MATHEMATICS OF COMPUTATION
Volume 00, Number 0, Pages 000–000
S 0025-5718(XX)0000-0

IMPROVED COMPUTATION OF FUNDAMENTAL DOMAINS FOR ARITHMETIC FUCHSIAN GROUPS

JAMES RICKARDS

Abstract. A practical algorithm to compute the fundamental domain of an arithmetic Fuchsian group was given by Voight, and implemented in Magma. It was later expanded by Page to the case of arithmetic Kleinian groups. We combine and improve on parts of both algorithms to produce a more efficient algorithm for arithmetic Fuchsian groups. This algorithm is implemented in PARI/GP, and we demonstrate the improvements by comparing running times versus the live Magma implementation.

1. Introduction

Let Γ be a discrete subgroup of $\mathrm{PSL}(2,\mathbb{R})$, which acts on the hyperbolic upper half plane $\mathbb{H}$. Assume that the quotient space $\Gamma\backslash\mathbb{H}$ has finite hyperbolic area $\mu(\Gamma)$, and denote the hyperbolic distance function on $\mathbb{H}$ by d. Let $p \in \mathbb{H}$ have trivial stabilizer under the action of Γ. Then the space

$$D(p) := \{z \in \mathbb{H} : d(z,p) \leq d(gz,p) \text{ for all } g \in \Gamma\}$$

forms a fundamental domain for $\Gamma\backslash\mathbb{H}$, and is known as a Dirichlet domain. It is a connected region whose boundary is a closed hyperbolic polygon with finitely many sides. Two examples of Dirichlet domains are given in Figure 1; the boundaries of the domains are in green, interiors in grey, and they are displayed in the unit disc model of hyperbolic space.

Explicitly computing fundamental domains has many applications, including:

- Computing a presentation for Γ with a minimal set of generators (Theorem 5.1 of [Voi09]);
- Solving the word problem with respect to this set of generators (Algorithm 4.3 of [Voi09]);
- Computing Hilbert modular forms ([DV13]);
- Efficiently computing the intersection number of pairs of closed geodesics ([Ric21]).

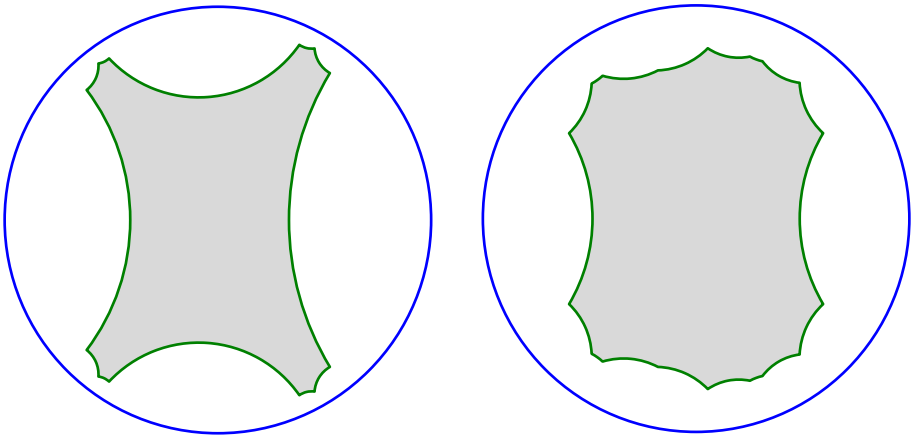
In [Voi09], John Voight published an algorithm to compute $D(p)$. The algorithm has two main parts:

2020 *Mathematics Subject Classification.* Primary 11Y40; Secondary 11F06, 20H10, 11R52.

Key words and phrases. Shimura curve, fundamental domain, quaternion algebra, algorithm.

I thank John Voight and Aurel Page for their useful discussions and comments. I also thank Bill Allombert for his help and suggestions with PARI/GP, and the anonymous reviewers for their helpful comments. This research was supported by an NSERC Vanier Scholarship at McGill University. The author is currently partially supported by NSF-CAREER CNS-1652238 (PI Katherine E. Stange).

©XXXX American Mathematical Society



(A) $F = \mathbb{Q}$, $\mathfrak{D} = 21$. (B) $F = \mathbb{Q}(\sqrt{5})$, $\mathrm{Nm}_{F/\mathbb{Q}}(\mathfrak{D}) = 61$.

FIGURE 1. Dirichlet domains corresponding to norm 1 unit groups of maximal orders in a quaternion algebra of discriminant $\mathfrak{D}$ over a number field F.

- Element enumeration: algebraic algorithms to produce non-trivial elements of Γ, which are added to a set G. This step is given for *arithmetic Fuchsian groups*, which are commensurable with unit groups of maximal orders in a quaternion algebra over a totally real number field that is split at exactly one real place.
- Geometry: geometric algorithms used to compute the fundamental domain of $\langle G \rangle$. This step is valid for all Fuchsian groups Γ.

The process terminates once the hyperbolic area of $\langle G \rangle \backslash \mathbb{H}$ equals $\mu(\Gamma)$ (computed with an explicit formula).

These algorithms were implemented in Magma [BCP97], and do a reasonable job with small cases, but do not scale very well. An improvement to the enumeration algorithms was given by Page in [Pag15]: he replaces the deterministic element generation by a probabilistic algorithm, which in practice performs significantly better. He also generalized the setting from arithmetic Fuchsian groups to arithmetic Kleinian groups, and has a Magma implementation available from his website.

In this paper, we aim to further improve the computation by improving the geometric part (as described by Voight), and refining the element enumeration (as described by Page). These algorithms have been implemented by the author in PARI/GP ([PAR22]), and are publicly available in a GitHub repository ([Ric22]). Sample running times comparing the live Magma implementation and the PARI implementation are found in Table 1 (all computations were run on the same McGill University server). The degree and discriminant of the base number field F, norm to $\mathbb{Q}$ of the discriminant of the algebra, area, number of sides, and running times are recorded.

