U.P.B. Sci. Bull., Series A, Vol. 85, Iss. 3, 2023 ISSN 1223-7027

HEEGNER POINTS, FRICKE INVOLUTION AND ALGORITHMS COMPLEXITY

Radu Gaba[1], Vladimir Olteanu[2]

This paper deals with the development of faster computer programs by mean of which the authors improve the complexity orders of the algorithms of [1] used by Cânepă and Gaba to classify the fixed points of the action of Fricke's involution w_n on the open modular curves $Y_0(n)$ as well as a certain class of Heegner points: the pairs $(E, E/C)$, where E are complex elliptic curves for which there exist cyclic subgroups $C \leq (E, +)$ of order n such that the elliptic curves E and E/C are isomorphic. We compute the complexity orders of the algorithms of [1] as well as the complexity orders of the new algorithms. Moreover, we provide an exhaustive comparison of the results obtained upon running the code on the same computer.

2000 Mathematics subject classification: Primary: 11G07, 11G15, 11Y16, Secondary: 14D22

Keywords: *elliptic curve, Fricke involution, algorithm, cyclic subgroup*

1. **Introduction**

Throughout this paper we will denote by $\mathcal{H}$ the upper half plane, $\mathcal{H} := \{z \in \mathbb{C}, \mathrm{Im}(z) > 0\}$ and by $\mathcal{F} := \left\{z = x + iy \in \mathbb{C} : -\frac{1}{2} \leq x < \frac{1}{2} \text{ and either } |z| \geq 1 \text{ if } x \leq 0 \text{ or } |z| > 1 \text{ if } x > 0\right\}$ the fundamental domain for the action of $SL_2(\mathbb{Z})$ on $\mathcal{H}$, action given by $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \tau = \frac{a\tau+b}{c\tau+d}, \tau \in \mathcal{H}, \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbb{SL}_2(\mathbb{Z})$. $SL_2(\mathbb{Z})\backslash\mathcal{H}$ is identified under this action with isomorphic classes of elliptic curves over $\mathbb{C}$. Let E be a complex elliptic curve given by the lattice L, $E = \mathbb{C}/L$, and C a cyclic subgroup of order $n < \infty$ of $(E, +)$. That is, C is a subgroup of order n of the n-torsion subgroup of E: $E[n] := \{P \in E : [n]P = O\} = \ker([n] : E \to E) = (\frac{1}{n}L)/L \subset \mathbb{C}/L$. The group E/C has a structure of Riemann variety due to the fact that C acts effectively and properly discontinuous on E and this structure is compatible with the natural projection denoted by $\pi : E \to E/C$. It is known that the isogeny π is unramified of degree n: $\deg\pi = |\pi^{-1}(O)| = |C| = n$ (see [7], Theorem 3.4). By $Y_0(n)$ one denotes the open modular curve defined as the quotient space $\Gamma_0(n)/\mathcal{H}$, equivalently $Y_0(n)$ is the set of orbits $\{\Gamma_0(n)\tau : \tau \in \mathcal{H}\}$, where $\Gamma_0(n)$ is the "Nebentypus" congruence subgroup of level n of $SL_2(\mathbb{Z})$, which acts on $\mathcal{H}$ from the left:

$$\Gamma_0(n) = \{\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbb{SL}_2(\mathbb{Z}) | \; c \equiv 0 (\mathrm{mod} n)\}.$$

An enhanced elliptic curve for $\Gamma_0(n)$ is by definition an ordered pair (E, C) where E is a complex elliptic curve and C is a cyclic subgroup of order n of E. Moreover, two pairs (E, C) and (E', C') are equivalent if there is some isomorphism $E \cong E'$ taking C to C'. The set of such equivalence classes is denoted by:

[1]Institute of Mathematics "Simion Stoilow" of the Romanian Academy, P.O. BOX 1-764 RO-014700 Bucharest e-mail: radu.gaba@imar.ro

[2] e-mail: vladimir.olteanu@proton.me

$$S_0(n) := \{\text{enhanced elliptic curves for } \Gamma_0(n)\}/\sim.$$

Moreover, an element of $S_0(n)$ is an equivalence class $[E, C]$ and $S_0(n)$ is a moduli space of isomorphism classes of complex elliptic curves and n-torsion data (see [4] for details).

