JSIAM Letters Vol.11 (2019) pp.1–4 ©2019 Japan Society for Industrial and Applied Mathematics JSIAM Letters

Counting points on hyperelliptic curves of genus 2 with real models

Yukihiro Uchida[1]

[1] Department of Mathematical Sciences, Graduate School of Science, Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji, Tokyo 192-0397, Japan

E-mail *yuchida@tmu.ac.jp*

Received September 29, 2018, Accepted December 9, 2018

Abstract

A model of a hyperelliptic curve is called an imaginary or real model if it has one or two points at infinity, respectively. In this paper, we propose an algorithm to count points on hyperelliptic curves of genus 2 with real models defined over finite fields of large characteristics. We also estimate the complexity of our algorithm and prove that it has the same order as that of a previously known algorithm for imaginary models.

Keywords point counting, hyperelliptic curves, real models

Research Activity Group Algorithmic Number Theory and Its Applications

1. Introduction

Counting points on algebraic curves defined over finite fields is an important problem in algorithmic number theory. For example, we need to count points on curves and their Jacobian varieties to construct algebraic curve cryptography. When the characteristic of the field of definition is small, we can count points on hyperelliptic curves efficiently by using p-adic algorithms proposed by Satoh [1], Kedlaya [2], and others. When the characteristic is large, ℓ-adic algorithms are suitable.

Schoof [3] proposed an ℓ-adic algorithm to count points on elliptic curves using division polynomials. Pila [4] generalized Schoof's algorithm to arbitrary Abelian varieties, however, Pila's algorithm is not practical. Gaudry and Harley [5] and Gaudry and Schost [6] proposed practical algorithms to count points on hyperelliptic curves of genus 2 with imaginary models by using division polynomials defined by Cantor [7].

In general, a hyperelliptic curve of genus g defined over a field of characteristic not equal to 2 has a model $y^2 = f(x)$, where f has no multiple roots and $\deg f = 2g+1$ or $2g+2$. We say that the model is imaginary or real if $\deg f$ is odd or even, respectively. The model is imaginary if and only if it has only one point at infinity. Note that Kedlaya's algorithm [2] was extended to hyperelliptic curves with real models by Harrison [8].

In this paper, we propose an algorithm to count points on hyperelliptic curves of genus 2 with real models. Our algorithm is a generalization of Schoof's algorithm. More precisely, the division polynomials defined by the author [9] are used in our algorithm instead of Cantor's division polynomials.

Note that Cantor's division polynomials are not defined for real models. As described in Section 2, a hyperelliptic curve has an imaginary model if and only if it has a rational Weierstrass point. Therefore, our algorithm can be applied to more general hyperelliptic curves of genus 2 than a previously known algorithm which uses Cantor's division polynomials.

This paper is organized as follows. In Section 2, we recall basic facts on hyperelliptic curves of genus 2 and Kummer surfaces. We also introduce the division polynomials constructed by the author [9]. In Section 3, we propose an algorithm to count points on hyperelliptic curves of genus 2 with real models. In Section 4, we estimate the complexity of our algorithm. In Section 5, we give a remark that we can improve our algorithm by using ℓ^n-torsion points. In Section 6, we give a conclusion.

2. Hyperelliptic curves of genus 2

We first recall real and imaginary models of hyperelliptic curves of genus 2. We refer the reader to [10] for details. Let F be a field of characteristic not equal to 2. Let C be a hyperelliptic curve of genus 2 defined over F with a model $y^2 = f(x)$, where $f(x) \in F[x]$ has no multiple roots and $\deg f = 5$ or 6. The model $y^2 = f(x)$ is imaginary or real if $\deg f = 5$ or $\deg f = 6$, respectively.

An imaginary model has one point at infinity and a real model has two points at infinity. Let ∞ be a point at infinity. For a real model, we denote by $\overline{\infty}$ the other point at infinity. Let $\iota\colon C \to C$ be the hyperelliptic involution. Then $\iota(x,y) = (x,-y)$ for a finite point $(x,y) \in C$. When the model is imaginary, $\iota(\infty) = \infty$. When the model is real, $\iota(\infty) = \overline{\infty}$ and $\iota(\overline{\infty}) = \infty$. We say that P is a Weierstrass (or ramified) point if $\iota(P) = P$.