In Section 2, we detail the geometric portion of the algorithm, and compare the theoretical running time with Voight's paper. Taking element generation as an oracle, we give the general algorithm to compute the fundamental domain in Section

TABLE 1. Running times (in seconds) of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	Sides	t(MAGMA)	t(PARI)	$\frac{\text{t(MAGMA)}}{\text{t(PARI)}}$
1	1	33	20.943	17	13.190	0.022	599.5
1	1	793	753.982	640	15727.170	1.718	9154.3
2	33	37	226.195	222	297.750	0.946	314.7
2	44	79	571.770	552	4182.640	3.142	1331.2
3	473	99	418.879	406	104146.830	4.382	23767.0
4	14656	17	469.145	454	2487.800	12.107	205.5
5	5763833	1	4490.383	4294	2746313.540	1242.494	2210.3
7	20134393	119	1507.964	1446	2236865.680	1234.850	1811.4

3. Section 4 specializes Page's enumeration to our setting, and investigates optimal selection of required constants. Finally, Section 5 gives more sample running times over a range of $\mu(\Gamma)$ for $\deg(F) \leq 4$.

2. Geometry

Instead of working in the upper half plane, it is better to work with the unit disc model, $\mathbb{D}$ (final results can be transferred back if desired). To this end, fix a $p \in \mathbb{H}$ which has trivial stabilizer under the action of Γ. Consider the map $\phi : \mathbb{H} \to \mathbb{D}$ given by

$$\phi(z) = \frac{z-p}{z-\overline{p}},$$

which is the conformal equivalence between $\mathbb{H}$ and $\mathbb{D}$ that sends p to 0. The group Γ now acts on $\mathbb{D}$ via $\Gamma^\phi := \phi\Gamma\phi^{-1} \subseteq \mathrm{PSU}(1,1)$.

2.1. Normalized boundary.

Definition 2.1. For a non-identity element $g = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma^\phi$, define the *isometric circle* of g to be

$$I(g) := \{z \in \mathbb{D} : |cz + d| = 1\}.$$

As the point p has trivial stabilizer under Γ, $c \neq 0$, hence $I(g)$ is an arc of the circle with radius $\frac{1}{|c|}$ and centre $\frac{-d}{c}$. By convention, the arc runs counterclockwise from the initial point to the terminal point. Furthermore, define

$$\mathrm{int}(I(g)) := \{z \in \mathbb{D} : |cz + d| < 1\}, \qquad \mathrm{ext}(I(g)) := \{z \in \mathbb{D} : |cz + d| > 1\},$$

to be the interior and exterior of $I(g)$ respectively. See Figure 2 for an example.

The isometric circle has the following alternate characterization:

$$d(z,0) < d(gz,0) \qquad \text{if and only if} \qquad z \in \mathrm{ext}(I(g)).$$

The analogous statements with $<$ replaced by $=$ or $>$ and $\mathrm{ext}(I(g))$ replaced by $I(g)$ or $\mathrm{int}(I(g))$ respectively also hold (Proposition 1.3b of [Voi09]).

Definition 2.2. Let $G \subset \Gamma^\phi \backslash \{1\}$ be a subset, and define the *exterior domain* of G to be

$$\mathrm{ext}(G) = \overline{\bigcap_{g \in G} \mathrm{ext}(I(g))}.$$

In particular, $D(0) = \mathrm{ext}(\Gamma^\phi \backslash \{1\})$.

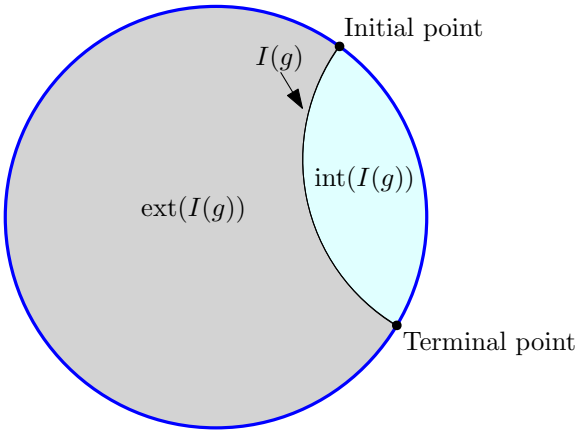


FIGURE 2. $I(g)$ example

Following Voight, let $G \subset \Gamma^\phi \backslash \{1\}$ be finite, and let $E = \text{ext}(G)$. Then E is bounded by a generalized hyperbolic polygon, with sides being:

- subarcs of $I(g)$ for $g \in G$ (proper side);
- arcs of the unit circle (infinite side);

and vertices being:

- intersections of $I(g)$ with $I(g')$ for $g \neq g'$ (proper vertex);
- intersections of $I(g)$ with the unit circle (vertex at infinity).

The polygon E can be neatly expressed via its normalized boundary.

Definition 2.3. A *normalized boundary* of E is a sequence $U = g_1, g_2, \ldots, g_k$ such that:

- $E = \text{ext}(U)$;
- The counterclockwise consecutive proper sides of E lie on $I(g_1)$, $I(g_2)$, ..., $I(g_k)$;
- The vertex v with minimal argument in $[0, 2\pi)$ is either a proper vertex with $v \in I(g_1) \cap I(g_2)$, or a vertex at infinity with $v \in I(g_1)$.

