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

Elie Eid

COMPUTING ISOGENIES BETWEEN JACOBIANS OF HYPERELLIPTIC CURVES OF ARBITRARY GENUS VIA DIFFERENTIAL EQUATIONS

ELIE EID

Abstract. Let p be an odd prime number and ℓ be an integer coprime to p. We survey an algorithm for computing explicit rational representations of $(\ell,\dots,\ell)$-isogenies between Jacobians of hyperelliptic curves of arbitrary genus over an extension K of the field of p-adic numbers $\mathbb{Q}_p$. The algorithm has a quasi-linear complexity in ℓ as well as in the genus of the curves.

1. Introduction

Over the last few years there has been a growing interest in computational aspects of abelian varieties, especially Jacobians of algebraic curves. When such a variety is given, a first task is to compute the number of points on it in some finite field [Sch95, BGG+17]. A way to achieve this efficiently is to work with isogenies. In addition to point counting, the computation of isogenies has many applications in number theory and cryptography [CEL12, CL13, FT19, CS20].
In order to have optimal algorithms for computing isogenies, in particular those which are defined over finite fields, several approaches have been suggested. One of them consists in reducing the problem to the computation of a solution of a nonlinear differential equation [Elk97, CE15], possibly after having lifted the problem to the p-adics [LS08, LV16, CEL20, Eid20]. In this work, we focus on p-adic algorithms that compute the explicit form of a rational representation of an isogeny between Jacobians of hyperelliptic curves for fields of odd characteristic.

Let k be a field of characteristic different from 2 and $\ell > 1$ and $g > 1$ two integers. Let C (resp. C_1) be a genus g hyperelliptic curve over k and let J (resp. J_1) be its Jacobian. We assume that there exists a separable $(\ell,\dots,\ell)$-isogeny $I : J_1 \to J$ defined over k and we are interested in computing one of its rational representations. Let us recall briefly the definition of a rational representation and how we compute it (see [Eid20] for more details). Let P be a Weierstrass point on C_1 and $j_P : C_1 \to J_1$ the Jacobi map with origin P. The morphism $I \circ j_p$ induces a morphism $I_P : C_1 \to C^{(g)}$, where $C^{(g)}$ is the g-th symmetric power of C. When a coordinate system for C_1 and C is fixed, the morphism I_P is given by its Mumford representation, which consists of a pair of polynomials $(U(z), V(z))$ with the following property: if $I_P(Q) = \{R_1,\dots,R_g\}$ (for some $Q \in C_1$), then $U(x(R_i)) = 0$ and $V(x(R_i)) = y(R_i)$, for all $i = 1,\dots,g$. Here $x(R_i)$ and $y(R_i)$ denote the coordinates of the point R_i. The $2g$ coefficients of the two polynomials U and V can be represented as rational fractions over k in one variable and they form what we call a rational representation of I.
We assume that C (resp. C_1) is given by the affine model $C : y^2 = f(x)$ (resp. $C_1 : v^2 = f_1(u)$). Let Q be a non-Weierstrass point on C_1 such that $I_P(Q) = \{(x_1^{(0)}, y_1^{(0)}),\dots,(x_g^{(0)}, y_g^{(0)})\}$ contains g distinct points and does not contain neither a point at infinity nor a Weierstrass point. Let t be a formal parameter of C_1 at Q and let $\{(x_1(t), y_1(t)),\dots,(x_g(t), y_g(t))\}$ be the image of $Q(t)$ by I_P. The action of I_P on the spaces of holomorphic differentials of C_1 and $C^{(g)}$ gives the

following differential system whose unknown is $X(t) = (x_1(t), \dots, x_g(t)) \in \bar{k}[\![t]\!]$.

