An update on p-adic point counting

Jan Tuitman

KU Leuven

13/02/2018

Introduction

Zeta functions

Let $X/\mathbb{F}_q$ be an algebraic variety over finite field.

Definition (zeta function)

$$Z(X,T) = \exp\left(\sum_{i=1}^{\infty} |X(\mathbb{F}_{q^i})| \frac{T^i}{i}\right).$$

$|X(\mathbb{F}_{q^i})|$ denotes number of points on X with values in $\mathbb{F}_{q^i}$.

Theorem (Weil conjectures)

Suppose that X is smooth and projective of dimension m. Then:

- $Z(X,T) = \frac{P_1 P_3 \ldots P_{(2m-1)}}{P_0 P_2 \ldots\ldots P_{2m}}$, *where* $P_i = \prod_j (1 - \alpha_{ij} T) \in \mathbb{Z}[T]$.
- *The transformation* $t \to q^m/t$ *maps the* $\alpha_{i,j}$ *to the* $\alpha_{2m-i,j}$.
- $|\alpha_{i,j}| = q^{i/2}$ *for all* j, *where* $|\cdot|$ *denotes the complex absolute value.*

Computing zeta functions

Problem

Compute $Z(X, T)$ efficiently.

Naive algorithm: compute enough of the $X(\mathbb{F}_{q^i})$ by trying all values. Far too slow to be useful in practice!

For curves:

- Schoof's ℓ-adic method: in theory for all curves (Pila), in practice only for elliptic curves (genus 2 with some extra structure: Gaudry, Kohel, Smith, Schost,...).
- Generic group methods (Baby Step Giant Step, Sutherland). Better than naive counting, but still restricted to small fields and low genus.

We will not say anything more about these and focus on p-adic algorithms in the rest of this talk.

Applications

1) Discrete Logarithm Problem in cryptography:

Let $X/\mathbb{F}_q$ an algebraic curve, J its Jacobian variety, so $J(\mathbb{F}_q)$ is a finite abelian group.

DLP: given $P, Q \in J(\mathbb{F}_q)$ find $n \in \mathbb{Z}$ such that $n * P = Q$. Weak if $|J(\mathbb{F}_q)|$ only has small prime factors. However, $|J(\mathbb{F}_q)|$ can be deduced easily from $Z(X, T)$, by evaluating its numerator at 1.

2) Compute invariants of varieties and data about conjectures, e.g.:

- Picard numbers of K3 surfaces,
- (conjectures about) L-functions,
- Sato-Tate and other distributions,
- Langlands program, modularity.

p-adic cohomology

Let:

- X be an algebraic variety of dimension m over $\mathbb{F}_q$ with $q = p^a$,
- $\mathbb{Q}_q$ the unique unramified extension of degree a of $\mathbb{Q}_p$,
- $\mathbb{Z}_q$ its valuation ring,
- $\sigma \in \mathrm{Gal}(\mathbb{Q}_q/\mathbb{Q}_p)$ the unique lift of $x \mapsto x^p$ in $\mathrm{Gal}(\mathbb{F}_q/\mathbb{F}_p)$.

Fact (Lefschetz formula)

Can define p-adic cohomology groups $H^i_{rig}(X)$ and $H^i_{rig,c}(X)$: finite dimensional $\mathbb{Q}_q$ vector spaces with σ-semilinear action of p-th power Frobenius map F_p on X (sending coordinates to p-th powers) such that

$$Z(X,T) = \prod_{i=0}^{2m} \det(1 - TF_p^a | H^i_{rig,c}(X))^{(-1)^{i+1}}.$$

Curves

Kedlaya's algorithm

Let:

- $\mathbb{F}_q$ a finite field of odd characteristic with $q = p^a$,
- $Q \in \mathbb{F}_q[x]$ monic of degree $2g+1$ without repeated roots,
- X smooth projective curve of genus g defined by $y^2 = Q(x)$.

Theorem (Kedlaya, 2001)

Can determine $Z(X,T)$ by computing $H^1_{rig}(X)$ with action of F_p in time

$$O((pg^4a^3)^{1+\epsilon}).$$

Completely implemented in Magma by Harrison (also even degree models and characteristic 2). Something available in SAGE and PARI as well...

Note: input size is about $\log(p)ga$, so complexity in p is quite bad. All p-adic algorithms suffer from this, but dependence on p can be improved.

Harvey's improvements (I)

The complexity of Kedlaya's algorithm in p can be improved using Baby Step Giant Step to carry out cohomological reductions more efficiently.

Let ω be an exponent for matrix multiplication.

