Computing Hasse-Witt matrices of hyperelliptic curves in average polynomial time

David Harvey and Andrew Sutherland

ANTS XI — August 9, 2014
Motivation

Let $C/\mathbb{Q}$ be a smooth projective curve of genus $g$.

For each prime $p$ of good reduction we have the trace of Frobenius

$$t_p = p + 1 - N_p \in [-2g\sqrt{p}, 2g\sqrt{p}],$$

where $N_p = \# C(\mathbb{F}_p)$, and the normalized trace

$$x_p = t_p / \sqrt{p} \in [-2g, 2g].$$

What is the distribution of $x_p$?
Exceptional trace distributions of genus 2 curves $C/\mathbb{Q}$

Harvey (UNSW) and Sutherland (MIT)
$L$-polynomial distributions

For a smooth projective curve $C/\mathbb{Q}$ of genus $g$ and a prime $p$ of good reduction for $C$ we have the zeta function

$$Z_p(T) := \exp \left( \sum_{k=1}^{\infty} \frac{N_k T^k}{k} \right) = \frac{L_p(T)}{(1 - T)(1 - pT)},$$

where $L_p \in \mathbb{Z}[T]$ has degree $2g$. The normalized $L$-polynomial

$$\bar{L}_p(T) := L_p(T/\sqrt{p}) = \sum_{i=0}^{2g} a_i T^i \in \mathbb{R}[T]$$

is monic, reciprocal ($a_i = a_{2g-i}$), and unitary (roots on the unit circle). The coefficients $a_i$ satisfy the Weil bounds $|a_i| \leq \binom{2g}{i}$.

We may now consider the distribution of $a_1, a_2, \ldots, a_g$ as $p$ varies.
Harvey (UNSW) and Sutherland (MIT)

Computing Hasse–Witt matrices
## Computing zeta functions

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves

<table>
<thead>
<tr>
<th>algorithm</th>
<th>$g = 1$</th>
<th>$g = 2$</th>
<th>$g = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>point enumeration</td>
<td>$p \log p$</td>
<td>$p^2 \log p$</td>
<td>$p^3 \log p$</td>
</tr>
<tr>
<td>group computation</td>
<td>$p^{1/4} \log p$</td>
<td>$p^{3/4} \log p$</td>
<td>$p^{5/4} \log p$</td>
</tr>
<tr>
<td>$p$-adic cohomology</td>
<td>$p^{1/2} \log^2 p$</td>
<td>$p^{1/2} \log^2 p$</td>
<td>$p^{1/2} \log^2 p$</td>
</tr>
<tr>
<td>CRT (Schoof-Pila)</td>
<td>$\log^5 p$</td>
<td>$\log^8 p$</td>
<td>$\log^{12} p$</td>
</tr>
</tbody>
</table>

(complexity (ignoring factors of $O(\log \log p)$))
Computing zeta functions

Algorithms to compute $L_p(T)$ for low genus hyperelliptic curves

<table>
<thead>
<tr>
<th>algorithm</th>
<th>complexity $g = 1$</th>
<th>complexity $g = 2$</th>
<th>complexity $g = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>point enumeration</td>
<td>$p \log p$</td>
<td>$p^2 \log p$</td>
<td>$p^3 \log p$</td>
</tr>
<tr>
<td>group computation</td>
<td>$p^{1/4} \log p$</td>
<td>$p^{3/4} \log p$</td>
<td>$p^{5/4} \log p$</td>
</tr>
<tr>
<td>$p$-adic cohomology</td>
<td>$p^{1/2} \log^2 p$</td>
<td>$p^{1/2} \log^2 p$</td>
<td>$p^{1/2} \log^2 p$</td>
</tr>
<tr>
<td>CRT (Schoof-Pila)</td>
<td>$\log^5 p$</td>
<td>$\log^8 p$</td>
<td>$\log^{12} p$</td>
</tr>
</tbody>
</table>

(see [Kedlaya-S, ANTS VIII]).
An average polynomial-time algorithm

All of these methods perform separate computations for each $p$. But we want to compute $L_p(T)$ for all good $p \leq N$ using reductions of the same curve in each case. Can we take advantage of this?

Theorem (H 2012)

There exists a deterministic algorithm that, given a hyperelliptic curve $y^2 = f(x)$ of genus $g$ with a rational Weierstrass point and an integer $N$, computes $L_p(T)$ for all good primes $p \leq N$ in time $O(g^8 + \epsilon N \log^3 + \epsilon N^2)$, assuming the coefficients of $f \in \mathbb{Z}[x]$ have size bounded by $O(\log N)$.

Average time is $O(g^8 + \epsilon \log^4 + \epsilon N)$ per prime, polynomial in $g$ and $\log p$. 
An average polynomial-time algorithm

All of these methods perform separate computations for each $p$. But we want to compute $L_p(T)$ for all good $p \leq N$ using reductions of the same curve in each case. Can we take advantage of this?

