MXM - Vectors with Smallest Slope Final Paper

Jake Neau, Arden Kim, Mele Pluviose, Mariya Siddiqui

I. Concepts

Translation Surfaces

Our work has primarily been on different translation surfaces in the strata of the octagon. The easiest way to think about a translation surface is by identifying certain sides of a two-dimensional polygon and "gluing" them together to form a three-dimensional shell. In the simplest case of the square, this would create a torus, as seen in Figure 1. In the case of the octagon, and thus the shapes we are working with, they would become a double torus.

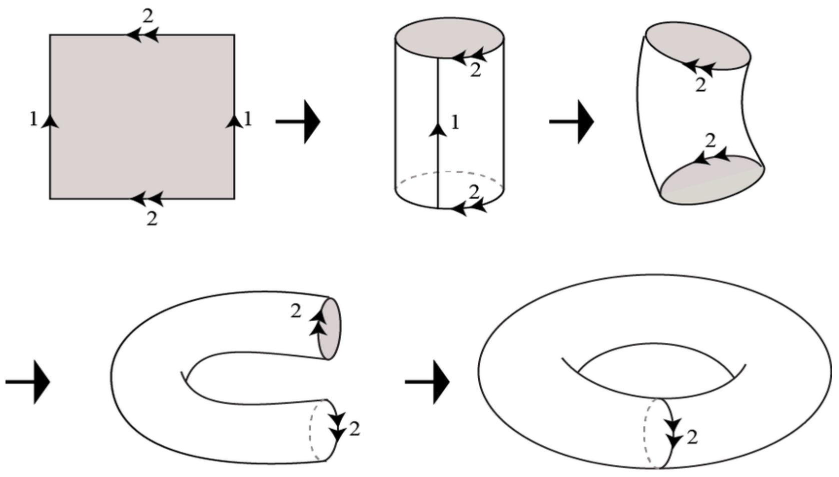


Figure 1

Cone Points:

When you fold specific shapes like the octagon, all of their angles will come together at one point. In the octagon case, you have eight angles of 3π/4 that add up to a 6π angle at that point. Because you can only fit a 2π angle at one point, we say the point has 2(2π) excess angles. This area of extra angle is called a cone point and will be helpful later on for defining saddle connections. Different surfaces can have other cone points and different excess angles at those cone points. If two surfaces happen to have the same number of cone points, the same number of cone points, and the same genus (number of holes in the 3D shape), they are in the same strata, and we can treat them the same way. Our surfaces are all in the strata of the octagon, so they have one cone point with 2(2π) excess angle and are genus two.

Saddle Connections:

Before saddle connections are described, it is helpful to explain how you can wrap lines around a translation surface. If you imagine the case of the square becoming a torus, you can imagine trying to wrap a string around the torus in many directions. Maybe you wrap vertically or horizontally around the torus many times before coming back to the starting point. Perhaps you wrap around once and come back to the same point. How would these lines translate over to the two-dimensional square? If you think back to identifying sides and "gluing" them together, the line where those sides meet would exist on the torus. Every time you cross the line on the torus, this would equate to wrapping around to the opposite identified side on the two-dimensional surface. See Figure 2 for details. This can be done for any translation surface. Every time a line on the two-dimensional surface reaches a side, it wraps around to the other side identified. This makes it helpful in describing saddle connections. Saddle connections are straight lines connecting cone points. The line can go between the same cone point. This line will have a slope defined by its horizontal component and vertical component that it travels.

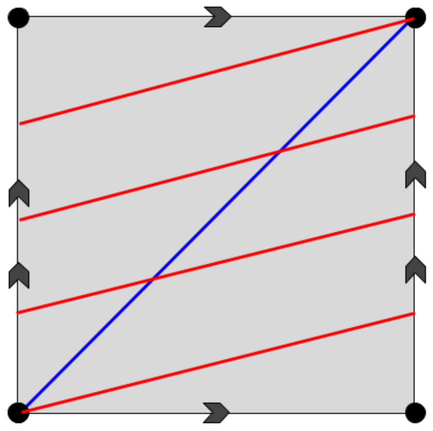


Figure 2

Assume that in the figure, the square is the unit square. The red line has a horizontal component of four as it wraps around four times, and the blue line has a horizontal component of one as it only wraps around once. Both lines have a vertical component of one.

