

CONSTRUCTING POLYNOMIAL KNOTS

MATTHEW WRIGHT

ABSTRACT. Shastri proved[3] that every knot can be expressed as the image of a parametric function $t \mapsto (x(t), y(t), z(t))$, where x , y , and z are polynomials in t . However, it is difficult based on his proof to actually find a polynomial knot of a given knot type. We present an algorithm for converting a piecewise linear parameterization of a knot into a polynomial parameterization of a knot of the same type, and prove that it works in general. We also show how this algorithm can be modified to give compact polynomial knots and (in some cases) trigonometric knots.

1. PRELIMINARIES

Definition 1. Let $\kappa(t) = (x(t), y(t))$ be a continuous parametric function from $\mathbb{R} \rightarrow \mathbb{R}^2$. A pair (a, b) is a *crossing* of $x(t)$ and $y(t)$ if $a \neq b$, $x(a) = x(b)$, and $y(a) = y(b)$.

If I is an interval in the domain of κ , we say that a crossing (a, b) is in I if $a, b \in I$; similarly, if I and J are intervals, (a, b) is in (I, J) if $a \in I$ and $b \in J$.

Example 2. $\kappa(t) = (t^2, t^3 - t)$ has two crossings: $(-1, 1)$ and $(1, -1)$ (note that we consider these crossings distinct, even though they correspond to the same double point).

$\kappa(t) = (t^2, 0)$ has infinitely many crossings, at $\{(t, -t) \mid t \in \mathbb{R} \setminus \{0\}\}$.

Lemma 3. Let $\kappa : t \mapsto (x(t), y(t), z(t))$ be a continuous injective map from $\mathbb{R} \rightarrow \mathbb{R}^3$, and let $\hat{z}(t)$ be a continuous function with the property that, for all crossings (a, b) of $x(t)$ and $y(t)$, we have $\hat{z}(a) > \hat{z}(b)$ if and only if $z(a) > z(b)$. Then κ has the same knot type as $\hat{\kappa}(t) = (x(t), y(t), \hat{z}(t))$.

Proof. The knots have the same projection on the xy -plane; because of the conditions on \hat{z} , they also have the same crossings — in other words, they have the same diagram. Since a diagram determines a knot up to equivalence, κ and $\hat{\kappa}$ have the same knot type. \square

2. THE NAÏVE ALGORITHM

In this section, we provide a first attempt at an algorithm for finding polynomial parameterizations of knots (note that we are dealing with knots whose ends go off to infinity, not compact knots). While the algorithm presented in this section is flawed and does not work in general, it contains the main ideas of the correct algorithm.

At the heart of the algorithm is the following theorem, which allows us to replace a single component of a parametric function with a polynomial under certain conditions:

Theorem 4. *Let $\kappa(t) = (x(t), y(t), z(t))$ be a continuous injective function from $\mathbb{R} \rightarrow \mathbb{R}^3$ such that x and y have a finite number of crossings. Then there is a polynomial $\hat{z}(t)$ such that $\hat{\kappa}(t) = (x(t), y(t), \hat{z}(t))$ has the same knot type as κ .*

Proof. By Lemma 3, it is enough for $\hat{z}(t)$ to satisfy, for all crossings (a, b) of $x(t)$ and $y(t)$, the condition that $\hat{z}(a) > \hat{z}(b)$ if and only if $z(a) > z(b)$. We may therefore choose as $\hat{z}(t)$ the interpolating polynomial satisfying $\hat{z}(a) = z(a)$ and $\hat{z}(b) = z(b)$ for each crossing (a, b) . \square

Remark 5. Once we know that the polynomial $\hat{z}(t)$ exists, we can use linear programming to find $\hat{z}(t)$ of minimum degree. We can also find $\hat{z}(t)$ of any specific form that can be specified by linear constraints (having only even powers, for example) if one of that form exists.

Specifically, if we want to find $\hat{z}(t)$ of degree k , take a general degree- k polynomial $\hat{z}(t) = a_k x^k + \dots + a_2 x^2 + a_1 x$ (we do not need a constant term). Now create a linear program as follows: for each crossing (a, b) of $x(t)$ and $y(t)$ satisfying $z(a) > z(b)$ (ie. all crossings in which the overcrossing is listed first), add a constraint of the form $\hat{z}(a) - \hat{z}(b) \geq 1$. Since $\hat{z}(t)$ can be multiplied by any arbitrary value without changing the knot type of $(x(t), y(t), \hat{z}(t))$, this linear program has a solution if and only if there exists $\hat{z}(t)$ of degree k such that $(x(t), y(t), z(t))$ and $(x(t), y(t), \hat{z}(t))$ have the same knot type.

