Computing L-series coefficients of hyperelliptic curves

Kiran S. Kedlaya and Andrew V. Sutherland

Massachusetts Institute of Technology

May 19, 2008

Demonstration

The distribution of Frobenius traces

Let C be a genus g curve defined over $\mathbb{Q}$. We may compute

$$\#C/\mathbb{F}_p = p - a_p + 1,$$

for each $p \leq N$ where C has good reduction, and plot the distribution of $a_p/\sqrt{p}$ over the interval $[-2g, 2g]$.

What does the picture look like for increasing values of N?

`http://math.mit.edu/~drew`

The object of interest

The numerator of the zeta function

$$Z(C/\mathbb{F}_p;T)=\exp\left(\sum_{k=1}^{\infty} c_k T^k/k\right)=\frac{\mathbf{L_p(T)}}{(1-T)(1-pT)}.$$

The polynomial $L_p(T)$ has integer coefficients

$$L_p(T)=p^g T^{2g}+a_1 p^{g-1} T^{2g-1}+\cdots+a_g T^g+\cdots+a_1 T+1.$$

$L_p(t)$ determines the order of the Jacobian $\#J(C/\mathbb{F})_p=L_p(1)$, the trace of Frobenius $a_p=-a_1$, and $L(C,s)=\prod L_p(p^{-s})^{-1}$.

A computational challenge

The task at hand

Compute $L_p(T)$ for all $p \leq N$ where C has good reduction.

We will assume C is hyperelliptic, genus $g \leq 3$, of the form

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

where $f(x) \in \mathbb{Q}[x]$ has degree $2g + 1$ (one point at ∞).

Some questions

- Which algorithm should we use? (all of them)
- How big can we make N, in practice? $(10^{12}, 10^8, 10^7)$

The complexity is necessarily exponential in N.
We expect to compute many $L_p(T)$ for reasonably small p.

Algorithms

Point counting

Compute $\#C/F_p$, $\#C/F_{p^2}, \ldots, \#C/F_{p^g}$.
Time: $O(p)$, $O(p^2)$, $O(p^3)$.

Algorithms

Point counting

Compute $\#C/F_p$, $\#C/F_{p^2}, \ldots, \#C/F_{p^g}$.
Time: $O(p)$, $O(p^2)$, $O(p^3)$.

Generic group algorithms

Compute $\#J(C/F_p) = L_p(1)$ and $\#J(\tilde{C}/\mathbb{F}_p) = L_p(-1)$.
Time: $O(p^{1/4})$, $O(p^{3/4})$, $O(p^{5/4})$.

Algorithms

Point counting

Compute $\#C/F_p$, $\#C/F_{p^2}$,...,$\#C/F_{p^g}$.
Time: $O(p)$, $O(p^2)$, $O(p^3)$.

Generic group algorithms

Compute $\#J(C/F_p) = L_p(1)$ and $\#J(\tilde{C}/\mathbb{F}_p) = L_p(-1)$.
Time: $O(p^{1/4})$, $O(p^{3/4})$, $O(p^{5/4})$.

p-adic methods

Compute Frobenius charpoly $\chi(T) = T^{-2g}L_p(T) \bmod p^k$.
Time: $\tilde{O}(p^{1/2})$.

Algorithms

Point counting

Compute $\#C/F_p$, $\#C/F_{p^2}, \ldots, \#C/F_{p^g}$.
Time: $O(p)$, $O(p^2)$, $O(p^3)$.

Generic group algorithms

Compute $\#J(C/F_p) = L_p(1)$ and $\#J(\tilde{C}/\mathbb{F}_p) = L_p(-1)$.
Time: $O(p^{1/4})$, $O(p^{3/4})$, $O(p^{5/4})$.

***p*-adic methods**

Compute Frobenius charpoly $\chi(T) = T^{-2g}L_p(T) \bmod p^k$.
Time: $\tilde{O}(p^{1/2})$.

Polynomial-time algorithms exist (Schoof-Pila) but are impractical.*

Strategy

Genus 1

Use $O(p^{1/4})$ generic group algorithm.

