Computing isogenies between Jacobians of hyperelliptic curves of arbitrary genus via differential equations

Elie Eid
Univ. Rennes 1

Effective Methods in Algebraic Geometry

June 7-11 2021

Why do we need to compute isogenies ?

Isogeny

An **isogeny** between two abelian varieties is a surjective morphism of abelian varieties of finite kernel.

Applications

- Improve point counting algorithms (*e.g.* SEA for elliptic curves).
- DLP transfer.
- Isogeny-based cryptography (isogeny graphes).
- Applications in number theory : construction of irreducible polynomials, normal bases extensions, *etc.*.

How to compute isogenies ?

Isogeny and differential equations (elliptic curves)

- Introduced by Elkies for elliptic curves (1998).
- Bostan-Morain-Salvy-Schost 08, elliptic curves over fields of large characteristic.
- Lercier-Sirvent 08 and Lairez-Vaccon 16, elliptic curves over finite fields of odd characteristic.
- Caruso-E.-Lercier 20, elliptic curves over fields of characteristic two.

Isogeny and differential equations (hyperelliptic curves)

- Couveignes-Ezome 14, Milio 19 and Kieffer-Page-Robert 20, jacobians of hyperelliptic curves of genus 2 and 3 over fields of large characteristic.
- E. 20, Jacobians of hyperellitpic curves of small genus over fields of odd characteristic. Complexity to compute a rational representation of an $(\ell,\ldots,\ell)$-isogeny : $\widetilde{O}(g^4\ell)$.

1. Isogenies between Jacobians of hyperelliptic curves
   - Hyperelliptic curves and their Jacobians
   - Rational representation of an isogeny
   - ODE associated with a rational representation

2. Solving the ODS
   - Newton iteration
   - Solving the ODS in small characteristic fields
   - Achieving quasi-optimality
   - Implementation

Plan

1. Isogenies between Jacobians of hyperelliptic curves
   - Hyperelliptic curves and their Jacobians
   - Rational representation of an isogeny
   - ODE associated with a rational representation
2. Solving the ODS
   - Newton iteration
   - Solving the ODS in small characteristic fields
   - Achieving quasi-optimality
   - Implementation

Let g be an integer ≥ 2. Let $H : y^2 = f(x)$ be a hyperelliptic curve of genus g defined over a field k ($\mathrm{char}(k) \neq 2$) and $J(H)$ be its Jacobian.

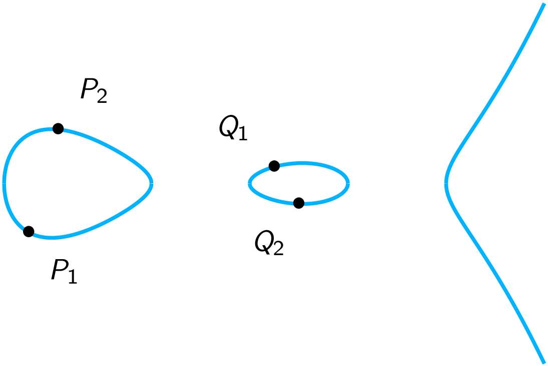
We assume that $\deg(f) = 2g+1$ and let P_∞ be the unique point at infinity of H.

We represent an element $[D]$ of $J(H)$ as a list of g points in H :

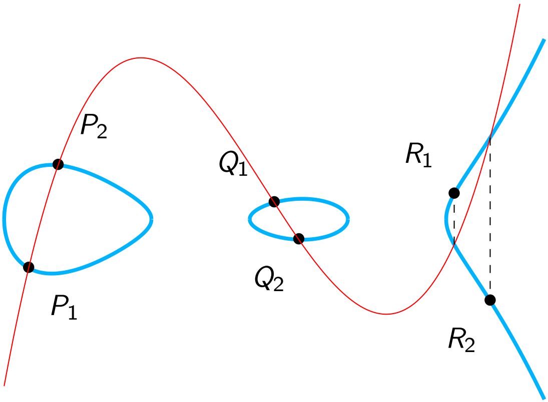
- There exists a unique $r \leq g$ such that $[D] = [(P_1 + \ldots + P_r) - r \cdot P_\infty]$.
- $[D]$ is uniquely represented by the list $\{P_1, \ldots, P_r, P_\infty, \ldots, P_\infty\}$.

Remark : If D is k-rational, then the P_i can be defined over an extension of k of degree $O(g)$.

