Computing zeta functions of algebraic curves using Harvey's trace formula

Madeleine Kyng
UNSW

ANTS-XV
University of Bristol
August 2022

The problem

Develop a practical point-counting algorithm that can take as input a completely arbitrary curve.

- **Input**: An absolutely irreducible polynomial $\bar{F} \in \mathbb{F}_q[x, y]$ defining a plane curve X.
- **Output**: The zeta function $Z_{\widetilde{X}}(T)$ of the nonsingular projective curve $\widetilde{X}$ that has the same function field as X.

Many practical algorithms have been developed for specific classes of curves (e.g., elliptic, hyperelliptic, nondegenerate).

Schoof's algorithm

The best-known point-counting algorithm is **Schoof's algorithm** for elliptic curves.

This was the first polynomial-time algorithm for point-counting on elliptic curves.

> An ℓ-**adic** algorithm — we count $\mathbb{F}_q$-points on E by computing $\mathrm{tr}(\phi) \pmod{\ell}$ modulo enough small primes ℓ to recover $\#E(\mathbb{F}_q)$ via CRT.
>
> Has time complexity $\widetilde{O}((\log q)^5)$.

Descendants of Schoof have time complexity $\log(q)^{C(g)}$
— impractical for curves of genus $g > 2$.

Kedlaya's algorithm

For $g > 2$, we would use a **Kedlaya-style *p*-adic algorithm** when applicable.

Kedlaya's algorithm [2001] applied to hyperelliptic curves.

> Kedlaya's algorithm has time complexity $\widetilde{O}(g^4 n^3 p)$.

Kedlaya's algorithm was soon generalised to work for larger classes of curves; **superelliptic curves**, C_{ab} **curves**, **nondegenerate plane curves**.

Descendents of Kedlaya have time complexity polynomial in g and $n = \log_p(q)$.

Tuitman's algorithm

Tuitman's algorithm [2016] is the most general of the Kedlaya-style algorithms.
It can be applied to any $\bar{F}$ for which a "good" lift to characteristic zero is provided.

- **Input**: A "good lift" $F \in K[x,y]$ of $\bar{F} \in \mathbb{F}_q[x,y]$,

  where K is a degree n number field in which p is inert,

  defining a nonsingular curve $\widetilde{X}$ over the field $\mathbb{Z}_K/p\mathbb{Z}_K \cong \mathbb{F}_q$.

- **Output**: The zeta function $Z_{\widetilde{X}}(T)$ of $\widetilde{X}$.

- **Complexity**: $\widetilde{O}(pd_1^6 d_2^4 n^3)$ where $d_1 = \deg_y(F)$ and $d_2 = \deg_x(F)$.

Applicability of Tuitman

Lifting problem

Given $\bar{F} \in \mathbb{F}_q[x, y]$, how does one find a lift $F \in K[x, y]$ for Tuitman?

For $p > 2$ there always exists a "good" lift to $\mathbb{Z}_q$ (not K), but in some cases it is difficult to compute.

Limitation of Tuitman's algorithm:

At present, Tuitman's algorithm cannot handle every $\bar{F}$, because there is no known method for computing a "good" lift for an **arbitrary** $\bar{F}$.

Recent progress on lifting

When $\bar{F}$ defines a non-singular curve in the toric surface associated with $\Delta(\bar{F})$, a naive lift of $\bar{F}$ almost always works.

Castryck, Tuitman and Vermuelen expanded the class of curves that Tuitman can deal with.

Castryck and Tuitman [2017] developed procedures for lifting curves of genus $g \leq 5$.

Castryck and Vermeulen [2020] developed procedures for lifting $\bar{F}$ with $\deg_y(\bar{F}) \leq 5$.

The new algorithm

Our algorithm:

- Can accept **any** input $\bar{F}$.
- Has time complexity competitive with Tuitman's (though exponents are worse).
- Is relatively easy to understand and implement.

Main theorem

The new algorithm has the following properties:

- **Input**: *An absolutely irreducible polynomial* $\bar{F} \in \mathbb{F}_q[x, y]$ *defining a plane curve* X.
- **Output**: *The zeta function* $Z_{\widetilde{X}}(T)$ *of the nonsingular projective curve* $\widetilde{X}$ *that has the same function field as* X.
- **Complexity**: $\widetilde{O}(d^{c_2} n^{c_1} p^{\frac{1}{2}})$ *where* $q = p^n$, $d = \deg(\bar{F})$, *and* c_1, c_2 *are positive real constants.*

Example 1

Consider one of the examples with $\deg_y(\bar{F}) = 5$ from my paper.

