Recent progress on point-counting algorithms

David Harvey

University of New South Wales

1st October 2013, University of Sydney

Contents

- The point-counting problem
- "Counting points on hyperelliptic curves in average polynomial time" (H., 2012 preprint, to appear in Annals of Mathematics)
- "Zeta functions of nondegenerate toric hypersurfaces via controlled reduction in p-adic cohomology" (Costa–H.–Kedlaya, in preparation)

No algorithm details today... just complexity bounds and example computations.

The point-counting problem

Notation:

$p =$ a prime

$q = p^a$, $a \geq 1$, a prime power

$\mathbf{F}_q =$ finite field with q elements

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

$n =$ dimension of X

For $r \geq 1$, define

$$N_r(X) = \#X(\mathbf{F}_{q^r}).$$

For example, if X is the affine variety defined as the simultaneous zero set of a collection of polynomials $F_1, \ldots, F_t \in \mathbf{F}_q[x_1, \ldots, x_m]$, then $N_r(X)$ is the number of solutions $(x_1, \ldots, x_m) \in (\mathbf{F}_{q^r})^m$ to $F_1 = \cdots = F_t = 0$.

Problem 1

Given X and r, compute $N_r(X)$.

Trivial solution: test every $(x_1, \ldots, x_m)$!

Revised Problem 1

Given X and r, compute $N_r(X)$ *efficiently*.

"Efficiently" = favourable complexity (time and/or space) as function of input parameters.

Problem 2

Given X, compute $N_r(X)$ for *all* $r \geq 1$.

What does it mean to compute an infinite sequence of integers?

The *zeta function* of X is the generating function

$$Z_X(T) = \exp \sum_{r \geq 1} \frac{N_r(X)}{r} T^r \in \mathbf{Z}[[T]].$$

Dwork's theorem: actually $Z_X(T) \in \mathbf{Q}[T]$.

Revised Problem 2

Given X, compute numerator and denominator of $Z_X(T)$. (Efficiently!)

Bombieri gave explicit bounds for the degrees of the numerator and denominator.

So Problem 2 admits an effective algorithmic solution.

Counting points on hyperelliptic curves in average polynomial time

Now suppose X is a hyperelliptic curve over $\mathbf{F}_p$ of genus g.

Explicitly:

$$y^2 = f(x)$$

where $f \in \mathbf{F}_p[x]$ has degree $2g+1$ or $2g+2$.

(We are assuming that $p \neq 2$.)

Zeta function looks like:

$$Z_X(T) = \frac{P(T)}{(1-T)(1-pT)},$$

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

Algorithms for computing $Z_X(T)$ in this case include:

- Enumerate points over $\mathbf{F}_p, \mathbf{F}_{p^2}, \ldots, \mathbf{F}_{p^g}$: complexity $p^{g+o(1)}$.
- Use group structure: saves factor of $p^{O(1)}$.
- Compute Frobenius action on ℓ-torsion points of Jacobian: complexity $(\log p)^{g^{2+o(1)}}$ (Schoof, Pila, Adleman–Huang).
- Monsky–Washnitzer cohomology: complexity $p^{1+o(1)}g^{4+o(1)}$ (Kedlaya, 2001) or $p^{1/2+o(1)}g^{11/2+o(1)}$ (H., 2007).
- many others...

Unfortunate fact

There is no known algorithm that computes $Z_X(T)$ in time polynomial in both $\log p$ and g, i.e. polynomial in the input size.

Aside: some record computations for 'random' input:

- Genus 1 (Sutherland, 2010): $p \sim 10^{5000}$.
- Genus 2 (Gaudry–Schost, 2012): $p \sim 10^{38}$.
- Genus 3 (H., 2008): $p \sim 10^{16}$.
- Genus 4 (H., 2008): $p \sim 10^{13}$.

The first two use mainly ℓ-adic algorithms; the latter two mainly p-adic.

The closest we have to a polynomial time algorithm is:

Theorem (H. 2012)

Let X be a hyperelliptic curve of genus g over $\mathbf{Q}$.

For a prime p, let X_p be its reduction modulo p.

