Computing L-series of hyperelliptic curves in moderate genus

David Harvey

University of New South Wales

2nd October 2015
ICERM

(joint work with Drew Sutherland and Jan Tuitman)

Introduction

$g \geq 1$

$f \in \mathbf{Z}[x]$, degree $2g+1$ or $2g+2$, squarefree over $\mathbf{Q}[x]$

$p =$ odd prime

$X_p : y^2 = f(x)$ over $\mathbf{F}_p$, hyperelliptic curve of genus g for almost all p

Zeta function at a good prime p:

$$\exp\left(\sum_{r\geq 1} \frac{|X_p(\mathbf{F}_{p^r})|}{r} T^r\right) = \frac{L_p(T)}{(1-T)(1-pT)}$$

$L_p \in \mathbf{Z}[T]$, degree $2g$

GOAL: given f and a bound N, compute $L_p(T)$ for all $p < N$.

Applications: Sato–Tate conjecture, etc.

Previous algorithms:

- Schoof (1985), Pila (1990), Adleman–Huang (2001):

$$(\log p)^{O(g^2 \log g)} \quad \text{for single } p.$$

  Polynomial in $\log p$, exponential in g.

- Kedlaya (2001), H. (2007):

$$g^{O(1)} p^{1/2+\varepsilon} \quad \text{for single } p.$$

  Polynomial in g, exponential in $\log p$.

- H. (2014):

$$g^{8+\varepsilon} N (\log N)^{3+\varepsilon} \quad \text{for all } p < N.$$

  Polynomial in both g and $\log p$, on average over all p.

No fully polynomial time algorithm is known (i.e., for one prime at a time).

The g^8 dependence is pretty ugly — that algorithm will never be implemented.

New algorithm (H.–Sutherland–Tuitman, in preparation):

$$g^{4+\varepsilon} N (\log N)^{3+\varepsilon} \quad \text{for all } p < N,$$

i.e., we reduce g^8 to g^4.

An efficient implementation is on the way (by the end of the semester).

H.–Sutherland (2014): implemented baby version for $g = 2, 3$, just for the Hasse–Witt matrix, i.e., only recovers $L_p(T)$ modulo p.

Timings for computing $L_p(T) \pmod{p}$ for all $p < 2^{30}$:

- $g = 2$. Old: 1.4 years (`smalljac`). New: 1.3 days.
- $g = 3$. Old: 3.8 years (`hypellfrob`). New: 4 days.

With the new algorithm we expect to see similar improvements for say $g = 4, 5, 6$.

Plan for this talk:

- I will describe a simplified version of the algorithm that yields only the Hasse–Witt matrix (not quite the same as the H.–Sutherland papers)
- The accumulating remainder tree... in one slide!
- I will sketch how to extend the algorithm to obtain the full zeta function.

But first...

Happy 6th birthday Zachary!

Born 2nd October 2009 in New York City (note correct time zone).

The Hasse–Witt matrix

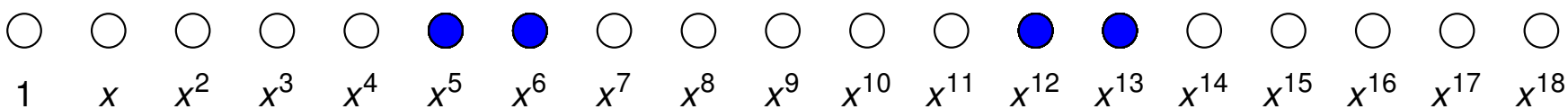
The Hasse–Witt matrix is a $g \times g$ matrix over $\mathbf{F}_p$:

$$W_p = (f^{(p-1)/2})_{pi-j} \bmod p \qquad 1 \le i, j \le g$$

The connection with $L_p(T)$ is

$$L_p(T) = \det(I - TW_p) \pmod{p}.$$

Example for deg $f = 6$, $g = 2$, $p = 7$:



How to compute the coefficients of $f^{(p-1)/2}$?

Using the identities $f^{n+1} = f \cdot f^n$ and $\left(f^{n+1}\right)' = (n+1) f' \cdot f^n$, we can deduce a **recurrence** of order $d = \deg f$ for the coefficients of f^n.

Example for $d = 6$:

$$\begin{bmatrix} (f^{(p-1)/2})_{k-5} \\ (f^{(p-1)/2})_{k-4} \\ (f^{(p-1)/2})_{k-3} \\ (f^{(p-1)/2})_{k-2} \\ (f^{(p-1)/2})_{k-1} \\ (f^{(p-1)/2})_{k} \end{bmatrix} = \frac{1}{2kf_0} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & (6p+6-2k)f_6 \\ 2kf_0 & 0 & 0 & 0 & 0 & (5p+5-2k)f_5 \\ 0 & 2kf_0 & 0 & 0 & 0 & (4p+4-2k)f_4 \\ 0 & 0 & 2kf_0 & 0 & 0 & (3p+3-2k)f_3 \\ 0 & 0 & 0 & 2kf_0 & 0 & (2p+2-2k)f_2 \\ 0 & 0 & 0 & 0 & 2kf_0 & (p+1-2k)f_1 \end{bmatrix}^T \begin{bmatrix} (f^{(p-1)/2})_{k-6} \\ (f^{(p-1)/2})_{k-5} \\ (f^{(p-1)/2})_{k-4} \\ (f^{(p-1)/2})_{k-3} \\ (f^{(p-1)/2})_{k-2} \\ (f^{(p-1)/2})_{k-1} \end{bmatrix}.$$

Let us write this as

$$V_k^p = \frac{1}{2kf_0} C_k^p V_{k-1}^p.$$

Remark: V_k^p and C_k^p have entries in $\mathbf{Z}$, not $\mathbf{F}_p$.

