Computing L-Series of genus 3 curves

Andrew V. Sutherland

Massachusetts Institute of Technology

July 1, 2017

Joint work with David Harvey; David Harvey and Maike Massierer; David Harvey; Andrew Booker, and David Platt.

The L-series of a curve

Let X be a nice (smooth, projective, geometrically integral) curve of genus g over $\mathbb{Q}$. The L-series of X is the Dirichlet series

$$L(X,s) = L(\mathrm{Jac}(X),s) := \sum_{n\geq 1} a_n n^{-s} := \prod_p L_p(p^{-s})^{-1}.$$

For primes p of good reduction for X we have the zeta function

$$Z(X_p;s) := \exp\left(\sum_{r\geq 1} \#X(\mathbb{F}_{p^r})\frac{T^r}{r}\right) = \frac{L_p(T)}{(1-T)(1-pT)},$$

and the L-polynomial $L_p \in \mathbb{Z}[T]$ in the numerator satisfies

$$L_p(T) = T^{2g}\chi_p(1/T) = 1 - a_pT + \cdots + p^g T^{2g}$$

where $\chi_p(T)$ is the charpoly of the Frobenius endomorphism of $\mathrm{Jac}(X_p)$.

The Selberg class with polynomial Euler factors

The Selberg class S^{poly} consists of Dirichlet series $L(s) = \sum_{n\geq 1} a_n n^{-s}$:

1. $L(s)$ has an analytic continuation that is holomorphic at $s \neq 1$;
2. For some $\gamma(s) = Q^s \prod_{i=1}^r \Gamma(\lambda_i s + \mu_i)$ and ε, the completed L-function $\Lambda(s) := \gamma(s)L(s)$ satisfies the functional equation

   $$\Lambda(s) = \varepsilon\overline{\Lambda(1-\bar{s})},$$

   where $Q > 0$, $\lambda_i > 0$, $\mathrm{Re}(\mu_i) \geq 0$, $|\varepsilon| = 1$. Define $\deg L := 2\sum_i^r \lambda_i$.
3. $a_1 = 1$ and $a_n = O(n^\epsilon)$ for all $\epsilon > 0$ (Ramanujan conjecture).
4. $L(s) = \prod_p L_p(p^{-s})^{-1}$ for some $L_p \in \mathbb{Z}[T]$ with $\deg L_p \leq \deg L$ (has an Euler product).

The Dirichlet series $L_{\mathrm{an}}(s, X) := L(X, s + \frac{1}{2})$ satisfies (3) and (4), and conjecturally lies in S^{poly}; for $g = 1$ this is known (via modularity).

Strong multiplicity one

Theorem (Kaczorowski-Perelli 2001)

If $A(s) = \sum_{n\geq 1} a_n n^{-s}$ and $B(s) = \sum_{n\geq 1} b_n n^{-s}$ lie in S^{poly} and $a_p = b_p$ for all but finitely many primes p, then $A(s) = B(s)$.

Corollary

If $L_{\mathrm{an}}(s, X)$ lies in S^{poly} then it is completely determined by (any choice of) all but finitely many coefficients a_p.

Henceforth we assume that $L_{\mathrm{an}}(s, X) \in S^{\mathrm{poly}}$.

Let $\Gamma_{\mathbb{C}}(s) = 2(2\pi)^s\Gamma(s)$ and define $\Lambda(X, s) := \Gamma_{\mathbb{C}}(s)^g L(X, s)$. Then

$$\Lambda(X, s) = \varepsilon N^{1-s}\Lambda(X, 2-s).$$

where the root number $\varepsilon = \pm 1$ and the analytic conductor $N \in \mathbb{Z}_{\geq 1}$ are determined by the a_p values (we view these as definitions).

Testing the functional equation

Let $G(x)$ be the inverse Mellin transform of $\Gamma_{\mathbb{C}}(s)^g = \int_0^\infty G(x)x^{s-1}dx$, and define

$$S(x) := \frac{1}{x}\sum a_n G(n/x),$$

so that $\Lambda(X,s) = \int_0^\infty S(x)x^{-s}dx$, and for all $x > 0$ we have

$$S(x) = \varepsilon S(N/x).$$

The function $G(x)$ decays rapidly, and for sufficiently large c_0 we have

$$S(x) \approx S_0(x) := \frac{1}{x}\sum_{n \le c_0 x} a_n G(n/x),$$

with an explicit bound on the error $|S(x) - S_0(x)|$.