See Figures 3 and 4 for an example of going from the set of $I(g)$ for $g \in G$ to its normalized boundary. While this process is "obvious" visually, we require an algorithm that a computer can execute. Algorithm 2.5 of [Voi09] accomplishes this task, and it can be summarized as follows: start by intersecting all $I(g)$ with $[0, 1]$ to find g_1. Next, given $g_1, g_2, \ldots, g_i$, find all intersections of $I(g_i)$ with $I(g)$ for $g \in G$, and choose the "best one". Repeat until $g_i = g_1$.

Taking $|G| = n$, the running time is $\text{O}(nk)$, where k is the number of sides of U. In practice, $k = \text{O}(n)$ is typical, which gives a running time of $\text{O}(n^2)$.

To improve this algorithm, consider the terminal points of $I(g_i)$, and observe that they are in order around the boundary of $\mathbb{D}$! This follows from the fact that $I(g)$ and $I(g')$ cannot intersect twice in $\mathbb{D}$. In particular, we can start by sorting G by the arguments of the terminal points of $I(g)$, to get $g_1, g_2, \ldots, g_n$. The final normalized boundary will then be $g_{i_1}, g_{i_2}, \ldots, g_{i_k}$, where $i_1 < i_2 < \cdots < i_k$. We can iteratively construct this sequence $i_1, i_2, \ldots, i_k$ in $\text{O}(n)$ steps, for a total running time of $\text{O}(n \log(n))$ (due to the sorting).

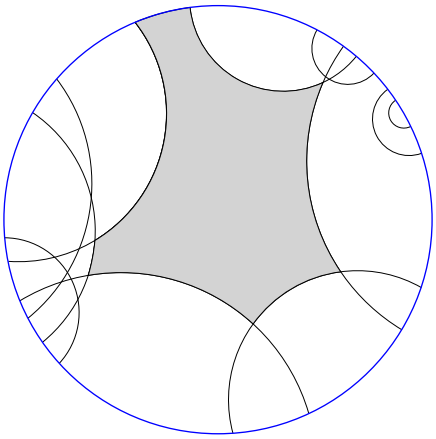


FIGURE 3. $I(g)$ for all $g \in G$

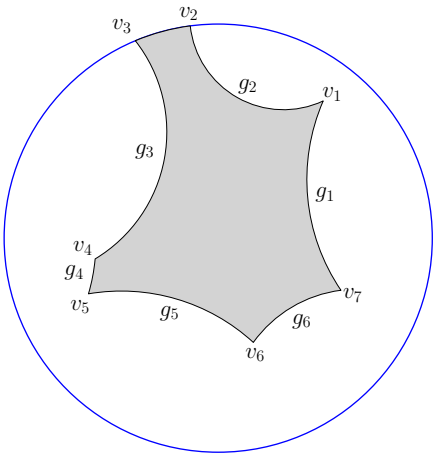


FIGURE 4. Normalized boundary of G

Algorithm 2.4. Given a finite subset $G \subset \Gamma^{\phi} \backslash \{1\}$, this algorithm returns the normalized boundary of $\text{ext}(G)$ in $O(n \log(n))$ steps. The letters U and V track sequences of elements of G and points in $\mathbb{D}$ respectively.

(1) Sort G by the arguments (in $[0, 2\pi)$) of the terminal points, to get $G = g_1, g_2, \ldots, g_n$ (take indices modulo n).
(2) Let H be the set of all g such that $I(g)$ intersects $[0, 1]$. If H is empty, go to step 3. Otherwise, go to step 4.
(3) Let $U = g_1$, let $i = 2$, let V be the terminal point of $I(g_1)$, and continue to step 5.
(4) Let $H' \subseteq H$ be the (non-empty) subset which gives the smallest intersection with $[0, 1]$. Let $g \in H'$ give the smallest angle of intersection of $I(g)$ with $[0, 1]$. Cyclically shift G so that $g_1 = g$, take $U = g_1$, let $i = 2$, and let V be the intersection of $I(g_1)$ with $[0, 1]$.
(5) If $i = n + 2$, delete g_1 from the end of U, and return U.
(6) Assume U ends with g, V ends with v, and compute $v' = I(g) \cap I(g_i)$.
    (a) If v' does not exist, then compare the terminal point of $I(g_i)$ with $I(g)$. If it lies in the interior, then increment i by 1, and go back to step 5. Otherwise, append g_i to U, append the initial point of $I(g)$ to V, append the terminal point of $I(g_i)$ to V, increment i by 1, and go back to step 5.
    (b) If v' is closer to the initial point of $I(g)$ than v, then append g_i to U, append v' to V, increment i by 1, and go back to step 5.
    (c) Otherwise, delete g from the end of U, delete v from the end of V, and go back to step 5.

Proof. If $I(g)$ does not intersect $[0, 1]$ for all $g \in G$, then $g = g_1$ must minimize the argument of the terminal point of $I(g)$. Otherwise, $g = g_1$ minimizes the intersection with $[0, 1]$, and taking the smallest angle (if this is not unique) guarantees the selection. This is the content of steps 1-4.

For the rest of the algorithm, U is the normalized boundary of $g_1, g_2, \ldots, g_{i-1}$, and V tracks the intersection of g with the previous side of U, where this side

may be infinite, or $[0,1]$ if $i = 2$. We intersect $I(g_i)$ with $I(g)$, and if there is no intersection, then we are either enclosed inside $I(g)$, or there is an infinite side (which must exist in all subsequent iterations due to the sorting of G). If v' is closer to the initial point of $I(g)$ than v, then we simply must add this new side in. Otherwise, this implies that the new side completely encloses the previous side, so backtrack by deleting the previous side and try again.

Our choice of g_1 guarantees that it will be a part of any partial normalized boundary, hence the sets U and V will never be empty. Furthermore, once we get to $i = n + 1$, we still need to go on, since $g_{n+1} = g_1$ may completely enclose some of the final isometric circles in U. Once we have finished with this step, we have completed the circle, and are left with the normalized boundary.