Genus 2

Use $O(p)$ point counting plus $O(p^{1/2})$ group operations.
Switch to $O(p^{3/4})$ group algorithm for $p > 10^6$.

Genus 3

Use $O(p)$ point counting plus $O(p)$ group operations.
Switch to $\tilde{O}(p^{1/2})$ p-adic plus $O(p^{1/4})$ group ops for $p > 10^5$.

"Elliptic and modular curves over finite fields and related computational issues", (Elkies 1997).

Point counting

Enumerating polynomials over $\mathbb{F}_p$

Define $\Delta f(x) = f(x+1) - f(x)$. Enumerate $f(x)$ from $f(0)$ via

$$f(x+1) = f(x) + \Delta f(x)$$

Enumerate $\Delta^k f(n)$ in parallel starting from $\Delta^k f(0)$.

Complexity

Requires only d additions per enumerated value, versus d multiplications and d additions using Horner's method.
Total for $y^2 = f(x)$ is $(d+1)p$ additions (no multiplications).

Generalizes to $\mathbb{F}_{p^n}$. Efficiently enumerates similar curves in parallel.

	Polynomial Evaluation		Finite Differences		Finite Differences $\times 32$	
$p \approx$	Genus 2	Genus 3	Genus 2	Genus 3	Genus 2	Genus 3
2^{18}	192.4	259.8	6.0	6.8	1.1	1.1
2^{19}	186.3	251.1	6.0	6.8	1.1	1.1
2^{20}	187.3	244.1	7.2	8.0	1.1	1.3
2^{21}	172.3	240.8	8.8	9.4	1.2	1.3
2^{22}	197.9	233.9	12.1	13.4	1.2	1.3
2^{23}	229.2	285.8	12.8	14.6	2.6	2.7
2^{24}	258.1	331.8	41.2	44.0	3.5	4.7
2^{25}	304.8	350.4	53.6	55.7	4.8	4.9
2^{26}	308.0	366.9	65.4	67.8	4.8	4.6
2^{27}	318.4	376.8	70.5	73.1	4.9	5.0
2^{28}	332.2	387.8	74.6	76.5	5.1	5.2

Point counting $y^2 = f(x)$ over $\mathbb{F}_p$ (CPU nanoseconds/point)

Generic group algorithms

High speed group operation

- Single-word Montgomery representation for $\mathbb{F}_p$.
- Explicit Jacobian arithmetic using *affine* coordinates. (unique representation of group elements)
- Modify generic algorithms to perform group operations "in parallel" to achieve $I \approx 3M$.

Randomization issues

The fastest/simplest algorithms are probabilistic.

Monte Carlo algorithms should be made Las Vegas algorithms to obtain provably correct results *and* better performance.

Non-group operations also need to be fast (e.g., table lookup).

		Standard			Montgomery		
g	p	$\times 1$	$\times 10$	$\times 100$	$\times 1$	$\times 10$	$\times 100$
1	$2^{20}+7$	501	245	215	239	89	69
1	$2^{25}+35$	592	255	216	286	93	69
1	$2^{30}+3$	683	264	217	333	98	69
2	$2^{20}+7$	1178	933	902	362	216	196
2	$2^{25}+35$	1269	942	900	409	220	197
2	$2^{30}+3$	1357	949	902	455	225	196
3	$2^{20}+7$	2804	2556	2526	642	498	478
3	$2^{25}+35$	2896	2562	2528	690	502	476
3	$2^{30}+3$	2986	2574	2526	736	506	478

Black box performance (CPU nanoseconds/group operation).

Computing the order of a generic abelian group

Computing the structure of G

Decompose G as a product of cyclic groups:

1. Compute $|\alpha|$ for random $\alpha \in G$ to obtain $\lambda(G) = \mathrm{lcm}|\alpha|$.
2. Using $\lambda(G)$, compute a basis for each Sylow p-subgroup, via discrete logarithms.

See Sutherland thesis (2007) for details (avoids SNF).

Benefits of working in Jacobians

Step 1 is aided by bounds on $|G|$ and knowledge of $|G| \mod \ell$.

Given $M \leq |G| < 2M$, step 2 takes $O(|G|^{1/4})$ group operations.