The curve has $g = 12, q = p = 101$, $d_1 = 5$, $d_2 = 35$, with Newton polygon:

On a computer with an Intel i9-12900K CPU, 128GB RAM, and RTX3090 GPU,

- Tuitman's code fails to compute $Z_{\widetilde{X}}(T)$ within 12 hours.
- Our code takes 2.3 minutes to compute $Z_{\widetilde{X}}(T)$.

Example 2

Consider one of the examples with $\deg_y(\bar{F}) = 8$ and $g > 5$ from my paper.

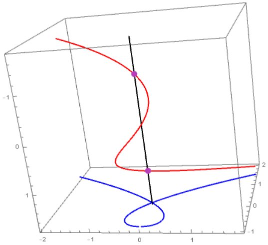
The curve has $g = 13, q = p = 13,\ d_1 = 8,\ d_2 = 32$, with Newton polygon:

We do not know how to find a good lift of this curve for Tuitman's algorithm.

On a computer with an Intel i9-12900K CPU, and RTX3090 GPU,

- Our code takes 10.5 minutes to compute $Z_{\widetilde{X}}(T)$.

The new algorithm



The new algorithm is composed of two sub-algorithms:

1. **CountPlaneModel**: count points on the plane model via **Harvey's trace formula**.
2. **ComputeCorrections**: determine the difference in point-counts for X and $\widetilde{X}$.

Step 1: Count points with Harvey's trace formula

Theorem

Let $\bar{F} \in \mathbb{F}_p[x, y]$, and let X be the curve cut out by $\bar{F}$. Let $\Gamma = \Delta(\bar{F})$.

Let k, λ be positive integers, assume $p > \frac{\lambda}{k}$, and let F be a lift of $\bar{F}$ to $\mathbb{Z}_p$ with $\Delta(F) = \Gamma$.

We have

$$|(X \cap \mathbb{T}^2)(\mathbb{F}_{p^k})| = (p^k - 1)^2 \sum_{s=0}^{\lambda} (-1)^s \binom{\lambda}{s} \operatorname{tr}(M_s^k) \pmod{p^\lambda},$$

where

$$(M_s)_{u,v} = [F^{(p-1)s}]_{pv-u}, \quad u, v \in s\Gamma.$$

.

Proof of trace formula

The idea of the proof is to set up an indicator function

$$H : (\mu_{p^k-1})^2 \to \mathbb{Z}/p^\lambda\mathbb{Z}$$

that indicates whether $(a, b) \in (\mu_{p^k-1})^2$ is a lift of a point in $X(\mathbb{F}_{p^k})$.

If $p > \frac{\lambda}{k}$, then $H = (1 - F^{p^k-1})^\lambda$ works.

This gives

$$\begin{aligned} N_k &= (p^k-1)^2 \sum_{s=0}^{\lambda} (-1)^s \binom{\lambda}{s} \left(\sum_{w \in \mathbb{Z}^2} [F^{s(p^k-1)}]_{(p^k-1)w} \right) \pmod{p^\lambda} \\ &= (p^k-1)^2 \sum_{s=0}^{\lambda} (-1)^s \binom{\lambda}{s} \operatorname{tr}(M_s^k) \pmod{p^\lambda} \end{aligned}$$

Implementation of trace formula

A straightforward implementation of Harvey's trace formula lets us compute

$$|X(\mathbb{F}_q)| \bmod p^\lambda, \ldots, |X(\mathbb{F}_{q^R})| \bmod p^\lambda$$

in time

$$\widetilde{O}(R\, n^2\, p^2\, \lambda^8\, \mathrm{Vol}(\Gamma)^3)$$

This time complexity assumes we compute the entries of the matrices M_s by computing the polynomials $F^{(p-1)s}$ in their entirety.

The p^2 can be improved to $p^{\frac{1}{2}}$ by using Harvey's deformation recurrences.

Step 2: Make corrections

We can efficiently count points on the plane model.
We now see how we can make corrections.

One approach to doing this is to first compute $Z_X(T)$, and then remove factors from the numerator whose roots have the wrong absolute value to get $Z_{\widetilde{X}}(T)$.

