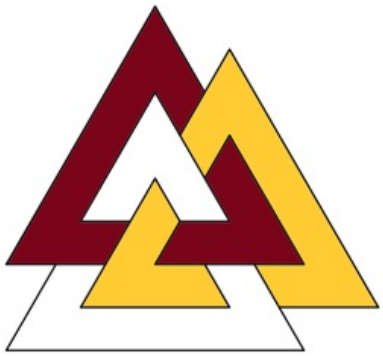
The Minnesota Journal of Undergraduate Mathematics

3D Printing Solid Möbius Surfaces

Travis C. Wert[1] and Danielle A. Brake[2]

[1]University of Notre Dame
[2]University of Wisconsin - Eau Claire



The Minnesota Journal of Undergraduate Mathematics

Volume 5 (2019-2020 Academic Year)

Sponsored by
School of Mathematics
University of Minnesota
Minneapolis, MN 55455

The Minnesota Journal of Undergraduate Mathematics

Volume 5 (2019-2020 Academic Year)

3D Printing Solid Möbius Surfaces

Travis C. Wert* [1] and Danielle A. Brake[2]

[1]University of Notre Dame
[2]University of Wisconsin - Eau Claire

Abstract. Möbius strips are parameterized explicitly by two variables, and have no thickness. However, surfaces with no thickness cannot be 3D-printed without additional post-processing to the discretization. Hence, we want equations for a naturally printable algebraic approximation of a Möbius strip that has thickness, referred to throughout as a solid Möbius surface. In this paper, we (re-)derive these algebraic equations, demonstrate Matlab code generating solid Möbius surfaces with an arbitrary number of twists, and use Numerical Algebraic Geometry to compute a smoothed numerical cellular decomposition of the objects. We conclude with 3D-printed results.

1. Introduction

Möbius strips have intrigued mathematicians, engineers, and the general public since being discovered by August Ferdinand Möbius in 1858. Beyond their mathematical beauty, Möbius strips have been used in consumer applications such as "infinity scarves".

Our primary goal in this work is to 3D print variations of Möbius strips, as artwork and as instructional aids. There are many ways to prepare a model for 3D printing. Programs such as `Blender` [7] or `OpenSCAD` [8] allow a modeler to input mathematical functions. However, the typical ways we see Möbius strips presented in mathematics is as twisted strips of "paper", with no thickness or interior volume. Hence, they are unprintable. One solution to this problem is to 'solidify' the surface, using the so-named modifier in `Blender`, but their current implementation struggles with both numerical data and non-orientable objects. `OpenSCAD`, a program for procedural boolean geometry, has Computational Geometry Algorithms Library (CGAL) [6] underlying its routines, but both `OpenSCAD` and `CGAL` struggle with numerically-generated data. Therefore, what we would really like is a mathematical object that is natively printable with minimal post-computation processing. That is, we want Möbius strips that have positive interior volume, and no singularities (e.g., self-crossings or cusps). We define a solid approximation of the Möbius strip as a *solid Möbius surface*.

A model to be 3D printed, which is fed to a type of program called a slicer, is commonly defined via faces and vertices, in the form of a triangulation, often in an `.stl` file. There

* *Corresponding author*

are many ways to compute a triangulation of a mathematical surface given implicitly in three variables, including isosurfaces [13] and subdivision methods [11]. We chose to use a tool from Numerical Algebraic Geometry called `Bertini_real` [4]. It uses the method of *homotopy continuation* implemented in `Bertini` [2] to repeatedly compute points on an N-dimensional algebraic variety, and connect them into a *numerical cellular decomposition* [3] – that is, a set of connections numerically built from points to form simplices of appropriate dimension. This paper is not about this software or algorithm, but rather an application of the decomposition software to a particular family of algebraic surfaces, so we leave most supporting details to the literature.

We have three primary tasks before us. First, we need to (re-)derive equations for algebraic surfaces approximating Möbius strips of arbitrary twist – not just the canonical 1-twist, but as many twists as one desires. Since 3D printing a model requires a well-defined interior based on outward facing normal vectors, we further require that such that a computed model can be 3D printed without need for post-computation "solidification" to overcome the problem of a surface with no interior. Second, we need to transform the surfaces into a 3D-printable digital format. Third, we shall print the solid Möbius surfaces.

We begin in Section 2 with a discussion of the process of making a surface into a 3D printed object. Section 3 computes the equations for the t-twisted solid Möbius surface, reproducing, amending, and extending earlier work. Numerical decomposition to compute 3D models is described in Section 4. Finally, in Section 5, we show our plastic results and conclude in Section 6.

2. Overview of printing a surface

Here we present a brief summary of what needs to be done to move from a surface to a 3D printed object. This is not by any means exhaustive, and a full explanation of all the techniques for doing this are beyond the scope of this article. We will briefly discuss the specific printer technologies we utilized to make our prints in Section 5.

The basic components of a 3D printing setup include a printer, materials, and tools; a software tool called a "slicer" which transforms models into machine instructions; and a digital model of the object which is to be printed [10]. Hence, a rather general procedure to print a mathematical object might be to

(1) obtain an equation or procedure to generate the model,

(2) generate the model – usually a triangulation, and

(3) slice and print.

However, slicers are particular about properties of the model, so that the instructions they generate are correct. Naturally, behavior varies between specific pieces of software in how they deal with model defects.

The primary requirement, and that of most interest here, is that a model must well-define an interior. That is, it must be water-tight, well-oriented, and define a non-empty volume. The water-tightness requirement implies that there are no holes – 'missing' triangles

or faces in the model. Well-orientedness means that the normal vector for each triangle points out from the interior, toward where there should be no material in the printed object. And non-zero volume requires that there is actually a place to put material. For example, a triangulated sphere with all normals pointing away from the origin and no holes is printable. But, if a triangle is missing or one triangle's normal vector points inside to the origin, the slicer may generate incorrect machine instructions.