Matrix Transformations:

Matrices can act on translation surfaces to change them. The easiest way to visualize this is to put the translation surface on a coordinate plane and apply the matrix to each corner. See Figure 3 for an example. If you can get one surface by cutting and pasting together another surface, we can treat them the same way. This makes it useful for us to turn some translation surfaces that are harder to work with into ones that are easier to work with.

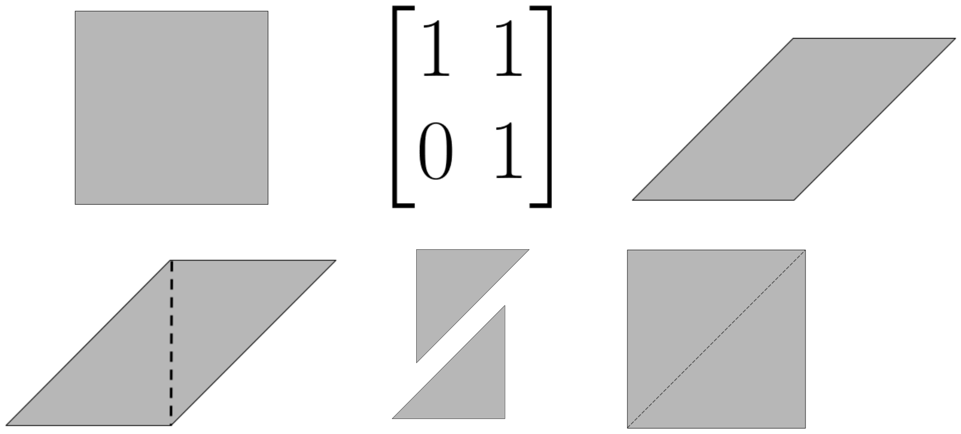


Figure 3

Horocycle Flow:

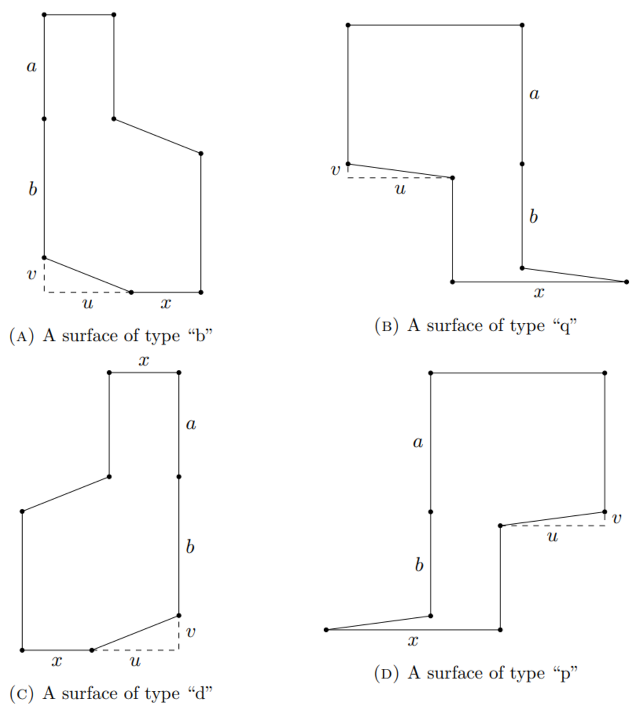
What we want to do with our surfaces is to analyze the distribution of different certain saddle connections. One way we can do this is with horocycle flow. When a translation surface undergoes horocycle flow, the slopes of the saddle connections are not preserved, but the difference in the slopes is. This is useful when considering that the horocycle flow matrix will make a horizontal saddle connection. Specifically, if you use the slope of a saddle connection in the field “S” in the hs matrix, that saddle connection will be the one to become horizontal. Then the slope difference between that saddle connection and a second saddle connection will now be the slope of the second saddle connection. We have been finding the saddle connection with the smallest slope and horizontal component less than or equal to 1 and using that to apply horocycle flow to our translation surface, then finding the next smallest slope and repeating this process.

$$h_s = \begin{bmatrix} 1 & 0 \\ -s & 1 \end{bmatrix}$$

Figure 4

Our surfaces:

As stated previously, we are working with four shapes in the strata of the octagon. We have four different types of surfaces: b, q, d, and p. What we have been trying to show is how applying a matrix transformation to one of these shapes will turn it into another one of these four shapes if we do some cutting and pasting. This would make it possible to keep doing horocycle flow multiple times on the shape.