If $\lambda(G) > M$, step 2 is unnecessary (often the case).

In genus 1, structure is not required, but it is necessary for $g > 1$.

Optimizing for distribution

Generalized Sato-Tate conjecture (Katz-Sarnak)

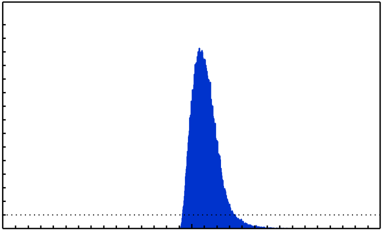
The distribution of $L_p(p^{-1/2}T)$ for a "typical" genus g curve is equal to the distribution of the characteristic polynomial of a random matrix in $USp(2g)$ (according to the Haar measure μ).

Optimized BSGS search

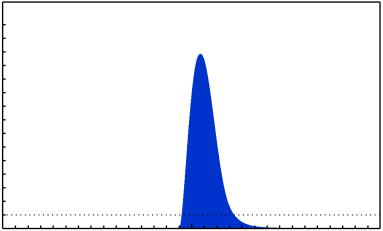
Using μ, we can compute the expected distance of a_1 (or better, a_2 given a_1) from its median value, and then choose an appropriate number of baby steps.

In genus 3 this reduces the expected search interval by a factor of 10.

$$y^2 = x^7 + 314159x^5 + 271828x^4 + 1644934x^3 + 57721566x^2 + 1618034x + 141421$$



Actual a_2 distribution



Predicted a_2 distribution

p-adic methods

Kedlaya's algorithm over a prime field

Approximates the $(2g \times 2g)$ matrix of the Frobenius action on the Monsky-Washnitzer cohomology, accurate modulo p^k:

$$\tilde{O}(pg^2k^2) = \tilde{O}(p)$$

Harvey's improvements

Apply BGS fast linear recurrence reduction to obtain:

$$\tilde{O}(p^{1/2}g^3k^{5/2} + g^4k^4 \log p) = \tilde{O}(p^{1/2})$$

Multipoint Kronecker substitution (Harvey 2007) improves polynomial multiplication by a factor of 3.

	Genus 2		Genus 3	
N	$\times 1$	$\times 8$	$\times 1$	$\times 8$
2^{16}	1	< 1	43	13
2^{17}	4	2	1:49	18
2^{18}	12	3	4:42	41
2^{19}	40	7	12:43	1:47
2^{20}	2:32	24	36:14	4:52
2^{21}	10:46	1:38	1:45:36	13:40
2^{22}	40:20	5:38	5:23:31	41:07
2^{23}	2:23:56	19:04	16:38:11	2:05:40
2^{24}	8:00:09	1:16:47		6:28:25
2^{25}	26:51:27	3:24:40		20:35:16
2^{26}		11:07:28		
2^{27}		36:48:52		

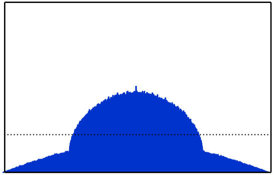
L-series computations in genus 2 and 3 (elapsed times)

N	PARI	Magma	smalljac v1	smalljac v2
2^{16}	0.26	0.29	0.07	0.04
2^{17}	0.55	0.59	0.15	0.08
2^{18}	1.17	1.24	0.30	0.16
2^{19}	2.51	2.53	0.62	0.31
2^{20}	5.46	5.26	1.29	0.63
2^{21}	11.67	11.09	2.65	1.30
2^{22}	25.46	23.31	5.53	2.68
2^{23}	55.50	49.22	11.56	5.57
2^{24}	123.02	104.50	24.31	11.66
2^{25}	266.40	222.56	51.60	24.54
2^{26}	598.16	476.74	110.29	52.07
2^{27}	1367.46	1017.55	233.94	111.24
2^{28}	3152.91	2159.87	498.46	239.32
2^{29}	7317.01	4646.24	1065.28	518.16
2^{30}	17167.29	10141.28	2292.74	1130.85

L-series computations in Genus 1 (CPU seconds)

Conclusion

All source code freely available under GPL.



drew@math.mit.edu