One of the main problems with mathematical surfaces, as opposed to models coming from engineering, for example, is that they often fail to well-define an interior. While a sphere is trivial to make into a printable digital model, a plane is not. The problem is that a plane fails to define an inside and out. Similarly, a Möbius strip fails to define an interior because it does not partition $\mathbb{R}^3$ into two disjoint subspaces, and furthermore is unorientable, causing another set of problems for the slicer. In order to overcome this barrier to printability, the maker must sacrifice mathematical correctness.

One tactic for making an object such as a Möbius strip printable is to "solidify" it. Perhaps one could take the Minkowski sum of the surface and a ball. The ball would add a smooth solidification, as it is "added" to every point of the strip, so that there is a smooth layer of connected spheres where we previously had only the strip. The Minkowski sum is implemented in OpenSCAD, for example. Or, one could make additional triangles which are offset from the original in the direction of the normal vectors, as in the Blender solidify surface modifier. Both methods offer challenges, in terms of software and problems arising from certain model phenomena, such as self-intersections.

We would like to be able to print something like a Möbius strip without the need for solidification. The model we want is immediately printable. Hence, we approximate the strip by a solid of revolution aesthetically similar to the original object, and call the resulting objects "solid Möbius surfaces".

3. Equations

In this section, we derive equations for solid Möbius surfaces of arbitrary twist. Our derivation of the k-**twisted solid Möbius surface** mirrors that of Stephan Klaus [9]. Rather than repeat all the details, we refer to Figure 1, explain some notation, and proceed to deriving the equations.

To create a Möbius strip, one would revolve an offset line segment about the z-axis while rotating it about its local origin some number of times, with $1/2$ of a rotation generating the classic 1-twisted strip. As discussed above, this generates a 3D model that is unprintable without additional solidification to give it a well-defined interior.

In contrast, to create a solid Möbius surface with positive interior volume, we use an offset ellipse instead of a line segment. Revolving an ellipse around the z-axis creates a flattened torus; rotating the ellipse around its center as it revolves induces twist into the generated surface. As depicted in Figure 1, let ϕ represent the angle of revolution of the ellipse about the z-axis in the global fixed xy-plane. Angle ψ represents rotation in the local moving tz-plane, and k represent the (integral) number of twists in the Möbius,

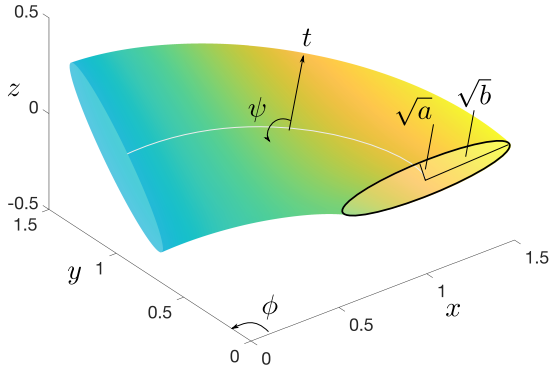


FIGURE 1. Coordinate system for deriving equations for k-twisted solid Möbius surface, which is essentially a twisted solid of revolution. A quarter of the surface is shown.

where the coupling of rotation and revolution comes from the equation

$$\psi = \frac{k}{2}\phi. \tag{1}$$

The ellipse has semi-axis lengths $\sqrt{a}$ and $\sqrt{b}$, with $0 < a, b < 1$. Note that if $a = b = 1$, then the solid of revolution is merely a standard torus; however if $a > 1$ or $b > 1$, then the surface will self-intersect.

When the ellipse is rotated in the moving tz-plane by angle ψ, it has equation

$$c^2(a(t-1)^2) + bz^2) + 2cs(a-b)(t-1)z + s^2(b(t-1)^2 + az^2) = ab, \tag{2}$$

where $C = \cos(\psi)$, $S = \sin(\psi)$, $x = t\cos(\phi)$, $y = t\sin(\phi)$, and $t^2 = x^2 + y^2$. The derivation for all k-twisted surfaces starts from (2), essentially by computing expressions for s^2, c^2, and $2cs$ in terms of C and S.

3.1. **Derivation of the 1-Twisted Solid Möbius Surface.** Given that the "standard" Möbius strip has one twist, we start our derivations there. To find the equation for the 1-Twisted solid Möbius surface, we set $k = 1 \Rightarrow \psi = \frac{\phi}{2}$. Due to the double-angle identities, we see that

$$\begin{aligned} C &= \cos(\psi) \\ &= \cos\left(2 \cdot \frac{\phi}{2}\right) \\ &= \cos^2\left(\frac{\phi}{2}\right) - \sin^2\left(\frac{\phi}{2}\right) \\ &= c^2 - s^2 \end{aligned} \qquad \begin{aligned} S &= \sin(\psi) \\ &= \sin\left(2 \cdot \frac{\phi}{2}\right) \\ &= 2\sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\phi}{2}\right) \\ &= 2sc \end{aligned}$$

where $c = \cos\left(\frac{\phi}{2}\right)$ and $s = \sin\left(\frac{\phi}{2}\right)$.

