A quick survey of average polynomial time point counting for curves

David Harvey

University of New South Wales

7th June 2017
PIMS Workshop on Computational Arithmetic Geometry
Simon Fraser University

The zeta function

Definition

Let X = smooth projective curve of genus g over $\mathbf{F}_p$.
The *zeta function* of X is the power series

$$Z(T) = \exp\left(\sum_{k=1}^{\infty} \frac{|X(\mathbf{F}_{p^k})|}{k} T^k\right) \in \mathbf{Q}[[T]].$$

It is actually a rational function of the form

$$Z(T) = \frac{L(T)}{(1-T)(1-pT)}$$

where $L(T) \in \mathbf{Z}[T]$ has degree $2g$.

Knowledge of $Z(T)$ is equivalent to knowledge of $L(T)$.

It is effectively computable: enough to compute $|X(\mathbf{F}_p)|, \ldots, |X(\mathbf{F}_{p^g})|$.

Example

Let X be the genus two hyperelliptic curve with affine equation

$$y^2 = x^5 + x + 1$$

over $\mathbf{F}_p$ where $p = 1000003$.

Then

$$|X(\mathbf{F}_p)| = 1000329, \qquad |X(\mathbf{F}_{p^2})| = 1000007333965,$$

which implies that

$$Z(T) = \frac{L(T)}{(1-T)(1-pT)}$$

where

$$L(T) = 1 + 325\,T + 719790\,T^2 + 325pT^3 + p^2T^4.$$

Global case

Now consider a smooth projective curve X of genus g over $\mathbf{Q}$.

Let X_p = reduction of X modulo p.

For all but finitely many primes, this reduction makes sense and yields a smooth projective curve of genus g over $\mathbf{F}_p$. For the rest of the talk, we ignore the "bad" primes.

Let $L_p(T)$ = corresponding L-polynomial for X_p.

Problem

Given curve $X/\mathbf{Q}$ and a bound N, compute $L_p(T)$ for all good $p < N$.

Applications: study Sato–Tate distributions, BSD conjecture.

Typically N is around 2^{20} or 2^{30}.

Counting points, one prime at a time

Some possible algorithms:

1. Naive point enumeration up to $\mathbf{F}_{p^g}$.
   Complexity $p^{O(g)}$.
2. Shanks–Mestre baby-step/giant-step.
   Complexity $p^{O(g)}$ (with better big-O constant).

These bounds are exponential in both g and $\log p$.

BSGS is quite effective in practice for small genus (especially $g \leq 2$) for a wide range of p. Highly optimised implementation `smalljac` by Sutherland.

Counting points, one prime at a time

3. Schoof–Pila.
   Complexity $(\log p)^{C_g}$ where C_g grows exponentially with g.
4. Kedlaya-type algorithms.
   Complexity $g^{O(1)} p^{1/2+\epsilon}$ (exponent of g depends on class of curve)

Polynomial in $\log p$ or g, but not both.

Major open problem: is it possible to obtain complexity polynomial in both g and $\log p$?

Schoof–Pila not competitive in the range of p under consideration.

Counting points, all primes simultaneously

Theorem (H. 2015, *Computing zeta functions of arithmetic schemes*)

Let X be a scheme of finite type over $\mathbf{Z}$. One may compute $Z_p(T)$ for all $p < N$ in time $O(N \log^{3+\epsilon} N)$.

Complexity is $O(\log^{4+\epsilon} N)$ on average per prime, where implied constant depends on X.

For curves, the dependence on g is essentially polynomial.

Algorithm spends most of its time multiplying matrices with huge integer entries (the *accumulating remainder tree*). Will not discuss further today.

The algorithm is not implemented and almost certainly not practical (although see recent work of Elsenhans–Jahnel where they use some ideas from this paper to handle some singular K3 surfaces...)

Plan for rest of the talk

I will discuss the status of *practical* average polynomial time algorithms for various classes of curves:

- in some cases we have actual implementations and you can read about them
- in some cases the algorithm is implemented but not yet written down
- in some cases the algorithm is written down but not yet implemented
- in some cases the algorithm is known, but is neither written down nor implemented
- in some cases we do not yet know a good algorithm.

I will *not* explain much about how these algorithms actually work.

Hyperelliptic curves

What hasn't been implemented

Let $g \geq 1$ and consider a hyperelliptic curve $y^2 = f(x)$ where $f \in \mathbf{Z}[x]$ is monic, squarefree, of degree $2g + 1$.

Theorem (H. 2014)

One may compute $L_p(T)$ for all good $p < N$ in time $g^{8+\epsilon} N \log^{3+\epsilon} N$.

This was the first average polynomial time algorithm published.

This algorithm is not implemented (and never will be).

Essentially a globalised version of Kedlaya's algorithm.

Note: $\log^{3+\epsilon} N$ term is really $\log^2 N \log^{1+\epsilon}(BN)$ where B is a bound for the absolute value of the coefficients of f. For the rest of the talk I will assume $\log B = O(\log N)$.

What hasn't been implemented

The horrendous $g^{8+\epsilon}$ term can be improved to a more manageable $g^{4+\epsilon}$, by using some ideas from my 2015 paper plus a few extra tricks.

(Joint work with Sutherland and Tuitman, following some discussions around 2014.)

This is not written down anywhere yet.

With high probability, a toy implementation exists somewhere on my computer.

An efficient implementation does not yet exist.

I expect it will be practical for moderate genus, say up to $g \leq 10$.

The Hasse–Witt matrix

So what has actually been implemented for hyperelliptic curves?

Consider a hyperelliptic curve X with equation $y^2 = f(x)$ over $\mathbf{Z}$.