As for the running time, the initial sorting is the bottleneck. If we assume that we start with a sorted G, then finding g_1 takes $\mathrm{O}(n)$ steps, and the rest of the algorithm also takes $\mathrm{O}(n)$ steps. Indeed, if we deleted e elements from U in a step, then we performed $e + 1$ intersections. Since each element of G can be added to U at most once, and each element of U can be deleted at most once, the result follows. □

Assume that Figure 3 contains $I(g_1), I(g_2), \ldots, I(g_{11})$, ordered by argument of terminal points, with $I(g_1)$ intersecting $[0,1]$. Figure 5 contains the partial normalized boundaries computed by Algorithm 2.4, and which values of i they correspond to.

The order of magnitude of improvement is typically $\mathrm{O}\left(\frac{n}{\log(n)}\right)$, which is a substantial improvement since Algorithm 2.4 is called a large number of times. Furthermore, we will often be in the situation where we have a computed normalized boundary E and a set G', and we want to compute the normalized boundary of $E \cup G'$. Instead of blindly running Algorithm 2.4 on $E \cup G'$, we can save time by "remembering" that E is a normalized boundary. We only have to sort G', and combine this sorted list with the presorted E. When we are iteratively going through the new sequence, we can copy the data from E when we are away from new sides coming from G'. This extra computational trick is most effective when the size of G' is a much smaller than the size of E.

Remark 2.5. John Voight has made similar observations with regards to optimizing the geometric algorithms (personal communication). This code was implemented in Magma, but is not in the live version.

2.2. **Reduction of points.** Given an enumeration of elements of Γ, Algorithm 2.4 is enough to compute the fundamental domain, since eventually we will have the boundary of $D(0)$ in our set G. However, this is far too slow in practice. One of the key ideas introduced by Voight in [Voi09] is the ability to efficiently compute $\mathrm{ext}(\langle G\rangle)$, not just $\mathrm{ext}(G)$. By only requiring a set of generators, we need to enumerate less elements, which brings the total computation time to reasonable levels.

Definition 2.6. Let $G \subset \Gamma^{\phi}\backslash\{1\}$ be finite. Call U a *normalized basis* of G if U is the normalized boundary for $\mathrm{ext}(\langle G\rangle)$.

Before describing the computation of the normalized basis, we consider the reduction of elements/points to a normalized boundary. Take G as above, let $z \in \mathrm{int}(\mathbb{D})$,

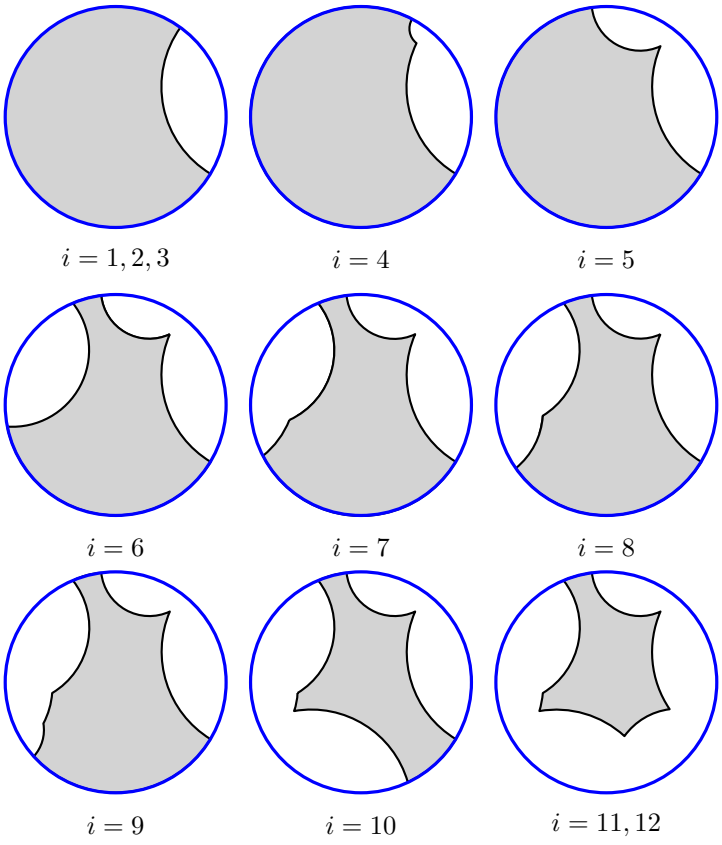


FIGURE 5. Partial normalized boundaries of Figure 3 computed with Algorithm 2.4

and consider the map

$$\begin{aligned}\rho : \Gamma &\to \mathbb{R}^{\geq 0} \\ \gamma &\to \rho(\gamma; z) = d(\gamma z, 0).\end{aligned}$$

Definition 2.7. An element $\gamma \in \Gamma$ is (G, z)-*reduced* if for all $g \in G$, we have $\rho(\gamma; z) \leq \rho(g\gamma; z)$.

Algorithm 4.3 of [Voi09] describes a process to reduce an element $\gamma \in \Gamma$, namely:

(1) Compute $\min_{g \in G} \rho(g\gamma; z)$. If this is greater than or equal to $\rho(\gamma; z)$, return γ.
(2) Replace γ by $g\gamma$, where g is the first element that attains the above minimum. Return to step 1.

This algorithm terminates to produce an element $\mathrm{red}_G(\gamma; z) := \delta\gamma$, where $\delta \in \langle G \rangle$ and $\mathrm{red}_G(\gamma; z)$ is (G, z)-reduced. If $z = 0$, write $\mathrm{red}_G(\gamma)$ for the reduction.