$$\begin{cases} H(X(t)) \cdot X'(t) = G(t) \\ y_i(t)^2 = f(x_i(t)),\ i = 1, \dots, g \\ X(0) = (x_1^{(0)}, \cdots, x_g^{(0)}) \\ Y(0) = (y_1^{(0)}, \cdots, y_g^{(0)}) \end{cases} \tag{1}$$

where $G(t) = (G_1(t), \dots, G_g(t)) \in k[\![t]\!]^g$ and $H(X(t))$ is the matrix defined by

$$H(x_1(t), \dots, x_g(t)) = \begin{pmatrix} x_1(t)/y_1(t) & x_2(t)/y_2(t) & \cdots & x_g(t)/y_g(t) \\ x_1(t)^2/y_1(t) & x_2(t)^2/y_2(t) & & x_g(t)^2/y_g(t) \\ \vdots & & & \vdots \\ x_1^{g-1}(t)/y_1(t) & x_2(t)^{g-1}/y_1(t) & \cdots & x_g(t)^{g-1}/y_g(t) \end{pmatrix} \tag{2}$$

Since the coefficients of $U(z)$ are rational fractions of degree at most $O(g\ell)$ [Eid20, Proposition 9], solving Equation (1) modulo $t^{O(g\ell)}$ allows to reconstruct all the components of the rational representation (note that the polynomial $V(z)$ can be recovered using the polynomial $U(z)$ and the equation of C).

Let p be an odd prime number. We assume that k is a finite field of characteristic p. Let K be an unramified extension of $\mathbb{Q}_p$ such that the residue field of K is k. In [Eid20], we have designed an algorithm that computes, after lifting Equation (1) over K, an approximation of its solution. This algorithm is based on the following Newton iteration:

$$X_{2m}(t) = X_m(t) + H(X_m(t))^{-1} \int (G - H(X_m(t)) \cdot X_m'(t))\, dt \tag{3}$$

which gives more and more accurate (for the t-adic distance) solutions of Equation (1). The complexity of this algorithm is quasi-linear with respect to ℓ but, unfortunately, it is at least quadratic in g (even if we note that the matrix $H(x_1(t), \dots, x_n(t))$ is a structured matrix). The main reason for this lack of efficiency is due to the fact that the components of the solution $X(t)$ of Equation (1) are power series over an unramified extension of degree g of K. However, the rational fractions of the rational representation are defined over the ring of integers $\mathcal{O}_K$ of K. This is where we loose an extra factor g.

In this article, we revisit the algorithm of [Eid20] and manage to lower its complexity in g and make it quasi-linear as well. For this, we work directly on the first Mumford coordinate $U(z) = \prod_{i=1}^{g} (z - x_i(t))$ which has the decisive advantage to be defined over the base field: we rewrite the Newton scheme (3) accordingly and design fast algorithms for iterating it in quasi-linear time. Our main theorem is the following

Theorem 1. *Let K be an unramified extension of $\mathbb{Q}_p$ and k its residue field. There exists an algorithm that takes as input:*

- *three positive integers g,n and N,*
- *A polynomial $f \in \mathcal{O}_K[z]$ of degree $2g+1$,*
- *a vector $X_0 = (x_1^{(0)}, \dots, x_g^{(0)})$ represented by the polynomial $U_0(z) = \prod_{i=1}^{g} (z - x_i^{(0)}) \in \mathcal{O}_K[z]$ such that, over k, $U_0(z)$ is separable,*
- *a vector $Y_0 = (y_1^{(0)}, \dots, y_g^{(0)})$ represented by the interpolating polynomial $V_0(z) \in \mathcal{O}_K[z]$ of the data $\{(x_1^{(0)}, y_1^{(0)}), \dots, (x_g^{(0)}, y_g^{(0)})\}$,*

- *a vector* $G(t) \in \mathcal{O}_K[\![t]\!]^g$,

and, assuming that the solution of Equation (1) *has coefficients in* $\mathcal{O}_L$ *with* L *an unramified extension of* K*, outputs a polynomial* $U(t,z) = \prod_{i=1}^{g}(z - x_i(t)) \in \mathcal{O}_K[\![t]\!][z]$ *such that* $X(t) = (x_1(t), \ldots, x_g(t))$ *is an approximation of this solution modulo* (p^N, t^{n+1}) *for a cost* $\tilde{O}(ng)$ *operations*[1] *in* $\mathcal{O}_K$ *at precision* $O(p^M)$ *with* $M = N + \lfloor \log_p(n) \rfloor$.