(A) A surface of type "b"
(B) A surface of type "q"
(C) A surface of type "d"
(D) A surface of type "p"

Figure 6

II. Methods

Approach One:

The first approach was to find the slopes of saddle connections based on the a, v, x, v, and u, and shape type parameters are shown in Figure 6. We would have a user input all these values and then generate possible cone points from our explicitly defined equations. We would then analyze the saddle connections to find the smallest one with a less than or equal to slope 1. We made a mock-up in Mathematica showing how creating these shapes would work and applying the hs matrix for variable values of s. You can see this example in Figure 7. Our process looped through getting a surface of one type, calculating the smallest slope from equations we found earlier, applying horocycle flow, and pasting the result into one of the four shapes we can work with. At this point, the cycle would start again, and a new shape would be found. We have pseudo code written for how we will handle this method. We need to work on getting the code written out and finished in the future. The demo made in Mathematica will be a good reference tool for how we might do this in a language like Python. We know least about right now how to paste the result of horocycle flow back into one of our four usable shapes, but we have looked at how it might be possible for some of our shape types.

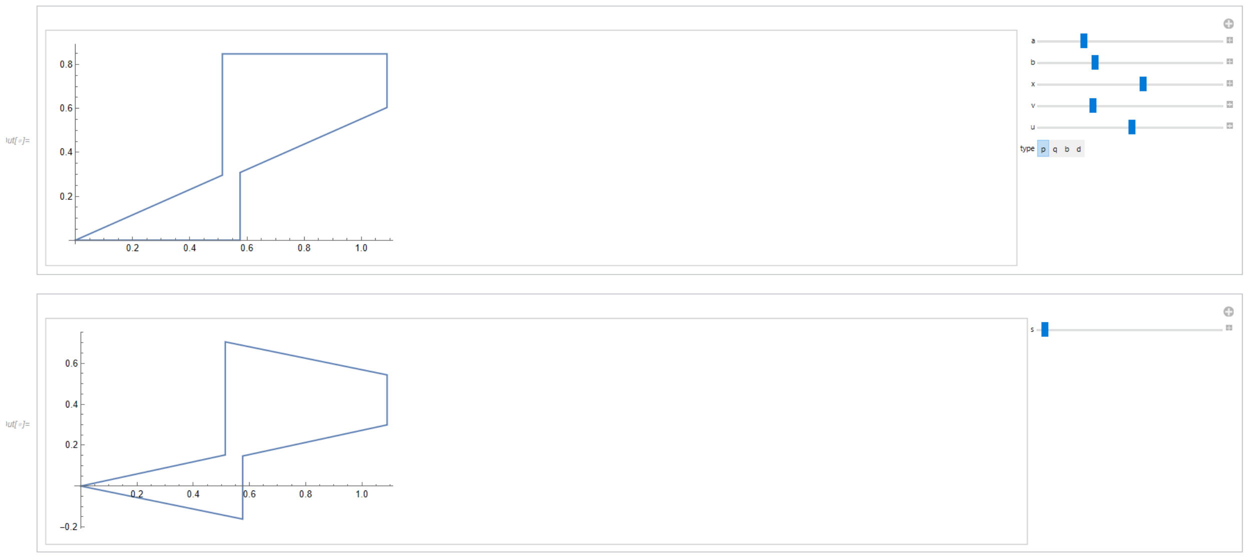
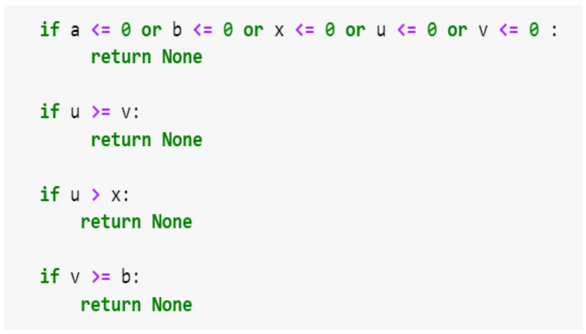