Proposition 4.4 of [Voi09] shows that if U is the normalized basis of G, then for almost all $z \in \mathbb{D}$, the element $\mathrm{red}_U(\gamma; z)$ is independent of all choices made. Furthermore, $\mathrm{red}_U(\gamma) = 1$ if and only if $\gamma \in \langle G \rangle$. Thus, if $\gamma \in \langle G \rangle$, this algorithm is a way to write γ^{-1} as a word in U.

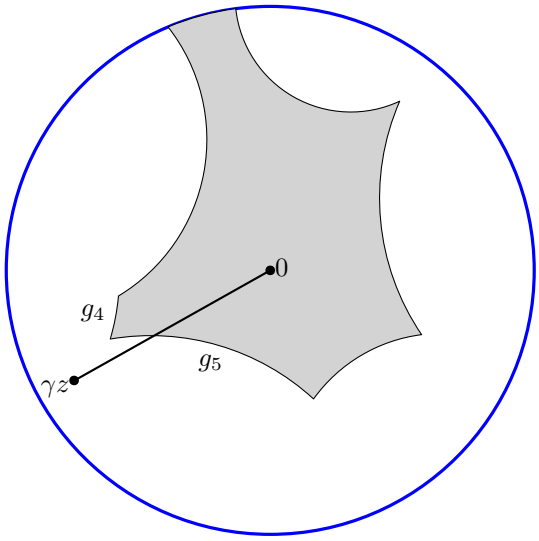


FIGURE 6. γz is in $\text{int}(I(g_4))$ and $\text{int}(I(g_5))$, and the line from 0 to γz passes through $I(g_5)$.

While the above algorithm is valid for all finite sets $G \subset \Gamma^\phi \backslash \{1\}$, it turns out that whenever we want to reduce an element, we will have the normalized boundary of G, denoted U, at our disposal! We make the following observations:

- When replacing γ with $g\gamma$, it is not necessary to have
$$\rho(g\gamma; z) = \min_{g' \in G} \rho(g'\gamma; z),$$
all that is required is $\rho(g\gamma; z) < \rho(\gamma; z)$. The algorithm will still terminate in approximately the same number of steps;
- By definition, $\rho(g\gamma; z) < \rho(\gamma; z)$ if and only if $\gamma z \in \text{int}(I(g))$;
- If $\gamma z \in \text{int}(I(g))$ for some $g \in G$, then the straight line from 0 to γz will intersect U on the proper side that is part of $I(g)$ for a possible g. See Figure 6 for a demonstration of this fact.

By finding where $\arg(\gamma z)$ should be inserted in the list of arguments of vertices of G (precomputed), we can determine a possible g for each step (or show that we are done) in $O(\log(n))$ steps. This is a large time save compared to the previous algorithm, where each step took $O(n)$ steps.

Algorithm 2.8. Let G be the normalized boundary of a finite subset of Γ, let the vertices of G be $v_1, v_2, \ldots, v_k$, let $z \in \text{int}(\mathbb{D})$, and let $\gamma \in \Gamma$. The following steps return δ, where $\text{red}_G(\gamma; 0) = \delta\gamma$ and $\delta \in \langle G \rangle$:

(1) Initialize $\delta = 1$, and $z' = \gamma z$.
(2) Use a binary search to determine the index i such that $\arg(v_i) \leq \arg(z') < \arg(v_{i+1})$.
(3) Let w be the intersection point of the side $v_i v_{i+1}$ (corresponding to $I(g)$) and the straight line (with respect to Euclidean geometry) between 0 and z'.

(4) If $|z'| \leq |w|$, return δ. Otherwise, replace (δ, z') by $(g\delta, gz')$, and return to step 2.

2.3. **Side pairing.** One crucial omission thus far is the notion of a side pairing for Dirichlet domains. Let P be a generalized hyperbolic polygon (allowing infinite sides), with proper sides S. Consider the set

$$\mathrm{SP} = \{(g, s, s') : s' = g(s)\} \subseteq \Gamma \times S \times S,$$

and call two sides (s, s') paired if there exists a $g \neq 1$ for which $(g, s, s') \in \mathrm{SP}$, where we allow a side to be paired with itself (note that that this relation is symmetric, but not necessarily reflexive). Say P has a *side pairing* if this relation induces a partition of S into singletons/pairs, i.e. for all $s \in S$ there exists a unique side $s' \in S$ with (s, s') paired.

As noted in Proposition 1.1 of [Voi09], the Dirichlet domain $D(p)$ has a side pairing. Furthermore, if G is a normalized boundary whose exterior domain has a side pairing, then $\mathrm{ext}(G)$ is the fundamental domain of $\langle G \rangle$. In particular, to compute the normalized basis of a finite set G, it suffices to compute a normalized boundary G' that has a side paring and $\langle G' \rangle = \langle G \rangle$.

In the case of Dirichlet domains, it is easy to see that if we have a side $s \subseteq I(g)$, then the only possible side s could be paired with is gs. In particular, given a normalized boundary G with vertices $v_1, v_2, \ldots, v_k$ and a proper side $s = v_i v_{i+1} \subseteq I(g)$, we can compute $g(v_i)$, search for its argument in the ordered list of $\arg(v_j)$, and then determine the j such that $g(v_i) = v_j$ (or determine that no such j exists). By computing $g(v_{i+1})$ and comparing it with v_{j-1}, we can determine if the side s is paired or not. Since all operations besides the set search are $\mathrm{O}(1)$ time, the the side pairing test takes $\mathrm{O}(\log(n))$ time, with $|G| = n$. If we try to find all paired sides, this will take $\mathrm{O}(n \log(n))$ time.