If C has an imaginary model defined over F, then C has a real model defined over F provided that F has more than 5 elements. On the other hand, C has an imaginary model defined over F if and only if it has an F-rational Weierstrass point (see [10, Proposition 2.1] or [11, Chapter 1, Section 1]). From now on, we mainly consider real models, however the following discussion is valid for imaginary models.

We explain Kummer surfaces following [11]. Let J be

the Jacobian variety of C. Then J is an Abelian surface. Identifying P and $-P$ on J, we obtain the Kummer surface K associated with J. The Kummer surface K is regarded as a hypersurface of degree 4 in $\mathbb{P}^3$. Let $\kappa\colon J \to K$ be the natural morphism. We write $\kappa(P) = (\xi_1(P) : \cdots : \xi_4(P))$. Let O be the identity of J. We may assume that $\kappa(O) = (0:0:0:1)$. Let G be a defining equation of K in $\mathbb{P}^3$. Then G is a homogeneous polynomial of degree 4. An explicit formula for G is given in [11, (3.1.8), (3.1.9)]. Note that $\deg_{\xi_4} G = 2$ in the formula.

Let $\overline{F}$ be an algebraic closure of F. There exist homogeneous polynomials δ_i in 4 variables and B_{ij} in 8 variables which satisfy the following: For all $P, Q \in J(\overline{F})$, there exist $c_1, c_2 \in \overline{F}^{\times}$ such that

$$\begin{aligned}&\xi_i(2P) = c_1\delta_i(\xi_1(P),\dots,\xi_4(P)),\\ &\xi_i(P+Q)\xi_j(P-Q) + \xi_i(P-Q)\xi_j(P+Q)\\ &\quad = 2c_2B_{ij}(\xi_1(P),\dots,\xi_4(P),\xi_1(Q),\dots,\xi_4(Q))\end{aligned}$$

for all $1 \le i, j \le 4$ (see [11, Chapter 3, Sections 4 and 5]).

The author [9] constructed polynomials which represent a multiplication map by using δ_i and B_{ij}. The following theorem follows immediately from [9, Theorem 3.3, Corollary 3.7, and Lemma 3.8].

Theorem 1 *For all $m \ge 1$ and $1 \le i \le 4$, there exist homogeneous polynomials $\mu_{m,i} \in F[\xi_1,\dots,\xi_4]$ of degree m^2 such that*

$$\mu_{1,i} = \xi_i, \quad \mu_{2,i} = \delta_i, \quad \mu_{2m,i} = \delta_i(\mu_{m,1},\dots,\mu_{m,4}),$$

$$\mu_{2m+1,i}\xi_i = B_{ii}(\mu_{m+1,1},\dots,\mu_{m+1,4},\mu_{m,1},\dots,\mu_{m,4})$$

in $F[\xi_1,\dots,\xi_4]/\langle G\rangle$. Furthermore, for all $P \in J(\overline{F})$,

$$\kappa(mP) = (\mu_{m,1}(\kappa(P)) : \cdots : \mu_{m,4}(\kappa(P))).$$

The polynomials $\mu_{m,i}$ can be computed by using arithmetic operations in $F[\xi_1,\dots,\xi_4]$ (see [9, Remark 3.5]).

3. Point counting

From now on, we assume that $F = \mathbb{F}_q$, where q is a power of an odd prime number p. We consider counting points on C and its Jacobian variety J.

We first recall some properties of the characteristic polynomial of the Frobenius map. We refer the reader to [12, Section 5.2] and [13, Chapter 5] for details. Let $\pi_q\colon J \to J$ be the Frobenius map. Then the characteristic polynomial $\chi(T)$ of π_q is of the form

$$\chi(T) = T^4 - s_1T^3 + s_2T^2 - qs_1T + q^2,$$

where s_1 and s_2 are integers. The order of $J(\mathbb{F}_q)$ satisfies $\#J(\mathbb{F}_q) = \chi(1)$. Moreover, when $\chi(T) = \prod_{i=1}^4(T-\alpha_i)$, we have $\#C(\mathbb{F}_{q^k}) = q^k + 1 - \sum_{i=1}^4 \alpha_i^k$ for any positive integer k. Hence it is sufficient to compute $\chi(T)$.