Example 6. As a trivial example, let $\kappa(t) = (t^3 - t, t^2, e^t)$, the image of which is an unknot. The crossings of $x(t)$ and $y(t)$ are $(-1, 1)$ and $(1, -1)$. By Theorem 4, if we let $\hat{z}(t)$ be defined such that $\hat{z}(-1) = e^{-1}$ and $\hat{z}(1) = e$, then $\hat{\kappa}(t) = (t^3 - t, t^2, \hat{z}(t))$ is also an unknot.

By the remark it is sufficient that $\hat{z}(t)$ satisfy $\hat{z}(1) - \hat{z}(-1) \geq 1$. To find out if there is a degree-1 polynomial that will work, we will let $\hat{z}(t) = at$; plugging into the inequality gives $2a \geq 1$ — in other words (since we can multiply $\hat{z}(t)$ by any positive number), $\hat{z}(t)$ can be any increasing linear function, as we would expect.

Lemma 4 suggests an algorithm for converting a parameterization of a knot into a polynomial parameterization: starting with $\kappa(t) = (x(t), y(t), z(t))$, first replace $z(t)$ with a polynomial $\hat{z}(t)$ without changing the knot type; then replace $y(t)$, and then replace $x(t)$.

The problem with this algorithm is that we may, after any of the steps, end up with an infinite number of crossings and not be able to proceed. The following example shows how this may happen:

Example 7. Let

$$\begin{aligned} x(t) &= \begin{cases} t & \text{if } t \leq 0 \\ -t & \text{if } t \geq 0 \end{cases} \\ y(t) &= t \\ z(t) &= e^t. \end{aligned}$$

When we try to apply the algorithm, we may first replace $z(t)$ by the polynomial $\hat{z}(t) = t^2$. When we compute the crossings of $x(t)$ and $\hat{z}(t)$ to be able to find $\hat{y}(t)$, though, we find that there are an infinite number of crossings (specifically, $(t, -t)$ is a crossing for any $t \neq 0$). The algorithm cannot proceed.

3. A CORRECT ALGORITHM

To correct the flaw in the naïve algorithm, we will give several modified versions of Lemma 4 that will give us stronger guarantees on the polynomials it produces. Together, these lemmas will show how we can go from any injective piecewise linear parameterization of a knot to a polynomial parameterization of the knot.

We will start by showing that if we start with piecewise linear functions, it is always possible to “patch up” any segments that have infinitely many crossings by adding conditions to $\hat{z}(t)$.

Lemma 8. *Let $\kappa : t \mapsto (x(t), y(t), z(t))$ be a continuous injective map from $\mathbb{R} \rightarrow \mathbb{R}^3$ such that $x(t)$ and $y(t)$ have a finite number of crossings. Assume further that $x(t)$ is piecewise linear and that no piece of $x(t)$ is constant.*

Then it is possible to construct a polynomial $\hat{z}(t)$ such that $\hat{\kappa}(t) = (x(t), y(t), \hat{z}(t))$ has the same knot type as κ , and so that there are a finite number of crossings of $x(t)$ and $\hat{z}(t)$.

Proof. Again, $\hat{z}(t)$ will be an interpolating polynomial; it will have all of the conditions as in the proof of Lemma 4; that is, we require that $\hat{z}(a) = z(a)$ and $\hat{z}(b) = z(b)$ for every crossing (a, b) of $x(t)$ and $y(t)$. This is enough to ensure that κ and $\hat{\kappa}$ have the same knot type, no matter what other conditions we add. We will now focus on trying to ensure that $x(t)$ and $\hat{z}(t)$ have a finite number of crossings.

Let $x_j(t)$ be the j^{th} piece of $x(t)$, and I_j the interval of values of t corresponding to the j^{th} piece. By assumption, $x_j(t) = at + b$ for some $a \neq 0$ and b ; in particular, $x_j(t)$ is a bijection and has inverse $x_j^{-1}(t) = (t - b)/a$, which is again a polynomial.