Important examples of isogenies are, of course, the multiplication-by-ℓ maps. Classical algorithms for computing them are usually based on Cantor algorithm for adding points on Jacobians (see for example [Can94, Abe18]). Although, they exhibit acceptable running time in practice, their theoretical complexity has not been well studied yet and experiments show that they become much slower when the genus gets higher. Actually, in many cases, we have observed that the algorithms of [Eid20] perform better in practice even if their theoretical complexity in g is not optimal. Consequently, even though the algorithms designed in the present paper use Kedlaya-Umans algorithm [KU11] as a subroutine and then could be difficult to implement in an optimized way, they appear as attractive alternatives for the computation of ℓ-division polynomials on Jacobians of hyperelliptic curves.

2. The main result

In this section, we sketch the proof of the main theorem by showing that the Newton iteration given in Equation (3) can be executed with quasi-linear time complexity to give the desired polynomial in Theorem 1. The precision analysis has been already studied in [Eid20].
Throughout this section, the letter p refers to a fixed odd prime number and the letter K refers to a fixed unramified extension of $\mathbb{Q}_p$ of degree d and k its residue field. Let $\mathcal{O}_K$ be the ring of integers of K.
We use *the fixed point arithmetic model* at precision $O(p^M)$ to do computations in $\mathcal{O}_K$ by representing an element in $\mathcal{O}_K$ by an expression of the form $x + O(p^M)$ with $x \in \mathcal{O}_K/p^M\mathcal{O}_K$. For instance, if $d = 1$, the quotient $\mathcal{O}_K/p^M\mathcal{O}_K$ is just $\mathbb{Z}/p^M\mathbb{Z}$. Additions, multiplications and divisions in this model all reduce to the similar operations in the exact quotient ring $\mathcal{O}_K/p^M\mathcal{O}_K$.
Let $\mathrm{M}(m)$ be the number of arithmetical operations required to compute the product of two polynomials of degree m in an exact ring. Standard algorithms allow us to take $\mathrm{M}(m) \in \tilde{O}(m)$.
Let $g > 1$ be an integer and let $G(t) \in \mathcal{O}_K[\![t]\!]$. Let also f be a polynomial of degree $2g+1$ and let $U_0(z) \in \mathcal{O}_K[z]$ be a polynomial of degree g which separable over k. For the sake of simplicity, we assume that U_0 is irreducible, therefore its splitting field L is an unramified extension of degree g of K. Let $x_1^{(0)}, \ldots, x_g^{(0)}$ be the roots of $U_0(z)$ in L and $X_0 = (x_1^{(0)}, \ldots, x_g^{(0)})$. For $i = 1, \ldots, g$, we assume that $f(x_i^{(0)})$ has a square root $y_i^{(0)}$ in $\mathcal{O}_L$. Take $Y_0 = (y_1^{(0)}, \ldots, y_g^{(0)})$ and let $V_0(z) \in \mathcal{O}_K[z]$ be the interpolating polynomial of the data $\{(x_1^{(0)}, y_1^{(0)}), \ldots, (x_g^{(0)}, y_g^{(0)})\}$. We assume that the unique solution $X(t) = (x_1(t), \ldots, x_n(t))$ of Equation (1) has coefficients in $\mathcal{O}_L$ when X_0 and Y_0 are the initial conditions.
Let $m \in \mathbb{N}$ and $n = 2m$. Let $X_m(t) = (x_1^{(m)}(t), \ldots, x_g^{(m)}(t))$ be an approximation of $X(t)$ modulo t^m represented by the minimal polynomial of $x_1^{(m)}$, $U_m(t,z) = \prod(z - x_i^{(m)}(t))$. We show in the next proposition that we can compute efficiently an approximation $X_n(t) = (x_1^{(n)}(t), \ldots, x_g^{(n)}(t))$ of $X(t)$ modulo t^n represented by the minimal polynomial $U_n(t,z)$ of $x_1^{(n)}(t)$ using Equation (3).

[1] The notation $\tilde{O}(-)$ means that we are hiding logarithmic factors.

Proposition 2. *Using the same notations as above, there exists an algorithm that computes $U_n(t,z)$ from $U_m(t,z)$ with time complexity $\tilde{O}(mg)$.*

Sketch of the proof. The algorithm performs the following steps.

