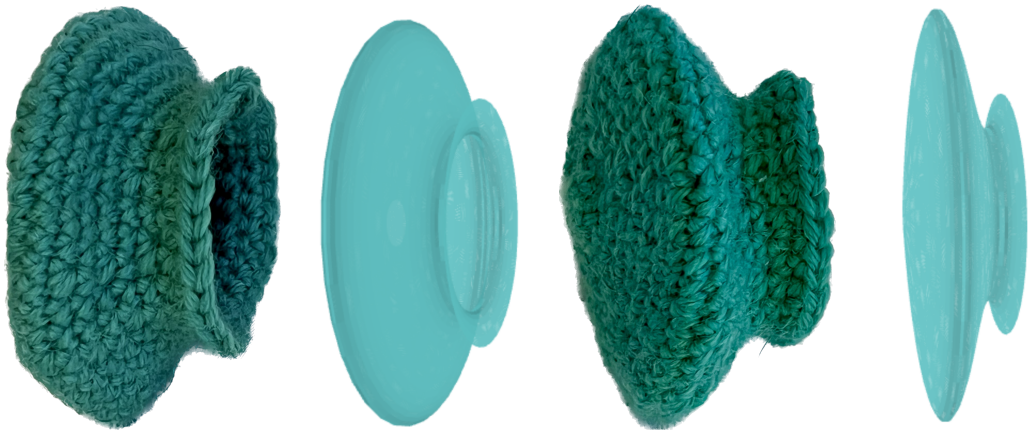
or end a closed shape when appropriate. If there is a root at both a and b, we further include instructions for stuffing the shape with fiberfill—a mathematical plushie! You can decide to turn *any* of your shapes into a plushie by crocheting a disk and using it to close an end (which we did for Figure 1(c)).

Output

Given $f(x) = x^3 + 2x^2 - 2x + 4$, $a = -3$. $b = 1$, $S = 22$, $R = 25$, $c = 0.18$, our program produces:

```
Row 0: Chain 6. join work, and Sc6.
Row 1:  *Inc* (6 times). (12 stitches)
Row 2:  Inc, *Sc1, Inc* (5 times), Sc1. (18 stitches)
Row 3:  *Sc2, Inc* (6 times). (24 stitches)
Row 4:  Sc1, Inc, *Sc3, Inc* (4 times), Sc6. (29 stitches)
Row 5:  Sc6, Inc, *Sc3, Inc* (5 times), Sc2. (35 stitches)
Row 6:  Inc, *Sc4, Inc* (5 times), Sc9. (41 stitches)
Row 7:  Sc3, Inc, *Sc5, Inc* (5 times), Sc7. (47 stitches)
Row 8:  Sc12, Inc, *Sc10, Inc* (3 times), Sc1. (51 stitches)
Row 9:  Sc6, Dec, *Sc10, Dec* (3 times), Sc7. (47 stitches)
Row 10:  Sc4, Dec, *Sc7, Dec* (4 times), Sc5. (42 stitches)
Row 11:  Dec, *Sc6, Dec* (4 times), Sc8. (37 stitches)
Row 12:  Sc3, Dec, *Sc5, Dec* (4 times), Sc4. (32 stitches)
Row 13:  Dec, *Sc4, Dec* (4 times), Sc6. (27 stitches)
Row 14:  Sc2, Dec, *Sc3, Dec* (4 times), Sc3. (22 stitches)
Row 15:  Sc6, Inc, *Sc4, Inc* (3 times). (26 stitches)
Row 16:  Sc1, Inc, *Sc4, Inc* (4 times), Sc4. (31 stitches)
Tie off
```

The result of following these instructions is in Figure 3. Note that we have crocheted a relatively small shape here; the larger the shape, the closer it will look to the idealized picture. Figure 1 has examples with larger *scale* values (Figure 1(c) is the largest shape in real life).



(a) (b) (c) (d)

Figure 3: *Figures (a) and (c) show different views of our crocheted model, using the instructions produced by our program, while (b) and (d) show our surface of revolution as pictured by CoCalc.*

Program Limitations

The decisions we made in the construction of our pattern program does lead to limitations. Ultimately, it is fairly easy to choose a set of inputs that will not lead to a good pattern. We have tried to mitigate this issue

by having the program provide feedback when inputs need to be adjusted.

It should be clear that crocheting impossibly complicated functions (such as $\sin(1/x)$ anywhere near $x = 0$) will not work. The user should use common sense in deciding what model is reasonable to make. Besides this, there are some situations that will cause errors in the program.

Steep Increases or Decreases

On occasion, there will be a change in stitch count from row to row that either more than doubles or more the halves the number of stitches, meaning using only Inc/Dec is not enough. Our program will recognize this problem and make a note of it at the top of the pattern. Most of the time, this occurs when the function gets close to the x-axis. There are a couple ways to change your inputs to avoid this issue:

- Add a positive constant to your chosen function. There is always a constant that will solve the problem, since adding such a constant will increase the number of stitches to work with, but will not change the difference in stitches between rows.
- Sometimes a change to $scale$ or multiplying your function by a constant can help. In situations where the problem is small, a small tweak can change the rounding enough for the problem to disappear.

Approximation

To create a program that works for more functions, we used "guess and check" to solve some equations to an accuracy of 0.01. In many cases, this works quite well. However, using functions with features that are closer together that 0.01 will not be noticed by our program.

When functions have a very rapid increase, there may be two x-landmarks that are exactly the same to two decimal places, or end up looking unevenly spaced because of the rounding. Since we always measure our arclength from the beginning of the shape, this kind of problem gets corrected and we rely on the forgiving nature of crochet to make a quality shape.

What More?

There are so many mathematical objects that can be crocheted. What other classes of mathematical objects might be amenable to computer generation? Our program is capable of producing a pattern for every quadric surface, besides the hyperbolic paraboloid. While we have devised a pattern for this surface [3], we ask: is there a way to automate pattern creation for surfaces that don't have circular cross-sections?

At a number of crossroads in the coding of this program, we made subjective decisions on how to proceed. Our aim has been to make a program that is broadly applicable, reasonably fast, and produces a pattern that would be enjoyable to crochet. We invite interested parties to make changes as they see fit to highlight the considerations they most care about. And of course, we invite crocheters to share their mathematical creations that arise from this program!

References

[1] M. Martinez, A. Taylor Lipnicki. *Crocheting Surfaces of Revolution*, 2023. https://github.com/meganmartinez/math_crochet, https://cocalc.com/github/meganmartinez/math_crochet

[2] A. Szczepański. "Knit Knit Revolution." In *Crafting by Concepts*, edited by S. Belcastro and C. Yackel, AK Peters, 2011.

[3] A. Taylor Lipnicki. *Little Quadric Lights*, 2022. Wool yarn, copper wire, and string lights. Bridges 2022, Aalto. http://gallery.bridgesmathart.org/exhibitions/2022-bridges-conference/lady_asphodel