Effective strong multiplicity one

Fix a finite set of small primes $\mathcal{S}$ (e.g. $\mathcal{S} = \{2\}$) and an integer M that we know is a multiple of the conductor N (e.g. $M = \Delta(X)$).

There is a finite set of possibilities for $\varepsilon = \pm 1$, $N|M$, and the Euler factors $L_p \in \mathbb{Z}[T]$ for $p \in \mathcal{S}$ (the coefficients of $L_p(T)$ are bounded).

Suppose we can compute a_n for $n \le c_1\sqrt{M}$ whenever $p \nmid n$ for $p \in \mathcal{S}$.

We now compute $\delta(x) := |S_0(x) - \varepsilon S_0(N/x)|$ with $x = c_1\sqrt{N})$ for every possible choice of ε, N, and $L_p(T)$ for $p \in \mathcal{S}$. If all but one choice makes $\delta(x)$ larger than our explicit error bound, we know the correct choice.

For a suitable choice of c_1 this is guaranteed to happen.[1] One can explicitly determine a set of $O(N^\epsilon)$ candidate values of c_1, one of which is guaranteed to work; in practice the first one usually works.

[1] Subject to our assumptions; if it does not happen then we have found an explicit counterexample to the conjectured Langlands correspondence.

Conductor bounds

The formula of Brumer and Kramer gives explicit bounds on the p-adic valuation of the algebraic conductor N of $\mathrm{Jac}(X)$:

$$v_p(N) \le 2g + pd + (p-1)\lambda_p(d),$$

where $d = \lfloor \frac{2g}{p-1} \rfloor$ and $\lambda_p(d) = \sum i d_i p^i$, with $d = \sum d_i p^i$ with $0 \le d_i < p$.

g	$p=2$	$p=3$	$p=5$	$p=7$	$p>7$
1	8	5	2	2	2
2	20	10	9	4	4
3	28	21	11	13	6

For $g \le 2$ these bounds are tight (see `www.lmfdb.org` for examples).

For hyperelliptic curves N divides $\Delta(X)$. Smooth plane curves?

Algorithms to compute zeta functions

Given $X/\mathbb{Q}$ of genus g, we want to compute $L_p(T)$ for all good $p \leq B$.

	complexity per prime (ignoring factors of $O(\log\log p)$)		
algorithm	$g = 1$	$g = 2$	$g = 3$
point enumeration	$p \log p$	$p^2 \log p$	$p^3 (\log p)^2$
group computation	$p^{1/4} \log p$	$p^{3/4} \log p$	$p (\log p)^2$
p-adic cohomology	$p^{1/2} (\log p)^2$	$p^{1/2} (\log p)^2$	$p^{1/2} (\log p)^2$
CRT (Schoof-Pila)	$(\log p)^5$	$(\log p)^8$	$(\log p)^{12?}$
average poly-time	$(\log p)^4$	$(\log p)^4$	$(\log p)^4$

For $L(X,s) = \sum a_n n^{-s}$, we only need a_{p^2} for $p^2 \leq B$, and a_{p^3} for $p^3 \leq B$.
For $1 < r \leq g$ we can compute all a_{p^r} with $p^r \leq B$ in time $O(B \log B)$.

The bottom line: it all comes down to computing a_p's.

Warmup: average polynomial-time in genus 1

Let $X : y^2 = f(x)$ with $\deg f = 3, 4$ and $f(0) \neq 0$, and let f_k^n be the coefficient of x^k in f^n. Then $a_p \equiv f_{p-1}^{(p-1)/2} \bmod p$ for all good p.

The relations $f^{n+1} = f \cdot f^n$ and $(f^{n+1})' = (n+1)f' \cdot f^n$ yield the identity

$$kf_0 f_k^n = \sum_{1 \leq i \leq d} (n+1) - k) f_i f_{k-i}^n,$$

for all $k, n \geq 0$. Suppose for simplicity $\deg f = 3$, and define

$$v_k^n := [f_{k-2}^n, f_{k-1}^n, f_k^n], \qquad M_k^n := \begin{bmatrix} 0 & 0 & (3n+3-k)f_3 \\ kf_0 & 0 & (2n+2-k)f_2 \\ 0 & kf_0 & (n+1-k)f_1 \end{bmatrix},$$

so that we have the recurrence $v_k^n = \frac{1}{kf_0} v_{k-1}^n M_k^n$.

