# Counting points on curves in average polynomial time

David Harvey

University of New South Wales

20th February 2018
Workshop on numerical methods for algebraic curves
Le Centre Henri Lebesgue, Rennes

# 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

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

# 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 $2^{30}$.

# Example

Again take $X$ defined over $\mathbf{Q}$ by

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

The bad primes are 3, 7, 23, and for the good primes we have

$$\begin{aligned}
L_5(T) &= 1 + 10T^2 + 25T^4 \\
L_{11}(T) &= 1 - 4T + 14T^2 - 44T^3 + 121T^4 \\
L_{13}(T) &= 1 + T + 4T^2 + 13T^3 + 169T^4 \\
L_{17}(T) &= 1 + 4T + 22T^2 + 68T^3 + 289T^4 \\
L_{19}(T) &= 1 - 4T + 14T^2 - 76T^3 + 361T^4 \\
&\vdots
\end{aligned}$$

# 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 polynomial.

# Goal for today's talk

Today I will explain in detail how to compute $L_p(T)$ for all $p < N$ in time $O(N \log^{3+\epsilon} N)$, for the simplest nontrivial case: an elliptic curve of the form

$$y^2 = x^3 + bx^2 + cx, \qquad b, c \in \mathbf{Z},\ c(b^2 - 4c) \neq 0.$$

The $L$-polynomial for each $p$ has the form

$$L_p(T) = 1 + a_p T + pT^2,$$

where $|a_p| < 2\sqrt{p}$ (the Hasse–Weil bound).

We want to compute $a_p \in \mathbf{Z}$ for all good $p < N$.

# Why I would rather live in $\mathbf{P}^2(\mathbf{R})$

# Polynomial powers

**Lemma**

*Let $u_p$ be the coefficient of $x^{(p-1)/2}$ (the "central coefficient") in the polynomial*

$$(x^2 + bx + c)^{(p-1)/2}.$$

*Then*

$$a_p \equiv u_p \pmod{p}.$$

For $p \geq 17$, the bound $|a_p| < 2\sqrt{p}$ implies that $u_p \pmod{p}$ determines $a_p \in \mathbf{Z}$ unambiguously.

So it is enough to compute $u_p \pmod{p}$ for all $p < N$.

# Polynomial powers

Sketch of proof of lemma:

The definition of the zeta function implies that

$$a_p = p + 1 - |X(\mathbf{F}_p)|.$$

For each $t \in \mathbf{F}_p$, the number of points with $x$-coordinate equal to $t$ depends on whether $t^3 + bt^2 + ct$ is a square in $\mathbf{F}_p$. We get

$$t^3 + bt^2 + ct = \begin{cases} \text{zero in } \mathbf{F}_p & \implies 1 \text{ point}, \\ \text{square in } \mathbf{F}_p & \implies 2 \text{ points}, \\ \text{nonsquare in } \mathbf{F}_p & \implies 0 \text{ points}. \end{cases}$$

There is also one point at infinity.

# Polynomial powers

(sketch of proof, continued)

Thus

$$\begin{aligned} |X(\mathbf{F}_p)| &= 1 + \sum_{t=0}^{p-1} \left[ \left( \frac{t^3 + bt^2 + ct}{p} \right) + 1 \right] \\ &\equiv 1 + \sum_{t=0}^{p-1} (t^3 + bt^2 + ct)^{(p-1)/2} \pmod{p}. \end{aligned}$$

Now expand out the right hand side, and use the fact that

$$\sum_{t=0}^{p-1} t^k \equiv \begin{cases} -1 & \text{if } p-1 \mid k, \\ 0 & \text{otherwise.} \end{cases}$$

# Example

For a running example, let's take $y^2 = xf(x)$ where

$$f(x) = x^2 - 3x - 2.$$

We have

$$\begin{aligned}
p = 5: &\quad f^2 = x^4 - 6x^3 + \boxed{5x^2} + 12x + 4, \\
p = 7: &\quad f^3 = x^6 - 9x^5 + 21x^4 + \boxed{9x^3} - 42x^2 - 36x - 8 \\
p = 11: &\quad f^5 = \cdots - 150x^7 - 95x^6 + \boxed{477x^5} + 190x^4 - 600x^3 + \cdots, \\
&\vdots \\
p = 103: &\quad f^{51} = \cdots + \boxed{-2882250240953935920621757274295x^{51}} + \cdots \\
&\vdots
\end{aligned}$$

For $p < N$, the total amount of data in this picture is roughly $N^3$.