First note that the graph of $t \mapsto (x_j(t), \hat{z}(t))$ is the same as that of $t \mapsto (t, \hat{z}(x_j^{-1}(t)))$. Since the latter has no crossings at all, we have shown that $x(t)$ and $\hat{z}(t)$ have a finite number of crossings in I_j (none at all, in fact). By a similar argument we can show that if $x_j = x_k$ then there is no crossing in (I_j, I_k) .

We now consider all i and j with $j \neq k$ and $x_j \neq x_k$. If the graphs of $(x_j(t), \hat{z}(t))$ and $(x_k(t), \hat{z}(t))$ intersect at infinitely many places, there are infinitely many values of t for which the polynomial $\hat{z}(x_j^{-1}(t))$ is equal to $\hat{z}(x_k^{-1}(t))$; the two are therefore equal (distinct polynomials are equal at finitely many values). It follows that as long as we can add conditions to $\hat{z}(t)$ so that for some t we have $\hat{z}(x_j^{-1}(t)) \neq \hat{z}(x_k^{-1}(t))$, then $x(t)$ and $\hat{z}(t)$ will have a finite number of crossings in (I_j, I_k) . As it turns out, since only finitely many points that $\hat{z}(t)$ must interpolate, and since $x_j \neq x_k$, we can find t such that $x_j^{-1}(t) \neq x_k^{-1}(t)$ and neither is part of a point that $\hat{z}(t)$ must interpolate; we may then add to the conditions on $\hat{z}(t)$ that $\hat{z}(x_j^{-1}(t)) = 1$ and $\hat{z}(x_k^{-1}(t)) = 2$ (or any other pair of values we choose).

We have added conditions to $\hat{z}(t)$, then, so that for all j, k , there are finitely many crossings of $x(t)$ and $\hat{z}(t)$ in (I_j, I_k) ; $x(t)$ and $z(t)$ therefore have finitely many crossings. \square

For the final step of the algorithm, we will need the following lemma (which is still a conjecture; we have been unable to prove it):

Lemma 9. *If $x(t)$ and $y(t)$ are polynomials whose derivatives (in the complex plane) are never simultaneously zero, then they have a finite number of crossings.*

Proof. Let $f(t)$ be a polynomial. We will define an operator Δf as follows:

$$(\Delta f)(u, v) = \frac{f(u) - f(v)}{u - v}$$

where the term of $u - v$ in the numerator is cancelled by the denominator, so Δ takes polynomials in one variable to polynomials in two variables. For example, if $f(t) = t^3 - t^2$, $(\Delta f)(u, v) = u^2 + uv + v^2 - u - v$.

Several facts about Δ are easily shown¹:

- (1) For any polynomial f ,

$$(\Delta f)(t, t) = f'(t).$$

- (2) Δ is linear; that is, if f, g are polynomials and c a constant,

$$\Delta(f + g) = \Delta f + \Delta g$$

and

$$\Delta(cf) = c\Delta f.$$

- (3) If $f(t) = t^n$,

$$(\Delta f)(u, v) = \sum_{k=0}^{n-1} u^k v^{n-1-k}.$$

The last two facts give us a characterization of all polynomials that can be written as Δf for some f : they are exactly the polynomials in which every term of a given degree has the same coefficient.

Now observe that every crossing of $x(t)$ and $y(t)$ is a common solution to $\Delta x = 0$ and $\Delta y = 0$. Therefore, if $x(t)$ and $y(t)$ have infinitely many roots, the algebraic curves Δx and Δy intersect at infinitely many points; by Bézout's theorem, they must have a common factor — that is,

$$(\Delta x)(u, v) = a(u, v)r(u, v)$$

and

$$(\Delta y)(u, v) = b(u, v)r(u, v)$$

for some polynomials a, b, r in two variables. It follows that

$$x'(t) = a(t, t)r(t, t)$$

and

$$y'(t) = b(t, t)r(t, t),$$

which means that as long as $r(t, t)$ is not constant the derivatives have a common complex root.

The only issue, then, is that $r(t, t)$ may be constant; in that case, the common factor of Δx and Δy doesn't become a common factor of the derivatives. Fortunately, as we will see, such a factor cannot occur.

Let $p(t)$ be some polynomial, and assume that $(\Delta p)(u, v)$ factors as $q(u, v)r(u, v)$ with $r(t, t)$ constant. Then we can write

$$r(u, v) = (u - v)s(u, v) + 1$$

for some polynomial s , so

$$(\Delta p)(u, v) = (u - v)s(u, v)q(u, v) + q(u, v).$$

¹It turns out that Δ has many more interesting properties, including a chain rule and product rule similar to those of calculus.