**Theorem (H 2012)**

There exists a deterministic algorithm that, given a hyperelliptic curve $y^2 = f(x)$ of genus $g$ with a rational Weierstrass point and an integer $N$, computes $L_p(T)$ for all good primes $p \leq N$ in time

$$O\left(g^{8+\epsilon} N \log^{3+\epsilon} N \right),$$

assuming the coefficients of $f \in \mathbb{Z}[x]$ have size bounded by $O(\log N)$.

Average time is $O\left(g^{8+\epsilon} \log^{4+\epsilon} N \right)$ per prime, polynomial in $g$ and $\log p$. 
An average polynomial-time algorithm

<table>
<thead>
<tr>
<th>algorithm</th>
<th>complexity (ignoring factors of $O(\log \log p)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$g = 1$</td>
</tr>
<tr>
<td></td>
<td>$g = 2$</td>
</tr>
<tr>
<td></td>
<td>$g = 3$</td>
</tr>
<tr>
<td>point enumeration</td>
<td>$p \log p$</td>
</tr>
<tr>
<td>group computation</td>
<td>$p^{1/4} \log p$</td>
</tr>
<tr>
<td>$p$-adic cohomology</td>
<td>$p^{1/2} \log^2 p$</td>
</tr>
<tr>
<td>CRT (Schoof-Pila)</td>
<td>$\log^5 p$</td>
</tr>
<tr>
<td>Average polytime</td>
<td>$\log^4 p$</td>
</tr>
</tbody>
</table>

But is it practical?
<table>
<thead>
<tr>
<th>$N$</th>
<th>genus 2</th>
<th></th>
<th>genus 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>smalljac</td>
<td>paper</td>
<td>current</td>
<td>hypellfrob</td>
</tr>
<tr>
<td>$2^{14}$</td>
<td>0.2</td>
<td>0.4</td>
<td>0.1</td>
<td>6.8</td>
</tr>
<tr>
<td>$2^{15}$</td>
<td>0.6</td>
<td>1.1</td>
<td>0.3</td>
<td>15.6</td>
</tr>
<tr>
<td>$2^{16}$</td>
<td>1.7</td>
<td>2.8</td>
<td>0.8</td>
<td>37.6</td>
</tr>
<tr>
<td>$2^{17}$</td>
<td>5.6</td>
<td>6.8</td>
<td>1.8</td>
<td>95.0</td>
</tr>
<tr>
<td>$2^{18}$</td>
<td>20.2</td>
<td>16.8</td>
<td>4.7</td>
<td>250</td>
</tr>
<tr>
<td>$2^{19}$</td>
<td>76.4</td>
<td>39.7</td>
<td>11.1</td>
<td>681</td>
</tr>
<tr>
<td>$2^{20}$</td>
<td>257</td>
<td>94.4</td>
<td>26.0</td>
<td>1920</td>
</tr>
<tr>
<td>$2^{21}$</td>
<td>828</td>
<td>227</td>
<td>61.4</td>
<td>5460</td>
</tr>
<tr>
<td>$2^{22}$</td>
<td>2630</td>
<td>534</td>
<td>142</td>
<td>16300</td>
</tr>
<tr>
<td>$2^{23}$</td>
<td>8570</td>
<td>1240</td>
<td>321</td>
<td>49400</td>
</tr>
<tr>
<td>$2^{24}$</td>
<td>28000</td>
<td>2920</td>
<td>729</td>
<td>152000</td>
</tr>
<tr>
<td>$2^{25}$</td>
<td>92300</td>
<td>6740</td>
<td>1660</td>
<td>467000</td>
</tr>
<tr>
<td>$2^{26}$</td>
<td>316000</td>
<td>15800</td>
<td>3800</td>
<td>1490000</td>
</tr>
</tbody>
</table>

Comparison of average polynomial time algorithm (as in the paper and currently) to \textbf{smalljac} in genus 2 and \textbf{hypellfrob} in genus 3.

(\textit{Intel Xeon E5-2670 2.6 GHz CPU seconds}).
The algorithm in genus 1

The Hasse invariant $h_p$ of an elliptic curve $y^2 = f(x) = x^3 + ax + b$ over $\mathbb{F}_p$ is the coefficient of $x^{p-1}$ in the polynomial $f(x)^{(p-1)/2}$.

We have $h_p \equiv t_p \mod p$, which uniquely determines $t_p$ for $p > 13$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$ and reduce the $x^{p-1}$ coefficient of $f(x)^{(p-1)/2}$ mod $p$ for each prime $p \leq N$. 
The algorithm in genus 1

The Hasse invariant $h_p$ of an elliptic curve $y^2 = f(x) = x^3 + ax + b$ over $\mathbb{F}_p$ is the coefficient of $x^{p-1}$ in the polynomial $f(x)^{(p-1)/2}$.

We have $h_p \equiv t_p \mod p$, which uniquely determines $t_p$ for $p > 13$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$ and reduce the $x^{p-1}$ coefficient of $f(x)^{(p-1)/2}$ mod $p$ for each prime $p \leq N$.

But the polynomials $f^n$ are huge, each has $\Omega(n^2)$ bits.
It would take $\Omega(N^3)$ time to compute $f, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$.