2.4. **Normalized basis.** Algorithm 4.7 of [Voi09] details how to compute the normalized basis of G, and is not modified here. We do record it, to demonstrate that whenever we apply Algorithm 2.8 to reduce an element, the corresponding normalized boundary has been pre-computed (and thus we are able to apply our improved algorithms).

Algorithm 2.9. [Normalized basis algorithm, Algorithm 4.7 of [Voi09]] Let $G \subset \Gamma^\phi \backslash \{1\}$ be finite. The normalized basis of G can be computed as follows:

(1) Let $G' = G \cup G^{-1} = \{g : g \text{ or } g^{-1} \in G\}$.
(2) Use Algorithm 2.4 to compute the normalized boundary of G', denoted U.
(3) Let $G'' = G'$. For each element $g \in G'$, compute $\mathrm{red}_U(g)$. If $\mathrm{red}_U(g) \neq 1$, extend G'' by $\mathrm{red}_U(g)^{-1}$.
(4) Compute the normalized boundary of G'', denoted U'.
(5) If $U' = U$ and $G'' = G'$, then continue. Otherwise, replace (G', U) by (G'', U'), and return to step 3.
(6) Compute the set of paired sides of U. If this is a side pairing, then return U. Otherwise, for all pairs (s, v) of an unpaired proper side s with unpaired vertex v (i.e. $s \subseteq I(g)$ and gv is not a vertex of U), compute $\mathrm{red}_{G'}(g; v)$ and add it to G'' (if v is a vertex at infinity, replace v by a nearby point in the interior of $\mathbb{D}$).
(7) Let $G' = G''$, and return to step 2.

3. The general algorithm (without enumeration)

With the geometry in hand, we describe the computation of the fundamental domain, since this will help guide us in considering the enumeration of elements of Γ.

Algorithm 3.1. Let Γ be a discrete subgroup of $\mathrm{PSL}(2,\mathbb{R})$, so that $\Gamma^\phi \subseteq \mathrm{PSU}(1,1)$. Let $f(\Gamma)$ denote an oracle that returns a finite set of elements of Γ. This algorithm returns $D(0)$, which is a fundamental domain of Γ.

(1) Compute $\mu(\Gamma)$ via theoretical means. Initialize $G = \emptyset$.
(2) Call $f(\Gamma)$ to generate $G' \subseteq \Gamma^\phi$.
(3) Use Algorithm 2.9 to compute U, the normalized basis of $G' \cup G$.
(4) Compute the hyperbolic area of U. If it is equal to $\mu(\Gamma)$, then return U. Otherwise, let $G = U$, and return to step 2.

Besides the oracle, this algorithm assumes two things: explicit computation of the hyperbolic area of U, and theoretical computation of $\mu(\Gamma)$. The first of these is easy to resolve: it is well known that the area of a hyperbolic polygon with n sides and angles $\alpha_1, \alpha_2, \ldots, \alpha_n$ is

$$\mu(P) = (n-2)\pi - \sum_{i=1}^{n} \alpha_i.$$

Therefore, the only requirements to compute a fundamental domain are a way to generate (enough) elements of the group, and a computation of the area (or another means to determine when we are done). In the case of arithmetic Fuchsian groups, an enumeration is described in the next section, and the area computation is classical (see Theorem 39.1.8 of [Voi21] for a statement and proof).

Remark 3.2. The most expensive part of the area computation for arithmetic Fuchsian groups is the computation of $\zeta_F(2)$, where F is the totally real number field. With larger fields, we need higher precision, and the cost to compute this zeta value accurately grows high. However, a precise value for $\zeta_F(2)$ is unnecessary! Indeed, as soon as we have a normalized basis with finite area, it will correspond to a finite index subgroup Γ' of Γ. In particular, $\mu(\Gamma') = [\Gamma : \Gamma']\mu(\Gamma) \geq 2\mu(\Gamma)$ if they are not equal. Thus, it suffices to check that the hyperbolic area is less than $2\mu(\Gamma)$, which requires very little precision. In practice, it is incredibly unlikely to end up with a $\Gamma' \neq \Gamma$; the algorithm will nearly always stop once the area is finite.

4. Enumeration

Let F be a totally real number field of degree n with ring of integers $\mathcal{O}_F$, and let B be a quaternion algebra over F that is split at exactly one infinite place. Consider B as being embedded in $\mathrm{Mat}(2,\mathbb{R})$ via the unique split infinite place, and let O be an order of B. Then the group $\Gamma_\mathrm{O} := \mathrm{O}^1/\{\pm 1\} \subseteq \mathrm{PSL}(2,\mathbb{R})$, the group of units in O of reduced norm 1, is an arithmetic Fuchsian group.

Definition 4.1. Label the Fuchsian group corresponding to O by the integer triple (n, d, N), where n is the degree of F, $d = \mathrm{disc}(F)$ is the discriminant of F, and $N = \mathrm{Nm}_{F/\mathbb{Q}}(\mathrm{discrd}(\mathrm{O}))$ is the norm to $\mathbb{Q}$ of the reduced discriminant of the order O. Call this triple the *data* associated to the group.

Note that the data does not necessarily uniquely label a Fuchsian group, since distinct number fields may have the same discriminant, and nonisomorphic algebras/orders may give the same N.

As in Section 2, fix $p = x + yi \in \mathbb{H}$, and let $\phi : \mathbb{H} \to \mathbb{D}$ be the corresponding map sending p to 0. For $g = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Mat}(2, \mathbb{R})$, define

$$f_g(p) := cp^2 + (d-a)p - b.$$

If g has norm 1, let $g^\phi = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \in \mathrm{SU}(1,1)$, and then a computation shows that

$$C = \frac{-\overline{f_g(p)}}{2iy}, \text{ and the radius of } I(g) \text{ is } \frac{1}{|C|}.$$

In particular, if $M = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \in \mathrm{U}(1,1)$, define

$$f_M(p) := 2iy\overline{C},$$

so that $f_g(p) = f_{g^\phi}(p)$.