For a positive integer m, let $J[m] = \{P \in J(\overline{\mathbb{F}_q}) \mid mP = O\}$. For any m-torsion point $P \in J[m]$, we have

$$\pi_q^4(P) - s_1\pi_q^3(P) + s_2\pi_q^2(P) - qs_1\pi_q(P) + q^2P = O. \quad (1)$$

In order to compute $\chi(T)$, for each prime number ℓ, we find $(s_1 \bmod \ell, s_2 \bmod \ell)$ such that (1) holds for all $P \in J[\ell]$. In usual, a unique pair $(s_1 \bmod \ell, s_2 \bmod \ell)$ is determined by (1). Even if several pairs $(s_1 \bmod \ell, s_2 \bmod \ell)$ satisfy (1), we can determine $\chi(T) \bmod \ell$. For details, see [6, Section 3.4].

Algorithm 1 Point counting for curves of genus 2

Input: $C/\mathbb{F}_q\colon y^2 = f(x)$
Output: $\chi(T)$
1: $\ell_m \leftarrow$ a prime with $2\cdot3\cdot5\cdots\ell_m > 12q$.
2: **for** $\ell \leftarrow 2, 3, 5, \dots, \ell_m$ **do:**
3: Compute $J[\ell]$ (Algorithm 2).
4: Find $(s_1 \bmod \ell, s_2 \bmod \ell)$ by (1).
5: **end for**
6: Compute s_1 and s_2 by (2) and the Chinese remainder theorem.
7: **return** $T^4 - s_1T^3 + s_2T^2 - qs_1T + q^2$.

By the Weil conjectures for Abelian varieties, which were proved by Weil, we have

$$|s_1| \le 4\sqrt{q}, \quad |s_2| \le 6q. \quad (2)$$

Note that we can also use the following sharper bounds given by Elkies and Momose (see [14, Lemma 1]):

$$\lceil 2\sqrt{q}\,|s_1| - 2q\rceil \le s_2 \le \left\lfloor \frac{1}{4}s_1^2 + 2q \right\rfloor.$$

Let ℓ_m be a prime such that $2\cdot3\cdot5\cdots\ell_m > 12q$. Then we can compute the integers s_1 and s_2 from $\{(s_1 \bmod \ell, s_2 \bmod \ell) \mid \ell = 2,3,5,\dots,\ell_m\}$ by the Chinese remainder theorem. This algorithm is summarized in Algorithm 1. In practice, when we only need the order of $J(\mathbb{F}_q)$, Algorithm 1 is used with the baby step giant step algorithm.

We consider the computation of $J[\ell]$ for each prime number ℓ. In the case of imaginary models, we use the division polynomials defined by Cantor [7]. However, Cantor's division polynomials can be defined only for imaginary models. In the case of real models, we use the division polynomials $\mu_{\ell,i}$ defined by the author [9] (see Theorem 1).

Let $P \in J(\overline{\mathbb{F}_q})$. Then $P \in J[\ell]$ if and only if

$$\begin{aligned}\mu_{\ell,1}(\kappa(P)) &= \mu_{\ell,2}(\kappa(P))\\ &= \mu_{\ell,3}(\kappa(P)) = G(\kappa(P)) = 0.\end{aligned} \quad (3)$$

Therefore it is sufficient to solve (3) in order to compute $J[\ell]$. We solve (3) by using resultants. An algorithm to compute the set $\{P \in J[\ell] \mid \xi_1(P) \ne 0\}$ is shown in Algorithm 2. The points $P \in J[\ell]$ with $\xi_1(P) = 0$ can be computed by a similar algorithm.

If (1) holds for an ℓ-torsion $P \in J[\ell]$, then (1) also holds for $-P$ and the conjugates of P and $-P$ over $\mathbb{F}_q$. Therefore, when Algorithm 2 is called in Algorithm 1, it is efficient to find candidates of $(s_1 \bmod \ell, s_2 \bmod \ell)$ in step 18 in Algorithm 2 instead of creating the list L.

4. Complexity

In this section, we estimate the complexities of the algorithms described in Section 3. We first review the costs of algorithms used in our algorithms. We assume that q is sufficiently large. The cost of an arithmetic operation in $\mathbb{F}_q$ is $(\log q)^{1+o(1)}$ bit operations. Let $M(n)$ be

Algorithm 2 Computation of $J[\ell]$

Input: $C/\mathbb{F}_q$: $y^2 = f(x)$, ℓ: a prime number
Output: $\{P \in J[\ell] \mid \xi_1(P) \neq 0\}$