Given that $c^2 + s^2 = 1$ we have

$$\begin{aligned} C &= c^2 - s^2 \\ &= (1 - s^2) - s^2 \\ &= 1 - 2s^2 \\ s^2 &= \frac{1}{2}(1 - C) \end{aligned} \qquad \begin{aligned} C &= c^2 - s^2 \\ &= c^2 - (1 - c^2) \\ &= 2c^2 - 1 \\ c^2 &= \frac{1}{2}(1 + C) \end{aligned}$$

Using the above equations with (2), $C = \frac{x}{t}$, and $S = \frac{y}{t}$, we arrive at

$$\begin{aligned} c^2(a(t-1)^2) + bz^2) + 2cs(a-b)(t-1)z + s^2(b(t-1)^2 + az^2) &= ab \\ \frac{1}{2}(1+C)(a(t-1)^2) + bz^2) + S(a-b)(t-1)z + \frac{1}{2}(1-C)(b(t-1)^2 + az^2) &= ab \\ \frac{1}{2}\left(1 + \frac{x}{t}\right)(a(t-1)^2) + bz^2) + \frac{y}{t}(a-b)(t-1)z + \frac{1}{2}\left(1 - \frac{x}{t}\right)(b(t-1)^2 + az^2) &= ab \end{aligned}$$

Then multiply each side by $2t$ to cancel the denominator: [1]

$$(t+x)(a(t-1)^2) + bz^2) + 2y(a-b)(t-1)z + (t-x)(b(t-1)^2 + az^2) = 2abt. \tag{3}$$

The following three steps rearrange the equation so that even powers of t are on the left side and odd powers of t are on the right side. First, expand

$$(t+x)(at^2 - 2at + a + bz^2) + 2y(a-b)(tz - z) + (t-x)(bt^2 - 2bt + b + az^2) = 2abt.$$

Then, put the even on the left and odd on the right,

$$\begin{aligned} &(axt^2 - 2at^2 + ax + bxz^2) - 2y(a-b)(z) - (bxt^2 + 2bt^2 + bx + axz^2) \\ &\quad = 2abt + (-at^3 + 2atx - at - btz^2) - 2y(a-b)(tz) - (bt^3 + 2btx + bt + atz^2), \end{aligned}$$

and simplify by factoring to get

$$(a-b)(x(t^2 - z^2 + 1) - 2yx) - (2a + 2b)t^2 = -t((a+b)(t^2 + z^2 + 1) + 2(a-b)(yz - x) - 2ab).$$

Finally, square both sides and substitute $t^2 = x^2 + y^2$ to arrive at the equation for the **1-twisted solid Möbius surface**:

$$\begin{aligned} &\Big((a-b)(x(x^2 + y^2 - z^2 + 1) - 2yz) - (2a + 2b)(x^2 + y^2)\Big)^2 \\ &\quad - (x^2 + y^2)((a+b)(x^2 + y^2 + z^2 + 1) + 2(a-b)(yz - x) - 2ab)^2 = 0 \end{aligned} \tag{4}$$

[1] Equation (3) differs from the one presented by Klaus, where the right-hand side was $2a$.

3.2. **Derivation of the 2-Twisted Solid Möbius Surface.** Now the number of twists $k = 2$, and (1) becomes simply $\psi = \phi$, so substituting $c = \frac{x}{t}$ and $s = \frac{y}{t}$ into (2), we get

$$\frac{x^2}{t^2}(a(t-1)^2) + bz^2) + 2\frac{xy}{t^2}(a-b)(t-1)z + \frac{y^2}{t^2}(b(t-1)^2 + az^2) = ab.$$

Next, multiply both sides by t^2 to clear denominators

$$x^2(a(t-1)^2 + bz^2) + 2xy(a-b)(t-1)z + y^2(b(t-1)^2 + az^2) = abt^2.$$

Rearrange as we did in Section 3.1 by odd and even powers of t,

$$(t^2+1)(ax^2+by^2) + z^2(bx^2+ay^2) - 2xyz(a-b) - abt^2 = 2t(ax^2+by^2-xyz(a-b)).$$

Square the above equation and insert $t^2 = x^2 + y^2$ to arrive at the polynomial equation for the **2-twisted solid Möbius surface**.[2]

$$\begin{aligned}&\Big((x^2+y^2+1)(ax^2+by^2) + z^2(bx^2+ay^2) - 2(a-b)xyz - ab(x^2+y^2)\Big)^2 \\ &- 4\Big(x^2+y^2)(ax^2+by^2-xyz(a-b)\Big)^2 = 0\end{aligned} \tag{5}$$

3.3. **Derivation of the 3-Twisted Solid Möbius Surface.** To determine the equation for the 3-Twisted solid Möbius, we first set $\psi = \frac{3}{2}\phi$. Due to the double angle and triple angle identities, we have

$$C^3 - 3CS^2 = c^2 - s^2 \qquad \text{and} \qquad 3C^2S - S^3 = 2cs$$

Next, isolate c^2, s^2, and cs. Then substitute those variables into (2), multiply by $2t^3$ to clear the denominator, separate odd and even powers, and insert $x^2 + y^2 = t^2$ to arrive at the equation for **the 3-twisted solid Möbius surface**:

$$\begin{aligned}&\Big(-2(a+b)(x^2+y^2)^2 + (a-b)\Big((x^3-3xy^2+1-z^2) - 2z(3x^2y-y^3)\Big)\Big)^2 \\ &- (x^2+y^2)\Big((a+b)(x^2+y^2)(x^2+y^2+1+z^2) \\ &- 2(a-b)(x^3-3xy^2-z(3x^2y-y^3)) - 2ab(x^2+y^2)\Big) = 0\end{aligned} \tag{6}$$

[2]This equation differs from the one presented by Klaus, where they replaced the $(a-b)$ term with $(a+b)$ in what appears to be a simple transcription error.

3.4. **Derivation of the k-Twisted Solid Möbius Surface.** In the MATLAB code found in Appendix B, we begin the derivation of the k-twisted solid Möbius surface by using De Moivre's Formula to compute the corresponding algebraic terms for c^2, s^2, and $2cs$. We then substitute those statements as in previous sections, and multiply by $2t^k$ to cancel terms. Finally, we collect the even and odd terms, squaring both sides to determine the final equation. Due to limitations in MATLAB symbolic processing, the resulting equations are much more verbose than they need to be, but mathematically correct.

