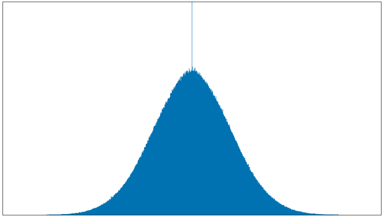
Counting points on superelliptic curves in average polynomial time

Andrew V. Sutherland

Massachusetts Institute of Technology

ANTS XIV



Why count points?

Let $X/\mathbb{Q}$ be a nice (smooth, projective, geometrically integral) curve of genus g.
For each prime p of good reduction for X (the good primes) we have

$$\#X_p(\mathbb{F}_p) = p + 1 - a_p,$$

where the trace of Frobenius a_p satisfies $|a_p| \leq 2g\sqrt{p}$.

Some open questions about the distribution of a_p as p varies:

- For a fixed integer t, how often is $a_p = t$? [Lang–Trotter]
- What is the distribution of $x_p := a_p/\sqrt{p}$? [Sato-Tate]
- Is the sign of a_p subject to a Chebyshev bias? [Mazur–Sarnak]
- Is the order of vanishing of $L(X, s)$ at $s = 1$ equal to the rank of $\mathrm{Jac}(X)$? [BSD]

Some partial answers are known for $g = 1$, but for $g > 1$ very little is known.

Why average polynomial time?

To "compute" $L(s)$ it is enough to know a_n for $n \le N$ with $N \propto \sqrt{\operatorname{cond}\operatorname{Jac}(X)}$.
We would like to be able to do this in quasi-linear time (as a function of N).

The a_n are determined by the a_{p^r} with $p^r \le N$, almost all of which are a_p's.
We can compute the a_{p^r} for $r > 1$ in $O(p)$ time using p-adic methods.
But we need to be able to compute the a_p in time polynomial in $\log p$ (on average).

What about bad primes?

- For distributional questions we can ignore the finitely many bad primes.
- For computing $L(X, s)$ we can deduce the a_{p^r} for bad p using the functional equation if we are willing to assume the Hasse–Weil conjecture holds for X (and prepared to compute lots of a_p, more than we might otherwise need).

Note: For $p \ge 16g^2$ it is enough to know $a_p \bmod p$, since $|a_p| \le 2g\sqrt{p}$.

Algorithms for hyperelliptic curves

Let $X/\mathbb{Q} : y^2 = f(x)$ with $d = \deg(f)$, then $g = (d - \gcd(d, 2))/2$.
We wish to compute a_p for good $p \leq N$ for some bound N. Three approaches:

1. Use Harvey's optimization [Har07] of Kedlaya's p-adic algorithm for each p.
   This costs $O(p^{1/2}(\log p)^2 g^\omega)$ per $p \leq N$, yielding $O(N^{3/2}(\log N)g^3)$.
2. Apply Abelard's optimization [Abe18] of Pila's ℓ-adic algorithm for each p.
   This costs $O((\log p)^{O(g)})$ per $p \leq N$, yielding $O(N(\log N)^{O(g)})$.
3. Apply the optimization [HS16] of Harvey's average polynomial-time algorithm.
   This costs $O((\log p)^4 g^3)$ per $p \leq N$ on average, or $O(N(\log N)^3 g^3)$.

Only the third has a running time that is quasi-linear in N and polynomial in g.

In practice it is much faster than the p-adic or ℓ-adic approaches for all values of g.
The best known value for the $O(g)$ ℓ-adic exponent is $5, 8, 14$ for $g = 1, 2, 3$.

Algorithms for superelliptic curves

Let $X/\mathbb{Q} : y^m = f(x)$ with $d = \deg(f)$, then $g = ((d-2)(m-1) + m - \gcd(m,d))/2$.
We wish to compute a_p for good $p \le N$ for some bound N. Three approaches:

1. Use the ANTS XIII [ABCMT19] generalization of Harvey's optimization of Kedlaya's p-adic for hyperelliptic curves.
   This costs $O(p^{1/2}(\log p)^2 m d^\omega)$ per $p \le N$, yielding $O(N^{3/2}(\log N) m d^\omega)$.
2. Use an optimization [AH01] of Pila's generalization of Schoof's algorithm.
   This costs $O((\log p)^{g^{O(1)}})$ per $p \le N$, yielding $O(N(\log N)^{g^{O(1)}})$.
3. Use the algorithm presented in this talk.
   This costs $O((\log p)^4 m d^3)$ per $p \le N$ on average, or $O(N(\log N)^3 m d^3)$.