# Recurrences

For each $n$, the coefficients of $f^n$ satisfy a linear recurrence.

Let

$$f^n = f_0^n x^{2n} + f_1^n x^{2n-1} + \cdots + f_{2n}^n.$$

Exercise: using the relations

$$f^{n+1} = f \cdot f^n, \qquad (f^{n+1})' = (n+1) f' \cdot f^n,$$

prove that

$$f_k^n = \frac{1}{k}\left((n-k+1) b f_{k-1}^n + (2n-k+2) c f_{k-2}^n\right).$$

# Recurrences

Problem: it's a different recurrence for each $n$!

$$f_k^n = \frac{1}{k}\left((n-k+1)bf_{k-1}^n + (2n-k+2)cf_{k-2}^n\right).$$

# Recurrences

Problem: it's a different recurrence for each $n$!

$$f_k^n = \frac{1}{k}\left((n-k+1)bf_{k-1}^n + (2n-k+2)cf_{k-2}^n\right).$$

But we only need the coefficients modulo $p$, and only for $n = (p-1)/2$:

$$f_k^{(p-1)/2} = \frac{1}{k}\left((-k+\tfrac{1}{2})bf_{k-1}^{(p-1)/2} + (-k+1)cf_{k-2}^{(p-1)/2}\right) \pmod{p}.$$

So now we have the same recurrence for each $p$.

# Recurrences

Let us rewrite the recurrence in vector form. Define

$$v_k^p := \begin{bmatrix} f_k^{(p-1)/2} \\ f_{k-1}^{(p-1)/2} \end{bmatrix} \in \mathbf{Z}^2.$$

Then

$$v_k^p = \frac{1}{2k} A_k v_{k-1}^p \pmod{p}$$

where

$$A_k := \begin{bmatrix} (-2k+1)b & (-2k+2)c \\ 2k & 0 \end{bmatrix}.$$

Notice that $A_k$ is defined over $\mathbf{Z}$, and no longer depends on $p$!!

# Recurrences

The initial conditions are easy: we have $v_0^p = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ for each $p$.

Therefore we have transformed the original problem into the problem of computing the matrix products

$$\begin{aligned}
A_1 &\qquad \pmod{3},\\
A_2A_1 &\qquad \pmod{5},\\
A_3A_2A_1 &\qquad \pmod{7},\\
&\vdots\\
A_{51}\cdots A_4A_3A_2A_1 &\qquad \pmod{103},\\
&\vdots
\end{aligned}$$

simultaneously, for all primes $p < N$.

# Example

For $f(x) = x^2 - 3x - 2$, we need to compute

$$
\begin{aligned}
&\begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} && \pmod 3, \\
&\begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} && \pmod 5, \\
&\begin{bmatrix} 15 & 8 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} && \pmod 7, \\
&\vdots \\
&\begin{bmatrix} 303 & 200 \\ 102 & 0 \end{bmatrix} \cdots \begin{bmatrix} 15 & 8 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} && \pmod{103}, \\
&\vdots
\end{aligned}
$$

Notice there are $O(N)$ rows, each row has $O(N)$ matrices, and the matrix entries have $O(\log N)$ bits.

# The accumulating remainder tree, in one slide

Suppose we want to compute:

$$\begin{aligned}
M_1 &\pmod{Q_1},\\
M_2 M_1 &\pmod{Q_2},\\
M_3 M_2 M_1 &\pmod{Q_3},\\
M_4 M_3 M_2 M_1 &\pmod{Q_4},\\
M_5 M_4 M_3 M_2 M_1 &\pmod{Q_5},\\
\cdots &\\
M_n M_{n-1} \cdots M_5 M_4 M_3 M_2 M_1 &\pmod{Q_n}.
\end{aligned}$$

Algorithm (assuming $n$ odd):

(1) multiply pairs of adjacent $M_i$'s and $Q_i$'s,

(2) recursively compute

$$\begin{aligned}
(M_2 M_1) &\pmod{Q_2 Q_3},\\
(M_4 M_3)(M_2 M_1) &\pmod{Q_4 Q_5},\\
\cdots &\\
(M_{n-1} M_{n-2}) \cdots (M_4 M_3)(M_2 M_1) &\pmod{Q_{n-1} Q_n},
\end{aligned}$$

(3) make the obvious corrections.

