Computing zeta functions of certain varieties in larger characteristic

David Harvey

New York University

16th March 2010

Effective methods in p-adic cohomology

Mathematical Institute, University of Oxford

Notation

$q = p^a$

$X =$ variety over $\mathbf{F}_q$

$$Z_X(T) = \exp\left(\sum_{r \geq 1} \frac{\#X(\mathbf{F}_{q^r})}{r} T^r\right) \in \mathbf{Q}(T)$$

Question: how to compute $Z_X(T)$ efficiently when p is "large"?

(Time = bit-complexity)

Previous results

Schoof (1985) and descendants: smooth projective curve of degree d in time

$$(a \log p)^{d^{O(1)}}.$$

Lauder (2004): degree d smooth hypersurface $X \subset \mathbf{P}^n$ in time

$$p^{2+\epsilon} \operatorname{poly}(d^n a).$$

(Note: dense input size is $d^n a \log p$ bits)

Previous results

H. (2007): genus g hyperelliptic curve in time

$$p^{0.5+\epsilon} \operatorname{poly}(ga).$$

e.g. $g = 3$, $q \approx 3 \times 10^{16}$ is feasible (30 hours on single CPU)

Minzlaff (2008): superelliptic curve in time

$$p^{0.5+\epsilon} \operatorname{poly}(ga).$$

Main question

Can we obtain $p^{0.5+\epsilon}$ complexity for varieties more general than superelliptic curves?

Main result

Theorem (almost)

Let $X \subset \mathbf{P}^n_{\mathbf{F}_q}$ be a smooth projective hypersurface of degree $d > n$. Then $Z_X(T)$ can be computed in time

$$p^{0.5+\epsilon} \operatorname{poly}((d^n a)^n)$$

(or something roughly of this form).

Still some details to be checked...

Implementation

A toy implementation in Sage has existed for about two weeks.

Ran on a random degree 4 hypersurface in $\mathbf{P}^3$:

$$Z_X(T) = ((1-T)(1-qT)(1-q^2T)P(T))^{-1}$$

where $\deg P = 21$.

Precision parameters selected experimentally to determine $P(T)$ unambiguously (not proved correct).

- $\mathbf{F}_{11}$: 29 hours. Checked $\#X(\mathbf{F}_{11^r})$ for $1 \leq r \leq 4$ against more-or-less naive point count in Magma.
- $\mathbf{F}_{101}$: 80 hours. Checked for $1 \leq r \leq 2$.
- $\mathbf{F}_{1009}$: 239 hours. Checked for $r = 1$.
- $\mathbf{F}_{10007}$: maybe in time for CRM workshop in April...

AKR algorithm

Algorithm based on AKR = Abbott–Kedlaya–Roe ("Bounding Picard numbers of surfaces using p-adic cohomology", 2007).

$X = \{f = 0\} \subset \mathbf{P}^n_{\mathbf{F}_q}$ defined by $f \in \mathbf{F}_q[x_0, \ldots, x_n]$, $\deg f = d$

$U = \mathbf{P}^n \backslash X$

$\sigma_q = q$-th power Frobenius

$P(T) = \det(1 - q^{-1}\sigma_q T | H^n_{\text{rig}}(U))$

Then

$$Z_X(T) = P(T)^{(-1)^n} \prod_{i=0}^{n-1} \frac{1}{1 - q^i T}.$$

So it suffices to compute $P(T) \in \mathbf{Z}[T]$.

AKR algorithm

Plan: compute in $H^n_{\text{rig}}(U)$ using the Monsky–Washnitzer theory.

$\widetilde{f} =$ lift of f to $\mathbf{Z}_q[x_0, \ldots, x_n]$

$\widetilde{U} =$ lift of U defined by $\widetilde{f}$, i.e. with coordinate ring

$$\widetilde{A} = \text{degree 0 piece of } \mathbf{Z}_q[x_0, \ldots, x_n, z]$$

where $z = \widetilde{f}^{-1}$, $\deg z = -d$.

