MOTION PICTURE OF 4D KNOTS

MATTHEW WALLOCH

Abstract. For the motion picture of a knot, we consider $\mathbb{R}^4 = \mathbb{R}^3 \times \mathbb{R}$ where we have some family Y_t spanning over $t \in \mathbb{R}$ and each Y_t is the projection of the knot into $\mathbb{R}^3$ "at time t" [1]. Then there are certain patterns we can recognize certain points which can tell us information about a knot [1]. A lot of mention of motion pictures is often in print and as a result of the medium, it is often a series of 2D projections which can limit how one views the knot. For this paper, we have created a program that allows us to choose a 1-knot and see the spun version of it through different projections into $\mathbb{R}^3$.

Contents

1. Knotting in 4d

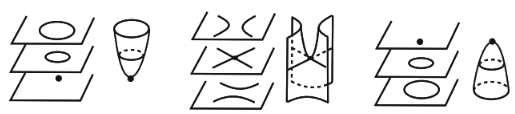
If we look at the traditional knot in 4D then we get something fascinating: everything is the unknot. This is because if you have your knot in 4 dimensions we can essentially just flip every crossing by dragging the overstrand into the 4th dimension, moving it to a position below the understrand, and moving it back to the 3-dimensional space at the knot is in, if we do this strategically enough we can then untie every knot. The knot we are used to are 1-knots, i.e. an embedding of S^1 in $\mathbb{R}^3$, in 4 dimensions we need to add something to our knot so it is not always trivial. It turns out all we need is a second dimension. A surface knot (or 2-knot) is a submanifold of R^4 that is homeomorphic to a closed connected surface. Which just means that if we zoom into our 2-knot it will look like the plane $\mathbb{R}^2$ at any point in our knot. Thus we can have 2-knots that are homeomorphic to S^2, T^2, the projective plane, or even the Klein bottle.

2. Making spins of knot

There are a few things we need to define before we begin spinning knots. The first is $\mathbb{R}^3_+$, this is essentially $\mathbb{R}^3$ with only the one half of an axis, in this case, we will consider $\mathbb{R}^3_+ = \{(x, y, z) \in \mathbb{R}^3 : x \geq 0\}$. Then we can define a tangle as a properly embedded arc in $\mathbb{R}^3_+$. We can see that if we close the ends of the tangle, we will get some knot, and thus we can convert a tangle to a knot and more importantly a knot to a tangle by attaching or detaching ends. Let k be a tangle for a knot K as described, we can spin k along the yz axis in 4 dimensions. Then we can take the union of every angle of rotation for the knot we get a 2-knot in $\mathbb{R}^4$. We call this new union as $spun(k)$, since K is a knot we get a 2-knot or the spun K. It is also of note that two knots that are not equivalent could have equivalent spun knots.

3. Motion Picture of a 2-knot in 4d space

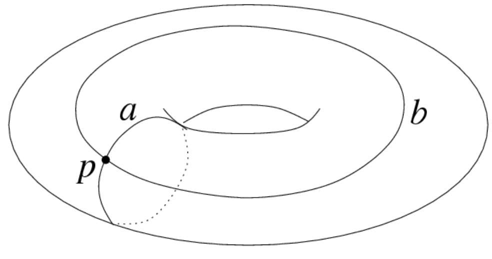
We can view $\mathbb{R}^4$ as $\mathbb{R}^3 \times \mathbb{R}$, and we can take a projection of $\mathbb{R}^4$ be the projection by crushing the $\mathbb{R}$ component. Then for the motion picture of subset $X \subset \mathbb{R}^4$ we can look for some $t \in \mathbb{R}$, a subset $Y_t = X \cap (\mathbb{R}^3 \times \{t\})$. This one parameter family is called the motion picture. In this sense we can define a line in R^4 and take the hyperplanes, i.e. the space orthogonal to the line gives us a cross-section. All of the cross sections give us the motion picture. The motion picture can also tell us the genus of the surface of the knot bounds. This is done by analyzing critical points which are summarized in the figure below from Kamada's *Surface-knots in 4-space*:



Thus we can see that a critical point can generally be described as either the appearance of a new curve, the splitting of a curve into multiple, multiple curves joining up, or the disappearance of a curve. It is important to know that the curves still bound a continuous space its just that the projections can make them look disjoint. For a genus g surface we need this splitting of curves and joining of curves to happen to make the "holes" in the surface.

4. Knot parameterization

Since 1-knots are embeddings of a sphere, we should be able to make a parameterization of the knot. One of the most accessible classes of knots to parameterize are the torus knots. These knots can be impeded in a torus and as such we can notate them with $T(p, q)$ where the knot wraps around q times the circle on the interior of the torus that goes the hole (circle a), and p times around the torus going through the axis of symmetry through the hole of the torus (circle b).