As in the hyperelliptic case, not only is the average polynomial-time approach asymptotically faster, it is faster in practice for essentially all values of d, m and N.

Note: Our definition of superelliptic curves coincides with the cyclic covers of $\mathbb{P}^1$ considered in [ABCMT19], which also requires f to be separable

The Cartier–Manin matrix

Let K be the function field of curve $X/\mathbb{F}_p$, and let Ω_K be its module of differentials. If we fix $x \in K$ so that $K/\mathbb{F}_p(x)$ is separable, every $z \in K$ can be written uniquely as

$$z = z_0^p + z_1^p x + \cdots z_{p-1}^p x^{p-1},$$

with $z_1 \in K^p$. The Cartier operator $\mathcal{C}\colon \Omega_K \to \Omega_K$ defined by $z dx \mapsto z_{p-1} dx$ satisfies

1. $\mathcal{C}(\omega_1 + \omega_2) = \mathcal{C}(\omega_1) + \mathcal{C}(\omega_2)$ for all $\omega_1, \omega_2 \in \Omega_K$;
2. $\mathcal{C}(z^p \omega) = z\,\mathcal{C}(\omega)$ for all $z \in K$ and $\omega \in \Omega_K$;
3. $\mathcal{C}(dz) = 0$ for all $z \in K$;
4. $\mathcal{C}(dz/z) = dz/z$ for all $z \in K^\times$.

and restricts to a semilinear operator on the space $\Omega_K(0)$ of regular differentials. The Cartier–Manin matrix $A_p \in \mathbb{F}_p^{g\times g}$ of X gives the action of $\mathcal{C}$ on a basis for $\Omega_K(0)$.

For a superelliptic curve $X\colon y^m = f(x)$ we use the basis $\omega := \{\omega_{ij} : mi + dj < md\}$, where $\omega_{ij} := \frac{1}{m} x^{i-1} y^{j-m} dx$ for $i, j \geq 1$. **Key fact**: $\operatorname{tr}(A_p) \equiv a_p \bmod p$.

Stöhr-Voloch

Write $k(X) = k(x)[y]/(F)$ with $F \in k[x][y]$. Then

$$\mathcal{C}\left(h\frac{dx}{F_y}\right) = \left(\nabla(F^{p-1}h)\right)^{1/p}\frac{dx}{F_y}, \qquad \text{where } \nabla := \frac{\partial^{2p-2}}{\partial x^{p-1}\partial y^{p-1}}.$$

If we now define $\omega_{k\ell} := x^{k-1}y^{\ell-1}\frac{dx}{F_y}$, with $k,\ell \geq 1$ and $k+\ell \leq \deg(F)-1)$, then

$$\mathcal{C}(\omega_{k\ell}) = \sum \left(F^{p-1}_{ip-k,\, jp-\ell}\right)^{1/p}\omega_{ij}.$$

Not all $\omega_{k\ell}$ are regular, $F(x,y) = y^m - f(x)$ requires $mk + d\ell < md$. The matrix of $\mathcal{C}$ is

$$A_p := [B^{j\ell}]_{j\ell}, \qquad B^{j\ell} := [(b^{j\ell}_{ik})^{1/p}]_{ik}, \qquad b^{j\ell}_{ik} := \begin{cases} f^{n_j}_{ip-k} & \text{for } (jp-\ell)m \in \mathbb{Z}_{\geq 0} \\ 0 & \text{otherwise} \end{cases}$$

where $j,\ell \leq m - \lfloor \frac{m}{d} \rfloor - 1$, $i \leq d - \lfloor \frac{dj}{m} \rfloor - 1$, $k \leq d - \lfloor \frac{d\ell}{m} \rfloor - 1$, and $n_j := p - 1 - \lfloor \frac{jp}{m} \rfloor$.