Then

$$H^n_{\text{rig}}(U) \cong H^n_{\text{dR}}(\widetilde{U}/\mathbf{Q}_q).$$

AKR algorithm

Explicit description of $H^n_{\mathrm{dR}}(\widetilde{U}/\mathbf{Q}_q)$ (Griffiths): let

$$\Omega = \sum_{i=0}^{n} (-1)^i x_i \, dx_0 \wedge \cdots (\text{omit } dx_i) \cdots \wedge dx_n.$$

Then $H^n_{\mathrm{dR}}(\widetilde{U}/\mathbf{Q}_q)$ is the quotient of

$$\langle z^m G \Omega : m \geq 1, \deg G = md - n - 1 \rangle$$

by

$$\left\langle (z^m \partial_i G - m z^{m+1} G \partial_i \widetilde{f}) \Omega : 0 \leq i \leq n, \, m \geq 1, \deg G = md - n \right\rangle.$$

(Here $\partial_i = \partial/\partial x_i$.)

AKR algorithm

A theorem of Macaulay implies that (since X smooth!)

$$(\partial_0 \widetilde{f}, \ldots, \partial_n \widetilde{f}) \supset (x_0, \ldots, x_n)^\alpha,$$

where $\alpha = (n+1)(d-2)+1$.

This yields a *reduction algorithm* for computing in H^n_{dR}: if $\deg F \geq \alpha$ then $F = \sum_i G_i(\partial_i \widetilde{f})$ for some G_i, and then

$$z^{m+1} F \Omega \sim \frac{1}{m} z^m \sum_i (\partial_i G_i) \Omega.$$

Some subset of the monomials for $\deg F < \alpha$ forms a basis for H^n_{dR}; can be computed explicitly by linear algebra.

AKR algorithm

What about Frobenius?

Let $A^\dagger$ = weak completion of $\widetilde{A}$: elements are power series $\sum_{j \geq 0} G_j z^j$, where $G_j \in \mathbf{Q}_q[x_0, \ldots, x_n]$, $\deg G_j = jd$, satisfying overconvergence condition $\liminf_{j \to \infty} v_p(G_j)/j > 0$.

Lift (absolute) Frobenius to $A^\dagger$ via $x_i^\sigma = x_i^p$ $(0 \leq i \leq n)$ and

$$\begin{aligned} z^\sigma &= \widetilde{f}^{-\sigma} \\ &= (\widetilde{f}^p - (\widetilde{f}^p - \widetilde{f}^\sigma))^{-1} \\ &= z^p (1 - z^p (\widetilde{f}^p - \widetilde{f}^\sigma))^{-1} \\ &= z^p \sum_{j \geq 0} z^{pj} (\widetilde{f}^p - \widetilde{f}^\sigma)^j. \end{aligned}$$

(Converges in $A^\dagger$ since $p \mid \widetilde{f}^p - \widetilde{f}^\sigma$.)

AKR algorithm

Furthermore

$$(z^t)^\sigma = z^{pt} \sum_{j \geq 0} \binom{t+j-1}{t-1} z^{pj} (\widetilde{f}^p - \widetilde{f}^\sigma)^j$$

and

$$\Omega^\sigma = p^n (x_0 \cdots x_n)^{p-1} \Omega.$$

Therefore we obtain series expansion for σ applied to a cohomology basis element:

$$(z^t x_0^{k_0} \cdots x_n^{k_n} \Omega)^\sigma.$$

AKR algorithm

Summary of AKR algorithm:

1. Compute a basis for H^n_{dR}
2. Compute series approximations for $(z^t x_0^{k_0} \cdots x_n^{k_n} \Omega)^\sigma$ for each basis element, to some p-adic and z-adic precision
3. Apply reduction algorithm to reduce each series back to basis elements
4. This yields matrix of absolute Frobenius; take product of conjugates to obtain matrix of q-th Frobenius
5. Characteristic polynomial is $P(T)$ to some p-adic precision

AKR algorithm

AKR did not analyse complexity.

Running time behaves at least like p^n, because algorithm works with dense polynomials like

$$\widetilde{f}^p - \widetilde{f}^\sigma$$

in n variables.

New algorithm

First modification: use a "sparse" series expansion.

$$
\begin{aligned}
(z^t)^\sigma &= z^{pt} \sum_{j \geq 0} \binom{t+j-1}{t-1} z^{pj} (\widetilde{f}^p - \widetilde{f}^\sigma)^j \\
&\equiv z^{pt} \sum_{j=0}^{N-1} \binom{t+j-1}{t-1} z^{pj} (\widetilde{f}^p - \widetilde{f}^\sigma)^j \qquad (\mathrm{mod}\ p^N) \\
&= z^{pt} \sum_{j=0}^{N-1} \binom{t+j-1}{t-1} (1 - z^p \widetilde{f}^\sigma)^j \\
&= \sum_{k=0}^{N-1} C_{t,k} z^{p(k+t)} \widetilde{f}^{k\sigma} \qquad \left[C_{t,k} = (-1)^k \textstyle\sum_{j=k}^{N-1} \binom{t+j-1}{t-1} \binom{j}{k} \right]
\end{aligned}
$$

Number of terms does not depend on p!