We could compute $Z_X(T)$ by counting points in extensions of degree up to Bombieri's bound.

We will do better than this.

Relationship between curves

Let $\overline{X}$ be the projective closure of the affine plane curve X defined by $\bar{F}$.

The normalisation morphism $\pi : \widetilde{X} \to \overline{X}$ restricts to an isomorphism

$$\pi : \widetilde{X} \setminus \pi^{-1}(S) \to \overline{X} \setminus S$$

where S is the subscheme of singular points on $\overline{X}$.

Thus we can compute corrections by counting points of certain 0-dimensional subschemes of $\overline{X}$ and $\widetilde{X}$ related to S.

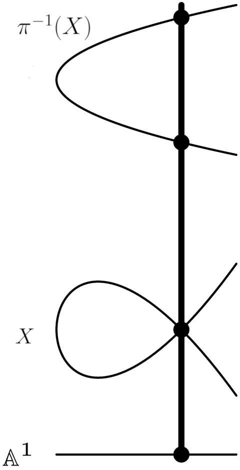
Computing corrections

To find the points on $\widetilde{X}$ lying above S we take advantage of existing **factorisation algorithms**, namely the Montes algorithm.

Given $\bar{p}(x) \in \mathbb{F}_q[x]$ and a function field $L = \mathbb{F}_q(x)[y]/(\bar{F})$, the Montes algorithm efficiently finds the factorisation of $\bar{p}(x)$ in the integral closure $\mathcal{O}$ of $\mathbb{F}_q[x]$ in L.

Complexity of Montes: $\widetilde{O}(d_1^3 + d_1^2 \log(q))$.

Computing corrections via factorisation



$$
\begin{tikzcd}
\mathbb{F}_q[\pi^{-1}(X)] \\
\mathbb{F}_q[X] \arrow[u, "\iota"'] \\
\mathbb{F}_q[x] \arrow[u, "\iota"']
\end{tikzcd}
$$

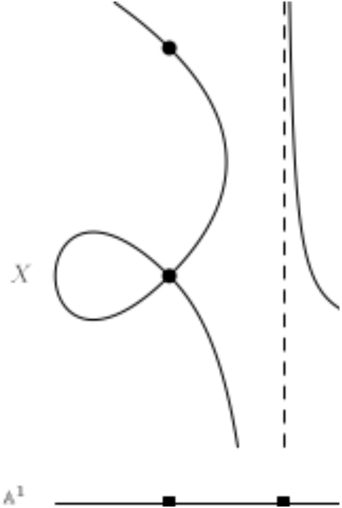
Consider the situation for the affine curve.

For simplicity, assume that $\bar{F}$ is monic in y.

The integral closure $\mathcal{O}$ of $\mathbb{F}_q[x]$ in L is the co-ordinate ring of $\pi^{-1}(X)$.

The prime ideal factors of $(x - x_0)$ in $\mathcal{O}$ are exactly the points on $\pi^{-1}(X)$ whose x-coordinate is x_0.

Computing corrections via factorisation



Let Y be the subscheme of $\mathbb{P}^1$ containing ∞, the zeros of $\overline{a_m}(x)$, and all x-coordinates of singular points of X.

We define $Z \subseteq \overline{X}$ to be the points in $\overline{X} \setminus X$ together with the points on X whose x-coordinate belongs to $Y \setminus \{\infty\}$.

We define $\widetilde{Z} \subseteq \widetilde{X}$ to be the points in $\widetilde{X}$ whose x-coordinate belongs to Y.

By our choice of Y and Z, we have

$$\widetilde{Z} = \pi^{-1}(Z).$$

Summary of our algorithm

The algorithm works as follows:

1. **Trace formula**: Count points on X in extensions of degree $k = 1, \dots, g$

   to p-adic precision $\lambda = \lfloor ag/2 + \log_p(4g) \rfloor + 1$.

2. **Resultant and GCD**: Locate the set Y of "bad" points on $\mathbb{P}^1$ for this X.

3. **Factorise polynomials over finite fields**: Adjust the point-counts from Step 1 to **remove** the points on X that lie above Y.

4. **Factorise primes in function field**: Adjust the point-counts from Step 3 to **add** the points on $\widetilde{X}$ that lie above Y.

5. Compute the zeta function $Z_{\widetilde{X}}(T)$.

Thanks for listening!