# Example

Initial problem for $N = 128$, with 63 rows:

$$
\begin{aligned}
&\begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} \quad (3), \\
&\begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} \quad (5), \\
&\begin{bmatrix} 15 & 8 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} \quad (7), \\
&\begin{bmatrix} 21 & 12 \\ 8 & 0 \end{bmatrix} \begin{bmatrix} 15 & 8 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} \quad (9), \\
&\begin{bmatrix} 27 & 16 \\ 10 & 0 \end{bmatrix} \begin{bmatrix} 21 & 12 \\ 8 & 0 \end{bmatrix} \begin{bmatrix} 15 & 8 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} \quad (11), \\
&\vdots \\
&\begin{bmatrix} 375 & 248 \\ 126 & 0 \end{bmatrix} \cdots \begin{bmatrix} 27 & 16 \\ 10 & 0 \end{bmatrix} \begin{bmatrix} 21 & 12 \\ 8 & 0 \end{bmatrix} \begin{bmatrix} 15 & 8 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 9 & 4 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix} \quad (127).
\end{aligned}
$$

# Example

First recursive step, 31 rows:

$$\begin{bmatrix} 35 & 0 \\ 12 & 0 \end{bmatrix} \quad (35),$$

$$\begin{bmatrix} 387 & 168 \\ 120 & 64 \end{bmatrix} \begin{bmatrix} 35 & 0 \\ 12 & 0 \end{bmatrix} \quad (99),$$

$$\begin{bmatrix} 1091 & 528 \\ 324 & 192 \end{bmatrix} \begin{bmatrix} 387 & 168 \\ 120 & 64 \end{bmatrix} \begin{bmatrix} 35 & 0 \\ 12 & 0 \end{bmatrix} \quad (195),$$

$$\vdots$$

$$\begin{bmatrix} 163715 & 88560 \\ 45012 & 29760 \end{bmatrix} \cdots \begin{bmatrix} 1091 & 528 \\ 324 & 192 \end{bmatrix} \begin{bmatrix} 387 & 168 \\ 120 & 64 \end{bmatrix} \begin{bmatrix} 35 & 0 \\ 12 & 0 \end{bmatrix} \quad (15875).$$

# Example

Second recursive step, 15 rows:

$$\begin{bmatrix} 15561 & 0 \\ 4968 & 0 \end{bmatrix} \quad (19305),$$

$$\begin{bmatrix} 2692297 & 1340976 \\ 805200 & 403200 \end{bmatrix} \begin{bmatrix} 15561 & 0 \\ 4968 & 0 \end{bmatrix} \quad (156009),$$

$$\vdots$$

$$\begin{bmatrix} 25150018761 & 13987917216 \\ 7115707800 & 3978428160 \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} 2692297 & 1340976 \\ 805200 & 403200 \end{bmatrix} \begin{bmatrix} 15561 & 0 \\ 4968 & 0 \end{bmatrix} \quad (236267625).$$

# Analysis

Number of recursion levels is $O(\log N)$.

At top level, have $O(N)$ matrices with $O(\log N)$-bit entries.

At each recursive level, half as many matrices, but entries have twice as many bits... so bit size at each level is still $O(N \log N)$.

Use FFT integer multiplication and division: cost is $O(N \log^{2+\epsilon} N)$ per level.

Total cost: $O(N \log^{3+\epsilon} N)$ bit operations (ignoring bit size of $b$ and $c$).

# Sample timings for hyperelliptic curves

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

| | |
|---|---|
| Baby-step/giant-step (`smalljac`) | 1.4 years |
| Average polynomial time | 1.3 days |

Genus 3, time to compute $L_p(T)$ for all $p < 2^{30}$:

| | |
|---|---|
| Accelerated Kedlaya (`hypellfrob`) | 3.8 years |
| Average polynomial time | 4.0 days |

(Timings from H. & Sutherland, 2016)

# Summary

- The “accumulating remainder tree” algorithm can be used to evaluate certain types of matrix products modulo many primes simultaneously.
- It is very memory intensive, and spends most of its time computing Fourier transforms of large integers.
- In the application to point counting, one must first express the point-counting problem in terms of such matrix products.

Thank you!