Assume for contradiction that such an s and q exist. Let the degree of $s(u, v)q(u, v)$ be n (so the degree of $q(u, v)$ is at most n). Denoting by a_k the coefficient of $u^k v^{n-k}$ in $s(u, v)q(u, v)$, we find that the coefficient of $u^k v^{n+1-k}$ in $(u - v)s(u, v)q(u, v)$ is $-a_0$ if $k = 0$; a_n if $k = n + 1$, and $a_{k-1} - a_k$ otherwise. Since these must all be equal, we have the following system of equations:

$$\begin{aligned} -a_0 &= a_0 - a_1 \\ a_{k-1} - a_k &= a_k - a_{k+1}, & 0 < k < n \\ a_{n-1} - a_n &= a_n \end{aligned}$$

which has no nonzero solution. This means that $s(u, v)q(u, v)$ could not have had degree n after all; this is a contradiction. \square

Remark 10. Lemma 9, as presented, requires that the polynomials have no common complex cusps. Proving the corresponding statement restricted to real cusps would simplify many of the steps below, but there is no obvious way to adapt the proof to work in that situation.

We'll be wanting to apply Lemma 9 to ensure that the polynomials at the last step of the algorithm have a finite number of crossings.

As we will see below, if Lemma 9 could be proved over the reals it would be quite straightforward to add conditions to our polynomials to satisfy its conditions. But since we have proved it only over \mathbb{C} , things are a bit more difficult: we can't in general interpolate at complex points without allowing the polynomial to have complex coefficients. However, the following lemma gives us a way of manipulating interpolating polynomials in a way that allows us to remove any complex cusps:

Lemma 11. *Let $p(t)$ be an interpolating polynomial, and let t_1, \dots, t_n be the values of t for which a point or derivative (or both) was specified. Then the polynomial*

$$q(t) = (t - t_1)^2(t - t_2)^2 \cdots (t - t_n)^2$$

has the following properties:

- (1) *For all $\alpha \in \mathbb{R}$, $p(t) + \alpha q(t)$ has the same values and derivatives as $p(t)$ at t_1, \dots, t_n .*
- (2) *All roots of $q'(t)$ are real.*

Proof.

- (1) By construction, $q(t) = q'(t) = 0$ at any of the points t_1, \dots, t_n ; since the derivative is a linear operator, the statement follows easily.
- (2) $q(t)$ has degree $2n$; $q'(t)$ has degree $2n - 1$. It suffices to find $2n - 1$ real roots of $q'(t)$.

Because $q(t)$ has a double root at $t = t_1, \dots, t_n$, $q'(t) = 0$ at any of these values. Furthermore, since $q(t) = 0$ at all of these values, by Rolle's theorem there is a point t at which $q'(t) = 0$ between any pair of these values.

The double roots therefore give us n roots of $q'(t)$, and there are $n - 1$ roots between them. These are our $2n - 1$ real roots. \square

Lemma 12. *Let $\kappa : t \mapsto (x(t), y(t), z(t))$ be a continuous injective map from $\mathbb{R} \rightarrow \mathbb{R}^3$ such that $x(t)$ and $z(t)$ have a finite number of crossings. Assume further that $z(t)$ is a polynomial.*

Then there exists $\hat{y}(t)$ such that $\hat{\kappa}(t) = (x(t), \hat{y}(t), z(t))$ has the same knot type as κ , and such that $\hat{y}(t)$ and $z(t)$ have a finite number of crossings.

Proof. We will construct two polynomials. The first, \tilde{y} , satisfies all conditions we would want on \hat{y} if Lemma 9 could be proved over the reals. The second, \bar{y} , is constructed from \tilde{y} and satisfies the conditions we want on \hat{y} with Lemma 9 stated as it is (over the complex numbers).

We will, as with previous constructions, require that $\tilde{y}(a) = y(a)$ and $\tilde{y}(b) = y(b)$ for every crossing (a, b) of $x(t)$ and $z(t)$ to ensure that we preserve the knot type. We will also add the condition that $\tilde{y}'(t) = 1$ for each root t of z . By construction, the real derivatives of $\tilde{y}(t)$ and of $z(t)$ are never simultaneously zero, so by Lemma 9 they have a finite number of crossings.