A genus 4 example

For $X\colon y^5 = f(x)$ with $\deg(f) = 3$ the Cartier–Manin matrix has the form

$$\begin{pmatrix} f_{p-1}^{(4p-4)/5} & f_{p-2}^{(4p-4)/5} & 0 & 0 \\ f_{2p-1}^{(4p-4)/5} & f_{2p-2}^{(4p-4)/5} & 0 & 0 \\ 0 & 0 & f_{p-1}^{(3p-3)/5} & 0 \\ 0 & 0 & 0 & f_{p-1}^{(2p-2)/5} \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & f_{p-1}^{(4p-3)/5} & 0 \\ 0 & 0 & f_{2p-1}^{(4p-3)/5} & 0 \\ 0 & 0 & 0 & 0 \\ f_{p-1}^{(2p-4)/5} & b_{p-2}^{(2p-4)/5} & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 0 & f_{p-1}^{(4p-2)/5} \\ 0 & 0 & 0 & f_{2p-1}^{(4p-2)/5} \\ f_{p-1}^{(3p-4)/5} & f_{p-2}^{(3p-4)/5} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f_{p-1}^{(3p-2)/5} \\ 0 & 0 & f_{p-1}^{(2p-3)/5} & 0 \end{pmatrix}.$$

for $p \equiv 1, 2, 3, 4 \bmod 5$. Here f_k^n denotes the coefficient of x^k in f^n.

Linear recurrences

Let $f = \sum_i f_i x^i$ with $f_0 \neq 0$. The identities $f^{n+1} = f \cdot f^n$ and $(f^{n+1})' = (n+1)f^n$ imply

$$\sum_i ((n+1)i - k) f_i f_{k-i}^n = 0,$$

For the exponents $n = ((m-j)p - (m-\ell))/m$ of interest to us (with $1 \le j, \ell < m$)

$$\sum_i (\ell i - mk) f_i f_{k-i}^n \equiv 0 \bmod p,$$

for all $n \in \mathbb{Z}_{\ge 0}$, $k \in \mathbb{Z}$. If we know $f_{k-1-d}^n, \ldots f_{k-1}^n$ we can compute f_k^n using

$$M_{k-1} := \begin{bmatrix} 0 & \cdots & 0 & (\ell r - mk) f_d \\ mkf_0 & \cdots & 0 & (\ell(d-1) - mk) f_{d-1} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & mkf_0 & (\ell - mk) f_1 \end{bmatrix}.$$

We want to compute $v_0 M_0 \cdots M_{k-1} \bmod m_k$ for $1 \le k \le N$, where $v_0 = [0, \ldots, 0, f_0^n]$.

Accumulating remainder forest

We can save both time and space by using a remainder forest rather than a tree. Given an initial vector (or matrix) v_0, matrices M_i, and moduli m_i, we compute

$$v_k := v_0 M_0 \cdots M_{k-1} \bmod m_k$$

for $1 \leq k \leq N$. by partitioning the N matrices and moduli into 2^κ blocks of size $n := N/2^\kappa$ and applying the remainder tree algorithm to each block.

Let $V := v_0$ and $m := m_1 \cdots m_N$, and for j from 1 to 2^κ proceed as follows:

1. Compute product trees of $M_{(j-1)n} \cdots M_{jn-1}$ and $m_{(j-1)n+1} \cdots m_{jn}$.
2. Starting with $V_{\text{parent}} := V$, work down the trees computing $V_{\text{left}} := V_{\text{parent}} \bmod m_{\text{left}}$ and $V_{\text{right}} := V_{\text{parent}} M_{\text{left}} \bmod m_{\text{right}}$ at each node.
3. Output $v_{(j-1)n}, \cdots, v_{jn-1}$ at the leaves.
4. Set $m \leftarrow m/(m_{(j-1)n+1} \cdots m_{jn})$ and $V \leftarrow V M_{(j-1)n} \cdots M_{jn-1} \bmod m$.