There is an explicit deterministic algorithm that computes $Z_{X_p}(T)$ for all (good reduction) primes $p < N$ in time

$$g^{8+o(1)} N \log^{3+o(1)} N.$$

In particular, the average time per prime is $g^{8+o(1)} \log^{4+o(1)} N$.

This is especially useful if you want to compute the L-series of the curve.

Implementation for genus 2 and 3 in progress with Andrew Sutherland (MIT).

I will give some examples comparing new code with:

- Sutherland's `smalljac` library — computes zeta functions of hyperelliptic curves of genus up to 3 over prime fields, for 'small' p, using a combination of naive point counting and group computations. Here 'small' means that Schoof-type algorithms are not yet competitive. To my knowledge, `smalljac` is the world champion for this regime, except as noted in the next bullet point.
- My `hypellfrob` library — solves same problem using p-adic algorithm with complexity $p^{1/2+o(1)}$. Faster than smalljac in genus 3 for $p \geq 2^{17}$.

Genus 2 example: take

$$y^2 = 164493x^6 + 271828x^4 + 314159x^3 + 141421x^2 + 915065x + 651482.$$

We compute $Z_{X_p}(T)$ for all $p < N$.

The new code breaks even with `smalljac` at $N \sim 2^{20}$.

At $N = 2^{20}$, new code takes 237s, `smalljac` takes 256s.

At $N = 2^{27}$, new code takes 10.3h, `smalljac` would take an estimated 100h.

For genus 3, take

$$y^2=120205x^8+577215x^6+164493x^5+271828x^4+314159x^3+141421x^2+915065x+283048.$$

New code breaks even with `smalljac` at only $N \sim 2^{12}$!

At $N = 2^{12}$, new code takes 0.5s, `smalljac` takes 0.6s.

At $N = 2^{17}$, new code takes 72s, `hypellfrob` is closest competitor at 215s.

At $N = 2^{24}$, new code takes 8h, `hypellfrob` would take an estimated 100h.

We are planning to go out to $N \sim 2^{30}$, where it becomes roughly one month vs 6 years.

Zeta functions of nondegenerate toric hypersurfaces via controlled reduction in p-adic cohomology

Let $\mathbf{F}_q[\mathbf{Z}^n] = \mathbf{F}_q[x_1, x_1^{-1}, \ldots, x_n, x_n^{-1}]$ be the ring of Laurent polynomials in n variables.

Multi-index notation: for $u = (u_1, \ldots, u_n) \in \mathbf{Z}^n$, we write f_u for the coefficient of $x^u = x_1^{u_1} \cdots x_n^{u_n}$ in f.

Let $f \in \mathbf{F}_q[\mathbf{Z}^n]$, $f \neq 0$. This defines a *toric hypersurface*, i.e. a hypersurface X in the torus $(G_m)^n = \operatorname{Spec} \mathbf{F}_q[\mathbf{Z}^n]$.

The number of points is simply

$$N_r(X) = \#\{(x_1, \ldots, x_n) \in (\mathbf{F}_{q^r}^*)^n : f(x_1, \ldots, x_n) = 0\}.$$

Recall that X is smooth if the system of equations

$$f = \frac{\partial f}{\partial x_1} = \cdots = \frac{\partial f}{\partial x_n} = 0$$

has no solution in the torus.

We need a slightly stronger condition, *nondegeneracy*, defined as follows.

Let $\operatorname{supp} f$ be the set of lattice points $u \in \mathbf{Z}^n$ such that $f_u \neq 0$.

Let Γ be a lattice polytope in $\mathbf{R}^n$ (the convex hull of a finite subset of $\mathbf{Z}^n$) containing $\operatorname{supp} f$; for example Γ could be the convex hull of $\operatorname{supp} f$ itself.

If γ is a face of Γ, let

$$f|_\gamma = \sum_{u \in \gamma \cap \mathbf{Z}^n} f_u x^u.$$

Then f is nondegenerate with respect to Γ if for every face γ, the system

$$f|_\gamma = \frac{\partial f|_\gamma}{\partial x_1} = \cdots = \frac{\partial f|_\gamma}{\partial x_n} = 0$$

has no solution in the torus.