Denote by Λ_τ the lattice $\mathbb{Z} + \mathbb{Z}\tau$, $\tau \in \mathcal{H}$ and by E_τ the elliptic curve $\mathbb{C}/\Lambda_\tau$. One has the following bijection (see [4], Theorem 1.5.1 for details):

$$S_0(n) \cong Y_0(n) \text{ given by } [\mathbb{C}/\Lambda_\tau, \langle 1/n + \Lambda_\tau\rangle] \mapsto \Gamma_0(n)\tau\ .$$

In this paper we develop faster algorithms than the ones previously developped by Cânepă and Gaba in [1] in order to classify on one hand the fixed points of the action of Fricke's involution on $Y_0(n)$ and a certain class of Heegner points namely: the pairs $(E, E/C)$, where E are complex elliptic curves for which there exist cyclic subgroups $C \leq (E, +)$ of order n such that the elliptic curves E and E/C are isomorphic. We compute the complexity orders of the old as well as of the new algorithms and provide a thorough comparison of the results obtained when running them on the same laptop. We point out that in [3] one improved the noncyclic case algorithm whereas in this paper we analyze the complexity and improve the algorithm used to classify the fixed points of the action of Fricke's involution w_n on $Y_0(n)$ as well as the cyclic case algorithm since it is natural to cover these remaining and different cases as well. As expected, the complexity order of the cyclic case algorithm is different than the complexity order of the noncyclic case algorithm.

The improved versions of those algorithms, developped in this paper, work correctly and faster in finding the fixed points of the action of the Fricke involution on $Y_0(n)$, points which though known, weren't studied in the above specified manner (see [1]). Note that this number of fixed points was previously computed by Ogg (see [8], Proposition 3) and Kenku (see [6], Theorem 2) and, for $n > 3$, it is $\nu(n) = h(-n) + h(-4n)$ if $n \equiv 3(\text{mod}4)$ and $\nu(n) = h(-4n)$ otherwise, where $h(-n)$ is the class number of primitive quadratic forms of discriminant $-n$ and $\nu(2) = \nu(3) = 2$.

2. **Preliminaries**

In [1], one provided a new method of classifying the fixed points of the action of the Fricke involution

$$w_n := \begin{pmatrix} 0 & -1 \\ n & 0 \end{pmatrix} \in GL_2(\mathbb{Q}^+)$$

on the open modular curves $Y_0(n)$. One firstly characterized the pairs $(E, E/C)$, where E are complex elliptic curves for which there exist cyclic subgroups $C \leq (E, +)$ of order n such that the elliptic curves E and E/C are isomorphic in Theorem 2.1. Next, upon imposing certain conditions (Theorem 2.3 of [1]), one also answered the question: "given a complex elliptic curve E, when can one find a cyclic subgroup of order n of E such that $(E, C) \simeq (E/C, E[n]/C)$.

More precisely, in [1] Cânepă and Gaba proved the following theorems:

Theorem 2.1. *([1], Theorem 1.1)*

Let E be a complex elliptic curve determined by the lattice $\langle 1, \tau\rangle$, $\tau \in \mathcal{H}$. Then:

i) $\exists C \leq (E, +)$ finite cyclic subgroup such that $\frac{E}{C} \simeq E \Leftrightarrow \exists u, v \in \mathbb{Q}$ such that $\tau^2 = u\tau + v$ with $\Delta = u^2 + 4v < 0$ (i.e. E admits complex multiplication);

ii) If τ satisfies the conditions of i) and $u = \frac{u_1}{u_2}, v = \frac{v_1}{v_2}, u_2 \neq 0, v_2 \neq 0, u_1, u_2, v_1, v_2 \in \mathbb{Z}, \mathrm{Gcd}(u_1, u_2) = \mathrm{Gcd}(v_1, v_2) = 1, d_2 = \mathrm{Gcd}(u_2, v_2)$, then:

$\exists C \leq (E, +)$ cyclic subgroup of order n which satisfies $\frac{E}{C} \simeq E \Longleftrightarrow \exists (a, b') \in \mathbb{Z}^2$ with $\mathrm{Gcd}(a, b') = 1$ such that $n = \det M$, where M is the matrix

$$M = \begin{pmatrix} a & A \\ b & B \end{pmatrix}$$

and $(a, A, b, B) = \left(a, \frac{u_2 v_1}{d_2} b', \frac{u_2 v_2}{d_2} b', a + \frac{u_1 v_2}{d_2} b'\right)$*;*

iii) The subgroup C *from ii) is* $C = \langle \frac{u_{11} + u_{21}\tau}{n} \rangle$*, where* u_{11}, u_{21} *are obtained in the following way: since* $\det M = n$ *and* $\mathrm{Gcd}(a, A, b, B) = 1$ *(one deduces easily this), the matrix* M *is arithmetically equivalent with the matrix:*

$$M \sim \begin{pmatrix} 1 & 0 \\ 0 & n \end{pmatrix},$$

hence

$$\exists U, V \in GL_2(\mathbb{Z}) \quad \textit{such that} \quad M = U \cdot \begin{pmatrix} 1 & 0 \\ 0 & n \end{pmatrix} \cdot V.$$

The elements u_{11}, u_{21} *are the first column of the matrix*

$$U = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix}.$$

Theorem 2.2. *([1], Theorem 2.3) Let* E *be an elliptic curve defined over* $\mathbb{C}$ *satisfying the conditions of 2.1 i). Then the following are equivalent:*

$$\{\exists C \leq (E, +) \textit{ cyclic subgroup of order } n \textit{ of } E \textit{ such that } (E, C) \sim (\tfrac{E}{C}, \tfrac{E[n]}{C})\} \iff$$
$$\{\exists (a, b') \in \mathbb{Z}^2 \textit{ with } \gcd(a, b') = 1 \textit{ such that } \det(M) = n \textit{ and } n | \mathrm{Tr}(M)\} \iff$$
$$\{\exists (a, b') \in \mathbb{Z}^2, \textit{ with } \gcd(a, b') = 1 \textit{ such that } \det(M) = n \textit{ and } M^2 \equiv O_2 (\mathrm{mod} n)\},$$

where M *is the matrix from 2.1 ii) and* $\mathrm{Tr}(M)$ *the trace of* M*.*

This number of fixed points of the action of Fricke's involution w_n on $Y_0(n)$ is $\nu(n) = h(-n) + h(-4n)$ if $n \equiv 3 (\mathrm{mod} 4)$ for $n > 3$ and $\nu(n) = h(-4n)$ otherwise. One can also obtain this number by using the second algorithm of [1]. In [1] Cânepă and Gaba also developped the algorithms classifying these points and implemented them in Magma. The non-cyclic case has been studied by them in [2].

We briefly recall now this algorithm developed in [1] for the classification of the fixed points of the Fricke's involution action.

The complete details can be found in [1] (pages 496-498). Note that we keep the same notations. Moreovever, we provide here sufficient details to make the exposure clear enough while following [1]. It is known that the complex elliptic curves are of the form $\frac{\mathbb{C}}{L}$ for some $L = \mathbb{Z} + \mathbb{Z}\tau \subset \mathbb{C}$ where $\tau \in \mathcal{F} = \Big\{ z = x + iy \in \mathbb{C} : -\frac{1}{2} \leq x < \frac{1}{2}$ and either $|z| \geq 1$ if $x \leq 0$ or $|z| > 1$ if $x > 0 \Big\}$.