Theorem (Harvey, 2006)

$Z(X,T)$ can be computed in time

$$O\left(\left(p^{1/2}g^{\omega+5/2}a^{7/2}+\log(p)g^8a^5\right)^{1+\epsilon}\right).$$

This is a lot better in p and still polynomial in g, n. Implemented:

- in Sage by Harvey,
- in Magma by Minzlaff (even for superelliptic curves).

Harvey's improvements (II)

For a hyperelliptic curve over $\mathbb{Z}$, i.e. such that $Q \in \mathbb{Z}[x]$, can exploit overlap between computations for Q mod p for various p (of good reduction), to obtain algorithm with polynomial runtime per prime.

Let $||Q||$ denote bound on absolute value coefficients of Q and $X_p = X \otimes \mathbb{F}_p$ the reduction mod p.

Theorem (Harvey, 2014)

$Z(X_p, T)$ can be determined for all $p \leq N$ in time per prime

$$O\left(g^{8+\epsilon} \log^3(N) \log^{1+\epsilon}(||Q||N)\right).$$

Not implemented. In this form not expected (by Harvey) to be practical. However, Harvey-Sutherland have developed practical version for $g \leq 3$.

General curves

What about more general curves? I have developed a version of Kedlaya's algorithm for general curves.

Suppose that a plane model of a 'good' lift to characteristic zero of the curve $X/\mathbb{F}_q$ with $q = p^a$ is defined by a polynomial $Q \in \mathbb{Z}_q[x, y]$ which is

- irreducible mod p,
- monic in y,
- of degree d_x in x and degree d_y in y.

Theorem (Tuitman, 2014-2016)

$Z(X, T)$ can be computed in time $O((pd_y^6 d_x^4 a^3)^{1+\epsilon})$.

Together with Castryck we have constructed 'good' gonality preserving lifts for all curves of genus $g \leq 5$. Completely implemented in Magma!

Example: generic curve of genus 5

With the latest version of Magma (v2.23):

```
> C:=RandomGenus5CurveNonTrigonal(FiniteField(101));
> C;
Curve over GF(101) defined by
36*$.1^2 + 21*$.1*$.2 + 30*$.2^2 + 7*$.1*$.3 + 28*$.2*$.3 + 26*$.1*$.4 +
    79*$.2*$.4 + 49*$.3*$.4 + 36*$.4^2 + 57*$.1*$.5 + 2*$.2*$.5 + 69*$.3*$.5 +
    66*$.4*$.5 + 35*$.5^2,
97*$.1^2 + 94*$.1*$.2 + 64*$.2^2 + 36*$.1*$.3 + 15*$.2*$.3 + 66*$.3^2 +
    39*$.1*$.4 + 22*$.2*$.4 + 34*$.3*$.4 + 85*$.4^2 + 98*$.1*$.5 + 67*$.2*$.5 +
    49*$.3*$.5 + 33*$.4*$.5 + 25*$.5^2,
57*$.1^2 + 6*$.1*$.2 + 11*$.2^2 + 9*$.1*$.3 + 87*$.2*$.3 + 73*$.3^2 + 64*$.1*$.4
    + 69*$.2*$.4 + 69*$.3*$.4 + 11*$.4^2 + 11*$.1*$.5 + 17*$.2*$.5 + 81*$.3*$.5
    + 79*$.4*$.5 + 50*$.5^2
> time Z:=ZetaFunction(C);
Time: 13.130
> Z;
(10510100501*t^10 - 624362406*t^9 + 51515050*t^8 - 4539445*t^7 + 1279872*t^6 -
    98962*t^5 + 12672*t^4 - 445*t^3 + 50*t^2 - 6*t + 1)/(101*t^2 - 102*t + 1)
```

Hypersurfaces

AKR algorithm

Let $X \subset \mathbb{P}^n_{\mathbb{F}_q}$ be a smooth projective hypersurface of degree d with $q = p^a$.

Abbott, Kedlaya and Roe (2006) proposed a Kedlaya type algorithm to determine $Z(X, T)$. by computing the cohomology $H^n_{rig}(U)$ of the complement $U = \mathbb{P}^n - X$ with its action of F_p.

The running time was not analysed but should be $(pd^n a)^{O(n)}$.

Since the input size is about $\log(p)d^n a$, this is not polynomial time even when p is fixed.

Costa, Kedlaya and Harvey are working on an improvement of this algorithm with better complexity in p. Costa has implemented part of this in C (in SAGE now).

Harvey's arithmetic scheme algorithm

Harvey (2015) proposed algorithms that avoid the use of cohomology and do not need any smoothness assumption on X. These algorithms are quite similar (but superior) to an earlier algorithm of Lauder and Wan (2006).

