Understanding math through code: an algebra project

Nicholas Gaubatz

September 20, 2023

Goals

- Main goal: to convince you through an example that coding can help you as a grad student to better understand things in your classes and research, even topics you wouldn't expect code to illuminate.
- Subgoal: to give an introduction to Riemann surfaces and their connection to function fields in algebra.
- Disclaimer: I'm not an expert. I'm showing how this project helped me go from no understanding of the topic to some understanding.

Overview of Presentation

1. Function fields
2. Riemann surfaces
3. n-sheeted covering spaces and n-sheeted branched coverings
4. Connections between all of these concepts
5. Constructing Riemann surfaces
6. My code

Note: most (if not all) of the content in this talk that is not code comes from Artin's *Algebra*.

Artin's *Algebra*

- Context: Last semester (spring 2023), I was taking Algebra II with Hal Schenck, using Artin's book, and I came across section 15.9 on function fields.
- This section didn't make sense to me and I wasn't going to devote too much time to it, but ...

The paragraph that started it all

Computing the Permutations

Given a polynomial $f(t, x)$, one wishes to determine the permutations σ_ν that define the gluing data of its Riemann surface. Two problems present themselves. First, the "local problem:" At each branch point p one must determine the permutation σ of the sheets that occurs when one circles that point. As we have seen, σ depends on the numbering of the sheets. Second, one must take care to use the same numbering for each branch point. This is the more difficult problem. A computer has no problem with it, but except in very simple cases, it is difficult to do by hand.

To compute the permutations, the computer chooses a "base point" b in the cut plane T and computes the n roots of the polynomial $f(b, x)$ numerically, with a suitable accuracy. It numbers these roots arbitrarily, say $\gamma_1, \ldots, \gamma_n$, and labels the sheets by calling S_i the sheet that contains the root γ_i. Then it walks to a point b_ν in the vicinity of a branch point p_ν, taking care not to cross any of the cuts. The roots γ_i vary continuously, and the computer can follow this variation by recomputing roots every time it takes a small step. This tells it how to label the sheets at the point b_ν. Then to determine the permutation σ_ν, the computer follows a counterclockwise loop ℓ_ν around p_ν, again recomputing roots as it goes along. Because the loop crosses the cut C_ν, the roots will have been permuted by σ_ν when the path returns to b_ν. In this way, the computer determines σ_ν. And because the numbering has been established at the base point b, it will be.the same for all of the branch points.

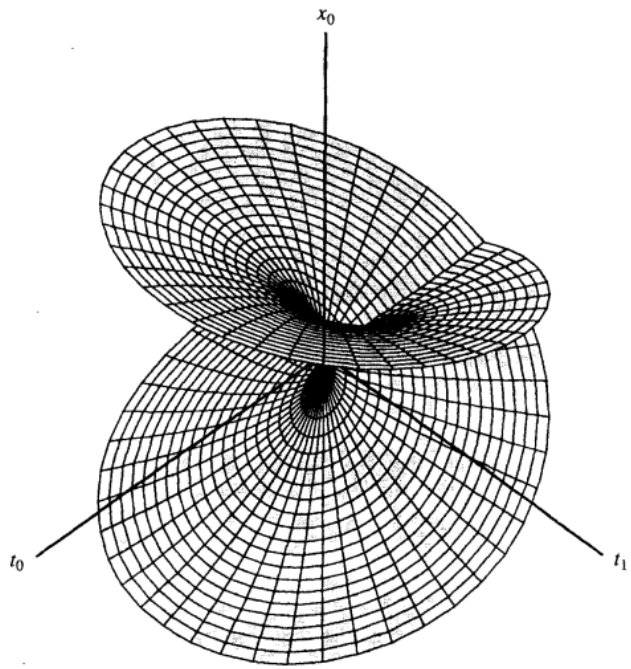
Needless to say, doing this by hand is incredibly tedious. We find ways to get around the problem in the examples we present below.

Interesting...

This page promises that a computer is much more efficient than us in computing data for these abstract objects. I decided to sit down and try to figure out what's going on.