If E is an elliptic curve satisfying the condition i) of Theorem 2.1, one can assume (up to isomorphism) that E is of the form $\frac{\mathbb{C}}{L}$ with $L = \mathbb{Z} + \mathbb{Z}\tau \subset \mathbb{C}$ and $\tau \in \mathcal{F}$. If $\tau^2 - u\tau - v = 0, u, v \in \mathbb{Q}, \Delta = u^2 + 4v < 0$ and $\tau \in \mathcal{F}$, then one further obtains $\tau = \frac{u \pm i\sqrt{|\Delta|}}{2}$, $-1 \leq u < 1$ and $|\Delta| \geq 3$.

Since $\Delta = u^2 + 4v < 0$, one has that $v < 0$. Moreover, one can assume without loss of generality that $v_2 > 0, v_1 < 0$ and $u_2 > 0$. Theorem 2.1, ii) leads to:

$$n = aB - bA = \Big(a + \frac{u_1 v_2}{2d} b'\Big)^2 - \frac{u_2^2 v_2^2 \Delta}{4d^2} b'^2 \quad (*) \tag{1}$$

Furthermore $d = \mathrm{Gcd}(u_2, v_2)$, $\Delta = \frac{u_1^2}{u_2^2} + 4\frac{v_1}{v_2}$ and let $u_2' := u_2/d$ and $v_2' := v_2/d$. Let also $v_1 := -v_1$ and remark that $u_2, v_2, v_1 > 0$ and also that $\Delta \leq -3$. One multiplies $(*)$ by 4 and hence obtain:

$$4n = (2a + u_1 v_2' b')^2 + v_2' b'^2 (4 v_1 d u_2'^2 - v_2' u_1^2) \tag{2}$$

This leads to the inequality $4n \geq v_2' b'^2 \cdot 4 v_1 d u_2'^2$ hence $n \geq v_2' b'^2 \cdot v_1 d u_2'^2$. Next denote by $\xi := 4 v_1 d u_2'^2 - v_2' u_1^2$ and note that $\Delta \leq -3$ is equivalent to $\frac{u_1^2}{d^2 u_2'^2} - 4\frac{v_1}{d v_2'} \leq -3$ and moreover to $u_1^2 v_2' - 4dv_1 u_2'^2 \leq -3d^2 u_2'^2 v_2'$ that is $-\xi \leq -3d^2 u_2'^2 v_2'$, i.e. $\xi \geq 3d^2 u_2'^2 v_2'$ $(**)$. From $(**)$ and (2) one gets that $4n \geq v_2' b'^2 \cdot 3d^2 u_2'^2 v_2'$ hence $4n/3 \geq v_2'^2 b'^2 \cdot d^2 u_2'^2$. Let $k := \sqrt{4n/3}$. One obtains next that u_2' runs from 1 to the integer part of k, $[k]$, v_2' from 1 to $[k/u_2']$, b' from 1 to $[k/u_2'/v_2']$ and d from 1 to $[k/u_2'/v_2'/b']$. Moreover, $-1/2 \leq \mathrm{Re}(\tau) < 1/2$ is equivalent to $-1/2 \leq u_1/(2u_2) < 1/2$ that is $-u_2 \leq u_1 < u_2$. Consequently u_1 runs from $-du_2'$ to $du_2' - 1$. Let $m := (2a + u_1 v_2' b')^2$. From (2) one obtains now that $4n + v_2'^2 b'^2 u_1^2 = m + 4v_2' b'^2 v_1 d u_2'^2 \geq 4v_2' b'^2 v_1 d u_2'^2$ and hence $v_1 \leq \frac{4n + v_2'^2 b'^2 u_1^2}{4v_2' b'^2 d u_2'^2}$. Consequently, v_1 will run from 1 to $[\frac{4n + v_2'^2 b'^2 u_1^2}{4v_2' b'^2 d u_2'^2}]$.