A general equation for the k-twisted solid Möbius surface is

$$\begin{aligned}
0 =&(x^2+y^2)^k\Big(a+b-2ab+(a-b)\mathfrak{Re}(((x+yi)/(x^2+y^2)^{1/2})^k)\\
&+(a+b)(x^2+y^2+z^2)(bz^2-az^2)\mathfrak{Re}(((x+yi)/(x^2+y^2)^{1/2})^k)\\
&+(a-b)\mathfrak{Re}(((x+yi)/(x^2+y^2)^{1/2})^k)(x^2+y^2)\\
&+(2bz-2az)\mathfrak{Im}(((x+yi)/(x^2+y^2)^{1/2})^k)\Big)^2-(x^2+y^2)\Big((2a+2b)(x^2+y^2)^{k/2}\\
&+2(a-b)\mathfrak{Re}((x/(x^2+y^2)^{1/2}+(yi)/(x^2+y^2)^{1/2})^k)(x^2+y^2)^{k/2}\\
&+2z(b-a)\mathfrak{Im}((x/(x^2+y^2)^{1/2}+(yi)/(x^2+y^2)^{1/2})^k)(x^2+y^2)^{k/2}\Big)^2
\end{aligned}$$

where $\mathfrak{Re}$ and $\mathfrak{Im}$ are the real and imaginary operators.

4. Decomposition

To print an object, we need a triangulation to feed to the slicer program, to generate machine instructions for the printer. To accomplish this goal, we used `Bertini_real` [4], a piece of software for real algebraic curves and surfaces, to compute models of the surfaces we wanted to print.

4.1. **Method.** A decomposition of a real algebraic set is a union of simplices such that the entire object is represented, using simplices of dimension equal to that of the decomposed the object. The surface decomposition algorithm we used in `Bertini_real` is essentially an implementation of the implicit function theorem (IFT) for a 2-dimensional object embedded in $\mathbb{R}^N$. Plainly spoken, the IFT tells us that we can use d parameters to parameterize a d-dimensional object, regardless of ambient dimension N, and off a set where the parameterization fails.

Rather than recount the full "numerical real cellular decomposition" algorithm here, we instead briefly describe the high-level procedure. For fuller details, see the original surface algorithm paper [3], or any of the papers on the implementation in `Bertini_real` [4, 5, 1]. The numerical path tracking routines coming from Numerical Algebraic Geometry are implemented in `Bertini` [2], the description of which is beyond the scope of this paper.

Since surfaces are 2-dimensional, we use two random orthogonal real linear projections $\pi_{1,2}(x) : \mathbb{R}^N \to \mathbb{R}$ as the parameters for the IFT. The first step is to compute the set where the IFT would fail – that set where our parameterization would fail the "vertical line test". This set is called the "critical curve", and consists of points and curves on the surface

where the number of points that project to a single value (π_1, π_2) changes. Secondly, smooth points in the centers of the faces and boundaries of the surface are computed via a sequence of surface-slicing curve decompositions. Finally, these smooth points are connected to each other to complete the decomposition of the surface. See Figure 2 for an illustration of a raw decomposition of the 1-twisted solid Möbius surface.

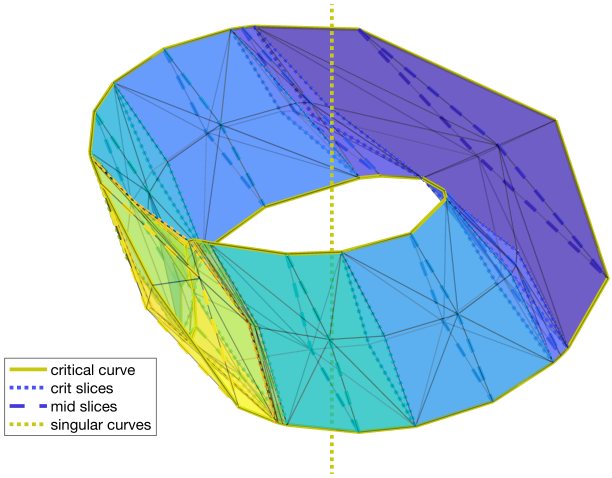


FIGURE 2. A numerical real cellular decomposition of the 1-Twisted solid Möbius surface, before sampling. The output is naturally a triangulation.

The end product is a set of points on the surface, and topologically correct connections between them. Taken as a static result, the decomposition produces a triangulation. However, the output from a decomposition with `Bertini_real` is more than a triangulation. It also contains information necessary to add more points, to refine the surface arbitrarily. Again, this is a high-level description, and we leave the many algorithmic and implementation details to the literature, particularly [4].

We below provide images of the decompositions of our surfaces alongside those generated from SURFER. `Bertini_real` differs significantly from SURFER. Whereas SURFER generates images of algebraic surfaces in three variables on the fly as the user views the object, `Bertini_real` computes a simplicial complex of 0-, 1-, and 2-cells in a single decomposition, after which the data is static (and available for refinement). SURFER is specialized software for viewing surfaces in 3D, whereas `Bertini_real` decomposes surfaces in $\mathbb{R}^N$. SURFER is very fast but gives only images, whereas `Bertini_real` is slower but gives a full N-dimensional refinable triangulation of the surface. That is, SURFER cannot be used to generate models to 3D print, but is an excellent preview tool for ensuring the surface you are working with is correct, before spending the time to perform a full decomposition with `Bertini_real`.

In our first attempts, decomposition of even the 1-twisted solid Möbius surface proved tricky because of a singular curve – that is, a curve of points satisfying the equation for the surface but on which the Jacobian matrix is singular. Notably, such a curve appears in all of the solid Möbius surfaces regardless of parameter values as the z-axis; that for

the 1-twist is depicted in Figure 2. To combat this, we added a new runtime option in `Bertini_real`, `-ignoresing`, allowing the user to ignore any singular curves.[3]

4.2. **Application.** Here we present visual results for the decomposition of four solid Möbius surfaces. An example input file, namely that for the 4-twist, can be found in Appendix A. The input file consists of settings for the tracker and decomposition software, and the system of equations. We note that a variety of tolerances had to be tightened to arrive at the final result, particularly for the 3- and 4-twist. The settings are documented in [2].