1: Let L be an empty list.
2: Compute $\mu_{\ell,i}$ for $i = 1, 2, 3$.
3: $R_i \leftarrow \mathrm{Res}_{\xi_4}(\mu_{\ell,i}(1, \xi_2, \xi_3, \xi_4), G(1, \xi_2, \xi_3, \xi_4))$ for $i = 1, 2, 3$.
4: $S_i \leftarrow \mathrm{Res}_{\xi_3}(R_i, R_3)$ for $i = 1, 2$.
5: $T \leftarrow \gcd(S_1, S_2)$.
6: Factorize T over $\mathbb{F}_q$.
7: **for** $\tilde{T} \leftarrow$ each irreducible factor of T **do:**
8: Let $\alpha_{\tilde{T}} \in \overline{\mathbb{F}_q}$ be a root of $\tilde{T}$.
9: $U_{\tilde{T}} \leftarrow \gcd(R_1(\alpha_{\tilde{T}}, \xi_3), R_2(\alpha_{\tilde{T}}, \xi_3), R_3(\alpha_{\tilde{T}}, \xi_3))$.
10: Factorize $U_{\tilde{T}}$ over $\mathbb{F}_q(\alpha_{\tilde{T}})$.
11: **for** $\tilde{U} \leftarrow$ each irreducible factor of $U_{\tilde{T}}$ **do:**
12: Let $\beta_{\tilde{U}} \in \overline{\mathbb{F}_q}$ be a root of $\tilde{U}$.
13: $V_{\tilde{U}} \leftarrow \gcd(\mu_{\ell,1}(\boldsymbol{\xi}), \mu_{\ell,2}(\boldsymbol{\xi}), \mu_{\ell,3}(\boldsymbol{\xi}), G(\boldsymbol{\xi}))$, where $\boldsymbol{\xi} = (1, \alpha_{\tilde{T}}, \beta_{\tilde{U}}, \xi_4)$.
14: Factorize $V_{\tilde{U}}$ over $\mathbb{F}_q(\alpha_{\tilde{T}}, \beta_{\tilde{U}})$.
15: **for** $\tilde{V} \leftarrow$ each irreducible factor of $V_{\tilde{U}}$ **do:**
16: Let $\gamma_{\tilde{V}} \in \overline{\mathbb{F}_q}$ be a root of $\tilde{V}$.
17: Find $P \in J(\overline{\mathbb{F}_q})$ such that $\kappa(P) = (1 : \alpha_{\tilde{T}} : \beta_{\tilde{U}} : \gamma_{\tilde{V}})$.
18: Append P, $-P$ and the conjugates of them over $\mathbb{F}_q$ to L.
19: **end for**
20: **end for**
21: **end for**
22: **return** L

a function such that we can multiply two polynomials in $\mathbb{F}_q[x]$ of degree less than n in at most $M(n)$ operations in $\mathbb{F}_q$. We assume that $M(n)/n \geq M(m)/m$ if $n \geq m > 0$ and that $M(mn) \leq m^2 M(n)$ for all $n, m > 0$. We can take $M(n) = O(n \log n \log\log n)$. See [15, Section 8.3] for more details on $M(n)$. Moreover, we can compute the product of two polynomials in r variables of total degree less than $n/2$ in $O(rM(n)\binom{n+r-1}{r}/n)$ operations in $\mathbb{F}_q$ (see [16, Section 6.1], here we use points in a geometric progression as evaluation points). We can compute the greatest common divisor and the resultant of two polynomials in $\mathbb{F}_q[x]$ of degree at most n in $O(M(n)\log n)$ operations in $\mathbb{F}_q$ (see [15, Corollaries 11.9 and 11.19]). We can factorize a polynomial in $\mathbb{F}_q[x]$ of degree n in $(n^{1.5+o(1)} + n^{1+o(1)} \log q)(\log q)^{1+o(1)}$ bit operations by a randomized algorithm given by Kedlaya and Umans [17]. Moreover, when $q = p^k$, the factorization can be done in $n^{1.5+o(1)}(\log q)^{1+o(1)} + k^{1+o(1)} n^{1+o(1)} (\log p)^{2+o(1)}$ bit operations (see [17, Remark in Section 8.3]).

We also need to estimate the complexity of computing the resultant of two multivariate polynomials of different total degrees. Since we could not find an appropriate reference for such a result while Bouzidi et al. [18, Lemma 4] gave a similar result for univariate polynomials over $\mathbb{Z}$, we give a brief proof of the following lemma.