Example: consider a curve C of degree d in $\mathbf{P}^2$, defined by a homogeneous polynomial $F(X, Y, Z)$ of degree d.

Put $f(x, y) = F(x, y, 1) \in \mathbf{F}_q[\mathbf{Z}^2]$.

Let Γ be the triangle with vertices $(0, 0)$, $(0, d)$, $(d, 0)$, so that $\operatorname{supp} f \subseteq \Gamma$.

Then f is nondegenerate with respect to Γ if and only if:

- C does not pass through the points $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$;
- C intersects the coordinate axes $X = 0$, $Y = 0$, $Z = 0$ transversally;
- C is smooth on the complement of the coordinate axes.

So nondegenerate loosely means: smooth plus 'smooth at infinity in every direction'.

Existing algorithms for computing $Z_X(T)$:

- Lauder–Wan: complexity $(pa \operatorname{vol} \Gamma)^{O(n)}$.
  The p^n is the real killer here.
  No smoothness hypotheses!
- Abbott–Kedlaya–Roe: probably same complexity.
  Requires X smooth.
- Sperber–Voight: similar complexity, but exponent of p depends on number of monomials. In worst case still p^n.
  Requires X nondegenerate.
- Lauder's deformation method: complexity $(pa \operatorname{vol} \Gamma)^{O(1)}$.
  Exponent of p can probably be improved from $O(1)$ to $1 + o(1)$.
  Requires X smooth.

Note: all these algorithms (except Sperber–Voight) are only stated for $\Gamma =$ simplex, but they probably can be generalised to arbitrary Γ.

The first version of our new algorithm has complexity roughly

$$p^{1+o(1)}(a\,\mathrm{vol}(\Gamma))^{O(n)}.$$

This is inferior to Lauder's algorithm for large n.

But our main advantage is that the space complexity is only

$$(\log p)(a\,\mathrm{vol}(\Gamma))^{O(n)}.$$

This allows us to handle examples with much larger p than any found in the literature.

Numerical example: a nondegenerate quartic in $\mathbf{P}^3$ (a K3 surface) over $\mathbf{F}_p$ with $p = 49999$.

C++/NTL code written by Edgar Costa for the $\Gamma =$ simplex case.

Surface is given by the randomly chosen polynomial

$$f = 25163x_0^4 + 9405x_0^3x_1 + 85x_0^2x_1^2 + 30034x_0x_1^3 + 21740x_1^4 + 14747x_0^3x_2 + 35394x_0^2x_1x_2 + 13683x_0x_1^2x_2 + 12720x_1^3x_2 + 36331x_0^2x_2^2 + 23023x_0x_1x_2^2 + 25667x_1^2x_2^2 + 7066x_0x_2^3 + 6479x_1x_2^3 + 8778x_2^4 + 40922x_0^3x_3 + 38119x_0^2x_1x_3 + 48775x_0x_1^2x_3 + 9720x_1^3x_3 + 20633x_0^2x_2x_3 + 41354x_0x_1x_2x_3 + 31769x_1^2x_2x_3 + 32904x_0x_2^2x_3 + 49443x_1x_2^2x_3 + 24957x_2^3x_3 + 37766x_0^2x_3^2 + 8622x_0x_1x_3^2 + 3377x_1^2x_3^2 + 15688x_0x_2x_3^2 + 10170x_1x_2x_3^2 + 19668x_2^2x_3^2 + 2486x_0x_3^3 + 13807x_1x_3^3 + 15264x_2x_3^3 + 27566x_3^4.$$