Example : g=2



Example : g=2



$$\{P_1, P_2\} + \{Q_1, Q_2\} = \{R_1, R_2\}.$$

Group law in practice : the Mumford representation

A "generic element" $\{(x_1,y_1),\dots,(x_g,y_g)\}$ in the Jacobian is represented by a pair of polynomials $(U(X),V(X))$ such that

$$U(X)=X^g+\sigma_1X^{g-1}+\cdots+\sigma_g=\prod_{i=1}^{g}(X-x_i)$$

and $V(X)=\rho_1X^{g-1}+\cdots+\rho_g$ is the interpolating polynomial of the set $\{(x_1,y_1),\dots,(x_g,y_g)\}$.

We represent a generic element in $J(H)$ by the $2g$-tuple $(\sigma_1,\cdots,\sigma_g,\rho_1,\cdots,\rho_g)$.

Let $H_1 : v^2 = f_1(u)$ and $H_2 : y^2 = f_2(x)$ two hyperelliptic curves of genus g over a field k (char$(k) \neq 2$).

We assume that there exists an isogeny $I : J(H_1) \rightarrow J(H_2)$.
Let $j_{P_\infty} : H_1 \rightarrow J(H_1)$ be **the Jacobi morphism** with origin P_∞.
$I \circ j_{P_\infty}$ induces a morphism I_{P_∞} defined as follow

$$\begin{array}{rcl} I_{P_\infty} : H_1 & \longrightarrow & J(H_2) \\ Q = (u,v) & \mapsto & I([Q - P_\infty]) \\ & & = (\sigma_1(u,v), \cdots, \sigma_g(u,v), \rho_1(u,v), \cdots, \rho_g(u,v)) \end{array}$$

It follows that I_{P_∞} can be represented by the $2g$ rational functions $\sigma_1(u,v), \cdots, \sigma_g(u,v), \rho_1(u,v), \cdots, \rho_g(u,v)$ on C_1.

Rational representation

We say that $(\sigma_1, \cdots, \sigma_g, \rho_1, \cdots, \rho_g)$ is a **rational representation** of I.

The case of an $(\ell,\ldots,\ell)$-isogeny

We assume that I is an $(\ell,\ldots,\ell)$-isogeny. Let $(\sigma_1,\ldots,\sigma_g,\rho_1,\ldots,\rho_g)$.

Proposition (bounding the degrees)

The degrees of the functions $\sigma_1,\ldots,\sigma_g$ on C_1 are bounded by $2g\ell$.
The degrees of the functions $\rho_1,\ldots,\rho_g$ on C_1 are bounded by $3g\ell$.

Remark : One can write $\sigma_i = A_i(u)/B_i(u)$ and $\rho_i = v \cdot C_i(u)/D_i(u)$. A_i, B_i, C_i and D_i are polynomials of degrees bounded by $g\ell$, $g\ell$, $\frac{3}{2}g\ell$ and $\frac{3}{2}g\ell$ respectively.

Action on spaces of holomorphic differentials

The action of the morphism I_P on the spaces $H^0(H_2^{(g)}, \Omega^1_{H_2^{(g)}})$ and $H^0(H_1, \Omega^1_{H_1})$ gives a linear map :

$$I_P^* : H^0(H_2^{(g)}, \Omega^1_{H_2^{(g)}}) \longrightarrow H^0(H_1, \Omega^1_{H_1})$$

We chose the following two bases of $H^0(H_1, \Omega^1_{H_1})$ and $H^0(H_2^{(g)}, \Omega^1_{H_2^{(g)}})$ resp. :

$$B_1 = \left\{ u^i \frac{du}{v} \, ; i \in \{0, \dots, g-1\} \right\}$$

et

$$B_2 = \left\{ \sum_{j=1}^{g} x_j^i \frac{dx_j}{y_j} \, ; i \in \{0, \dots, g-1\} \right\}.$$

Let $(m_{ij})_{ij}$ be the matrix of the linear map I_P^* in (B_2, B_1), this gives the following ODS

$$\left\{\begin{array}{ccccccc}
\dfrac{dx_1}{y_1} & + & \cdots & + & \dfrac{dx_g}{y_g} & = & \left(m_{11} + \ldots + m_{1g} \cdot u^{g-1}\right)\dfrac{du}{v} \\
\dfrac{x_1 \cdot dx_1}{y_1} & + & \cdots & + & \dfrac{x_g \cdot dx_g}{y_g} & = & \left(m_{21} + \ldots + m_{2g} \cdot u^{g-1}\right)\dfrac{du}{v} \\
 & & \vdots & & & & \vdots \\
\dfrac{x_1^{g-1} \cdot dx_1}{y_1} & + & \cdots & + & \dfrac{x_g^{g-1} \cdot dx_g}{y_g} & = & \left(m_{g1} + \ldots + m_{gg} \cdot u^{g-1}\right)\dfrac{du}{v} \\
y_1^2 = f_2(x_1), & & \cdots & & , y_g^2 = f_2(x_g). & &
\end{array}\right.$$