(1) Compute the degree $g-1$ polynomial $W_m(t,z) = \sum_{i=0}^{g-1} w_i^{(m)}(t)\, z^i$ such that

$$W_m(t,z) \equiv 1/f^2(z) \mod (t^m, U_m(t,z))$$

and $W_m(0,z) = 1/V_0(z) \mod U_0(z)$. Observe that it is the interpolating polynomial of the points:

$$\{(x_1^{(m)}, 1/y_1^{(m)}), \cdots, (x_g^{(m)}, 1/y_g^{(m)})\}.$$

Deduce $V_m(z) = f(z)W_m(z) \mod (t^m, U_m(z))$.

(2) Compute the Newton sums $s_i^{(m)}(t) = \sum_{j=1}^{g} (x_j^{(m)}(t))^i \mod t^m$ for $i = 1, \ldots, 2g-1$ and deduce $r_i^{(m)}(t) = \sum_{j=1}^{g} (x_j^{(m)}(t))^{i-1} (x_j^{(m)}(t))' \mod t^m$.

(3) Compute the two products $H(X_m(t))X_m'(t)$ and $H(X_m(t))X_m(t)$ as follows:

$$H(X_m(t))X_m'(t) = \begin{pmatrix} r_1^{(m)} & r_2^{(m)} & \cdots & r_g^{(m)} \\ r_2^{(m)} & r_3^{(m)} & & r_{g+1}^{(m)} \\ \vdots & & & \\ r_g^{(m)} & r_{g+1}^{(m)} & \cdots & r_{2g-1}^{(m)} \end{pmatrix} \begin{pmatrix} w_0^{(m)} \\ w_1^{(m)} \\ \vdots \\ w_{g-1}^{(m)} \end{pmatrix} \mod t^m$$

and

$$H(X_m(t))X_m(t) = \begin{pmatrix} s_1^{(m)} & s_2^{(m)} & \cdots & s_g^{(m)} \\ s_2^{(m)} & s_3^{(m)} & & s_{g+1}^{(m)} \\ \vdots & & & \\ s_g^{(m)} & s_{g+1}^{(m)} & \cdots & s_{2g-1}^{(m)} \end{pmatrix} \begin{pmatrix} w_0^{(m)} \\ w_1^{(m)} \\ \vdots \\ w_{g-1}^{(m)} \end{pmatrix} \mod t^m$$

(4) Compute $(F_1^{(m)}, \cdots, F_g^{(m)}) = H(X_m(t))X_m(t) - \int (G(t) - H(X_m(t))X_m'(t))\, dt$.

(5) Let $D_m(t,z) = F_1^{(m)} z^g + F_2^{(m)} z^{g-1} + \ldots + F_{g-1}^{(m)} z^2 + F_g^{(m)} z$. Compute $U_m(t,z)D_m(t,z) = q_{2g}^{(m)} z^{2g} + q_{2g-1}^{(m)} z^{2g-1} + \ldots + q_0^{(m)} \mod t^m$ and read off the polynomial $Q_m(t,z) = q_{2g}^{(m)} z^{g-1} + q_{2g-1}^{(m)} z^{g-2} + \ldots + q_{g+1}^{(m)}$.

(6) Compute $T_m(t,z) = \dfrac{Q_m(t,z)V_m(t,z)}{U_m'(t,z)} \mod (t^m, U_m(t,z))$.

(7) Compute $U_n(t,z)$ such that $U_n(t, T_m(t,z)) \equiv 0 \mod (t^m, U_m(t,z))$.