Finally, one has to make sure that the condition $\tau \in \mathcal{F}$ is entirely fulfilled by setting: ($u_1 > 0$ or $v_1 \geq v_2$) and ($u_1 \leq 0$ or $v_1 > v_2$). The cyclic case algorithm of [1] is therefore Algorithm 1 (see [1], page 498). Throughout the code, the substitutions made are $b := b'$, $u_2 := u_2/d$ and $v_2 := v_2/d$, where $d = \mathrm{Gcd}(u_2, v_2)$ and b', u_2, v_2 are defined in Theorem 2.1.

In Algorithm 1, one made the substitutions $b := b'$, $u_2 := u_2/d$ and $v_2 := v_2/d$, where $d = \mathrm{Gcd}(u_2, v_2)$ and b', u_2, v_2 are defined in Theorem 2.1. After modifying the previous code one computes the fixed points of the Fricke involution by adding a few conditions. That is, by using Theorem 2.2, one has that $\det(M) = n$ and $n | \mathrm{Tr}(M)$ or equivalently: $a^2 + au_1 v_2 b + u_2 v_2 d \cdot bu_2 v_1 b = n$ and $n | (2a + u_1 v_2 b)$. Algorithm 2 (fixed points of Fricke's involution) is obtained by inserting these two conditions in the cyclic case algorithm.

After modifying the first code by using the notations of Theorem 2.2, [1] obtained the second agorithm namely Algorithm 2 (see [1], page 499). Throughout the next section we will also compute their order of complexity and improve the algorithms.

3. Main Results

Note that the isomorphism $(\frac{E}{C} \simeq E)$ can only occur for non-singular projective curves of genus 1 (see for example [3], Lemma 1).

We are ready to compute now the complexity orders of the algorithms of [1].

Theorem 3.1. *The order of complexity of Algorithm 1 (C cyclic) is* $O(n^3 \cdot \log^2 \mathrm{n})$.

Proof. Line 2 is $O(k)$ time. One has that $O(\mathrm{Floor(k)}) = O(k)$, hence line 3 is $O(k)$ iterations. However, since $\sum_{u2=1}^{k} (k/u2) \to \mathrm{k} \cdot \ln(\mathrm{k})$, we obtain that lines 3 and 4 combined are $O(\mathrm{k} \cdot \log(\mathrm{k}))$ iterations (for this, note that $\sum_{u=1}^{k} (1/u) - \ln(\mathrm{k}) \to \gamma \approx 0.57$ hence $O(\sum_{u=1}^{k} (1/u)) = O(\ln(\mathrm{k})))$. Since Gcd is $O(\log(\mathrm{k}))$ time, one obtains that line 5 is $O(\log(\mathrm{k}))$ time. Since $(k/u_2/v_2)_{\max} = k$, it follows that Line 6 is $O(k)$ time (note that $k/u_2/v_2 = \frac{k}{u_2 \cdot v_2}$). Remark that $\sum_{b=1}^{k} (k/b) \to \mathrm{k} \ln(\mathrm{k})$, hence lines 6 and 7 combined are $O(\mathrm{k} \cdot \log(\mathrm{k}))$ iterations. Note that $d_{\max} = k$ for $u_2 = 1$ hence line 8 is $O(2 \cdot k)$ iterations, which is $O(k)$ iterations. Recall now that $n = 3k^2/4$. Line 9 is $O(\log(k^2)) = O(2 \cdot \log(k)) = O(\log(k))$ iterations. Observe that $(v_1)_{\max} = (4 \cdot n + v_2^2 \cdot u_1^2 \cdot b^2)/(4 \cdot v_2 \cdot b^2 \cdot d \cdot u_2^2) = 3k^2/(4 \cdot v_2 \cdot b^2 \cdot d \cdot u_2^2) + v_2^2 \cdot u_1^2 \cdot b^2/(4 \cdot v_2 \cdot b^2 \cdot d \cdot u_2^2) = 3k^2/(4 \cdot v_2 \cdot b^2 \cdot d \cdot u_2^2) + v_2 \cdot u_1^2/(4 \cdot d \cdot u_2^2)$. Consequently line 10 is $O(k^2)$ iterations. Line 11 is $O(\log(k^2)) = O(2 \cdot \log(k)) = O(\log(k))$ iterations. Remark that $O(\text{if } \mathrm{c}_1 \text{ then } \mathrm{c}_2 \text{ else } \mathrm{c}_3)$ is $O(\mathrm{c}_1) + \mathrm{Max}(O(\mathrm{c}_2), O(\mathrm{c}_3))$, which is $\mathrm{Max}(O(\mathrm{c}_1), O(\mathrm{c}_2), O(\mathrm{c}_3))$. It follows that line 12 is $O(1)$, line 13 is $O(1)$. IsSquare function is $O(\mathrm{sqrt})$ and since from line 13 one has that $m \leq 3k^2$, it follows that line 14 is $O(\mathrm{sqrt(k^2)}) = O(k)$. Line 15 is $O(1)$. Line 16 is $O(1)$ since *IsEven* is $O(1)$. Line 17 is $O(1)$. Remark that from line 17 one obtains $a < k \cdot \mathrm{sqrt}(3)/2 < k$. Since $b \leq k^2$ one obtains that $\mathrm{Gcd(a, b)} < \mathrm{k}$. Consequently one obtains that line 18 is $O(\log(\mathrm{k}))$. The remaining lines of the algorithm are $O(1)$. Using now the above, the order of complexity of the algorithm is $O(k + k \cdot \log(\mathrm{k})(\log(\mathrm{k}) + (k \cdot \log(\mathrm{k}) \cdot k \cdot (\log(\mathrm{k}) + k^2 \cdot (\log(\mathrm{k}) + k + \log(\mathrm{k}))))) = O(k + k \cdot \log(\mathrm{k})(\log(\mathrm{k}) + (k^4 \log(\mathrm{k}) \cdot (k + \log(\mathrm{k}))))) = O(k + k^5 \log^2(\mathrm{k}) \cdot (k + \log(\mathrm{k})))) = O(k + k^6 \log^2(\mathrm{k})) = O(k^6 \log^2(\mathrm{k})) = O(n^3 \cdot \log^2(\sqrt{\mathrm{n}})) = O(n^3 \cdot \log^2 \mathrm{n})$. □