Bertini setting name	value
randomseed	9
maxstepsbeforenewton	0
maxnewtonits	1
functiontolerance	1e-8
sharpendigits	30
endgamenum	2
numsamplepoints	10
endgamebdry	0.01
targettolmultiplier	10
securitylevel	1
condnumthreshold	1e300

TABLE 1. Common settings for the `Bertini_real` decompositions of the 1, 2, 3, and 4-twisted solid Möbius surfaces

Our settings were as follows, with exceptions for tracking tolerances and the ODE predictor. We summarize them in Table 1, and describe the most important ones below.

- The Cauchy endgame was used with 10 sample points (for an 11-gon); this means that the method for computing singular points uses the Cauchy integral formula, using an 11-sided regular polygon as a discretization for the curve of integration.
- In the path tracker, the "endgame boundary time" at which the tracker enters the special tracking mode called the endgame, was set to be 0.01.
- We allowed only a single Newton iteration during path tracking.
- We also prevented truncation of paths which appear to be diverging to infinity.

Sampling for each of the four surfaces we present below used the distance-adaptive method. In this mode, when the sampler computes additional points on the surface to make it appear smoother, it first estimates the next point. Then, it computes the actual point, and measures the difference between them – that is, the error. It does this until the error is less than some user-defined tolerance, or a maximum number of iterations is reached. We used a distance tolerance of 0.01, and a maximum number of sampling iterations of 7.

[3]Ignoring singular curves in general is a terrible idea. But, for the solid Möbius surfaces, the singular curves are naked – they do not contact the rest of the surface – and so are better ignored.

The sampler works by adding these additional points on curves on the faces called "ribs", which are constant when projected through π_1, and then triangulating adjacent ribs. The minimum and maximum number of sampled face ribs we allowed was between 7 and 20.

For each of the equations we used in the paper, we set $a = 0.01$ and $b = 0.23$, as this achieved nice-looking results in SURFER visualizations similar to that of Klause [9].

4.2.1. *1-Twisted Solid Möbius.* The 1-twist surface was decomposed using mostly default settings for `Bertini` and `Bertini_real`. Decomposition took about 187 seconds on 24 processors at the University of Notre Dame's Center for Research Computing (CRC) cluster computer. Tracking tolerances were `Bertini` defaults: `1e-5` and `1e-6` for pre- and in-endgame regions. The sampled surface has 305k vertices and 916k faces.

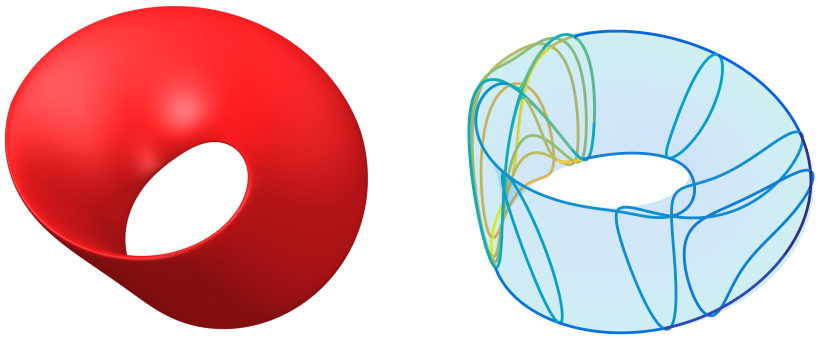


FIGURE 3. 1-Twisted solid Möbius surface. Left: SURFER image. Right: `Bertini_real` decomposition.

4.2.2. *2-Twisted Solid Möbius.* The 2-twist used the same settings as the 1-twist. Decomposition took 1341 seconds on 4 processors. The sampled surface has 366k vertices and 1.10M faces.

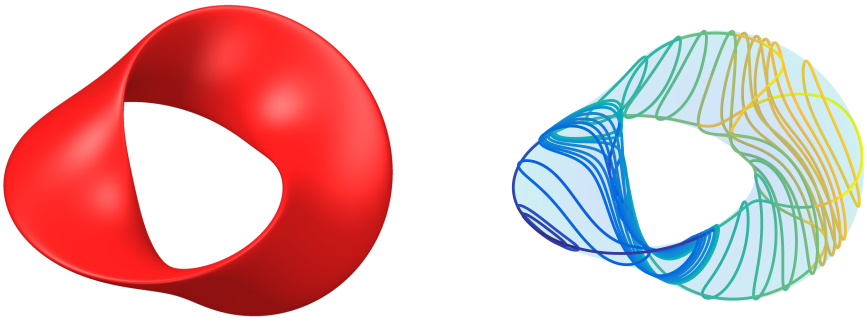


FIGURE 4. 2-Twisted Möbius surface. Left: SURFER image. Right: `Bertini_real` decomposition.

4.2.3. *3-Twisted Solid Möbius.* This decomposition used tracking settings tighter than default. The tolerance before the endgame boundary was 1e-9, and during the endgame 1e-10. The ODE predictor we used was a seventh-order Runge-Kutta-Verner method as implemented in `Bertini`; using higher-order predictors helps the decomposition algorithm run faster and more correctly. Computation time was 20834 seconds on 24 processors. The sampled surface has 408k vertices and 1.23M faces.