We now discuss briefly the complexity analysis. The polynomial W_m in step 1 can be efficiently computed by the classical Newton scheme for extracting square roots. Since, the coefficients of W_m and V_m are polynomials of degrees at most m defined over K, the complexity of this

step is $O(\mathrm{M}(m)\mathrm{M}(g))$. The computation of the Newton sums $s_i^{(m)}$ of U_m in step 2 is classical [BGVPS21] and can be carried out for a cost of $O(\mathrm{M}(m)\mathrm{M}(g))$ operations. In step 3, we are dealing with two Hankel matrix-vector products. This can be done in $O(\mathrm{M}(m)\mathrm{M}(g))$ operations in K [CKY89, Section 3a]. The polynomial T_m constructed in step 5 and step 6 interpolates the data $\{(x_1^{(m)}, x_1^{(n)}), \ldots, (x_g^{(m)}, x_g^{(n)})\}$ (see [KY89, Section 5] for more details), it can be computed in $O(\mathrm{M}(m)\mathrm{M}(g))$ as well. Step 7 computes U_n, the minimal polynomial of $x_1^{(n)}$. We make use of Kedlaya-Umans algorithm [KU11] to execute step 7; the resulting bit complexity is $\tilde{O}(mg)$. □

References

[Abe18] S. Abelard. *Comptage de points de courbes hyperelliptiques en grande caractéristique : algorithmes et complexité*. PhD thesis, 2018. Thèse de doctorat dirigée par Gaudry, Pierrick et Spaenlehauer, Pierre-Jean Informatique Université de Lorraine 2018. 3

[BGG+17] S. Ballentine, A. Guillevic, E. L. García, C. Martindale, M. Massierer, B. Smith, and J. Top. Isogenies for point counting on genus two hyperelliptic curves with maximal real multiplication. In *Algebraic geometry for coding theory and cryptography*, pages 63–94. Springer, 2017. 1

[BGVPS21] A. Bostan, L. González-Vega, H. Perdry, and E. Schost. Complexity issues on newton sums of polynomials. 02 2021. 5

[Can94] D. G. Cantor. On the analogue of the division polynomials for hyperelliptic curves. 1994(447):91–146, 1994. 3

[CE15] J.-M. Couveignes and T. Ezome. Computing functions on jacobians and their quotients. *LMS Journal of Computation and Mathematics*, 18(1):555–577, 2015. 1

[CEL12] J.-M. Couveignes, T. Ezome, and R. Lercier. A faster pseudo-primality test. *Rend. Circ. Mat. Palermo (2)*, 61(2):261–278, 2012. 1

[CEL20] X. Caruso, E. Eid, and R. Lercier. Fast computation of elliptic curve isogenies in characteristic two. working paper or preprint, March 2020. 1

[CKY89] J. F. Canny, E. Kaltofen, and L. Yagati. Solving systems of nonlinear polynomial equations faster. In *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, ISSAC '89, page 121–128, New York, NY, USA, 1989. Association for Computing Machinery. 5

[CL13] J.-M. Couveignes and R. Lercier. Fast construction of irreducible polynomials over finite fields. *Israel J. Math.*, 194(1):77–105, 2013. 1

[CS20] C. Costello and B. Smith. The supersingular isogeny problem in genus 2 and beyond. In *International Conference on Post-Quantum Cryptography*, pages 151–168. Springer, 2020. 1

[Eid20] E. Eid. Fast computation of hyperelliptic curve isogenies in odd characteristic, 2020. 1, 2, 3

[Elk97] N. Elkies. Elliptic and modular curves over finite fields and related computational issues. 1997. 1

[FT19] E. V. Flynn and Y. B. Ti. Genus two isogeny cryptography. In J. Ding and R. Steinwandt, editors, *Post-Quantum Cryptography*, pages 286–306, Cham, 2019. Springer International Publishing. 1

[KU11] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011. 3, 5

[KY89] E. Kaltofen and L. Yagati. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Symbolic and Algebraic Computation*, pages 467–474, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. 5

[LS08] R. Lercier and T. Sirvent. On Elkies subgroups of l-torsion points in elliptic curves defined over a finite field. *J. Théor. Nombres Bordeaux*, 20(3):783–797, 2008. 1

[LV16] P. Lairez and T. Vaccon. On p-adic differential equations with separation of variables. In *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, pages 319–323. ACM, New York, 2016. 1

[Sch95] R. Schoof. Counting points on elliptic curves over finite fields. *Journal de Théorie des Nombres de Bordeaux*, 7(1):219–254, 1995. 1

Elie Eid, Univ. Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France.
Email address: elie.eid@univ-rennes1.fr