\bar{y} needs the additional property that the complex derivatives of $\bar{y}(t)$ and $z(t)$ are never simultaneously zero. Let z_1, \dots, z_k be the roots of $z(t)$ in $\mathbb{C} \setminus \mathbb{R}$, and let

$$q(t) = (t - t_1)^2 \cdots (t - t_n)^2,$$

where t_1, \dots, t_n are values of t for which a point or derivative (or both) was specified for $y(t)$.

By Lemma 11, $q(t)$ has the property that $\tilde{y}(t) + \alpha q(t)$ satisfies all of the conditions we placed on $\tilde{y}(t)$; furthermore, $q'(z_i) \neq 0$ for any $1 \leq i \leq k$. This means that, for each i , there is only one value of α for which $y'(z_i) + \alpha q'(z_i) = 0$, which means that there are finitely many “bad” values of α for which $y(t) + \alpha q(t)$ and $z(t)$ have a cusp in $\mathbb{C} \setminus \mathbb{R}$. We can therefore find α so that $(y(t) + \alpha q(t), z(t))$ has no cusps in $\mathbb{C} \setminus \mathbb{R}$, and so that $\tilde{y}(t) + \alpha q(t)$ satisfies all of the conditions on $\tilde{y}(t)$ (including conditions that guarantee that there are no real cusps). Therefore, $\bar{y}(t) = \tilde{y}(t) + \alpha q(t)$ satisfies all of the conditions we wanted on $\hat{y}(t)$. \square

Lemma 13. *Let $x(t)$ and $y(t)$ be piecewise linear, and assume that there is no interval over which they are simultaneously constant. Then there is a rotation of the xy -plane so that no piece of $x(t)$ is constant.*

Proof. Let x_j and y_j be the j^{th} piece of x and y , respectively, and let I_j be the interval of values of t for which these pieces apply (we may assume without loss of generality that it is the same interval for x and for y , because we can always subdivide pieces). $x_j(t)$ has the form $a_x t + b_x$ and $y_j(t)$ has the form $a_y t + b_y$.

The x component of x_j, y_j under a rotation by θ is

$$\begin{aligned} & \cos(\theta)x_j(t) - \sin(\theta)y_j(t) \\ = & (a_x t \cos \theta - a_y t \sin \theta) + (b_x \cos \theta - b_y \sin \theta). \end{aligned}$$

The latter is constant in t if and only if $a_x \cos \theta - a_y \sin \theta = 0$, and there are only two values of θ (with $0 \leq \theta < 2\pi$) for which that equation can hold (intuitively, the two rotations that would cause the line segment to be parallel to the y -axis). Since there are a finite number of pieces, there is an angle by which we can rotate the xy -plane so that no piece of $x(t)$ is constant. \square

The correct algorithm is now clear. Given a continuous injective piecewise linear parameterization $\kappa = (x(t), y(t), z(t))$ of some knot such that $x(t)$ and $y(t)$ are piecewise linear and have a finite number of crossings:

- (1) Rotate κ about the z -axis so that no piece of $x(t)$ is constant (we can find a suitable angle using Lemma 13).

- (2) Apply Lemma 8 to replace $z(t)$ with a polynomial $\hat{z}(t)$ in such a way that the knot type is preserved, and such that there are a finite number of crossings of $x(t)$ and $\hat{z}(t)$.
- (3) Apply Lemma 12 to replace $y(t)$ with a polynomial $\hat{y}(t)$ in such a way that the knot type is preserved, and such that the derivatives of $\hat{y}(t)$ and of $\hat{z}(t)$ are never simultaneously zero and there are a finite number of crossings of $\hat{y}(t)$ and $\hat{z}(t)$.
- (4) Apply Lemma 4 to replace $x(t)$ with $\hat{x}(t)$ in such a way that the knot type is preserved.

Theorem 14. *The result of applying the algorithm above is a polynomial embedding.*

Proof. Since the derivatives of $\hat{y}(t)$ and $\hat{z}(t)$ are never simultaneously zero, the derivatives of $\hat{x}(t)$, $\hat{y}(t)$, and $\hat{z}(t)$ can never be simultaneously zero.