Average time per $p \leq N$ in milliseconds (2.8GHz CPU)

			$N = 2^{16}$		$N = 2^{20}$		$N = 2^{24}$		$N = 2^{28}$	
g	m	d	sage	new	sage	new	sage	new	sage	new
1	2	3	21	0.01	27	0.05	67	0.13	230	0.30
2	2	5	30	0.08	55	0.38	163	0.92	580	2.01
2	2	6	42	0.16	83	0.74	280	1.77	1070	3.92
3	2	7	53	0.24	112	1.29	307	3.12	1217	6.71
3	2	8	74	0.34	169	2.07	528	4.94	2106	10.57
3	3	4	34	0.05	61	0.26	178	0.70	702	1.63
3	4	3	32	0.03	58	0.15	165	0.37	601	0.89
3	4	4	49	0.09	101	0.44	343	1.14	1283	2.63
4	2	9	96	0.44	194	3.24	576	7.70	2214	15.90
4	2	10	138	0.55	319	4.65	974	10.98	3693	22.79
4	3	5	47	0.11	93	0.65	287	1.67	1105	3.68
4	3	6	71	0.18	152	1.28	535	3.20	2121	7.07
4	5	3	37	0.03	68	0.13	200	0.40	778	0.99
4	6	3	49	0.06	112	0.24	313	0.64	1184	1.53

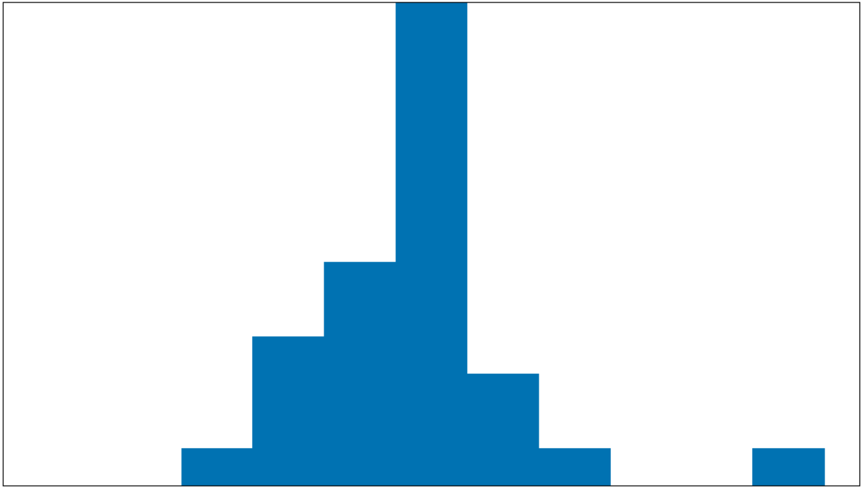
Choose your own adventure!

Questions you could now ask:

- Your cover slide seemd to promise a Sato-Tate histogram, where is it?!
- Remainder forests use a time/space trace-off, so they must be slower, right?
- What about arbitrary cyclic covers of $\mathbb{P}^1$? $C_{a,b}$ curves? Smooth plane curves?
- Traces aren't enough, I want the full zeta function!
  How do I compute that in average polynomial time?
- Something else entirely...

a1 histogram of $y^7 = x^3 + 4x^2 + 3x - 1$ for $p <= 2^{10}$

170 data points in 13 buckets, z1 = 0.829, out of range data has area 0.829



Moments: 1 -0.102 1.247 1.339 47.773 230.195 2943.015 20456.367 207226.170 1639084.370 15253280.851

click histogram to animate (requires adobe reader)

Optimal values of κ (2^κ trees in the forest) for various N

g	m	d	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}	2^{28}
1	3	3	12	14	12	12	12	12	12	12	12
1	2	3	12	13	10	11	10	10	10	10	10
1	2	4	12	14	10	11	10	10	10	10	10
2	2	5	12	14	11	11	10	10	10	10	11
2	2	6	12	14	16	11	10	10	10	10	11
3	2	7	12	14	16	12	10	10	10	11	11
3	2	8	12	14	16	11	10	10	10	11	11
3	4	3	12	14	12	12	12	12	12	12	12
3	3	4	12	14	13	13	12	12	12	12	12
3	4	4	12	14	13	13	12	12	12	12	12
4	2	9	12	14	16	12	10	10	10	10	11
4	3	5	12	14	16	13	12	12	12	12	12
4	3	6	12	14	16	13	12	12	12	12	13
4	2	10	12	14	16	18	10	10	10	11	11
4	5	3	12	14	15	15	15	14	14	14	14
4	6	3	12	14	12	12	12	12	12	12	12

Cyclic covers of $\mathbb{P}^1$

The p-rank of ramified covers of curves
Irene I. Bouw, Compositio. Mathematica **126** (2001), 295–322.