The complexity in p is very good: $O(p^{1+\epsilon})$, $O(p^{1/2+\epsilon})$ or average polynomial time, depending on the exact version of the algorithm.

However the complexity is again polynomial in d^{n^2} and a^n instead of d^n and a, like for AKR.

The algorithm is not expected to practical (by Harvey). However, a very special case (K3 surfaces which are double covers of $\mathbb{P}^2$) has been partially implemented in Magma by Elsenhans.

Deformation method (I)

Lauder (2004) proposes to put the smooth hypersurface $X \subset \mathbb{P}^n_{\mathbb{F}_q}$ with $q = p^a$ into a (smooth) family X_t/T over some open $T \subset \mathbb{P}^1$ with:

- X_0 a diagonal hypersurface,
- $X_1 = X$.

The relative cohomology $H^n_{rig}(X_t/T)$ is an overconvergent F-isocrystal:

- a p-adic differential equation (the Gauss-Manin connection),
- a Frobenius structure with matrix $\Phi(t)$.

The Frobenius structure is horizontal w.r.t the connection, so $\Phi(t)$ satisfies a p-adic differential equation.

Main idea: compute $\Phi(0)$ (easy, X_0 is diagonal) then solve the differential equation for $\Phi(t)$, finally find $\Phi(1)$ and deduce $Z(X, T)$.

Deformation method (II)

Pancratz and me (2013) improved this in terms of complexity (a bit) and in practice (a lot).

Let ω be an exponent for matrix multiplication and e the basis of the natural logarithm.

Theorem (Pancratz-Tuitman, 2012)

$Z(X,T)$ can be computed in time $O\left((pd^{n(\omega+4)}e^{n(\omega+1)}a^3)^{1+\epsilon}\right)$.

Note that this is polynomial time for fixed p unlike any of the other algorithms for hypersurfaces mentioned so far!

Implemented in C by Pancratz (in SAGE now, Costa and Flori).

My latest work (2017) improves the complexity of the deformation method in p from $O(p^{1+\epsilon})$ to $O(p^{1/2+\epsilon})$, resulting in the algorithm of best known complexity for smooth projective hypersurfaces.

New algorithm

Setup

Let $X_1 \subset \mathbb{P}^n_{\mathbb{F}_q}$ with $q = p^a$ be a smooth projective hypersurface of deg d.

$P_1 \in \mathbb{Z}_q[x_0, \ldots, x_n]$ homogeneous of degree d defining a lift of X_1.
Suppose d is not divisible by p.

$P_0 = a_0 x_0^d + \ldots + a_n x_n^d$ with $a_i \in \mathbb{Z}_q^\times$ defining diagonal hypersurface.

$P = (1-t)P_0 + tP_1 \in \mathbb{Z}_q[t][x_0, \ldots, x_n]$ defining deformation X_t/T.

Let U_t/T be the affine complement of X_t/T.

Fact

$$Z(X_1, T) = \frac{\det(1 - T(p^{-1}F_p)^a | H^n_{rig}(U_1))^{(-1)^n}}{(1-T)(1-qT)\ldots(1-q^{n-1}T)}$$

Relative cohomology

We fix an explicit basis $[\omega_1, \ldots, \omega_b]$ for $H^n_{rig}(U_t)$. $M \in M_{b\times b}(\mathbb{Q}_q(t))$ denotes the matrix of the Gauss–Manin connection ∇ and $\Phi(t)$ the matrix of $p^{-1}\mathrm{F}_p$ on $H^n_{rig}(U_1)$:

$$\nabla(\omega_j) = \sum_{i=1}^{b} M_{ij}\omega_i \otimes dt, \qquad p^{-1}\mathrm{F}_p(\omega_j) = \sum_{i=1}^{b} \Phi_{ij}\omega_i.$$

Let $C \in M_{b\times b}(\mathbb{Q}_q[[t]])$ denote solution to differential equation:

$$\left(\frac{d}{dt} + M\right) C = 0, \qquad C(0) = I.$$

Then it follows from the the theory that

$$\Phi(t) = C\Phi(0)(C^{\sigma}(t^p))^{-1}$$

We can deduce $\Phi(1)$ from $\Phi(0)$ by solving for C.

Bostan-Gaudry-Schost

Let:

- R be a commutative ring,
- $\mathrm{M}(d)$ number of operations for multiplying polynomials of degree at most d over R,
- $\mathrm{MM}(m)$ number of operations for multiplying $m \times m$ matrices over R,
- $A(x)$ be $m \times m$ matrix over $R[x]$ of degree at most 1.