Warmup: average polynomial-time in genus 1

We then have

$$v_k^n = \frac{1}{(f_0)^k k!} v_0^n M_1^n \cdots M_k^n.$$

We want to compute $a_p \equiv f_{2n}^n \bmod p$ with $n := (p-1)/2$.
This is just the last entry of the vector v_{2n}^n reduced modulo $p = 2n+1$.

Observe that $2(n+1) \equiv 1 \bmod p$, so $2M_k^n \equiv M_k \bmod p$, where

$$M_k := \begin{bmatrix} 0 & 0 & (3-2k)f_3 \\ kf_0 & 0 & (2-2k)f_2 \\ 0 & kf_0 & (1-2k)f_1 \end{bmatrix}$$

is an integer matrix whose entries do not depend on $p = 2n+1$, and

$$v_{2n}^n \equiv -\left(\frac{f_0}{p}\right) V_0 M_1 \cdots M_{p-1} \bmod p \qquad \text{(where } V_0 = [0,0,1]\text{)}.$$

Accumulating remainder tree

Given matrices $M_0, \ldots, M_{n-1}$ and moduli $m_1, \ldots, m_n$, to compute

$$\begin{aligned} M_0 &\bmod m_1 \\ M_0 M_1 &\bmod m_2 \\ M_0 M_1 M_2 &\bmod m_3 \\ M_0 M_1 M_2 M_3 &\bmod m_4 \\ &\ldots \\ M_0 M_1 \cdots M_{n-2} M_{n-1} &\bmod m_n \end{aligned}$$

multiply adjacent pairs and recursively compute

$$\begin{aligned} (M_0 M_1) &\bmod m_2 m_3 \\ (M_0 M_1)(M_2 M_3) &\bmod m_4 m_5 \\ &\ldots \\ (M_0 M_1) \cdots (M_{n-2} M_{n-1}) &\bmod m_n \end{aligned}$$

and adjust the results as required (for better results, use a forest).

Complexity analysis

Assume $\log|f_i| = O(\log B)$. The recursion has depth $O(\log B)$ and in each recursive step we multiply and reduce 3×3 matrices with integer entries whose total bitsize is $O(B \log B)$.

We can do all the multiplications/reductions at any given level of the recursion in $O(\mathsf{M}(B \log B)) = B(\log B)^{2+o(1)}$.

Total complexity is $B(\log B)^{3+o(1)}$, or $(\log p)^{4+o(1)}$ per prime $p \leq B$.

For a single prime p we do not have a polynomial-time algorithm, but we can give an $O(p^{1/2}(\log p)^{1+o(1)})$ algorithm using the same matrices.

This is a silly way to compute a_p in genus 1, but it turns out to be much faster than any other method currently available in genus 3.

Efficiently handling a single prime

Simply computing $V_0 M_1 \cdots M_{p-1}$ modulo p is surprisingly quick (faster than semi-naïve point-counting); it takes $p(\log p)^{1+o(1)}$ time.

But we can do better.

Viewing $M_k \bmod p$ as $M \in \mathbb{F}_p[k]^{3\times 3}$, we compute

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

with $r \approx \sqrt{p}$ and then instantiate $A(k)$ at roughly r points to get

$$M_1 M_2 \cdots M_{p-1} \equiv_p A(1)A(r+1)A(2r+1)\cdots A(p-r).$$

Using standard product tree and multipoint evaluation techniques this takes $O(\mathsf{M}(p^{1/2})\log p) = p^{1/2}(\log p)^{2+o(1)}$ time.

Bostan-Gaudry-Schost: $p^{1/2}(\log p)^{1+o(1)}$ time.

Genus 3 curves

The canonical embedding of a genus 3 curve into $\mathbb{P}^2$ is either

1. a degree-2 cover of a smooth conic (hyperelliptic case);
2. a smooth plane quartic (generic case).

Average polynomial-time implementations available for the first case:

- rational hyperelliptic model [Harvey-S 2014]
- no rational hyperelliptic model [Harvey-Massierer-S 2016].

New result (joint with Harvey): smooth plane quartics.

Prior work has all been based on p-adic cohomology:

[Lauder 2004], [Castryck-Denef-Vercauteren 2006], [Abott-Kedlaya-Roe 2006], [Harvey 2010], [Tuitman-Pancrantz 2013], [Tuitman 2015], [Costa 2015], [Tuitman-Castryck 2016], [Shieh 2016]

Current implementations of these algorithms are all $O(p^{1+o(1)})$.

The Hasse-Witt matrix of a hyperelliptic curve

Let $X_p/\mathbb{F}_p$ be a hyperelliptic curve $y^2 = f(x)$ of genus g (assume p odd).
As in the warmup, let f_k^n denote the coefficient of x^k in f^n.