With a few cups of coffee and blackboard, we can get a parameterization as:

$$(\cos(2p\pi t)(2+\cos(q\pi t))), \sin(2p\pi t)(2+\cos(2q\pi t)), \sin(2q\pi t), 0)$$

for $0 \leq t < 1$. In our instance, we are working with the trefoil which has the slightly altered parameterization of

$$(\cos(4\pi t)(2+\cos(2\pi t)) - 4), \sin(4\pi t)(2+\cos(6\pi t)), \sin(6\pi t), 0)$$

due to the fact, we need it to be in $\mathbb{R}^3_+$. We then can cut this parameterization to $\frac{1}{12} \leq t \leq \frac{11}{12}$, and then for $t < \frac{1}{12}$ and $t > \frac{11}{12}$ we can attach paths from the yz-axis to the cut trefoil to from the tangle.

FIGURE 1. The tangle in R^3_+.

5. Pseudocode Analysis

We will go over a brief algorithm that we used to compute the knot sets. We first need a parameterization of the trefoil,

$$(x_k(t), y_k(t), z_k(t), 0)$$

in 4 dimensions using the axis x,y,z, and w; this result was discussed in the previous section in more detail. Then we can rotate this knot about the yz-plane using a rotation matrix and which results in the parameterization:

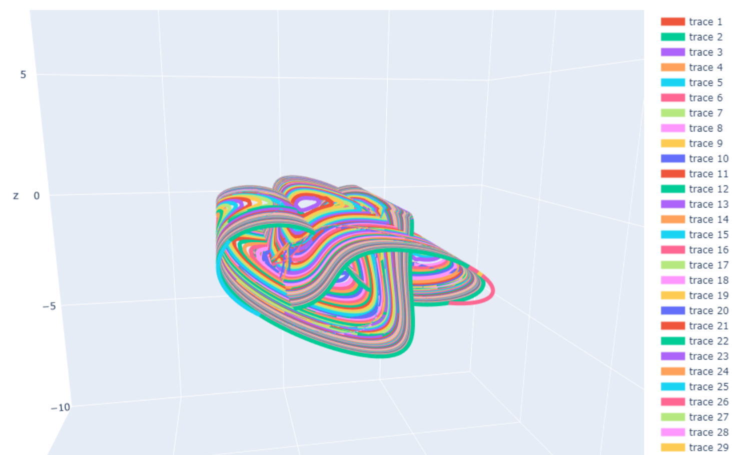
$$(\cos(\theta)x_k(t), y_k(t), z_k(t), \sin(\theta)x_k(t)) \text{ for } 0 < \theta \leq 2\pi.$$

Then for some value of w we get that $\sin(\theta)x_k(t)) = w$, so $\theta = \sin^{-1}(\frac{w}{x_k(t)})$. Because we moved the knot over $x_k(t) \neq 0$. Then we go through the t values and evaluate the inverse sin for each t, calculate the rotated knot point at that θ and t, and add these points to a list. We take that compiled list and run it through a series of computations that then sort the list by nearest neighbor while also looking out for a significant jump in x and z values that would correspond to a second closed curve appearing. If there is a significant jump another list of points is made, then we graph the set of points or sets of points and the graphing software will take care of adding the lines. One thing to note is that we did not parameterize the lines that connect the tangle to the zy-plane instead, we add these in after the rotation as it is easier computational-wise. The program used to graph is Plotly which has integrations with Dash which allows us to add it in an HTML environment.

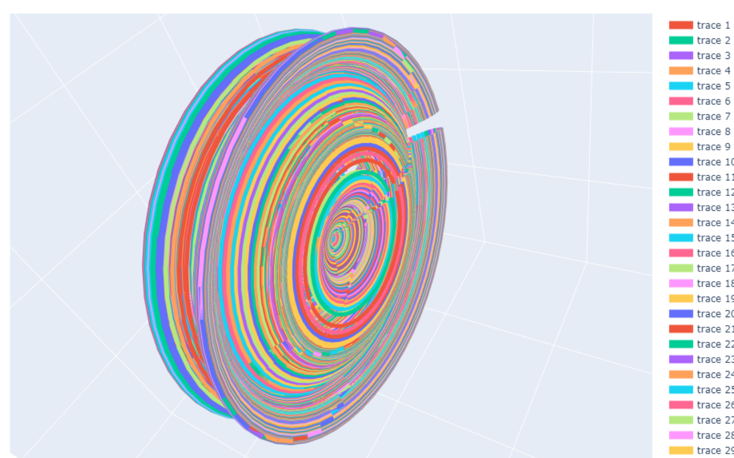
For going on the y-axis it is slightly altered. For some value of y, we get that $y_k(t) = y$, so we can use a numerical solver to estimate $y_k(t) - y = 0$ for t and we use this list of t values to plot the points. We then go through the parameterization fixing t and allowing θ to range over its domain. Then we run it through the same algorithm of sorting by nearest neighbor and the steps after.