Theorem (Bostan-Gaudry-Schost)

Suppose that some invertibility conditions are satisfied. Then for any positive integer N, the matrix product $A(1)A(2)\cdots A(N)$ can be computed in

$$O\left(MM(m)\sqrt{N} + m^2 M(\sqrt{N})\right)$$

ring operations in R.

Example: computing $k!$

To get some idea, how do we compute $k!$ in $O(k^{1/2+\epsilon})$ operations?

Suppose without loss of generality that $k = c^2$.

Define a polynomial

$$f(x) = (x+1)\dots(x+c).$$

Then

$$k! = f(0)f(c)\dots f((c-1)c).$$

However, a polynomial of degree c can be evaluated in c points in $O(\mathrm{M}(c)\log(c))$ operations by multipoint evaluation.

In this form the idea is due to the Chudnovsky brothers. BGS improve this by getting rid of the factor $\log(c)$ using that $0, c, \dots, (c-1)c$ are in arithmetic progression.

Diagonal fibre

There is an explicit formula involving factorials for $\Phi(0)$ (see paper with Pancratz).

The formula is very complicated, but essentially we need to compute

$$k(k+1)\dots(k+p-1) \bmod p^N$$

for a number of k and a p-adic precision N both independent of p.

Doing this naively takes time at least $O(p^{1+\epsilon})$.

Using BGS, we can get this down to $O(p^{1/2+\epsilon})$!

Differential equation (I)

Recall that we are looking for $C = \sum C_i t^i \in M_{b\times b}(\mathbb{Q}_q[[t]])$ such that

$$\left(\frac{d}{dt} + M\right) C = 0, \qquad C(0) = I.$$

Write $M = G/r$ with $G \in M_{b\times b}(\mathbb{Q}_q[t])$ and $r \in \mathbb{Z}_q[t]$. Moreover, denote

$$G = \sum G_i t^i \qquad r = \sum r_i t^i.$$

Then we get the following recurrence for the matrices C_i:

$$C_0 = I,$$

$$C_{i+1} = \frac{-1}{r_0(i+1)} \left(\sum_{j=i-\deg(G)}^{i} G_{i-j} C_j + \sum_{j=i-\deg(r)+1}^{i} r_{i-j+1}(jC_j) \right).$$

Differential equation (II)

Recall that $\Phi(t) = C(t)\Phi(0)(C^\sigma(t^p))^{-1}$ and we want to compute $\Phi(1)$.

The required t-adic precision is linear in p. So solving the recursion for the C_i naively takes time at least $O(p^{1+\epsilon})$.

We need to evaluate at $t = 1$ along the way, not at the end (too many terms). Solve the recurrence for $D_i = C_0 + \ldots + C_i$ instead. Need only D_i with $i \equiv i_0 \bmod p$ for some i_0.

The recurrence for D_i is not first order, and there is a denominator $i + 1$. Solve larger recurrence for $[D_i, D_{i+1}, \ldots, D_\kappa]$ where κ is the order and multiply by $(i + 1)!$.

Some other minor issues (related to convergence) can also be resolved and once again using BGS, we can compute $\Phi(1)$ from $\Phi(0)$ in time $O(p^{1/2+\epsilon})$.

Complexity

Let X be a generic hypersurface of degree d in projective space $\mathbb{P}^n_{\mathbb{F}_q}$ over a finite field $\mathbb{F}_q$ of characteristic p not dividing d and cardinality $q = p^a$.

Theorem (Tuitman, 2017)

$Z(X, T)$ can be computed in time

$$O\left(\left(p^{1/2}a^3d^{n(2\omega+3)}e^{n(\omega+1)}\right)^{1+\epsilon}\right).$$

So the complexity in p has been lowered from $O(p^{1+\epsilon})$ to $O(p^{1/2+\epsilon})$ at the (small) expense of increasing the complexity in d^n from $O(d^{n(\omega+4)})$ to $O(d^{n(2\omega+3)})$.

This new algorithm has the best known complexity for computing the zeta function of a smooth projective hypersurface.

Loose ends

- What about more general X? Can likely be extended to (some) nondegenerate hypersurfaces in toric varieties. The only problem is finding 'easy' starting fibres X_0 for a given Newton polytope.
- What about average polynomial time? Replacing BGS by accumulating remainder trees (Costa-Gerbicz-Harvey) this should be possible. However, I have not tried this yet.
- What about an implementation? Magma code to compute $\Phi(0)$ in time $O(p^{1/2+\epsilon})$ is already available on my website (using an implementation of BGS by Minzlaff). I have some proof of concept code for the whole algorithm which is available upon request :-) (very messy and not optimised at all).

To be continued....