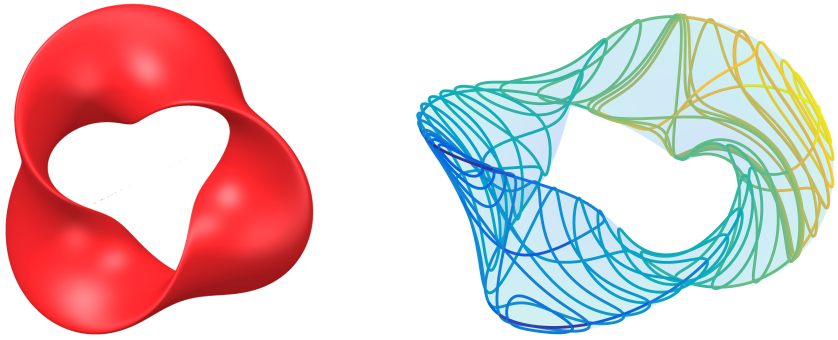


FIGURE 5. 3-Twisted solid Möbius surface. We can see hints of the singular line in the SURFER image (left), which was omitted from computation during decomposition with `Bertini_real` (right).

4.2.4. *4-Twisted Solid Möbius.* This decomposition used the same settings as the 3-twist. Decomposition with `Bertini_real` took 30535 seconds on 24 processors. The sampled surface has 572k vertices and 1.72M faces.

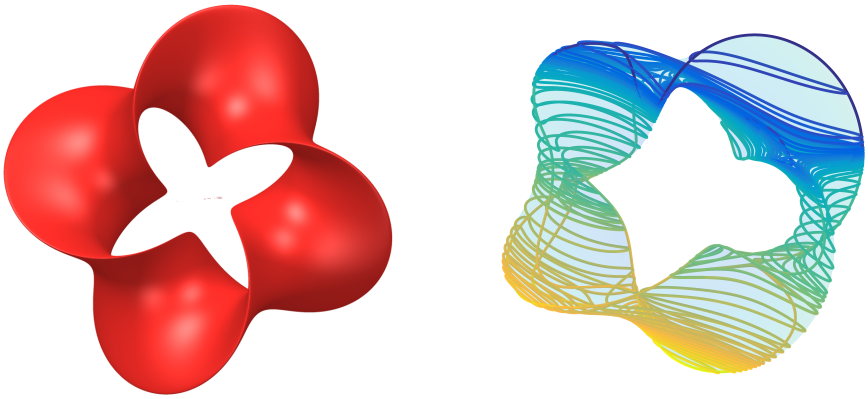


FIGURE 6. 4-Twisted solid Möbius surface. Left: SURFER image. Right: `Bertini_real` decomposition. As with the 3-twist, the barest hint of the singular line appears near the origin in the SURFER image.

5. Printing

The 3D printer we used was a Lulzbot TAZ4, with a 0.5mm extruder tip and stock hot-end. The filament is 3mm PLA from Push Plastic. We extruded at 200C, with a bed temperature of 60C. Bed treatment consisted of PEI sheeting with ABS juice for adhesion.

For mathematical objects like these surfaces, support tends to be the most challenging part of the printing process. We used manually placed support pillars in the closed-source slicing software `Simplify3D` [12] version 3.1.1. Support density was 20%, horizontal offset of 0.15mm, and a single vertical separation layer. Infill was rectangular with density 15%, with 2 perimeter layers, and 5 top and bottom layers. Layer height was uniformly 0.2mm. Print speed was 1500mm / minute, with slowdown and speedup for external perimeters and support respectively.

Print times were approximately 16 hours each for the 1- and 2-twist, and 21 hours each for the 3- and 4-twist. The objects are approximately 175mm in diameter, and 60mm thick. Figure 7 shows the results of the prints.

6. Future work

For future work in this area, we would like to continue the decomposition and printing of higher twist solid Möbius surfaces, and to investigate the growth of the degree and multiplicity of the singular curve. We also desire a larger printed family of Möbius surfaces, for parameters across the range of a and b values.

The exploration of parameter values is interesting for algorithmic reasons, beyond the aesthetic. As the parameters tend toward 0, the generated surface becomes arbitrarily thin, and this can only cause problems for the modeling software. The exploration of the case where a or b are greater than unity and the surface self-intersects would also be interesting.

Additionally, the same methodology used in this paper could be used to parameterize and 3D-print a solidified surface representing a Klein bottle, a surface bearing many similarities to the Möbius strip.

Acknowledgments

This research was supported in part by the Notre Dame Center for Research Computing through computational servers and library support, Sloan Research Fellowship BR2014-110 TR14, and NSF grant ACI-1460032.