Definition 4.2. Let $z_1, z_2 \in \mathbb{D}$, and fix $M_1, M_2 \in \mathrm{PSU}(1,1)$ so that $M_i(0) = z_i$ for $i = 1, 2$. For $g \in B$, define the quadratic form

$$Q_{z_1,z_2}(g) := \frac{1}{2y^2}\left|f_{M_2^{-1} g^\phi M_1}(p)\right|^2 + \mathrm{Tr}_{F/\mathbb{Q}}(\mathrm{nrd}(g)).$$

This is the analogue of Q_{z_1,z_2} from Definition 28 in [Pag15], and is similarly well defined (independent of M_1, M_2) and positive definite. In [Voi09], Voight defined the absolute reduced norm N, which satisfies

$$N(g) = 2y^2 Q_{0,0}(g).$$

The analogue of Proposition 30 of [Pag15] is Proposition 4.3.

Proposition 4.3. *If $g \in \Gamma_{\mathrm{O}}$, then*

$$Q_{z_1,z_2}(g) = \cosh(d(g^\phi z_1, z_2)) + n - 1.$$

Proof. Let $M = M_2^{-1} g^\phi M_1 = \left(\begin{smallmatrix} A & B \\ \overline{B} & \overline{A} \end{smallmatrix}\right)$, with $|A|^2 - |B|^2 = 1$. Since $\mathrm{nrd}(g) = 1$, it suffices to prove that

$$\cosh(d(gz_1^\phi, z_2)) - 1 = 2|B|^2.$$

Since the action of PSU is isometric with respect to hyperbolic distance,

$$d(g^\phi z_1, z_2) = d(M_2^{-1} g^\phi z_1, M_2^{-1} z_2) = d(M0, 0).$$

Applying a classical formula for the hyperbolic distance, we get

$$\cosh(d(gz_1^\phi, z_2)) - 1 = \frac{2|B/\overline{A}|^2}{1 - |B/\overline{A}|^2} = \frac{2|B|^2}{|A|^2 - |B|^2} = 2|B|^2,$$

as desired. □

In particular, if we pick z_1, z_2 such that gz_1 is close to z_2 for some $g \in \Gamma_{\mathrm{O}}$, then by enumerating vectors $g' \in \mathrm{O}$ such that $Q_{z_1,z_2}(g') \leq C$ and checking which of them have reduced norm 1, we can recover g. Since O is a positive definite rank $4n$ module over $\mathbb{Z}$, the small vectors of Q_{z_1,z_2} can be enumerated with the Fincke-Pohst algorithm, [FP85].

In [Voi09], the enumeration strategy was to solve $Q_{0,0}(g) \leq C$, and compute the normalized basis of the generated elements. If the area is larger than the target

area, then increase C and repeat. Since the boundary of the fundamental domain has finitely many sides and

$$Q_{0,0}(g) = \frac{2}{\mathrm{rad}(I(g))^2} + n$$

for $g \in \mathrm{O}^1$, for large enough C we obtain the fundamental domain. A downside to this strategy is the number of $g \in \mathrm{O}$ with $Q_{z_1,z_2}(g) \leq C$ grows like C^{2n}, the proportion of elements with $g \in \mathrm{O}^1$ shrinks, and thus the computation time blows up. The timing also has high variance, as the value of C you need can vary greatly for similar examples.

In [Pag15], Page introduced Q_{z_1,z_2} (defined for Kleinian groups, which we specialized down to Fuchsian groups), and used this new freedom to define a probabilistic enumeration that greatly outperforms the deterministic one in practice. The outline of his enumeration is as follows:

- Pick a set of random points Z;
- Solve $Q_{0,z}(g) \leq C$ for $z \in Z$;
- Take the small vectors with $\mathrm{nrd}(g) = 1$, and add them to your generating set;
- Compute the normalized basis, and if the area is not correct, repeat these steps.

This strategy requires a couple of choices that are not immediately obvious:

- What value of C is optimal?
- How do we pick the random points Z, and how many of them are chosen in each pass?

Heuristics for both questions are given, but it is not clear if they remain optimal for the case of arithmetic Fuchsian groups. Furthermore, the heuristics do not specify the constants, which are necessary for an efficient practical implementation. We explore these questions, with plenty of data as evidence, in the following sections.

Remark 4.4. One subtle part of this strategy is checking if $\mathrm{nrd}(g) = 1$. Since we are repeating this norm computation a large number of times, we pre-compute the Cholesky form of $\mathrm{nrd}(g)$, i.e. write it as a sum of (four) squares (see [FP85] for more details). This speeds the norm computations up by a factor of ≈ 10 over the PARI command "algnorm".

4.1. **Choosing random points.** As in [Pag15], we pick points uniformly from a hyperbolic disc of radius R. The choice of R needs to be large enough that the random point is approximately uniform in the fundamental domain. On the other hand, if R is too large, then precision issues may occur. We take $R = R_{\mathrm{O}}$, where

$$\mu(\text{disc of radius } R_{\mathrm{O}}) = \mu(\Gamma_{\mathrm{O}})^{2.1}.$$

This is the same choice as the implementation of [Pag15].

Remark 4.5. There are several papers on the diameter of $\Gamma\backslash\mathbb{H}$, which lend support to R_{O} being large enough. Unconditional results are given by Chu-Li in [CL16], and conditional results on the almost-diameter are given by Golubev-Kamber in [GK19]. Assuming the Selberg eigenvalue conjecture, their results imply that the almost-diameter of $\Gamma\backslash\mathbb{H}$ is bounded by

$$(1 + o(1)) \log(\mu(\Gamma\backslash\mathbb{H})),$$

when Γ is a congruence arithmetic Fuchsian group. An upcoming work by Steiner ([Ste]) gives this result unconditionally in certain cases.