Theorem 3.2. *The order of complexity of the Algorithm 2 (fixed points of Fricke's involution) is* $O(n^3 \cdot \log^2 \mathrm{n})$.

Proof. We compare it to the first algorithm and remark that the difference is given by Line 20 in which one imposes two extra conditions both being in $O(1)$. Consequently Line 20 is still $O(1)$ iterations.

□

We are ready to improve the two algorithms. For this, we introduce two helper classes and reimplement the algorithms in C++ . The source code is available upon request. The first class is called *GCDs*. It computes *Gcd*s using dynamic programming. It holds a $(k+1)\times(n+1)$ matrix which stores all *Gcd*s once they are computed. Computing the *Gcd*s for all pairs (i,j) with $i\leq k$ and $j\leq n$ is done in $O(k*n)$ time. Moreover the lookup cost is $O(1)$ once a *Gcd* has been already computed. Consequently, the amortized cost of all *GCDs* :: *gcd* calls is $O(k*n)=O(k^3)$. The second class is called *Squares*. This class features a single method called $sqrtIfPerfectSq$, which returns the square root of a number if the respective number is a perfect square and -1 otherwise. This class holds a $4*n+1$ size vector, initialized with -1; Next, for each i such that $i^2<4*n+1$, we populate the vector at index i^2 with i's. The instantiation of the *Squares* class is done in $O(4*n+1)=O(k^2)$ time. The calls to $sqrtIfPerfectSq$ are vector lookups, and hence in $O(1)$. We obtain below:

Theorem 3.3. *The order of complexity of the improved version of Algorithm 1 (case C cyclic) is* $O(n^2\sqrt{n}\cdot\mathrm{log}^2\mathrm{n})$.