The “initial condition” for the recurrence is easy to compute:

$$V_0^p = [0, 0, \ldots, 0, (f^{(p-1)/2})_0].$$

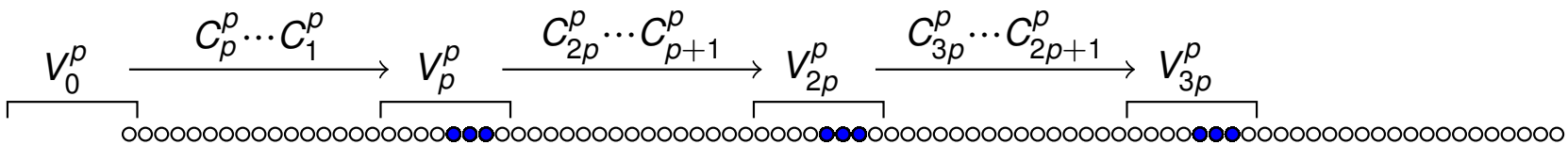
To get the first row of the Hasse–Witt matrix, we take p steps:

$$V_p^p = \frac{1}{2^p(1 \cdot 2 \cdots p)} C_p^p C_{p-1}^p \cdots C_2^p C_1^p V_0^p.$$

Then to get the second row, we take another p steps:

$$V_{2p}^p = \frac{1}{2^p((p+1)(p+2)\cdots(2p))} C_{2p}^p C_{2p-1}^p \cdots C_{p+2}^p C_{p+1}^p V_p^p.$$

And so on. Example with $g = 3$:



Despite the annoying multiples of p in the denominator, one can prove:

To obtain the Hasse–Witt matrix, it is enough to compute the products

$$C^p_{tp+p} \cdots C^p_{tp+2} C^p_{tp+1} \pmod{p^2}$$

for $t = 0, 1, \ldots, g-1$.

(For this we must assume that p is not too small compared to g.)

Here is a trick to remove the dependence on p and t.

(I did not yet know this trick when I wrote my first "average polynomial time" paper... it makes everything *so much easier!!!*)

Consider the j-th term:

$$C^p_{tp+j} = \begin{bmatrix} 0 & 0 & 0 & (4p+4-2(tp+j))f_4 \\ 2(tp+j)f_0 & 0 & 0 & (3p+3-2(tp+j))f_3 \\ 0 & 2(tp+j)f_0 & 0 & (2p+2-2(tp+j))f_2 \\ 0 & 0 & 2(tp+j)f_0 & (p+1-2(tp+j))f_1 \end{bmatrix}$$

Let us replace p and tp by **formal variables** P and T:

$$\widetilde{C}_j = \begin{bmatrix} 0 & 0 & 0 & (4P+4-2(T+j))f_4 \\ 2(T+j)f_0 & 0 & 0 & (3P+3-2(T+j))f_3 \\ 0 & 2(T+j)f_0 & 0 & (2P+2-2(T+j))f_2 \\ 0 & 0 & 2(T+j)f_0 & (P+1-2(T+j))f_1 \end{bmatrix}$$

We regard this as a matrix with entries in $\mathbf{Z}[P, T]/(P^2, PT, T^2)$.

Notice that $\widetilde{C}_j$ **does not depend on** p **or** t**!**

Now for each p we want to compute

$$\widetilde{C}_p \cdots \widetilde{C}_2 \widetilde{C}_1 \pmod{p^2}$$

We do this using the "accumulating remainder tree" (next slide).

This product should be thought of as a matrix with entries in

$$\mathbf{Z}[P, T]/(P^2, PT, T^2, p^2).$$

At the end we substitute $P = p$, $T = tp$ for each p, t to get the desired products

$$C^p_{tp+p} \cdots C^p_{tp+2} C^p_{tp+1} \pmod{p^2}.$$

The accumulating remainder tree, in one slide

Suppose we want to compute:

$$\begin{aligned}
M_1 &\pmod{Q_1},\\
M_2M_1 &\pmod{Q_2},\\
M_3M_2M_1 &\pmod{Q_3},\\
M_4M_3M_2M_1 &\pmod{Q_4},\\
M_5M_4M_3M_2M_1 &\pmod{Q_5},\\
&\cdots\\
M_nM_{n-1}\cdots M_2M_1 &\pmod{Q_n}.
\end{aligned}$$

Algorithm: (1) multiply adjacent M_i's and Q_i's, (2) recursively compute

$$\begin{aligned}
(M_2M_1) &\pmod{Q_2Q_3},\\
(M_4M_3)(M_2M_1) &\pmod{Q_4Q_5},\\
&\cdots\\
(M_{n-1}M_{n-2})\cdots(M_4M_3)(M_2M_1) &\pmod{Q_{n-1}Q_n},
\end{aligned}$$

and (3) make the obvious corrections.