Definition

The *Hasse–Witt* matrix for X_p is the $g \times g$ matrix over $\mathbf{F}_p$ with entries

$$(W_p)_{ij} = (f^{(p-1)/2})_{pi-j}, \qquad 1 \leq i, j \leq g.$$

(Here f_k^n means the coefficient of x^k in f^n.)

Can be interpreted as the matrix of the Cartier operator on the space of differentials on the curve over $\mathbf{F}_p$, with respect to a particular basis.

The Hasse–Witt matrix

The Hasse–Witt matrix determines the zeta function *modulo* p:

$$L_p(T) = \det(I - TW_p) \pmod{p}.$$

In particular, if p is not too small compared to g, then the Hasse–Witt matrix determines the *trace of Frobenius* and hence $|X(\mathbf{F}_p)|$.

Theorem (H. & Sutherland, 2016)

We may compute W_p for all $p < N$ in time $g^3 N \log^{3+\epsilon} N$.

The idea is to write down recurrences for the desired f_k^n, make them "independent of p", and feed them into the accumulating remainder tree.

We wrote an efficient implementation in C.

Timings for genus 2

Knowledge of $L_p(T) \pmod p$ determines $L_p(T) \in \mathbf{Z}[T]$ up to $O(1)$ possible candidates.

Can lift to correct $L_p(T)$ with $O(1)$ additional work (group computations in Jacobian).

Time to compute $L_p(T)$ for all $p < 2^{30}$:

Old code (`smalljac`)	1.4 years
New code	1.3 days

Note: `smalljac` is trivial to parallelise. New algorithm is much harder. This is work in progress.

Timings for genus 3

Knowledge of $L_p(T) \pmod p$ determines $L_p(T) \in \mathbf{Z}[T]$ up to $O(p^{1/2})$ possible candidates.

Can lift to correct $L_p(T)$ with $O(p^{1/4+\epsilon})$ additional work. In theory this dominates the computation for large enough p, but in practice it is quite fast (cf. Sutherland's awesome code).

Time to compute $L_p(T)$ for all $p < 2^{30}$:

Old code (`hypellfrob`)	3.8 years
New code	4.0 days

Timings for genus 10

Here we compute only $L_p(T) \pmod p$. Lifting step far too expensive.

Time to compute $L_p(T) \pmod p$ for all $p < 2^{24}$:

Old code (`hypellfrob`)	27 days
New code	3 days

Note: I believe g^3 can be improved to g^2. This is optimal in the sense that the Hasse–Witt matrix has g^2 entries. Therefore I expect we can improve these times by a factor of $O(10)$. A toy implementation exists (I think).

Other genus 3 curves

Classification of genus 3 curves over $\mathbf{Q}$

Every genus 3 curve over $\mathbf{Q}$ is one of the following:

- A smooth plane quartic
- A plain vanilla hyperelliptic curve as discussed earlier
- *Geometrically hyperelliptic case*: a curve that is hyperelliptic over $\bar{\mathbf{Q}}$ but does not have a plain vanilla hyperelliptic model over $\mathbf{Q}$.

Geometrically hyperelliptic case

Equations are

$$0 = g(X, Y, Z) \qquad \text{(homogeneous quadratic over } \mathbf{Z})$$
$$w^2 = f(X, Y, Z) \qquad \text{(homogeneous quartic over } \mathbf{Z})$$

Can generalise previous algorithm for computing the Hasse–Witt matrix (H., Massierer, Sutherland, 2016):

- Find quadratic extension $K/\mathbf{Q}$ where conic acquires rational point
- Convert to plain vanilla hyperelliptic model over K
- Run average polynomial time algorithm over O_K
- Yields $L_p(T)L_p(-T) \pmod{p}$ for all $p < N$
- Work slightly harder in lifting algorithm to get $L_p(T) \in \mathbf{Z}[X]$

Geometrically hyperelliptic case

Time to compute $L_p(T) \pmod p$ for all $p < 2^{30}$:

Old code (`hypellfrob`)	3.1 years
New code	23 days

The new code used 480 GB RAM.

Smooth plane quartics

This is an ongoing project with Sutherland.

If the curve is $f(X, Y, Z) = 0$, then the Hasse–Witt matrix is

$$(f^{p-1})_{pu-v} \pmod{p}, \qquad u, v \in \{(1,1,2), (1,2,1), (2,1,1)\}.$$

Note that f^{p-1} has $O(p^2)$ coefficients; we only need 9 of them.

We have developed recurrences for extracting the desired coefficients. This is more involved than before because it is a multivariate problem.

Current status: nothing written down yet. Efficient C implementation exists; can compute the trace. Cannot lift to $L_p(T) \in \mathbf{Z}[T]$ yet.

Code is being used by Sutherland and collaborators to study Sato–Tate distributions.

What next?

More general curves

Assume we are given as input a plane model of a curve over $\mathbf{Q}$.

We may assume the only singularities are ordinary double points.

One approach (ongoing project with Edgar Costa):

- Use trace formula from 2015 paper to count number of points on plane model; then analyse singularities modulo p to work out correction term.
- Need to assume model is "nondegenerate".
- Currently trying to do this one prime at a time. Some code exists, not optimised yet. Seems potentially very fast.
- Some unresolved issues with p-adic precision loss.
- Curve over $\mathbf{F}_p$ inherits singularities from $X/\mathbf{Q}$, so all primes are "bad". Some primes are "worse", i.e., have worse singularities. For these primes we probably need to change models over $\mathbf{Q}$.

More general curves

Another possible approach:

- Jan Tuitman has a generalisation of Kedlaya's algorithm that uses an arbitrary map from the curve to $\mathbf{P}^1$.
- His method works for a wide range of curves (I don't completely understand the conditions.)
- It is not implausible that his method can be adapted to average polynomial time. Not quite sure yet. Jan is thinking about it and we have had a few conversations.

Thank you!