Talk notes :
Theta functions and isogenies between abelian surfaces

Jean Kieffer (Harvard)
CARAMBA seminar, Nancy, France, 21 nov. 2022

1 Motivation : isogeny graphs

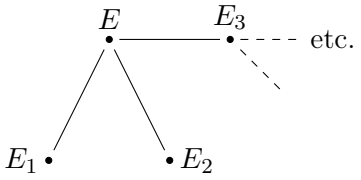
Base field : $k = \mathbb{F}_q$ or number field ($\mathbb{Q}$ or finite extension).
E, E' elliptic curves $/k$.
Def : $f : E \to E'$ is an *ℓ-isogeny* (ℓ prime) if

- f is a morphism of curves, given by algebraic formulas,
- f respects the group structure : $f(P+Q) = f(P) + f(Q)$,
- $\deg(f) = \ell$: either degree of polynomials giving f, or degree as a map, i.e. $\ell = \#\ker(f)$.

Then $\ker(f) \subset E[\ell]$ is a cyclic subgroup ; this isogeny–subgroup correspondence is a bijection.

———-

Construct *isogeny graph* from a vertex E by taking all E' linked to E by an ℓ-isogeny defined over k as neighboring vertices, and continuing until the (finite) connected component is exhausted :



———-

This definition makes sense in a more general setting : *abelian varieties*.
A is an abelian variety of dim. g means : A is a "nice" variety of dimension g (smooth, projective), with group law $+ : A \to A$.
There is a notion of ℓ-isogeny $f : A \to A'$, corresponding to subgroups $\ker(f) \subset A[\ell]$.
We can construct similar isogeny graphs.
"First case" after elliptic curves is that of Jacobians of genus 2 curves over k, a.k.a. principally polarized abelian surfaces.

(Not said during the talk : there are oversimplifications here. The notion of ℓ-isogeny is usually formulated in the setting of abelian varieties endowed with principal polarizations (or at least a polarization of degree prime to ℓ), in which case $\ker(f)$ is assumed to be maximal isotropic in $A[\ell]$ for the Weil pairing, and isomorphic to $(\mathbb{Z}/\ell\mathbb{Z})^2$. These are the ℓ-isogeny graphs we consider in the talk. We ignore issues of twists, issues about genus 2 curves only defined over a quadratic extension of k, and issues about field of definitions of isogenies in the presence of extra automorphisms.)

———-

Why study these graphs ?
Case $k = \mathbb{F}_q$: they appear in cryptography, in particular isogeny graphs of supersingular elliptic curves.

— Expansion properties : small diameter, random walks mix rapidly.

— Small-degree isogenies ($\ell = 2, 3$) are cheap to compute.

— Hard problem is to find a path between two points. No attack is currently known, either in classical or quantum setting, that is better than attacks against generic graphs.

— However : slower than other cryptographic families.

In dimensions $g \geq 2$, probably no constructive cryptography. Isogeny graphs could still be used for cryptanalysis : cf. recent destruction of SIDH.
Case $k = \mathbb{Q}$: of interest to number theorists, cf. analogues of Mazur's isogeny theorem. As we will see, algorithmic results over $\mathbb{Q}$ can be useful over $\mathbb{F}_p$ by reduction.

2 Computing neighbors

Two basic problems :

1. (Neighbors) Given A, compute all neighbors in ℓ-isogeny graph.
2. (Quotient) Given A and $K \subset A[\ell]$, compute quotient A/K.

Writing down explicit formulas giving the isogeny is usually no harder than solving 1. or 2. Many algorithms are known for 2. for abelian surfaces. In this talk, concentrate on 1.
Goals of this section :

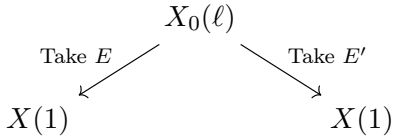
— Explain usual method for elliptic curves using modular polynomials.

— Explain why the method cannot work as is in dimensions $g \geq 2$.

— Explain how a reasonable algorithm reduces to the evaluation of *theta functions*.

———-

Modular polynomials. Isomorphism class of E/k is encoded by j-invariant $j(E) \in k$: "the moduli space $X(1)$ of elliptic curves is a line".
We have a geometric picture :



where $X_0(\ell)$ is a *modular curve* : points of $X_0(\ell)$ correspond to the data of E with an ℓ-isogeny $E \to E'$.
The *modular polynomial* $\Phi_\ell \in \mathbb{Z}[X, Y]$ is an equation for $X_0(\ell) \subset X(1) \times X(1)$, the same for every field k. Therefore :

$$\Phi_\ell(j(E), j(E')) = 0 \iff E, E' \text{ are neighbors in } \ell\text{-isogeny graph.}$$

Algorithm to compute neighbors :

1. Get precomputed Φ_ℓ ; reduce to k.
2. Evaluate to get $\Phi_\ell(j(E), Y) \in k[Y]$.
3. Compute roots.

(This works for ℓ moderately large (say ≤ 500). For larger ℓ, we can use a direct evaluation algorithm like the one I'm about to discuss.)