Lemma 2 *Let $f, g \in \mathbb{F}_q[y_1, \ldots, y_r, x]$ be polynomials of total degree n and m, respectively. We assume that $n \geq m$ and that q is sufficiently large. Then we can compute $\mathrm{Res}_x(f, g)$ in*

$$O\left(\binom{nm+r}{r}\left(\frac{rM(nm)}{m} + M(n)\log n\right)\right)$$

operations in $\mathbb{F}_q$.

Proof We compute $\mathrm{Res}_x(f, g)$ by using the algorithms for evaluation and interpolation given by van der Hoeven and Schost [16, Theorem 1]. Note that there exists a geometric progression of sufficient length in $\mathbb{F}_q$. Since the total degree of $\mathrm{Res}_x(f, g)$ is at most nm, it is sufficient to evaluate $\mathrm{Res}_x(f, g)$ at $\binom{nm+r}{r}$ points. We first evaluate the coefficients of f and g at $\binom{nm+r}{r}$ points. The cost of evaluation is

$$O\left(\binom{nm+r}{r} r\frac{M(nm)}{nm}(n+m+2)\right) = O\left(\binom{nm+r}{r}\frac{rM(nm)}{m}\right)$$

operations in $\mathbb{F}_q$. Next, we compute the resultant of f and g evaluated these points. This can be done in $O\left(\binom{nm+r}{r} M(n)\log n\right)$ operations in $\mathbb{F}_q$. Finally, we interpolate these values to obtain $\mathrm{Res}_x(f, g)$, which can be done in $O\left(\binom{nm+r}{r} rM(nm)/(nm)\right)$ operations in $\mathbb{F}_q$. Summarizing the above, we obtain the lemma. **(QED)**

We first consider the complexity of Algorithm 2. The polynomials $\mu_{\ell,i}$ can be computed in $\ell^{6+o(1)}$ operations in $\mathbb{F}_q$. The total degrees of $\mu_{\ell,i}(1, \xi_2, \xi_3, \xi_4)$ and $G(1, \xi_2, \xi_3, \xi_4)$ are at most ℓ^2 and 4, respectively. Hence the resultants in step 3 can be computed in $\ell^{6+o(1)}$ operations in $\mathbb{F}_q$ by Lemma 2. Similarly, the resultants in step 4 are computed in $\ell^{6+o(1)}$ operations in $\mathbb{F}_q$ since the total degree of R_i is at most $4\ell^2$. The greatest common divisor in step 5 is computed in $\ell^{4+o(1)}$ operations in $\mathbb{F}_q$ since $\deg S_i \leq 16\ell^4$. The factorization in step 6 takes $(\ell^{6+o(1)} + \ell^{4+o(1)} \log q)(\log q)^{1+o(1)}$ bit operations since $\deg T \leq 16\ell^4$.

Let $d_{\tilde{T}} = \deg \tilde{T}$ for an irreducible factor $\tilde{T}$ of T. For each $\tilde{T}$, we can compute U in $\ell^{2+o(1)} d_{\tilde{T}}^{1+o(1)}$ operations in $\mathbb{F}_q$ since $[\mathbb{F}_q(\alpha_{\tilde{T}}) : \mathbb{F}_q] = d_{\tilde{T}}$. Let $d_{U_{\tilde{T}}} = \deg U_{\tilde{T}}$. We can factorize $U_{\tilde{T}}$ over $\mathbb{F}_q(\alpha_{\tilde{T}})$ in

$$d_{\tilde{T}}^{1+o(1)}\left[d_{U_{\tilde{T}}}^{1.5+o(1)}(\log q)^{1+o(1)} + d_{U_{\tilde{T}}}^{1+o(1)}(\log q)^{2+o(1)}\right]$$

bit operations. Let β be a root of $U_{\tilde{T}}$. Then $(\alpha_{\tilde{T}}, \beta)$ is a solution of $R_1(\xi_2, \xi_3) = R_2(\xi_2, \xi_3) = 0$. Hence, by Bézout's theorem, we have $\sum_{\tilde{T}} d_{\tilde{T}} d_{U_{\tilde{T}}} \leq 16\ell^4$. Therefore the bit complexity of the factorizations of all $U_{\tilde{T}}$ in step 10 is

$$\sum_{\tilde{T}} d_{\tilde{T}}^{1+o(1)}\left[d_{U_{\tilde{T}}}^{1.5+o(1)}(\log q)^{1+o(1)} + d_{U_{\tilde{T}}}^{1+o(1)}(\log q)^{2+o(1)}\right] \leq \ell^{6+o(1)}(\log q)^{1+o(1)} + \ell^{4+o(1)}(\log q)^{2+o(1)}.$$