Let Q be a point on H_1 and $I_P(Q) = \{P_1, \dots, P_g\}$. We assume that $I_P(Q)$ is generic.
Around the point Q, the ODS can be written of the form

$$\begin{cases} H(x_1(t), \dots, x_g(t)) \cdot X'(t) = G(t) \\ x_1(0), \dots, x_g(0) = x_{P_1}, \dots, x_{P_g}. \end{cases}$$

The matrix $H(x_1, \dots, x_g(t))$ is given by

$$H(x_1, \dots, x_g) = \begin{pmatrix} x_1'(t)/y_1(t) & \dots & \dots & x_g'(t)/y_g(t) \\ x_1(t)x_1'(t)/y_1(t) & \dots & \dots & x_g(t)x_g'(t)/y_g(t) \\ \vdots & & & \vdots \\ x_1(t)^{g-1}x_1'(t)/y_1(t) & \dots & \dots & x_g(t)^{g-1}x_g'(t)/y_g(t) \end{pmatrix}$$

Remark : Finding $X(t) \mod t^{2g\ell+1}$ allows to reconstruct the rational representation.

Plan

$$\begin{cases} H(X(t)) \cdot X'(t) = G(t) \\ x_1(0), \ldots, x_g(0) = x_{P_1}, \ldots, x_{P_g}. \end{cases}$$

We use a Newton iteration to solve it :

Proposition

Let $n \geq 0$ be an integer. If $X_n(t)$ is a solution of the ODS modulo t^{n+1}, then

$$X_{2n+1} = X_n + (H(X_n))^{-1} \int (G - H(X_n) \cdot X_n')\, dt$$

is a solution modulo t^{2n+2}.

Assume that I is defined over a finite field of characteristic $p > 0$ ($p \neq 2$).
If $p < 2g\ell$, then the ODS have more than one solution because :

$$\int t^{p-1} dt = ?$$

In order to have a unique solution :

- We lift the ODS : a finite extension $\mathbb{Q}_p$.
- Solve then reduce modulo p.
- Solving in $\mathbb{Q}_p \implies$ loss of p-adic precision.
  - $\longrightarrow$ An optimal bound of the loss of p-adic precision is already found.

Complexity ?

$$X_{2n+1} = X_n + (H(X_n))^{-1} \int (G - H(X_n) \cdot X_n')\, dt$$

The components of vector X_n are defined in $L[\![t]\!]$ where L is an extension of k of degree at most $O(g)$.

Naive algorithm complexity : $\widetilde{O}(g^4\ell)$ operations in k (quasi-optimal in ℓ but not in g).

How do we achieve quasi-optimality in g ?

Goal : $\widetilde{O}(g^2\ell)$ instead of $\widetilde{O}(g^4\ell)$ operations in k.

First idea : The matrix $H(x_1,\ldots,x_g)$,

$$H(x_1,\ldots,x_g)=\begin{pmatrix} 1/y_1(t) & \ldots & \ldots & 1/y_g(t) \\ x_1(t)/y_1(t) & \ldots & \ldots & x_g(t)/y_g(t) \\ \vdots & & & \vdots \\ x_1(t)^{g-1}/y_1(t) & \ldots & \ldots & x_g(t)^{g-1}/y_g(t) \end{pmatrix}$$

is a structured matrix.

→ Complexity of the Newton iteration : $\widetilde{O}(g^3\ell)$ instead of $\widetilde{O}(g^4\ell)$ operations in k.

Second idea : We rewrite the Newton iteration to compute the polynomial $U(X,t) = \prod_{i=1}^{g} (X - x_i(t)) \in k[\![t]\!][x]$ instead of computing $(x_1, \dots, x_g) \in L[\![t]\!]$.

$\longrightarrow$ Complexity of the Newton iteration : $\widetilde{O}(g^2\ell)$ instead of $\widetilde{O}(g^3\ell)$ operations in k.

Overall algorithm

$$X_{2n+1} = X_n + (H(X_n))^{-1} \int (G - H(X_n) \cdot X_n')\, dt.$$

Step 1 : Compute $H(X_n) \cdot X_n'$ and $H(X_n) \cdot X_n$ $\widetilde{O}(gn)$

Step 2 : Compute $F_n = H(X_n) \cdot X_n + \int (G - H(X_n) \cdot X_n')$ $\widetilde{O}(gn)$

Step 3 : Solve $H(X_n) \cdot X_{2n+1} = F_n$ (Shoup + Kedlaya-Umans) $\widetilde{O}(gn)$

Difficult to implement the quasi-optimal algorithm in an optimized way since it uses the Kedlaya-Umans algorithm.

$$g = 2, 5, 7$$