The Hasse–Witt matrix of X_p is $W_p := [f^n_{pi-j}]_{ij} \in \mathbb{F}_p^{g\times g}$ with $n = (p-1)/2$.
In genus $g = 3$ we have

$$W_p := \begin{bmatrix} f^n_{p-1} & f^n_{p-2} & f^n_{p-3} \\ f^n_{2p-1} & f^n_{2p-2} & f^n_{2p-3} \\ f^n_{3p-1} & f^n_{3p-2} & f^n_{3p-3} \end{bmatrix}.$$

This is the matrix of the p-power Frobenius acting on $H^1(C_p, \mathcal{O}_{C_p})$
(and the Cartier-Manin operator acting on regular differentials).
As proved by Manin, we have

$$L_p(T) \equiv \det(I - TW_p) \bmod p;$$

in particular, $a_p \equiv \operatorname{tr} W_p \bmod p$. For $p > 144$ this yields $a_p \in [-6\sqrt{p}, 6\sqrt{p}]$.

Hyperelliptic average polynomial-time

As in our warmup, assume $f(0) \neq 0$ and define $v_k^n := [f_{k-d+1}^n, \ldots, f_k^n]$. The last g entries of v_{2n}^n form the first row of W_p, and we have

$$v_{2n}^n = -\left(\frac{f_0}{p}\right) V_0 M_1 \cdots M_{p-1} \bmod p \qquad \text{(where } V_0 = [0, \ldots, 0, 1]\text{)}.$$

Compute the first row of W_p for good $p \leq B$ in $O(g^2 B (\log B)^{3+o(1)})$ time.

To get the remaining rows, consider the isomorphic curve $y^2 = f(x + a)$ whose Hasse-Witt matrix $W_p(a) = T(a) W_p T(-a)$ is conjugate to W_p via

$$T(a) := \left[\binom{j-1}{i-1} a^{j-1}\right]_{ij} \in \mathbb{F}_p^{g \times g}.$$

Given the first row of $W_p(a)$ for g distinct values of a we can compute all the rows of W_p. Total complexity is $O(g^3 B (\log B)^{3+o(1)})$.

The Hasse-Witt matrix of a smooth plane quartic

Let $X_p/\mathbb{F}_p$ be a smooth plane quartic defined by $f(x,y,z) = 0$.
For $n \geq 0$ let $f^n_{i,j,k}$ denote the coefficient of $x^i y^j z^k$ in f^n.

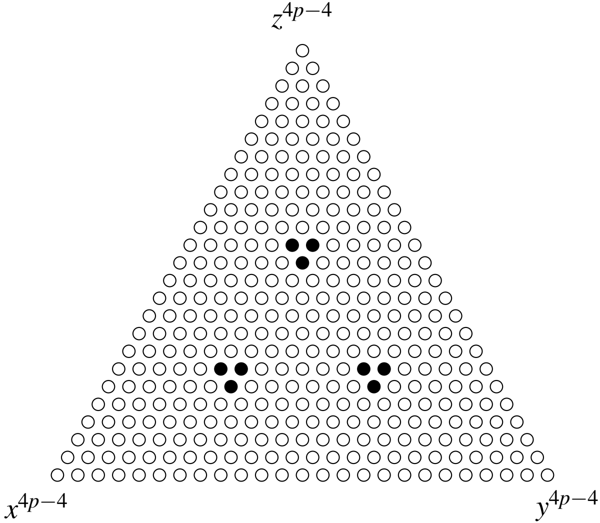
The Hasse–Witt matrix of X_p is the 3×3 matrix

$$W_p := \begin{bmatrix} f^{p-1}_{p-1,p-1,2p-2} & f^{p-1}_{2p-1,p-1,p-2} & f^{p-1}_{p-1,2p-1,p-2} \\ f^{p-1}_{p-2,p-1,2p-1} & f^{p-1}_{2p-2,p-1,p-1} & f^{p-1}_{p-2,2p-1,p-1} \\ f^{p-1}_{p-1,p-2,2p-1} & f^{p-1}_{2p-1,p-2,p-1} & f^{p-1}_{p-1,2p-2,p-1} \end{bmatrix}.$$

This case of smooth plane curves of degree $d > 4$ is similar.

More generally, given a singular plane model for any nice curve (equivalently, a defining polynomial for its function field) one can use the methods of Stohr-Voloch to explicitly determine W_p.