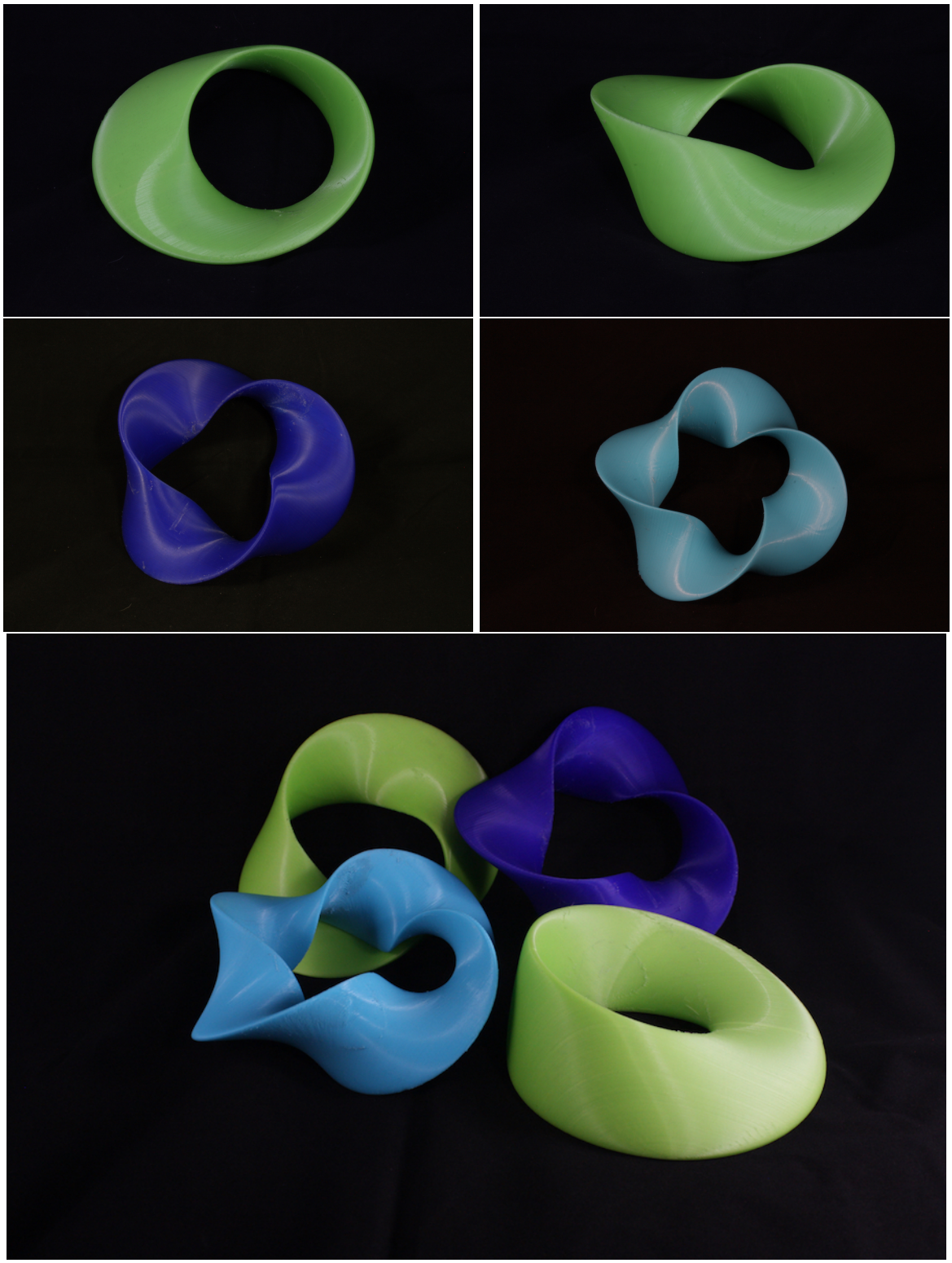


FIGURE 7. 3D prints of the smooth algebraic solid Mobiüs surfaces, twists 1 through 4, decomposed and smoothed using `Bertini_real`. Printed in PLA on a TAZ4.

References

[1] Daniel J Bates, Danielle A Brake, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler. On computing a cell decomposition of a real surface containing infinitely many singularities. In *International Congress on Mathematical Software*, pages 246–252. Springer, 2014.

[2] Daniel J Bates, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler. *Numerically solving polynomial systems with Bertini*, volume 25. SIAM, 2013.

[3] Gian Mario Besana, Sandra Di Rocco, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler. Cell decomposition of almost smooth real algebraic surfaces. *Numerical Algorithms*, 63(4):645–678, 2013.

[4] Danielle A Brake, Daniel J Bates, Wenrui Hao, Jonathan D Hauenstein, Andrew J Sommese, and Charles W Wampler. Algorithm 976: Bertini_real: Numerical decomposition of real algebraic curves and surfaces. *ACM Transactions on Mathematical Software (TOMS)*, 44(1):10, 2017.

[5] Danielle A Brake, Daniel J Bates, Wenrui Hao, Jonathan D Hauenstein, Andrew J Sommese, and CharlesW Wampler. Bertini_real: Software for one-and two-dimensional real algebraic sets. In *International Congress on Mathematical Software*, pages 175–182. Springer, 2014.

[6] CGAL. Computational Geometry Algorithms Library, 2017.

[7] The Blender Foundation. Blender, 2017.

[8] Marius Kintel. OpenSCAD, 2017.

[9] Stephan Klaus. Solid Möbius strips as algebraic surfaces, 2009.

[10] Liza Wallach Kloski and Nick Kloski. *Getting Started with 3D Printing: A Hands-on Guide to the Hardware, Software, and Services Behind the New Manufacturing Revolution*. Maker Media, Inc, first edition, 2016.

[11] Jörg Peters and Ulrich Reif. *Subdivision surfaces*. Springer, 2008.

[12] Simplify3D. Simplify3D, 2017.

[13] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, 11(3):201–227, 1992.