———-

In dimension 2, the geometric picture is the same except that A has three independent coordinates : j_1, j_2, j_3.
The analogue of Φ_ℓ is $\Psi_\ell \in \mathbb{Q}(X_1, X_2, X_3)[Y]$ (actually three of these fractions).
They are too large to be computed explicitly as soon as $\ell \geq 5$: their size is $O(\ell^{15} \log \ell)$. We pay the price of a larger number of variables. To compare, Φ_ℓ has size $O(\ell^3 \log \ell)$.
What could we do ?

1. Purely algebraic method : write down equation for $A[\ell]$, compute factorization to find any rational subgroups, apply quotient algorithm. This is also very expensive.
2. Better : directly compute the evaluation
$$\Psi_\ell(j_1(A), j_2(A), j_3(A), Y) \in k[Y]$$
This univariate polynomial has reasonable size : $O(\ell^6(h+\log \ell))$ where h is the height/number of digits of $j_i(A)$ (say $\in \mathbb{Q}$).

———-

Algorithm for direct evaluation uses an *analytic method.* Disclaimer : the math here to explain how we reduce to computing theta functions is more involved. Assume A is over $\mathbb{F}_p$.

1. Lift $j_i(A)$ to $x_1, x_2, x_3 \in \mathbb{Q}$, viewed as subfield of $\mathbb{C}$.
2. Compute genus 2 curve $C/\mathbb{C}$ whose Jacobian has coordinates x_i.
3. Compute *period matrix* of $C : \tau \in \mathbb{H}_2$ (i.e. complex, symmetric, $\mathrm{Im}(\tau)$ positive definite) such that
$$\mathrm{Jac}(C) \simeq \mathbb{C}^2/(\mathbb{Z}^2 \oplus \tau\mathbb{Z}^2).$$
4. Enumerate period matrices $\tau' \in \mathbb{H}_2$ such that the complex tori $\mathbb{C}^2/(\mathbb{Z}^2 \oplus \tau'\mathbb{Z}^2)$ are all the complex abelian surfaces ℓ-isogenous to $\mathrm{Jac}(C)$. (Obtained by the action of certain symplectic matrices on τ.) We have :
$$\Psi_\ell(x_1, x_2, x_3, Y) = \prod_{\tau'} (Y - j_1(\tau')).$$
5. Recognize that the result actually lives in $\mathbb{Q}[Y]$, and reduce mod p. (In fact, we can separate numerator and denominator to only recognize integral coefficients : this will make certification possible later on.)

Critical step is 4. Write down the modular function j_1 in terms of *theta functions* : for any $g \geq 2$, any $a, b \in \{0,1\}^g$,
$$\theta_{a,b}(\tau) = \sum_{n \in \mathbb{Z}^g + \frac{a}{2}} \exp(i\pi(n^t \tau n + b^t n)).$$
(rapidly decreasing exponential terms).
We reduced our initial number theory problem to the question of evaluating theta functions which is pure numerical analysis.

———-

Dupont, Labrande–Thomé : heuristically, $\theta_{a,b}(\tau)$ can be computed up to 2^{-N} in *quasilinear* time $O(\mathcal{M}(N) \log N)$, where $\mathcal{M}(N)$ is the time to multiply N-bit integers. This works in pratice, and is critical to reach the required precisions. Also critical to obtain a reasonable complexity estimate.
In the rest of the talk, we explain how one can get a *certified implementation* of this algorithm.

3 Computing theta functions

The "naive" algorithm (compute partial sums of exponential series) is not quasi-linear time : $O(\mathcal{M}(N)N^{g/2}\log N)$. When optimized, it is still better for practical applications when $g = 1$; not so in higher dimensions.
Instead, the quasilinear algorithm is based on *arithmetic-geometric means* (AGM). 2 main steps :

1. From the input of θ, certain AGMs can recover the period matrix τ.
2. Use *Newton iterations* around 1. to go in the other direction $\tau \rightsquigarrow \theta$.

To discuss certified implementations, we enter into details for each step.

———-

First discuss Step 1 (I shortened this part sharply in the actual talk).
Start with projective theta values $\theta_{0,b}(\tau/2)/\theta_{0,0}(\tau/2)$, indexed by $b \in \{0,1\}^g$: i.e. $2^g - 1$ coordinates. By duplication, get $\theta^2_{a,b}(\tau)/\theta^2_{0,0}(\tau)$ for all a, b. For well-chosen symplectic matrices $N \in \mathrm{Sp}_4(\mathbb{Z})$, we compute $\theta^2_{0,0}(\tau)$ as follows :

— Get $\lambda\theta^2_{0,b}(N\tau)$ using transformation formula, for some common unknown $\lambda \in \mathbb{C}^\times$.