Lemma (Lemma 5.1)

Let $X\colon y^m = (x-x_1)^{a_1}(x-x_2)^{a_2}\cdots(x-x_r)^{a_r}$ be a cyclic cover of $\mathbb{P}^1$ over an algebraically closed field of characteristic p. If $i' \equiv pi \bmod m$ then the $(i,j),(i',j')$ coefficient of the Hasse-Witt matrix of X is given by

$$(-1)^N \sum_{n_1+\cdots n_r=N} \binom{[p\langle\frac{ia_1}{\ell}\rangle]}{n_1}\cdots\binom{[p\langle\frac{ia_r}{\ell}\rangle]}{n_r} x_1^{n_1}\cdots x_r^{n_r},$$

where $N = p(\|i\|+1-j) - (\|i'\|+1-j')$, and if $i' \not\equiv pi \bmod m$ then it is zero.

Here $\langle z\rangle$ and $[z]$ denote the fractional and integers parts of $z \in \mathbb{Q}$.

Harvey's results for arithmetic schemes

Theorem (Harvey 2014)

Let X be an arithmetic scheme. The following hold:

1. *There is a deterministic algorithm that, given a prime p, outputs $Z_{X_p} \in \mathbb{Q}[T]$ in $p(\log p)^{1+o(1)}$ time using $O(\log p)$ space.*
2. *There is a deterministic algorithm that, given a prime p, outputs $Z_{X_p} \in \mathbb{Q}[T]$ in $\sqrt{p}\,(\log p)^{2+o(1)}$ time using $O(\sqrt{p}\log p)$ space.*
3. *There is a deterministic algorithm that, given an integer N outputs $Z_{X_p} \in \mathbb{Q}[T]$ for all $p \leq N$ in time $N(\log N)^{3+o(1)}$ using $O(N\log^2 N)$ space.*

In these complexity estimates, X is fixed, only p or the bound N are part of the input (the arithmetic scheme X is effectively "hardwired" into the algorithm).

If one constrains X and fixes its representation (e.g. a smooth plane curve), one can make the dependence on X completely explicit.

This theorem is not merely an existence statement, its gives explicit algorithms.

Complexity analysis for smooth plane curves

There are four ways to compute $M_s \bmod p^e$ for $1 \le s \le e$;

1. Apply $M_s = [F^{s(p-1)}_{p\vec{v}-\vec{u}}]$; time $g^5 p^2 (\log p)^{1+o(1)}$.
   (multivariate Kronecker: $\sum_{0\le s\le e}((dsp)^2)^3 e(\log p)^{1+o(1)} = g^5p^2(\log p)^{1+o(1)}$)
2. Use $Q(k,\ell)$ to compute rows of M_s using mat-vec mults: time $g^{11}p(\log p)^{1+o(1)}$.
   ($\sum_{0\le s\le e}((ds)^2 p((ds)^2)^2 e(\log p)^{1+o(1)} = g^{11}p(\log p)^{1+o(1)}$)
3. Apply BGS to compute $Q(k,\ell)$ products: time $g^{14}\sqrt{p}(\log p)^{2+o(1)}$.
   (as above, but now we need matrix-matrix mults, dimension is $O(g^3)$)
4. Use an average polynomial time approach for $p \le N$: time $g^{14}N(\log N)^{3+o(1)}$.

Except for 1, these dominate the time to compute $Z_{C_p}(T)$ given the $M_s \bmod p^e$.
In case 1 we obtain a total complexity of $(g^5p^2 + g^{11}\log p)(\log p)^{1+o(1)}$.