6. Results and Working Code

The code and potential updates will be hosted here. For an online version of the fast plotter, you can access it here. We can see the crushed axis portions below:

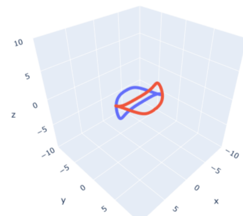


(A) The projection of the crushed w-axis

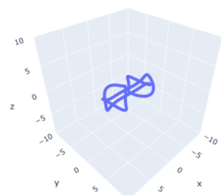


(B) The projection of the crushed y-axis.

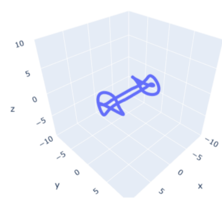
For the w-axis motion picture, we get:



(A) Between -6 and -3, and 3 and 6 we see there are two unlinked curves.

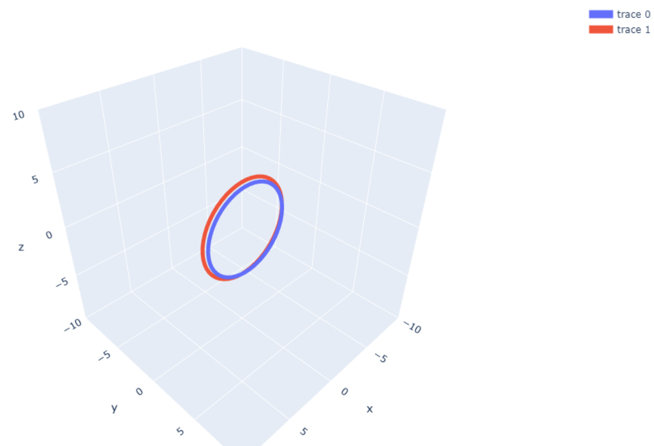


(B) At -3 and 3 we see that these curves hit a critical point and come together as one curve.

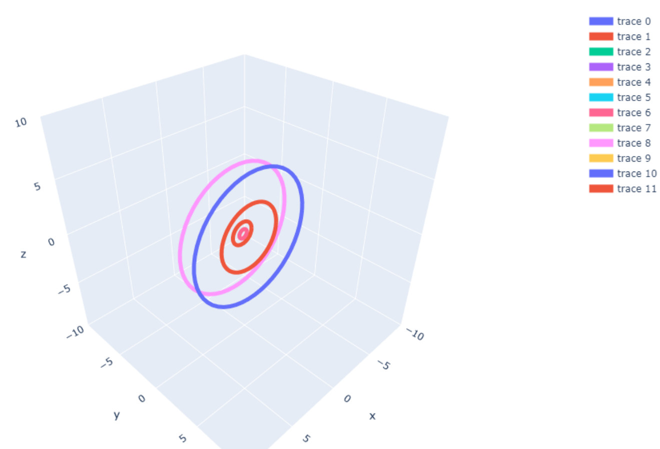


(C) Between -3 and 3 we see this connected sum of two trefoils spread apart going to 0 and coming back together heading away from 0.

For the y-axis motion picture, we get:



(A) Between -2.8 and -1.75 and 1.75 and 2.8 we see two distinct circles that begin to move slightly.



(B) Between -1.75 and 1.75 we see multiple circles that seem to move in and out of each other and then disappear and reappear seemingly randomly.

From looking at the w-axis motion picture we can see that we start with two distinct curves, that then merge into one, then split into two curves. Thus from looking at the critical points, we can conclude that the spun trefoil is a knotted S^2.

7. Conclusion and future directions

For the next iteration, we plan to add more torus knots, or just even have the user choose a torus knot to spin. Another hopefully mathematically simple, addition would be to include a twist to the spin. In this, we rotate the knot in 3D along a line through the knot while also spinning in 4 dimensions, this should be somewhat simple due to the fact that rotations can be easily done through matrix multiplication, however, finding out what values of θ and t that correspond to each value of w. Another addition would be spinning non-torus knots, although parameterization might be more difficult, especially for generalizing to all knots.

References

[1] Seiichi Kamada. *Surface-knots in 4-space*. en. 1st ed. Springer Monographs in Mathematics. Singapore, Singapore: Springer, Apr. 2017.