Let $d_{\tilde{U}} = \deg \tilde{U}$ for an irreducible factor $\tilde{U}$ of $U_{\tilde{T}}$. Since $\sum_{\tilde{U}} d_{\tilde{U}} \leq d_{U_{\tilde{T}}}$, we have $\sum_{\tilde{T}} \sum_{\tilde{U}} d_{\tilde{T}} d_{\tilde{U}} \leq 16\ell^4$. For each $\tilde{U}$, we can compute V in $\ell^{2+o(1)}(d_{\tilde{T}} d_{\tilde{U}})^{1+o(1)}$ operations in $\mathbb{F}_q$ since $[\mathbb{F}_q(\alpha_{\tilde{T}}, \beta_{\tilde{U}}) : \mathbb{F}_q] = d_{\tilde{T}} d_{\tilde{U}}$. Hence the cost of computing $V_{\tilde{U}}$ for all $\tilde{T}$ and $\tilde{U}$ is $\ell^{6+o(1)}$ operations in $\mathbb{F}_q$. Let $d_{V_{\tilde{U}}} = \deg V_{\tilde{U}}$. We can factorize $V_{\tilde{U}}$ over

$\mathbb{F}_q(\alpha_{\tilde{T}}, \beta_{\tilde{U}})$ in

$$(d_{\tilde{T}} d_{\tilde{U}})^{1+o(1)} \Big[d_{V_{\tilde{U}}}^{1.5+o(1)} (\log q)^{1+o(1)} + d_{V_{\tilde{U}}}^{1+o(1)} (\log q)^{2+o(1)} \Big]$$

bit operations. Since $d_{V_{\tilde{U}}} \leq \deg_{\xi_4} G \leq 2$, we have $\sum_{\tilde{T}} \sum_{\tilde{U}} d_{\tilde{T}} d_{U_{\tilde{T}}} d_{V_{\tilde{U}}} \leq 32\ell^4$. Hence the cost of the factorizations of all $V_{\tilde{U}}$ in step 14 is $\ell^{6+o(1)} (\log q)^{1+o(1)} + \ell^{4+o(1)} (\log q)^{2+o(1)}$ bit operations. The costs of the remaining steps are negligible.

Summarizing the above argument, we conclude that the expected cost of Algorithm 2 is $\ell^{6+o(1)} (\log q)^{1+o(1)} + \ell^{4+o(1)} (\log q)^{2+o(1)}$ bit operations.

Next, we consider the cost of Algorithm 1. For $P \in J(\overline{\mathbb{F}_q})$, we denote by $\mathbb{F}_q(P)$ the minimal field of definition for P. Let $S_P = \{P, \pi_q(P), \pi_q^2(P), \dots\}$. Then S_P is finite and $\#S_P = [\mathbb{F}_q(P) : \mathbb{F}_q]$. Let R be a complete set of representatives of $\{S_P \mid P \in J[\ell]\}$. Then the relation (1) holds for all $P \in J[\ell]$ if and only if (1) holds for all $P \in R$. For each $P \in J[\ell]$, the cost of finding ($s_1 \bmod \ell, s_2 \bmod \ell$) satisfying (1) is $O(\ell + \log q)$ operations in $\mathbb{F}_q(P)$. Since $\sum_{P \in R} [\mathbb{F}_q(P) : \mathbb{F}_q] \leq \#J[\ell] = \ell^4$, the cost of step 4 is $O(\ell + \log q) \ell^{4+o(1)} (\log q)^{1+o(1)}$ bit operations.

By the prime number theorem, we have $\ell = O(\log q)$. Hence the bit complexity for each ℓ is $(\log q)^{7+o(1)}$. Since the number of prime numbers ℓ is $O(\log q)$, the bit complexity of the whole algorithm is $(\log q)^{8+o(1)}$. Therefore we have the following theorem.

Theorem 3 *The expected complexity of Algorithm 1 is* $(\log q)^{8+o(1)}$ *bit operations.*

Note that the algorithm of Gaudry and Schost [6] has the same complexity.

5. Using ℓ^n-torsion points

As noted by Gaudry and Schost [6], we can improve the algorithm by using ℓ^n-torsion points. This does not affect its asymptotic complexity.