Proof. In the improved version of Algorithm 1 the *Gcd* and $IsSquare$ are replaced by the classes *GCDs* :: *gcd* and *Squares* :: $sqrtIfPerfectSq$ respectively as oposed to the non-optimized version. The key point is that in this manner we front-load the costs, and then treat all subsequent calls as being $O(1)$. The operations concerning *GCDs* are in $O(k^3)$, and the operations concerning *Squares* are in $O(k^2)$. Analysing now the improved algorithm: line 2 is in $O(k)$ as in Algorithm 1. Lines 3 and 4 are $O(\mathrm{k}\cdot\mathrm{log(k)})$ iterations. Lines 6 and 7 are also $O(\mathrm{k}\cdot\mathrm{log(k)})$ iterations. Line 8 is $O(k)$ iterations. Line 9 is $O(1)$ as subsequent call of *GCD*. Line 10 is $O(k^2)$ iterations. The remaining lines are $O(1)$. We obtain that the order of complexity is $O(k^3)+O(k^2)+O(k)+O(\mathrm{k}\cdot\mathrm{log(k)})\cdot\mathrm{O(k\cdot log(k))\cdot O(k)(O(1)+O(k^2))} = \mathrm{O(k^3)+O(k^5\cdot log^2k)=O(k^5\cdot log^2k)=O(n^2\sqrt{n}\cdot log^2n))}$.

□

Similarly we obtain Theorem 3.4 below:

Theorem 3.4. *The order of complexity of the improved version of Algorithm 2 (fixed points of the Fricke involution) is* $O(n^2\sqrt{n}\cdot\mathrm{log}^2\mathrm{n})$.

Proof. The Algorithm 2 is derived from Algorithm 1 by restricting solutions to the case when $a^2+au_1v_2b+u_2v_2d\cdot bu_2v_1b=n$ and $n|(2a+u_1v_2b)$ in the innermost loop. This check is however in $O(1)$. Consequently, the order of complexity will be the same as the one obtained for the optimized version of Algorithm 1. This ends the proof.

□

The main function of the second improved algorithm namely Algorithm 2 (Fricke's involution fixed points) will contain the code below (the complete C++ code is available upon request). Deleting the lines 33 and 34 will give us Algorithm 1 (cyclic case).

4. **Examples**

We provide now several examples of the fixed points of the Fricke's involution as well as the numbers of classes of CM elliptic curves E which admit cyclic subgroups C of order n such that $E\cong E/C$. The examples are gathered in Table 1 and Table 2. The extended set of values also contains the set provided in [1]. However, the runs were made on a different

Algorithm 3.1 Fricke's involution fixed points in C++ Input: any random integer number; Output: $u_1/u_2, v_1/v_2$.

```
list⟨pair⟨double, double⟩⟩ fricke(int n)
{
list⟨pair⟨double, double⟩⟩ result;
int $k = \mathrm{sqrtInt}(4 * n/3)$;
GCDs gcds(k + 1, n + 1);
Squares squares(4 ∗ n + 1);
for(int u2 = 1; u2 <= k; u2 + +) {
    for(int v2 = 1; v2 <= k/u2; v2 + +) {
        if (gcds.gcd(u2, v2)! = 1)
        continue;
        for(int b = 1; u2 <= k/u2/v2; b + +) {
            for(int d = 1; u2 <= k/u2/v2/b; d + +) {
                for(int u1 = −d ∗ u2; u1 <= d ∗ u2 − 1; u1 + +) {
                if (gcds.gcd(u1, d ∗ u2)! = 1)
                continue;
                  int v1max = (4 ∗ n + sq(v2) ∗ sq(u1) ∗ sq(b))/(4 ∗ v2 ∗ sq(b) ∗ d ∗ sq(u2));

                    for(int v1 = 1; v1 <= v1max; v1 + +) {
                    if (gcds.gcd(v1, d ∗ v2)! = 1)
                      continue;
                        if (sq(u1)/sq(d)/(u2 ∗ u2) − 4 ∗ v1/d/v2 > −3);
                            continue;
                        int m = 4 ∗ n − v2 ∗ sq(b) ∗ (4 ∗ v1 ∗ d ∗ sq(u2) − v2 ∗ sq(u1));
                        int y = squares.sqrtIfPerfectSq(m);
                        if (y == −1)
                            continue;
                        if ((y − u1 ∗ v2 ∗ b)%2! = 0)
                            continue;
                        int a = (y − u1 ∗ v2 ∗ b)/2;
                        gcdAB = gcds.gcd(a, b);
                        if (gcdAB! = 1)
                            continue;
                        if ((u1 > 0||v1 >= d ∗ v2)&&(u1 <= 0||v1 > d ∗ v2))
                        if (((a ∗ a + a ∗ u1 ∗ v2 ∗ b + u2 ∗ v2 ∗ d ∗ b ∗ u2 ∗ v1 ∗ b) == n)
                                                    &&((2 ∗ a + u1 ∗ v2 ∗ b)%n == 0)))

                            result.push_back({1.0 ∗ u1/d/u2, −1.0 ∗ v1/d/v2});
                    }
                }
            }
        }
    }
}
return result;
}
```