New algorithm

Thus $(z^t x_0^{k_0} \cdots x_n^{k_n} \Omega)^\sigma$ is approximated by a sum of terms of the form

$$z^{pm} x_0^{pr_0 - 1} \cdots x_n^{pr_n - 1} \Omega,$$

i.e. on a lattice with distance p between terms.

Number of terms is about $O((Nd)^n)$.

To recover zeta function, need $N = O(d^n a)$ (?).

New algorithm

Second modification: use "controlled reduction".

Consider a differential $z^{m+1}x^u F\Omega$, where $\deg F = \alpha$.

(Multi-index notation: $x^u = x_0^{u_0} \cdots x_n^{u_n}$.)

Write $F = \sum G_i(\partial_i \widetilde{f})$ for polynomials G_i, then

$$z^{m+1}x^u F\Omega \sim \frac{z^m}{m} \sum_i \partial_i(x^u G_i)\,\Omega = \frac{z^m}{m} \frac{x^u}{x_0 \cdots x_n} H\Omega,$$

where $\deg H = \alpha - d + n + 1$.

New algorithm

Provided that $d > n$, we can rewrite this as (for example)

$$\frac{z^m}{m} \frac{x^u}{x_0^{d-n} x_1 \cdots x_n} (x_0^{d-n-1} H)\Omega = \frac{z^m}{m} x^{u'} F' \Omega,$$

where $\deg F' = \alpha$.

(Result is slightly different if some $u_i \approx 0$, need to reduce in a different "direction", but basically same idea.)

Now iterate this process with F' and u'; number of terms remains bounded as m decreases.

Already this technique reduces complexity from p^n to p^1!

If $d \leq n$, unclear how to control growth of number of terms.

New algorithm

Third modification: use "accelerated reduction".

Consider the reduction described above, i.e.

$$z^{m+1} x^u F\Omega \Longrightarrow \frac{z^m}{m} x^{u'} F'\Omega.$$

Let $\overline{F}$ and $\overline{F}'$ denote the vectors of coefficients. Then

$$\overline{F}' = S(m)\overline{F}$$

where $S(m)$ is a matrix of *linear polynomials* in m over $\mathbf{Z}_q$.

To iterate the reduction, want to compute the matrix product

$$S(m-p+1)\cdots S(m-1)S(m).$$

New algorithm

Fortunately there is a fast algorithm for computing such products!

Let $S(m)$ be an $r \times r$ matrix of linear polynomials over a ring R.

Computing $S(k) \cdots S(1)S(0)$ naively requires

$$O(kr^\omega)$$

ring operations ($\omega \leq 3$ is exponent of matrix multiplication).

Better algorithm of Chudnovsky–Chudnovsky (1988), using fast polynomial arithmetic, obtains

$$O(k^{0.5+\epsilon} r^\omega)$$

ring operations.

(See also improvements by Bostan–Gaudry–Schost (2004).)

New algorithm

Summary of new algorithm:

1. Compute a basis for H^n_{dR}
2. Compute *sparse* series approximations for $(z^t x_0^{k_0} \cdots x_n^{k_n} \Omega)^\sigma$ for each basis element
3. Compute reduction matrices moving between adjacent lattice points
4. Apply reduction matrices to terms from step 2 to compute absolute Frobenius matrix; get $P(T)$ as before.

For large p, all the work is in step 3.

Open questions

1. Can the algorithm be made more practical?
   - countless opportunities for optimisation
   - implementation grunt work
2. Can $(d^n a)^n$ be reduced to $d^n a$?
   - maybe combine with deformation techniques? (can we deform in $p^{0.5}$ time?)
3. Can we drop the condition $d > n$?
   - would have expected $d \leq n$ to be *easier!*
4. What about smooth affine varieties?
5. Can we drop smoothness condition?
6. Can we do better than $p^{0.5}$?
   - does anyone know how to compute $(p-1)! \bmod p^2$ faster than $O(p^{0.5+\epsilon})$?

Thank you!