So this is a terrible idea...
The algorithm in genus 1

The Hasse invariant $h_p$ of an elliptic curve $y^2 = f(x) = x^3 + ax + b$ over $\mathbb{F}_p$ is the coefficient of $x^{p-1}$ in the polynomial $f(x)^{(p-1)/2}$.

We have $h_p \equiv t_p \mod p$, which uniquely determines $t_p$ for $p > 13$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$ and reduce the $x^{p-1}$ coefficient of $f(x)^{(p-1)/2} \mod p$ for each prime $p \leq N$.

But the polynomials $f^n$ are huge, each has $\Omega(n^2)$ bits. It would take $\Omega(N^3)$ time to compute $f, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$.

So this is a terrible idea...

But we don’t need all the coefficients of $f^n$, we only need one, and we only need to know its value modulo $p = 2n + 1$. 
A better approach

Let \( f(x) = x^3 + ax + b \), and let \( f^n_k \) denote the coefficient of \( x^k \) in \( f(x)^n \). Using \( f^n = f \cdot f^{n-1} \) and \( (f^n)' = nf'f^{n-1} \), one obtains linear relations

\[
(n + 2)f^n_{2n-2} = n \left( 2af^n_{2n-3} + 3bf^n_{2n-2} \right),
\]
\[
(2n - 1)f^n_{2n-1} = n \left( 3f^n_{2n-4} + af^n_{2n-2} \right),
\]
\[
2(2n - 1)bf^n_{2n} = (n + 1)af^n_{2n-4} + 3(2n - 1)bf^n_{2n-3} - (n - 1)a^2f^n_{2n-2}.
\]

These allow us to compute the vector \( v_n = [f^n_{2n-2}, f^n_{2n-1}, f^n_{2n}] \) from the vector \( v_{n-1} = [f^{n-1}_{2n-4}, f^{n-1}_{2n-3}, f^{n-1}_{2n-2}] \) via multiplication by a \( 3 \times 3 \) matrix \( M_n \):

\[
v_n = v_0M_1M_2 \cdots M_n.
\]

For \( n = (p - 1)/2 \), the Hasse invariant of the elliptic curve \( y^2 = f(x) \) over \( \mathbb{F}_p \) is obtained by reducing the third entry \( f^{2n}_n \) of \( v_n \) modulo \( p \).
Computing $t_p \mod p$

To compute $t_p \mod p$ for all odd primes $p \leq N$ it suffices to compute

\[
M_1 \mod 3
\]
\[
M_1M_2 \mod 5
\]
\[
M_1M_2M_3 \mod 7
\]
\[\vdots\]
\[
M_1M_2M_3 \cdots M_{(N-1)/2} \mod N
\]

Doing this na"ïvely would take $O(N^{2+\epsilon})$ time.

But it can be done in $O(N^{1+\epsilon})$ time using a remainder tree. For best results, use a remainder forest.
The algorithm in genus $g$.

The *Hasse-Witt* matrix of a hyperelliptic curve $y^2 = f(x)$ over $\mathbb{F}_p$ of genus $g$ is the $g \times g$ matrix $W_p = [w_{ij}]$ with entries

$$w_{ij} = f_{p^{i-j}}^{(p-1)/2} \mod p \quad (1 \leq i, j \leq g).$$

The $w_{ij}$ can each be computed using recurrence relations between the coefficients of $f^n$ and those of $f^{n-1}$, as in genus 1.
The algorithm in genus $g$.

The \emph{Hasse-Witt} matrix of a hyperelliptic curve $y^2 = f(x)$ over $\mathbb{F}_p$ of genus $g$ is the $g \times g$ matrix $W_p = [w_{ij}]$ with entries

$$w_{ij} = f_{p^{i-j}}^{(p-1)/2} \mod p \quad (1 \leq i, j \leq g).$$

The $w_{ij}$ can each be computed using recurrence relations between the coefficients of $f^n$ and those of $f^{n-1}$, as in genus 1.

The congruence

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

allows us to determine the coefficients $a_1, \ldots, a_g$ of $L_p(T)$ modulo $p$.

The algorithm can be extended to compute $L_p(T)$ modulo higher powers of $p$ (and thereby obtain $L_p \in \mathbb{Z}[T]$), but for $g \leq 3$ it is faster in practice to derive $L_p(T)$ from $L_p(T) \mod p$ using computations in $\text{Jac}(C)$. 
Theorem (HS 2014)

Given a hyperelliptic curve $y^2 = f(x)$ of genus $g$, and an integer $N$, one can compute the Hasse-Witt matrices $W_p$ for all good primes $p \leq N$ in

$$O\left(g^{2+\epsilon} N \log^{3+\epsilon} N\right) \text{ time and } O(g^2 N) \text{ space},$$

provided that $g$ and $\log \|f\|$ are sufficiently small relative to $N$.

The time bound has improved by a factor of $g^{3-\epsilon}$ since the paper. The complexity is quasi-linear in the output size.

This should extend to computing $L_p \in \mathbb{Z}[T]$ in $O(g^{4+\epsilon} N \log^{3+\epsilon} N) \text{ time}$.

In progress: generalize to non-hyperelliptic curves.