Let $P_1 \in J[\ell]$. It is sufficient to compute $P_{k+1} \in J(\overline{\mathbb{F}_q})$ satisfying $\ell P_{k+1} = P_k$ for $k = 1, 2, \dots, n-1$. By Theorem 1, we have

$$(\mu_{\ell,1}(\kappa(P_{k+1})) : \cdots : \mu_{\ell,4}(\kappa(P_{k+1}))) = \kappa(P_k). \quad (4)$$

We obtain $\kappa(P_{k+1})$ by solving (4) and $G(\kappa(P_{k+1})) = 0$. The equations can be solved by using resultants.

6. Conclusion

In this paper, we proposed an algorithm to count points on hyperelliptic curves of genus 2 with real models. We also proved that the expected cost of our algorithm has the same order as the algorithm of Gaudry and Schost [6]. It is future work to improve our algorithm in practice by avoiding factorizations and to construct a hyperelliptic curve suitable for cryptography by using our algorithm.

Acknowledgments

The author would like to thank the referee for useful comments. This work was supported in part by JSPS KAKENHI Grant Number JP25800023.

References

[1] T. Satoh, The canonical lift of an ordinary elliptic curve over a finite field and its point counting, J. Ramanujan Math. Soc., **15** (2000), 247–270.
[2] K. S. Kedlaya, Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology, J. Ramanujan Math. Soc., **16** (2001), 323–338.
[3] R. Schoof, Elliptic curves over finite fields and the computation of square roots mod p, Math. Comp., **44** (1985), 483–494.
[4] J. Pila, Frobenius maps of abelian varieties and finding roots of unity in finite fields, Math. Comp., **55** (1990), 745–763.
[5] P. Gaudry and R. Harley, Counting points on hyperelliptic curves over finite fields, in: Proc. of ANTS-IV, W. Bosma ed., LNCS, Vol. 1838, pp. 313–332, Springer-Verlag, Berlin, 2000.
[6] P. Gaudry and É. Schost, Genus 2 point counting over prime fields, J. Symbolic Comput., **47** (2012), 368–400.
[7] D. G. Cantor, On the analogue of the division polynomials for hyperelliptic curves, J. Reine Angew. Math., **447** (1994), 91–145.
[8] M. C. Harrison, An extension of Kedlaya's algorithm for hyperelliptic curves, J. Symbolic Comput., **47** (2012), 89–101.
[9] Y. Uchida, Canonical local heights and multiplication formulas for the Jacobians of curves of genus 2, Acta Arith., **149** (2011), 111–130.
[10] M. J. Jacobson, Jr., R. Scheidler and A. Stein, Cryptographic aspects of real hyperelliptic curves, Tatra Mt. Math. Publ., **47** (2010), 31–65.
[11] J. W. S. Cassels and E. V. Flynn, Prolegomena to a Middlebrow Arithmetic of Curves of Genus 2, London Mathematical Society Lecture Note Series, 230, Cambridge University Press, Cambridge, 1996.
[12] H. Cohen, G. Frey, R. M. Avanzi, C. Doche, T. Lange, K. Nguyen and F. Vercauteren, Handbook of Elliptic and Hyperelliptic Curve Cryptography, Discrete Mathematics and its Applications, Chapman & Hall/CRC, Boca Raton, FL, 2006.
[13] H. Stichtenoth, Algebraic Function Fields and Codes, 2nd ed., Graduate Texts in Mathematics, 254, Springer-Verlag, Berlin, 2009.
[14] K. Matsuo, J. Chao and S. Tsujii, Baby step giant step algorithms in point counting of hyperelliptic curves, IEICE Trans. Fundamentals, **E86-A** (2003), 1127–1134.
[15] J. von zur Gathen and J. Gerhard, Modern Computer Algebra, 3rd ed., Cambridge University Press, Cambridge, 2013.
[16] J. van der Hoeven and É. Schost, Multi-point evaluation in higher dimensions, Appl. Algebra Engrg. Comm. Comput., **24** (2013), 37–52.
[17] K. S. Kedlaya and C. Umans, Fast polynomial factorization and modular composition, SIAM J. Comput., **40** (2011), 1767–1802.
[18] Y. Bouzidi, S. Lazard, G. Moroza, M. Pouget, F. Rouillier and M. Sagraloff, Solving bivariate systems using rational univariate representations, J. Complexity, **37** (2016) 34–75.