Target coefficients of f^{p-1} for $p = 7$:



Coefficient relations

Let $\partial_x = x\frac{\partial}{\partial x}$ (degree-preserving). The relations

$$f^{p-1} = f \cdot f^{p-2} \qquad \text{and} \qquad \partial_x f^{p-1} = -(\partial_x f) f^{p-2}$$

yield the relation

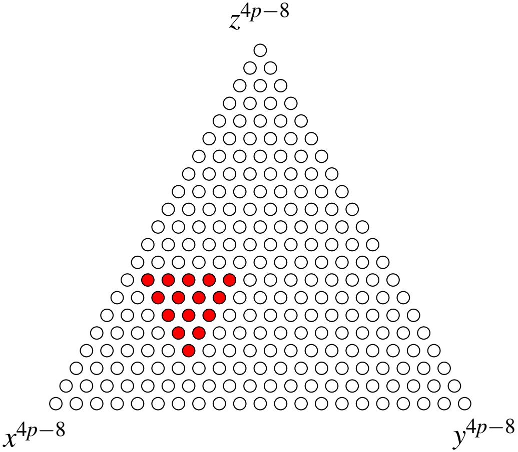
$$\sum_{i'+j'+k'=4} (i+i') f_{i',j',k'} f^{p-2}_{i-i',j-j',k-k'} = 0.$$

among nearby coefficients of f^{p-2} (a triangle of side length 5).

Replacing ∂_x by ∂_y yields a similar relation (replace $i+i'$ with $j+j'$).

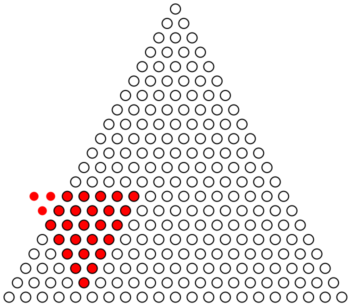
Coefficient triangle

For $p = 7$ with $i = 12, j = 5, k = 7$ the related coefficients of f^{p-2} are:



Moving the triangle

Now consider a bigger triangle with side length 7.
Our relations allow us to move the triangle around:



An initial “triangle” at the edge can be efficiently computed using coefficients of $f(x, 0, z)^{p-2}$.

Computing one Hasse-Witt matrix

Nondegeneracy: we need $f(1,0,0), f(0,1,0), f(0,0,1)$ nonzero and $f(0,y,z), f(x,0,z), f(x,y,0)$ squarefree (easily achieved for large p).

The basic strategy to compute W_p is as follows:

- There is a 28×28 matrix M_j that shifts our 7-triangle from y-coordinate j to $j+1$; its coefficients depend on j and f.
  In fact a 16×16 matrix M_i suffices (use smoothness of C).
- Applying the product $M_0 \cdots M_{p-2}$ to an initial triangle on the edge and applying a final adjustment to shift from f^{p-2} to f^{p-1} gets us one column of the Hasse-Witt matrix W_p.
- By applying the same product (or its inverse) to different initial triangles we can compute all three columns of W_p.

We have thus reduced the problem to computing $M_1 \cdots M_{p-2} \bmod p$, which we already know how to do, either in $p^{1/2}(\log p)^{1+o(1)}$ time, or in average polynomial time $(\log p)^{4+o(1)}$.

Cumulative timings for genus 3 curves

Time to compute $L_p(T) \bmod p$ for all good $p \leq B$.

B	`spq-Costa-AKR`	`spq-HS`	`ghyp-MHS`	`hyp-HS`	`hyp-Harvey`
2^{12}	18	1.4	0.3	0.1	1.3
2^{13}	49	2.4	0.7	0.2	2.6
2^{14}	142	4.6	1.7	0.5	5.4
2^{15}	475	9.4	4.6	1.0	12
2^{16}	1,670	21	11	2.1	29
2^{17}	5,880	47	27	5.3	74
2^{18}	22,300	112	62	14	192
2^{19}	78,100	241	153	37	532
2^{20}	297,000	551	370	97	1,480
2^{21}	1,130,000	1,240	891	244	4,170
2^{22}	4,280,000	2,980	2,190	617	12,200
2^{23}	16,800,000	6,330	5,110	1,500	36,800
2^{24}	66,800,000	14,200	11,750	3,520	113,000
2^{25}	244,000,000	31,900	28,200	8,220	395,000
2^{26}	972,000,000	83,300	62,700	19,700	1,060,000

(Intel Xeon E7-8867v3 3.3 GHz CPU seconds).