If $\hat{y}(a) = \hat{y}(b)$ and $\hat{z}(a) = \hat{z}(b)$ but $a \neq b$, by construction either $\hat{x}(a) > \hat{x}(b)$ or $\hat{x}(a) < \hat{x}(b)$. The map is therefore injective. \square

4. OUR IMPLEMENTATION, AND EMPIRICAL OBSERVATIONS

We have implemented the naïve algorithm as a Mathematica notebook. Even though the algorithm has the possibility of getting “stuck,” it has succeeded in every case we have tried (and has produced embeddings). Instead of using interpolating polynomials, our implementation uses linear programming to minimize degrees at each step (note that this *does not* guarantee minimal-degree embeddings, as we will show below). We have also written a program to automatically generate a piecewise linear parameterization of a knot given its braid representation (doing so is quite straightforward; we will not go into the details here).

The algorithm has given very good results for low-crossing knots. From the braid notation for a trefoil it was able to produce a polynomial trefoil of degrees (3, 4, 5); from braid notation of a figure-8, the polynomials had degrees (4, 5, 6). Both triples of degrees are provably minimal (see [2]). The trefoil produced in this way is relatively nice; the figure-8 generated automatically from the braid notation, though, has extremely close crossings (it is very difficult to verify from a graph that it really is a figure-8).

One might wonder, since the naïve algorithm produced a minimum-degree trefoil and figure-8, if the embedding (if it produces one) is always of minimum degree. As it turns out, the algorithm as presented gives no such guarantees — the algorithm is “greedy” and tries to minimize the degrees of polynomials at each stage, but there are cases in which increasing the degrees of the lowest-degree polynomials lets us decrease the degree of the highest-degree polynomial. For example, consider the following xy -projection of the 6_2 knot, given by Ashley Brown[1]:

$$\begin{aligned} x(t) &= t(t^2 - 4)(t^2 - 11) \\ y(t) &= t^4 - 12t^2. \end{aligned}$$

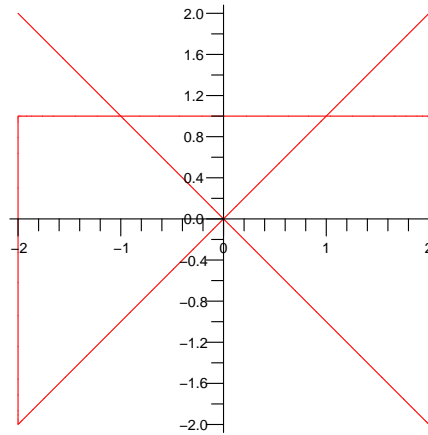
The lowest-degree polynomial $z(t)$ so that $(x(t), y(t), z(t))$ forms the 6_2 knot has degree 11 (as can be shown with linear programming). However, if we add a term of $t^6/10^3$ to $y(t)$, it is possible to find $z(t)$ of degree 9. We can therefore not expect the greedy algorithm to give minimal degrees in general.

The exact piecewise linear function that we start with seems to make a big difference — simply rotating the automatically-generated piecewise linear parameterization of the figure-8 by 90 degrees caused the algorithm to produce a qualitatively nicer parameterization (again of minimal degree). Manually drawing the piecewise linear parameterization gave an even nicer figure-8, with more space in the crossings and not straying as far from the crossings between them.

5. EXAMPLE: AUTOMATICALLY GENERATING A TREFOIL

In this section, we will demonstrate the (naïve) algorithm by running it on a piecewise linear parameterization of the trefoil knot.

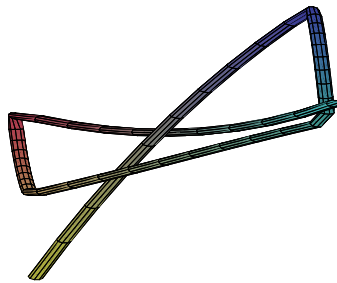
The following is a graph of the projection of the parameterization we are using:



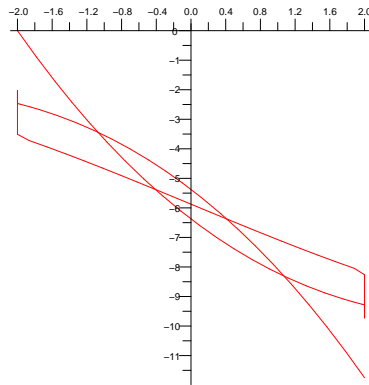
Using linear programming, we find $z(t)$ of minimal degree to produce a trefoil from this projection;

$$z(t) = -16.79444444074648t + 8.666666664760259t^2 - 1.155555553018169t^3$$

works. The following is a view of the result:



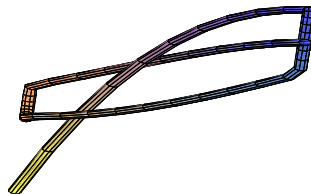
The next step is to replace $y(t)$ with a polynomial; again, we do so by computing crossings and using linear programming. Projecting the knot above to the xz -plane, we get the following:



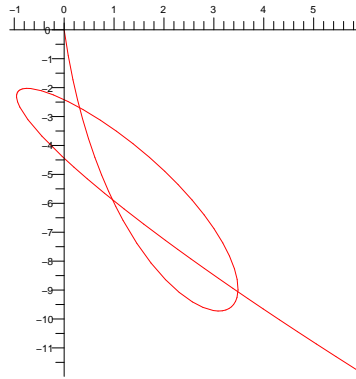
Note that there are two places in which we have an infinite number of crossings, at the top-left and bottom-right of the graph (remember that the implementation is of the naïve algorithm). However, if we ignore those regions and consider only the isolated crossings, everything works out in this case and we can proceed. In this case, we get

$$y(t) = 1.3695689767360817t + 2.444694780594364t^2 - 1.569630571850854t^3 + 0.2145508646188345t^4.$$

Here is the result (here $x(t)$ is piecewise linear; $y(t)$ and $z(t)$ are both polynomials):



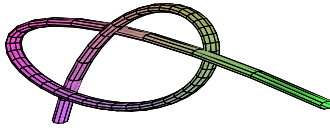
Finally, we can project the curve to the yz -plane and get the following:



and we can use

$$\begin{aligned}
 x(t) = & -0.34802129782162444t - 0.8796271508402143t^2 \\
 & -1.5391825576798006t^3 + 0.9839675055434827t^4 \\
 & -0.1314411413430609t^5
 \end{aligned}$$

to end up with our polynomial trefoil:



6. GENERALIZING THE ALGORITHM

The algorithm we have presented gives polynomial parameterizations of long knots. However, by changing it slightly, we can get other kinds of parameterizations of knots.

For example, we can change the algorithm slightly to prove that *compact* polynomial knots exist for any knot type; that is, for any knot type, we can find an injective parametric function $\hat{\kappa} : [0, 1] \rightarrow \mathbb{R}^3$ where each component is a polynomial and $\hat{\kappa}(0) = \hat{\kappa}(1)$ (we can also guarantee that $\hat{\kappa}'(0) = \hat{\kappa}'(1)$). To do so, start with a piecewise linear parameterization of a closed knot (that is, a piecewise linear function $\kappa : [0, 1] \rightarrow \mathbb{R}^3$ satisfying $\kappa(0) = \kappa(1)$). Now run the same algorithm as before, except consider only crossings in $[0, 1]$ at each step and ensure that each intermediate polynomial has the same value at 0 and at 1 (we can enforce this condition with interpolating polynomials). Because all of the lemmas used in proving the correct algorithm can easily be adapted to this new setting, we have a constructive proof that compact polynomial knots exist.

We can also choose a different set of basis functions. So far, we have used linear programming to replace a component of a parametric function with one of the form

$$\hat{z}(t) = a_1 t + a_2 t^2 + \cdots + a_n t^n$$

without changing the knot type. We could instead have found a function of the form

$$\begin{aligned} \hat{z}(t) = & a_1 \cos(t) + b_1 \sin(t) + \\ & a_2 \cos(2t) + b_2 \sin(2t) + \cdots + a_k \cos(kt) + b_k \sin(kt) \end{aligned}$$

Turning this idea into an algorithm for finding trigonometric parameterizations of knots requires some extra work: we must again ensure that the number of crossings (in each period, at least) is finite at each step.

7. ACKNOWLEDGEMENTS

The research presented here would not have been possible without Alan Durfee and Donal O'Shea, the advisors for the knot theory group at the Mount Holyoke REU, and Susan Durst, Emily List, Charles Siegel, and Katie Toman, the other members of the group.

REFERENCES

1. Ashley Brown, *Examples of polynomial knots*.
2. Alan Durfee and Donal O'Shea, *Polynomial knots*.
3. Anant Shastri, *Polynomial representations of knots*, *Tôhoku Math. J.* **44** (1992), 11–17.