Computing zeta functions and L-functions
Lecture 3

Andrew V. Sutherland

June 25, 2019

CMI-HIMR Summer School in Computational Number Theory

Computing Frobenius traces of elliptic curves

Let $E/\mathbb{Q}$ be an elliptic curve $y^2 = f(x)$ and let p an odd prime of good reduction.

$$\#E_p(\mathbb{F}_p) = p + 1 - a_p = 1 + \sum_{x_0 \in \mathbb{F}_p} \left(1 + \left(\frac{f(x_0)}{p}\right)\right)$$

Reducing modulo p yields

$$a_p \equiv -\sum_{x_0 \in \mathbb{F}_p} \left(\frac{f(x_0)}{p}\right) \equiv -\sum_{x_0 \in \mathbb{F}_p} f(x_0)^{(p-1)/2} \equiv -\sum_{x_0 \in \mathbb{F}_p} f^{(p-1)/2}(x_0) \bmod p, \qquad (*)$$

which determines $a_p \in \mathbb{Z}$ for all $p > 13$ (since $|a_p| \le 2\sqrt{p}$). For $k > 0$ we have

$$\sum_{x_0 \in \mathbb{F}_p} x_0^k = \begin{cases} -1 & (p-1)|k \\ 0 & \text{otherwise} \end{cases}$$

(sums of non-trivial roots of unity vanish), and $\deg f^{(p-1)/2} < 2(p-1)$, so (*) reduces to

$$a_p \equiv f_{p-1}^{(p-1)/2} \bmod p.$$

where $f_{p-1}^{(p-1/)2}$ is the coefficient of x^{p-1} in $f^{(p-1)/2}$; this is the Hasse invariant of E_p.

Recurrence relations

Now let $f \in \mathbb{Z}[x]$ have degree r. For $n \geq 0$ and $k \in \mathbb{Z}$, let f_k^n be the coefficient of x_k in f^n. The relations $f^{n+1} = f \cdot f^n$ and $(f^{n+1})' = (n+1)f' \cdot f^n$ yield the identities

$$f_k^{n+1} = \sum_{0 \leq i \leq r} f_i f_{k-i}^n \qquad \text{and} \qquad k f_k^{n+1} = (n+1) \sum_{0 \leq i \leq r} i f_i f_{k-i}^n.$$

Multiplying the first identity by k and subtracting the second yields the linear recurrence

$$k f_0 f_k^n = \sum_{1 \leq i \leq r} \big((n+1)i - k\big) f_i f_{k-i}^n,$$

We can express this recurrence in terms of $v_k^n \in \mathbb{Z}^r$ and $R_k^n \in \mathbb{Z}^{r \times r}$, which for $r = 3$ look like

$$v_k^n := [f_{k-2}^n, f_{k-1}^n, f_k^n], \qquad R_k^n := \begin{bmatrix} 0 & 0 & (3n+3-k)f_3 \\ k f_0 & 0 & (2n+2-k)f_2 \\ 0 & k f_0 & (n+1-k)f_1 \end{bmatrix}.$$

Provided $f_0 \neq 0$, we have $v_k^n = (k f_0)^{-1} v_{k-1}^n = (k!(f_0)^k)^{-1} v_0^n R_1^n \cdots R_k^n$ for all $k, n \geq 0$.

Computing the Hasse invariant

Now consider $f(x) = x^3 + Ax + B$, where $E : y^2 = x^3 + Ax + B$ has good reduction at p.
Let us assume $f_0 = B \neq 0$ (or work with $g = f/x$ and $g^n = f/x^n$ which is even easier).

To compute $a_p \equiv f_{p-1}^{(p-1)/2} \bmod p$, it suffices to compute $v_{2n}^n \bmod (2n+1)$ for $n = (p-1)/2$.
We have $2(n+1) \equiv 1 \bmod p$ and now define

$$M_k := 2R_k^n \bmod p = \begin{bmatrix} 0 & 0 & (3-2k)f_3 \\ kf_0 & 0 & (3-2k)f_2 \\ 0 & kf_0 & (1-2k)f_1 \end{bmatrix} \bmod p,$$

which we not is independent of n. We then have

$$v_{2n}^n \equiv \frac{1}{(2n)!f_0^{2n}} v_0^n \frac{1}{2^{2n}} M_1 M_2 \cdots M_{2n} \equiv -v_0^n M_1 \cdots M_{p-1} \bmod p,$$

where we have used $(2n)! = (p-1)! \equiv -1 \bmod p$ and $a^{2n} = a^{p-1} \equiv 1 \bmod p$ for $p \nmid a$.
Computing $a_p \bmod p$ reduces to computing $M_1 \cdots M_{p-1} \bmod p$ and $v_0^n = [0, 0, f_0^n] \bmod p$.

Complexity analysis for a single prime p

If we simply evaluate the matrix product $M_1 \cdots M_{p-1} \bmod p$ we obtain a bit-complexity of

$$O(p\,\mathsf{M}(\log p)) = O(p \log p \log\log p)$$

which we is already slightly better than naïve point counting (even with a fast implementation of the Legendre symbol rather than counting square roots of $f(x_0)$ for each $x_0 \in \mathbb{F}_p$).

To improve this, let us view $M_k \bmod p$ as $M(k) \in \mathbb{F}_p[k]^{3\times 3}$, fix $s := \lfloor\sqrt{p-1}\rfloor$, and define

$$A(k) := M(k)M(k+1)\cdots M(k+s-1) \in \mathbb{F}_p[k]^{3\times 3},$$

We can then compute the desired matrix product as

$$M_1M_2\cdots M_{p-1} \equiv_p A(1)A(s+1)A(2s+1)\cdots A((s-1)s+1)M_{s^2+1}\cdots M_{p-1}$$

Using a product tree to compute $A(k)$, and standard multipoint evaluation yields a complexity of $p^{1/2}(\log p)^{2+o(1)}$. Applying the algorithm of Bostan-Gaudry-Schöst improves this to

$$p^{1/2}(\log p)^{1+o(1)}$$

Accumulating remainder tree

Given integer matrices $A_0, \ldots, A_{n-1}$ and integer moduli $m_1, \ldots, m_n$, we want to compute

$$C_j := A_0 \cdots A_{j-1} \bmod m_j$$

for $1 \le j \le n$. We now define $B_i = A_{2i}A_{2i+1}$ and $n_i = m_{2i}m_{2i+1}$ (pad as needed to make n even). We now recursively compute $D_i = B_0 \cdots B_{i-1} \bmod n_i$ for $1 \le i < n/2$, put $C_1 = A_0 \bmod m_1$ and

$$C_2 i = D_i \bmod m_{2i} \qquad \text{and} \qquad C_{2i+1} = (D_i \bmod m_{2i+1})A_{2i}.$$

The recursion depth is $O(\log n)$, and the total number of bits at each level is roughly the same (the matrix dimension is fixed, so the bit size of a product of two matrices is the sum of the bit sizes of the factors plus $O(1)$ bits; the total bit size increases by $O(\log n)$ bits over the course of the algorithm).

If we assume the entries of the matrices A_i and the moduli m_i have $O(\log n)$ bits (this is true in our application, since the f_i are fixed and $k \le n$), then the total complexity is $O(\mathsf{M}(n \log n) \log n)$ or

$$O(n(\log n)^3 \log\log n) \qquad \text{or} \qquad O((\log p)^4 \log\log p) \text{ per prime}$$

This is asymptotically faster than both Schoof's algorithm and the expected running time of SEA, even under the best-case heuristic assumptions for SEA (by a factor of $(\log\log p)^2$)