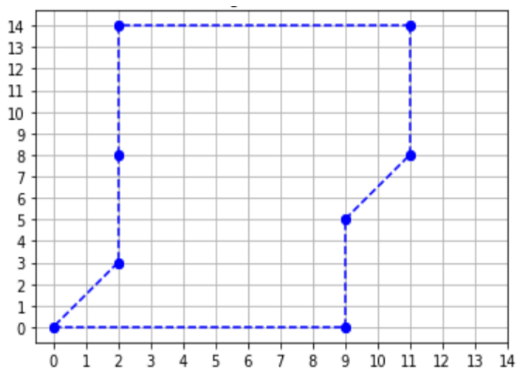
Figure 7

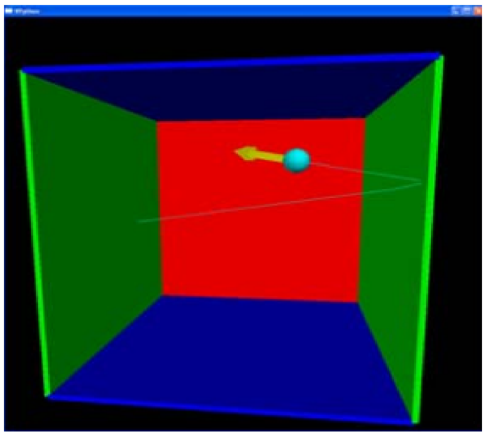
Approach Two:

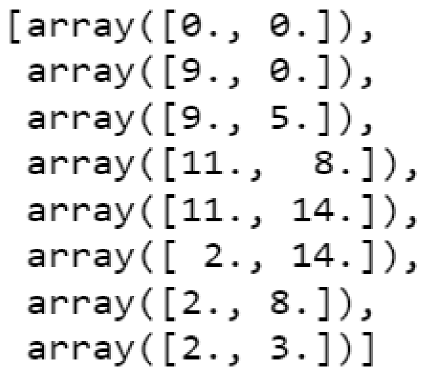
The second approach was to use a 3D simulation. So first, we'd get the values of a,b,v,u, and x of any particular BDPQ surface. We'd then set conditions to figure out restrictions on the surface. An example is that x must be greater than u for PQ surfaces. Next, we'd create coordinate points based on user input, matrix translations, flips Among the y-axis, and rotations. So essentially, we have the BDPQ baseline shapes, but the user would transform these shapes if desired. After the transformation is complete, we would use these coordinate points to make walls on a 3D surface. The ball starts at the origin (0,0) and then bounces off the walls until it reaches another coordinate based on the user's choosing. The ball cannot touch the destination point directly from the start. It has to make a series of loops through the wall – that is, once it reaches an endpoint of a wall, it will reappear on another wall that is opposed to the original one. Once reappearing, the ball will move in the same trajectory as the original slope. If the ball reaches the destination, we will collect this slope and add it to the series. After several tries with different slopes, we will see which one is the smallest and has a horizontal component less than or equal 1. Below are some code, sample coordinates, and the bigger picture of what we're trying to achieve.



```
if a <= 0 or b <= 0 or x <= 0 or u <= 0 or v <= 0 :
     return None

if u >= v:
     return None

if u > x:
    return None

if v >= b:
    return None
```







```
[array([0., 0.]),
 array([9., 0.]),
 array([9., 5.]),
 array([11.,  8.]),
 array([11., 14.]),
 array([ 2., 14.]),
 array([2., 8.]),
 array([2., 3.])]
```

Figure 8

Ball in a box2 - Southeastern Louisiana University. (n.d.). Retrieved May 12, 2022, from https://www2.southeastern.edu/Academics/Faculty/rallain/allain/plab223/docs/Ball_in_a_box.pdf