Function fields

Why do we care about Riemann surfaces in an algebra class?



They're connected to function fields, a type of field extension.

Function fields

Definition

Function fields are extensions of the field $\mathbb{C}(t)$ of rational functions with complex coefficients.

For example, if $F = \mathbb{C}(t)$, $F(\sqrt{2}) \cong F[x]/\langle x^2 - 2\rangle$. Example elements:

$$\sqrt{2} \quad , \quad \frac{i}{2+i} + i\sqrt{2} \quad , \quad 1 + \frac{2}{i}\sqrt{2} \quad , \quad \dots$$

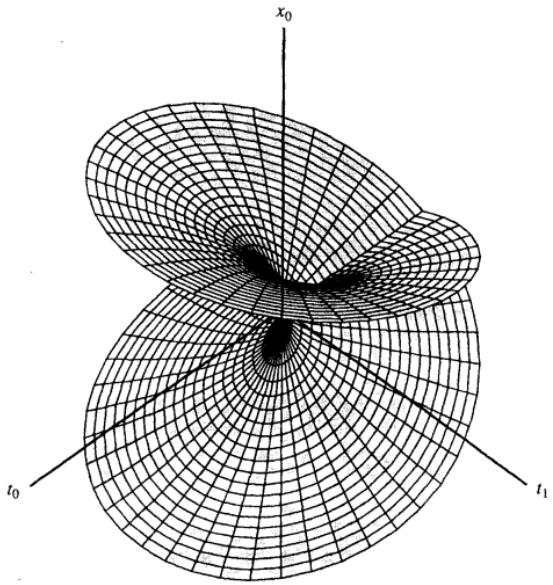
Our goal: understand isomorphism classes of these fields.

Riemann surfaces

Definition

The Riemann surface of a polynomial $f(t, x)$ is the locus of zeros of f in $\mathbb{C}^2$.

For example: the Riemann surface of $x^2 - t$ is below (in 3-space):



n-sheeted covering spaces

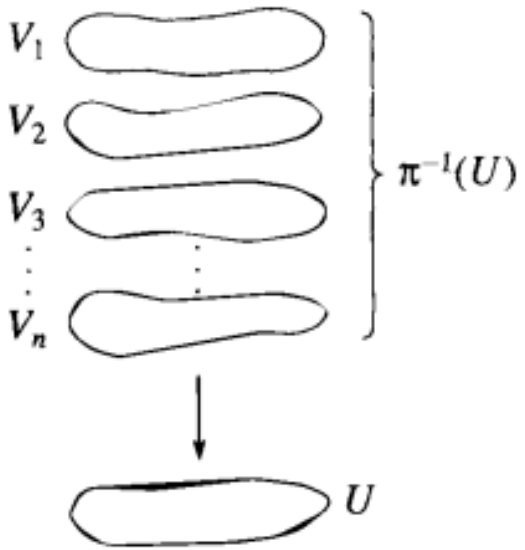
We'll describe Riemann surfaces here as *branched coverings* involving n-*sheeted covering spaces*.

Definition

*Let X and T be Hausdorff spaces. A continuous map $\pi : X \to T$ is an n-***sheeted covering space*** if every fiber consists of n points, and if it has this property: let x_0 be a point of X and let $\pi(x_0) = t_0$. Then π maps an open neighborhood U of x_0 in X homeomorphically to an open neighborhood V of t_0 in T.*

n-sheeted covering spaces

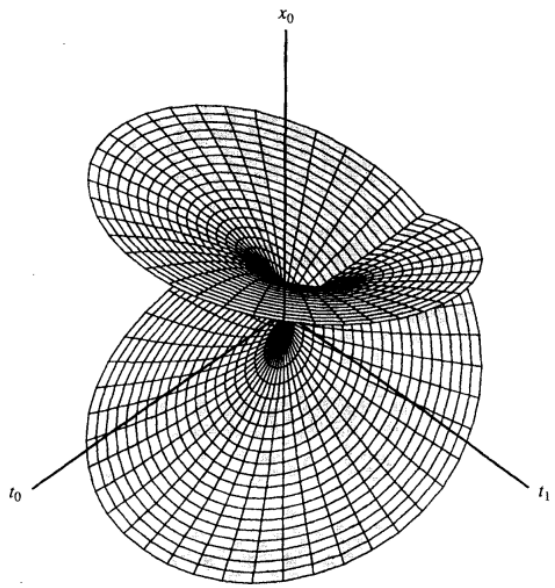
Here's an example of an n-sheeted covering space:



Branched coverings

Definition

A map $\pi : X \to T$ (where T is the complex plane) is an ***n*-sheeted branched covering** *if X contains no isolated points, the fibers of π are finite, and there is a finite set Δ of points of T called* **branch points**, *such that the map $(X - \pi^{-1}\Delta) \to (T - \Delta)$ is an n-sheeted covering space.*



The connection

Theorem

Let $f(t, x)$ be an irreducible polynomial in $\mathbb{C}[t, x]$ which has positive degree n in the variable x. The Riemann surface of f is an n-sheeted branched covering of the complex plane T.

The Riemann Existence Theorem

Theorem (Riemann Existence Theorem)

There is a bijective correspondence between isomorphism classes of function fields of degree n over $\mathbb{C}(t)$ and isomorphism classes of connected, n-sheeted branched coverings of T, such that the class of the field extension K defined by an irreducible polynomial $f(t, x)$ corresponds to the class of its Riemann surface X.

i.e., If we know that the Riemann surfaces of two degree n polynomials are "similar," then their function fields are isomorphic.

How can we make sense of these Riemann surfaces?

Constructing Riemann surfaces

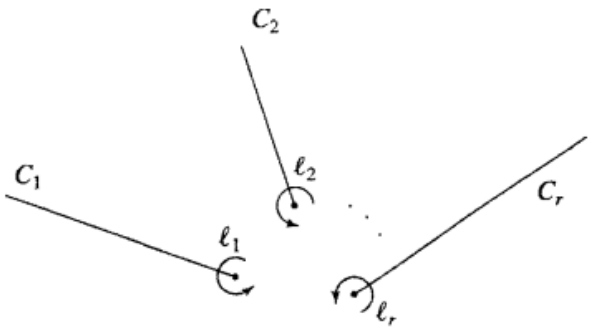
- Fact: every n-sheeted branched covering $X \to T$ is isomorphic to one constructed by a "cut-and-paste process."
- Note that if we take the Riemann surface of $x^2 - t$ and cut (separate) along the "double locus" (the negative real t-axis), we get two surfaces that each project bijectively to T.
- We can turn this around:

Constructing Riemann surfaces (cont.)

- Stack two copies S_1, S_2 of the complex plane over T and cut them open along the negative real axis.
- Label the two sides of the cut on each sheet as side A and side B.
- Glue side A of sheet S_1 to side B of sheet S_2, and vice versa (can't do this in 3-space).

The "cut-and-paste process"

- In general, take each branch point p_ν in T (there can be multiple), construct nonintersecting half-lines (or rays) C_ν from p_ν to infinity, and cut open both T and the n sheets of X over these lines.



The Cut Plane T.

The "cut-and-paste process" (cont.)

- Then we label the n sheets of X as $S_1, \ldots, S_n$ and stack them over T.
- On the complex plane T, we make a loop ℓ_ν that circles a branch point p_μ in the counterclockwise direction.
- Label the side of C_ν we pass through first as side A and the other side as side B.
- Label the corresponding sides of sheet S_i as side A_i and side B_i.