Computing the whole zeta function

So far we have only computed $L_p(T) \pmod{p}$.

How do we get all of $L_p(T) \in \mathbf{Z}[T]$?

It suffices to compute $L_p(T) \pmod{p^\lambda}$ where $\lambda = O(g)$.

One option: use p-adic cohomology (as in Kedlaya's algorithm).

We will use instead a “trace formula” adapted from my paper *Computing zeta functions of arithmetic schemes* (in press):

$$|X_p(\mathbf{F}_{p^r})| = (p^r - 1)\left(1 + \sum_{\substack{s=1 \\ s \text{ odd}}}^{2\lambda-1} \alpha_s \operatorname{tr} A_s^r\right) \pmod{p^\lambda}.$$

where

$$\alpha_s = \sum_{i=0}^{\lambda-1}\sum_{j=0}^{\lambda}(-1)^{j+1}2^{1-i-j}\binom{\lambda}{j}\binom{\lambda+i-1}{i}\binom{i+j}{s}$$

and where A_s is the matrix

$$(A_s)_{i,j} = (f^{s(p-1)/2})_{pi-j}, \qquad 0 \le i, j \le \lfloor sd/2 \rfloor.$$

The complexity is dominated by computing the matrices $A_1, A_3, \ldots, A_{2\lambda-1}$.

For $\lambda = 1$ the trace formula reduces to computing the Hasse–Witt matrix (more or less), i.e., it involves the same coefficients of $f^{(p-1)/2}$.

For $\lambda = 2$ we also need to compute coefficients of $f^{3(p-1)/2}$.

For $\lambda = 3$ we also need $f^{5(p-1)/2}$. And so on.

In general for $f^{s(p-1)/2}$ we need to work with a recurrence matrix that looks something like

$$C_k^{p,s} \stackrel{?}{=} \begin{bmatrix} 0 & 0 & 0 & (4sp-4(s-2)-2k)f_4 \\ 2kf_0 & 0 & 0 & (3sp-3(s-2)-2k)f_3 \\ 0 & 2kf_0 & 0 & (2sp-2(s-2)-2k)f_2 \\ 0 & 0 & 2kf_0 & (sp-(s-2)-2k)f_1 \end{bmatrix}$$

PROBLEM: this is not sufficiently p-adically continuous with respect to s, so we cannot use the "formal variable" trick to make the formula independent of s.

The reason is that the powers $s(p-1)/2$ are spaced out by multiples of $p-1$ instead of p.

It is not hard to fudge things to make it work.

Instead of computing $f^{s(p-1)/2}$ directly, we work with $f^{(sp-2\lambda+1)/2}$.

For example, if $\lambda = 3$, we compute coefficients of:

- $f^{(p-5)/2}$. This is not what we need for the trace formula, but it is easy to get coefficients of $f^{(p-1)/2}$ from those of $f^{(p-5)/2}$.
- $f^{(3p-5)/2}$. Again it is easy to adjust to get coefficients of $f^{3(p-1)/2}$.
- $f^{(5p-5)/2}$, which is already what we want.

We end up with the following recurrence matrices:

$$C_{tp+j}^{p,s} = \begin{bmatrix} 0 & 0 & 0 & (4sp-4(2\lambda-3)-2(tp+j))f_4 \\ 2(tp+j)f_0 & 0 & 0 & (3sp-3(2\lambda-3)-2(tp+j))f_3 \\ 0 & 2(tp+j)f_0 & 0 & (2sp-2(2\lambda-3)-2(tp+j))f_2 \\ 0 & 0 & 2(tp+j)f_0 & (sp-(2\lambda-3)-2(tp+j))f_1 \end{bmatrix}$$

Now we can apply the formal variable trick:

$$\widetilde{C}_j = \begin{bmatrix} 0 & 0 & 0 & (4S-4(2\lambda-3)-2(T+j))f_4 \\ 2(T+j)f_0 & 0 & 0 & (3S-3(2\lambda-3)-2(T+j))f_3 \\ 0 & 2(T+j)f_0 & 0 & (2S-2(2\lambda-3)-2(T+j))f_2 \\ 0 & 0 & 2(T+j)f_0 & (S-(2\lambda-3)-2(T+j))f_1 \end{bmatrix}$$

The entries are in $\mathbf{Z}[S,T]/(S,T)^{\lambda+1}$.

Finally one can show that it suffices to compute the products

$$\widetilde{C}_p \cdots \widetilde{C}_2 \widetilde{C}_1 \pmod{p^{\lambda+1}}$$

simultaneously for all $p < N$.

Thank you!