n	the cyclic case	old CPU time	new CPU time
2	3	0.001s	0.001s
39	54	0.016s	0.001s
101	108	0.016s	0.002s
1457	1498	0.811s	0.036s
2012	3039	1.326s	0.042s
2022	4055	3.167s	0.047s
2023	2425	2.995s	0.052s
5772	12558	6.755s	0.251s
12383	14856	21.403s	0.724s
124071	165782	804.918s	20.993s
253124	443302	2134.828s	38.011s

Table 1. Number of classes of complex elliptic curves, the cyclic case: computations for various n

n	fixed points of w_n	old CPU time	new CPU time
2	2	0.001s	0.001s
39	8	0.016s	0.001s
101	14	0.016s	0.002s
1457	24	0.811s	0.043s
2012	42	1.326s	0.050s
2022	24	3.026s	0.057s
2023	36	2.948s	0.062s
5772	48	6.396s	0.244s
12383	184	20.733s	0.747s
124071	708	703.689s	24.990s
253124	456	2078.292s	44.340s

Table 2. Number of fixed points of Fricke's involution

machine than the one used in [1] namely the computations were done using Magma 2.19-9 as well as C++ on the same Lenovo i3-3110M laptop at 2.40 GHz and 8 GB RAM. For each n we have also recorded the CPU time the processor took to complete these calculations with the old code (written in Magma) as well as with the new one (written in C++).

5. **Conclusions**

In this paper we improve the orders of complexity of the algorithms of [1] and provide examples as well as an extended comparison between the old CPU time recorded and the new one. The new code implemented in C++ is available upon request and can be further used in problems involving the Fricke involution in the theory of complex elliptic curves. Moreover, we also emphasize here the utility of studying elliptic curve quotients.

REFERENCES

[1] B. Canepa, R. Gaba, *On some special classes of complex elliptic curves and related algo- rithms*, Mathematical Reports 16(66), 4 (2014), 477-502.

[2] B. Canepa, R. Gaba, *A generalization of a fixed point theorem for CM elliptic curves*, U.P.B. Sci. Bull., Series A (2019), 3-12.

[3] B. Canepa, R. Gaba, V. Olteanu, *CM complex elliptic curves and algorithms comple- xity*, U.P.B. Sci. Bull., Series A (2022), 123-130.

[4] F. Diamond, J. Shurman, *A first course in modular forms*, Graduate Texts in Mathematics, volume 228, Springer, New-York, 2005.
[5] D. Husemoeller, *Elliptic curves*, Graduate Texts in Mathematics, volume 111, Springer, New-York, 2004.
[6] M. A. Kenku, *Atkin-Lehner involutions and class number residuality*, Acta Arithmetica, 23 (1977), 1-9.
[7] R. Miranda, *Algebraic curves and Riemann surfaces*, Graduate Studies in Mathematics, volume 5, AMS.
[8] A. P. Ogg, *Hyperelliptic modular curves*, Bulletin de la S.M.F. (1974), 449-462.