Gluing

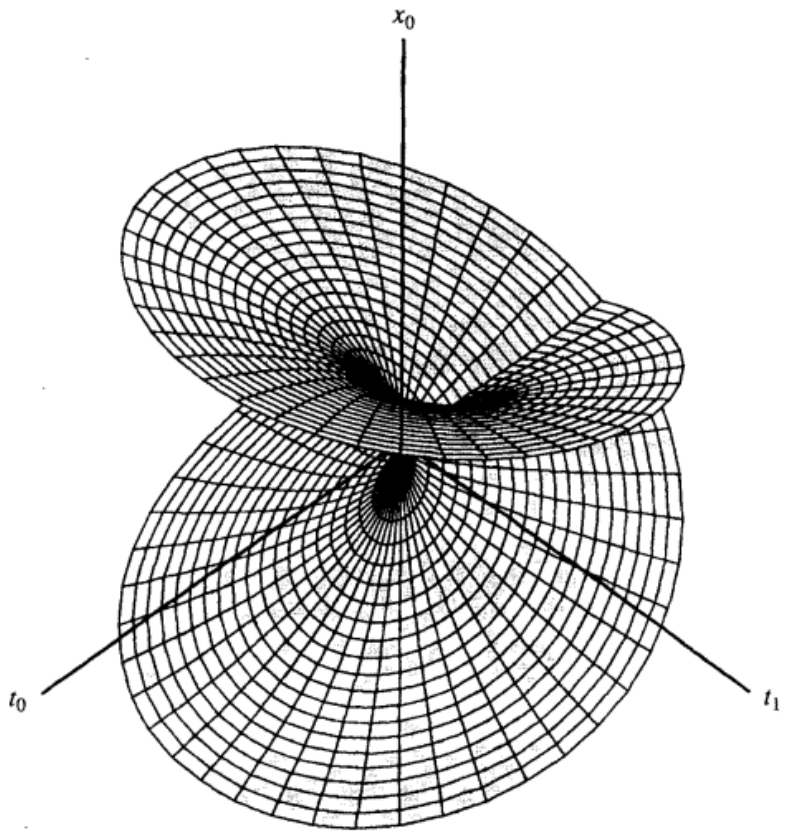
Gluing X amounts to gluing each side A_i to some side B_j.

Thus, we can describe a Riemann surface with two pieces of information:

- Branch points $p_1, \ldots, p_k$
- Gluing data: permutations $\sigma_1, \ldots, \sigma_k$ (around branch point p_ν, side A_ν is glued to side $B_{\sigma_\nu(i)}$).

Example

For our example $f(t, x) = x^2 - t$, there is one branch point $p_1 = (0, 0)$, and gluing data $\sigma_1 = (12)$.



The algorithm

Computing the Permutations

Given a polynomial $f(t, x)$, one wishes to determine the permutations σ_ν that define the gluing data of its Riemann surface. Two problems present themselves. First, the "local problem:" At each branch point p one must determine the permutation σ of the sheets that occurs when one circles that point. As we have seen, σ depends on the numbering of the sheets. Second, one must take care to use the same numbering for each branch point. This is the more difficult problem. A computer has no problem with it, but except in very simple cases, it is difficult to do by hand.

To compute the permutations, the computer chooses a "base point" b in the cut plane T and computes the n roots of the polynomial $f(b, x)$ numerically, with a suitable accuracy. It numbers these roots arbitrarily, say $\gamma_1, \ldots, \gamma_n$, and labels the sheets by calling S_i the sheet that contains the root γ_i. Then it walks to a point b_ν in the vicinity of a branch point p_ν, taking care not to cross any of the cuts. The roots γ_i vary continuously, and the computer can follow this variation by recomputing roots every time it takes a small step. This tells it how to label the sheets at the point b_ν. Then to determine the permutation σ_ν, the computer follows a counterclockwise loop ℓ_ν around p_ν, again recomputing roots as it goes along. Because the loop crosses the cut C_ν, the roots will have been permuted by σ_ν when the path returns to b_ν. In this way, the computer determines σ_ν. And because the numbering has been established at the base point b, it will be.the same for all of the branch points.

Needless to say, doing this by hand is incredibly tedious. We find ways to get around the problem in the examples we present below.

My code

Let's examine my program. I'll give an overview of each of the steps, and maybe dive deeper into some of the less complicated functions.

What I want you to take away

- If you don't understand everything, that's ok!
- It took me spending some hours on this project to understand the pieces I do now.
- I want you to be curious about things you encounter. Ask yourself if coding up some examples or algorithms can help your understanding.

Thank you!