Appendices

Appendix A. Sample Input File for Four-Twisted Solid Möbius

```
CONFIG

tracktype: 1; % do numerical irreducible decomposition in Bertini
tracktolbeforeeg: 1e-9;
tracktolduringeg: 1e-10;
randomseed: 2; %fix the random number generator's seed value
maxstepsbeforenewton: 0; % do newton corrector steps every time step
maxnewtonits: 1; %take at most one newton step
finaltol: 1e-13; %compute solutions to at least 13 digits
functiontolerance: 1e-8; %function residuals are zero if smaller
sharpendigits: 30; %sharpen using Newton's method to 30 digits, when possible
endgamenum: 2; %use the Cauchy endgame
numsamplepoints: 10; %use 10+1 points in the endgame
endgamebdry: 0.01; % switch to endgame at t = 0.01
odepredictor: 8; % use a higher-order ODE predictor
targettolmultiplier: 10; %numerical fuzzyness factor for zero testing
securitylevel: 1; % don't truncate paths appearing to go to infinity
condnumthreshold: 1e300; %points are singular only if they appear multiply

END;

INPUT

constant a,b;
a = 0.01;
b = 0.23;

variable_group x, y, z;
function f;

f = (a*(x^2 + y^2)^2 + a*(x^2 + y^2)^3 + b*(x^2 + y^2)^2 + b*(x^2 + y^2)^3 +
    a*x^4 + a*y^4 - b*x^4 - b*y^4 + a*z^2*(x^2 + y^2)^2 + b*z^2*(x^2 + y^2)^2 -
    6*a*x^2*y^2 + 6*b*x^2*y^2 - a*x^4*z^2 - a*y^4*z^2 + b*x^4*z^2 + b*y^4*z^2 -
    2*a*b*(x^2 + y^2)^2 + a*x^4*(x^2 + y^2) + a*y^4*(x^2 + y^2) - b*x^4*(x^2 +
    y^2) - b*y^4*(x^2 + y^2) + 6*a*x^2*y^2*z^2 - 6*b*x^2*y^2*z^2 -
    6*a*x^2*y^2*(x^2 + y^2) + 6*b*x^2*y^2*(x^2 + y^2) + 8*a*x*y^3*z -
    8*a*x^3*y*z - 8*b*x*y^3*z + 8*b*x^3*y*z)^2 - (x^2 + y^2)*(2*a*x^4 + 2*a*y^4
    - 2*b*x^4 - 2*b*y^4 + (x^2 + y^2)^2*(2*a + 2*b) - 12*a*x^2*y^2 +
    12*b*x^2*y^2 + 8*a*x*y^3*z - 8*a*x^3*y*z - 8*b*x*y^3*z + 8*b*x^3*y*z)^2;

END;
```

Appendix B. k-twisted Solid Möbius Matlab code

Here we provide a Matlab function for computing a symbolic variable containing the equation for the k-twisted solid Möbius surface. This code was tested against Matlab 2016a, and also appears in the test/examples folder in Bertini_real.

```
function answer = compute_twisted_mobius(twist)
%Created by Travis C. Wert at the University of Notre Dame under Danielle A.
%Brake on 4.26.17

syms x y z t a b c2 s2 two_cs c s C S left1 left1a left1b left2 right1 right1a
    right1b right2

E_psi = c2*(a*(t-1)^2+b*z^2)+two_cs*(a-b)*(t-1)*z+s2*(b*(t-1)^2+a*z^2)-a*b;
    %from original paper found here
    https://imaginary.org/sites/default/files/moebiusband.pdf

assume(C,'real'); %assists with the substitutions later
assume(S,'real');
left1 = real(expand((C+1i*S)^twist)); %derived from De Moivre's formula
left2 = imag(expand((C+1i*S)^twist));

%There will be three equations that we need, c^2, s^2, and 2cs
%right1a will get s^2, right1b will get c^2, and right2 will get 2cs

right1 = c^2 - s^2;
right2 = 2*c*s;

right1a = subs(right1,c^2,1-s^2);
left1a = -(left1-1)/2;
left1a = subs(left1a,C,x/t);
left1a = subs(left1a,S,y/t);
left1a = expand(left1a);

right1b = subs(right1,s^2,1-c^2);
left1b = (left1+1)/2;
left1b = subs(left1b,C,x/t);
left1b = subs(left1b,S,y/t);
left1b = expand(left1b);

left2 = subs(left2,S,y/t);
left2 = subs(left2,C,x/t);

%substitute each of the left hand sides into E_psi
E_psi = subs(E_psi,{c2,two_cs,s2}, ...
    {left1b, ...
    left2, ...
    left1a});

E_psi = (E_psi)* (2*t^twist); %cancel the denominator for the equation
```

```
E_psi = expand(E_psi);
simplify(E_psi);

%gather the odd and even coefficients of t
[ct,tt] = coeffs(E_psi,t);

even = sym(0); %seed the loop
odd = sym(0);

for ii = 1:length(tt)
   deg = feval(symengine,'polylib:degree',char(tt(ii)),'t');
   if mod(deg,2) == 1 %odd
       odd = odd + ct(ii)*tt(ii);
   else % even
       even = even + ct(ii)*tt(ii);
   end
end

assume(a>0 | a<1);
assume(b>0 | b<1);
odd = collect(odd/t,t); %took out a t term, to be added back later
even = collect(even,t);
even = simplify(even);

%square both sides
even = even*even;
odd = odd*odd;

even = subs(even,t,sqrt(x^2+y^2));
odd = subs(odd,t,sqrt(x^2+y^2))*(x^2+y^2); %here the original t term (now t^2)
    is added back

answer = even-odd;
end
```

Appendix C. 10-Twisted Solid Möbius

The Matlab code found in Appendix B can also be used to compute Möbius surfaces of arbitrary twist, an example of which can be seen in Figure 8 with the 10-twisted solid Möbius surface:

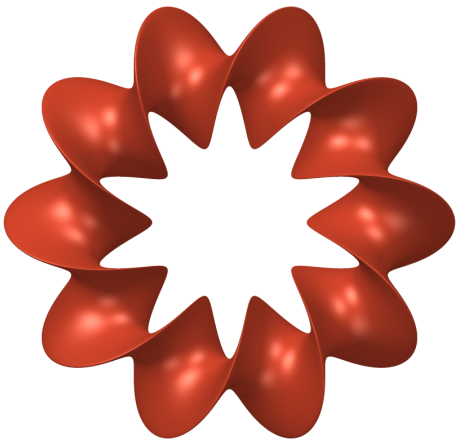


Figure 8. 10-Twisted solid Möbius surface rendered in SURFER.

.

Student biographies

Travis C. Wert: (*Corresponding author:* twert@alumni.nd.edu) Travis Wert graduated cum laude from the University of Notre Dame in spring 2017 with majors in Finance and Applied and Computational Mathematics and Statistics. Following graduation, he spent two years in strategy consulting before moving to a private investment firm, utilizing his undergraduate mathematics background to solve complex business problems.