Its zeta function is

$$Z_X(T) = \frac{1}{(1-T)(1-pT)(1-p^2T)P(T)}$$

where

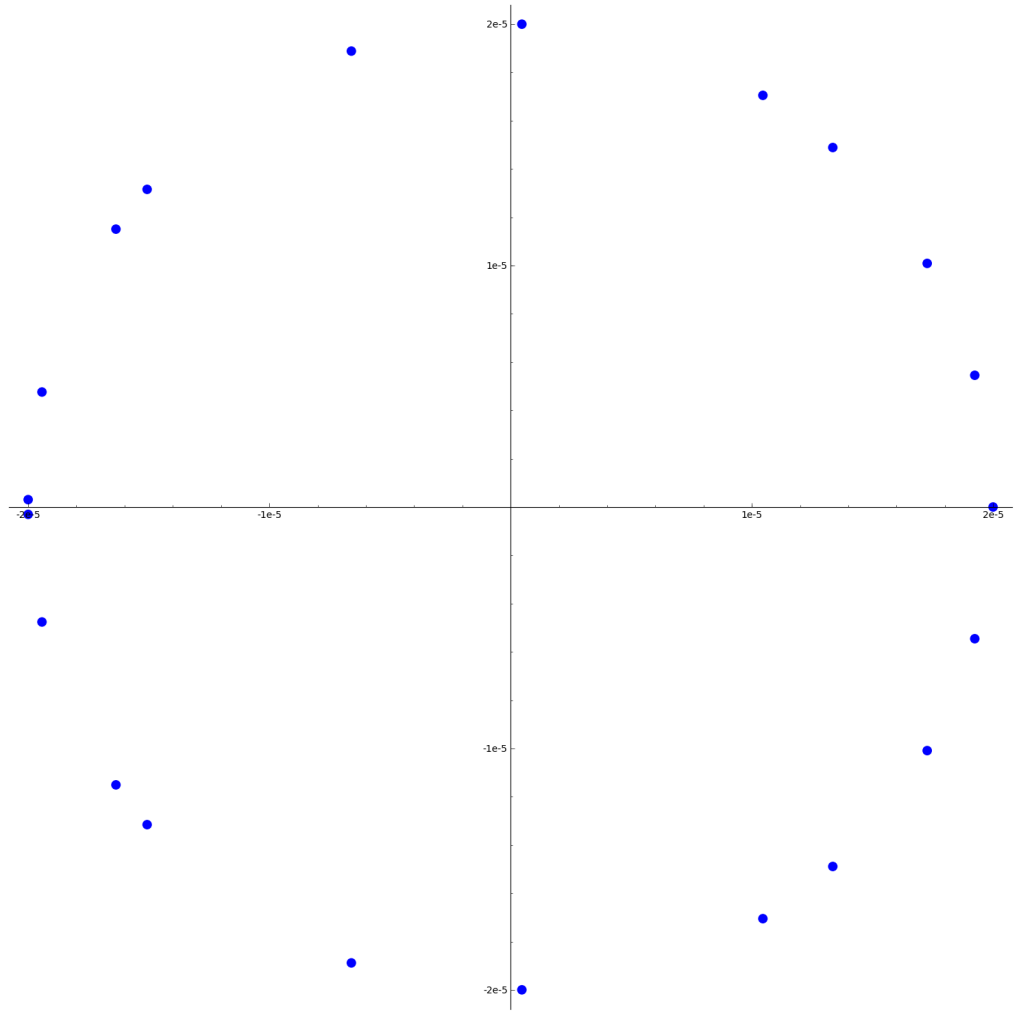
$$P(T) = 1 + a_1 T + a_2 p T^2 + \cdots + a_{10} p^9 T^{10} \\ - a_{10} p^{10} T^{11} - \cdots - a_2 p^{18} T^{19} - a_1 p^{19} T^{20} - p^{21}$$

and

$$(a_1, \ldots, a_{10}) = (33264, -81893, -32490, 86146, 23017, \\ -55214, -22632, -2392, 43164, 47726)$$

Computed in about 5.75 hours, using about 1.5 GB RAM.

Here are the complex roots, illustrating the Riemann hypothesis component of the Weil conjectures (all roots have absolute value $1/p$):



A second version of our algorithm has complexity

$$p^{1/2+o(1)}(a\,\mathrm{vol}(\Gamma))^{O(n)}.$$

For large p this is the fastest known algorithm for computing zeta functions of such varieties.

Finally, there is also an 'average polynomial time' version, i.e. the input is a hypersurface defined over $\mathbf{Q}$, and the average complexity for each prime $p < N$ is

$$(\log N)^{4+o(1)}(a\,\mathrm{vol}(\Gamma))^{O(n)}.$$

These have not been implemented yet.

Thank you!