— Construct the AGM sequence whose n-th term is

$$(\lambda\theta^2_{0,b}(2^n N\tau))_{b\in\{0,1\}^g}.$$

At each step, use the duplication formula :

$$\theta^2_{0,b}(\tau) = \frac{1}{2^g}\sum_{b'\in(\mathbb{Z}/2\mathbb{Z})^g} \theta_{0,b'}(\tau)\theta_{0,b+b'}(\tau).$$

Thus we have to extract *square roots* at each step : I will come back to this.

— The limit is $(\lambda, \dots, \lambda)$. Thus we get λ, hence $\theta^2_{0,0}(N\tau)$ from the very first term of the sequence.

In particular, get $\theta^2_{0,b}(\tau)$ from $N = I_{2g}$. We recover information on τ (some polynomial in its coefficients) by applying the transformation formula again. A good choice is to take :

$$N = \begin{pmatrix} \mathrm{Diag}(\delta_i) & \mathrm{Diag}(1-\delta_i) \\ \mathrm{Diag}(\delta_i - 1) & \mathrm{Diag}(\delta_i) \end{pmatrix}, \quad \text{where } \delta_i \in \{0,1\} \text{ for } 1 \le i \le g.$$

———-

For the actual implementation of step 1 : can control convergence rate of AGM sequences. We also have to find the correct sign choices when taking square roots.

— Can be precomputed in the context of evaluating theta functions.

— Theorem (K. 2022) : sign choices can be predicted (they are always "good", i.e. contained in a common quarter plane) in the case $g = 2$.

— In general, sign choices can be precomputed by computing τ to a low precision by numerical integration ; however the dependency of the cost in terms of τ is hard to control.

In the end, this AGM method gives analytic functions F, G in the following diagram (where $N = 2^g - 1$, and G is polynomial in coefficients of τ) :

$$\begin{array}{ccc} & & \mathbb{H}_g \\ & & \Big\downarrow G \\ \mathbb{C}^N \supset \Omega & \xrightarrow{F} & \mathbb{C}^N \end{array}$$

$$\left(\frac{\theta_{0,b}(\tau/2)}{\theta_{0,0}(\tau/2)}\right) \longmapsto G(\tau).$$

with Ω open. (Or at least we get such a diagram around any point where the method actually works, e.g. none of the theta values we hit are equal to zero).

4 Newton iterations

Given an exact τ, we can hope to compute theta values at high precision as follows :

- Start with $x = (\theta_{0,b}(\tau/2)/\theta_{0,0}(\tau/2))$ up to 2^{-p} for a small p, computed using the naive algorithm ;
- Compute $F(x)$ as before ; it is not exactly $G(\tau)$;
- Compute (an approximation of) $dF(x)$ using finite differences ;
- Correct x using the usual Newton procedure, which is hopefully closer to the real theta values at τ.

Even if we *observe* convergence in practice, it is in no way a *proof* in the mathematical sense that the result is correct.

———-

To turn this into a *certified* algorithm (K. 2022), it is sufficient to collect :

- A lower bound for the size of a polydisk (ball) Ω where F is defined ;
- An upper bound on $|F|$ on Ω ; thus we also get upper bounds on $|dF|, |d^2F|$ using 1. and Cauchy's formula ;
- An upper bound on $|dF^{-1}(G(\tau))|$.

These explicit values can be determined a priori for small g, or on the fly during the computation. In the end, the provable error bounds closely match with experimental precision losses.
However, I can only show that $|dF^{-1}|$ is uniformly bounded when $g \leq 2$. In higher dimensions, we can still run certified Newton iterations at any point where they actually work.

Formal theorem : there exists an algorithm which, on the input of an exact $\tau \in \mathcal{F}_2$ (Siegel fundamental domain), computes $\theta^2_{a,b}(\tau)$ for all a, b up to an error of 2^{-N}, in quasi-linear time $O(\mathcal{M}(N) \log N)$, uniformly in τ ; we know all the explicit constants necessary to implement it in a computer.

———-

Comments on implementation. I wrote an `acb_theta` module https://github.com/j-kieffer/arb for the C library Arb https://arblib.org/ featuring :

— State-of-the-art versions of the naive algorithm, for low precisions ; (summing exponential terms over a tight ellipsoid, using only 2 multiplications per term asymptotically) ;

— Certified quasi-linear algorithms for higher precisions.

This is still work in progress, and should soon be merged into the Arb master branch. I plan on comparing performance with existing implementations in CMH ($g = 2$, Pari/GP ($g = 2$), Arb ($g = 1$), Magma (any g but unoptimized algorithm), Julia (any g but low precisions only),...

The more general case of theta *functions* (as opposed to theta *constants* as in this talk) is also covered.

———-

A previously untractable application, joint work in progress with Raymond van Bommel, Shiva Chidambaram, Edgar Costa (MIT) : the isogeny class over of principally polarized abelian surfaces with LMFDB label `349.a`, containing the Jacobian of

$$y^2 = x^6 + 2x^5 + 3x^4 - x^2 + 2x + 1$$

contains exactly two vertices linked by an isogeny of minimal degree $13^4 = 28,561$.
A research project is to employ these techiques for SEA-style point counting records for abelian surfaces.