Remark 4.6. Based on Remark 4.5, an exponent of $1+\epsilon$ in the definition of R_{O} would be sufficient to pick up the whole fundamental domain when O is Eichler. Since we do not want the random point to be biased, we increased the exponent to 2.1 for R_{O}. This is not a precise choice, merely a choice that worked sufficiently well in practice. For non-Eichler orders, a larger exponent might be required, or an alternate approach where the fundamental domain for a maximal order containing O is computed, and a coset enumeration algorithm is applied to push the domain down.

4.2. **Choice of C.** Choosing the correct value of C in the computation of $Q_{0,z}(g') \leq C$ is extremely important. If C is too small, then it will take too many trials to get an element of reduced norm 1, and if C is too large, each individual trial will take too long.

First, we compute the probability of success of a trial. Assume that z is chosen sufficiently randomly, take $g \in \mathrm{O}^1$, and let $x = d(g^\phi(0), z)$ be the distance between $g^\phi(0)$ and z. By Proposition 4.3, the trial $Q_{0,z}(g') \leq C$ will find g if and only if

$$\cosh(x) = \frac{e^x + e^{-x}}{2} \leq C + 1 - n.$$

Therefore we find g if and only if the hyperbolic disc of radius $\cosh^{-1}(C+1-n)$ about z contains $g^\phi(0)$. The expected number of such g's is thus the area of this disc divided by $\mu(\Gamma_{\mathrm{O}})$. Since a hyperbolic disc of radius R has area

$$4\pi \sinh(R/2)^2 = \pi(e^R + e^{-R} - 2) = 2\pi(\cosh(R) - 1),$$

we derive Heuristic 4.7.

Heuristic 4.7. The expected number of elements of O^1 that the trial $Q_{0,z}(g') \leq C$ outputs is

$$\mathbb{E}(\text{number of elements found}) = \frac{2\pi(C-n)}{\mu(\Gamma_{\mathrm{O}})}.$$

To demonstrate this heuristic, we run the trial $Q_{0,z}(g') \leq C$ across a range of C's in two algebras. The points z are chosen uniformly at random from a disc of radius R_{O}. In Figures 7 and 8, we display the results, including the straight line predicted by Heuristic 4.7 (we only count the non-trivial elements found). The R^2 values of the heuristic are 0.98840348 and 0.96506302 respectively. This data also lends support to our chosen radius being sufficiently large to model a random point of the fundamental domain.

Remark 4.8. If C is small, then we will find at most one element of O in each trial. Indeed, if we found two elements g_1, g_2, then we must have $d(0, g_1^{-1}g_2(0)) \leq 2\cosh^{-1}(C+1-n)$, i.e. 0 is close to $g_1^{-1}g_2(0)$. Since the action of Γ_{O}^ϕ is discrete, this gives a minimum bound on C for this behaviour to occur. In practice, the optimal value of C will typically be smaller than this, so we can stop a trial if we find a single element. Note that even if C were just large enough, the second element found will be useless after the first time! An element is only useful if it is not in the span of all previously found elements. In this case, the quotient $g = g_1^{-1}g_2$ will be constant, so g_2 is in the span of g_1 and all previously found elements if we already have g.

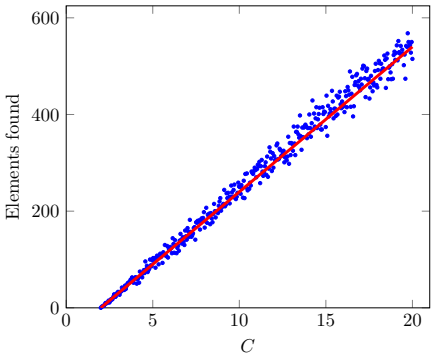


FIGURE 7. Elements found in 1000 trials of $Q_{0,z}(g') \leq C$ for curve $(2, 21, 101)$.

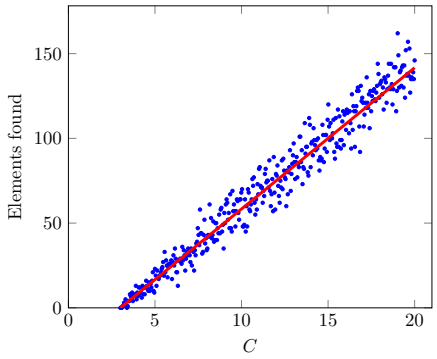


FIGURE 8. Elements found in 1000 trials of $Q_{0,z}(g') \leq C$ for curve $(3, 229, 106)$.

Remark 4.9. We can test $Q_{0,0}(g') \leq C$ at the start in an attempt to pick up some of these elements with large isometric circle radii. This is typically more useful when $\mu(\Gamma_{\mathrm{O}})$ is small, and becomes more effective as the degree of the number field grows (as the cost to generate a single element becomes high very quickly). We will use the same value of C as for the tests $Q_{0,z}(g') \leq C$.

To perform the enumeration of $Q_{0,z}(g') \leq C$, there are two distinct parts. Part one is the setup of Fincke-Pohst, i.e. computing the a series of matrix reductions to minimize failures in the enumeration. This part is independent of C. Part two consists of the actual enumeration, and we assume that the time taken is proportional to the number of vectors enumerated. This leads to Heuristic 4.10.

Heuristic 4.10. The time to complete the enumeration of $Q_{0,z}(g') \leq C$ is

$$A + BC^{2n},$$

where A, B are constants depending on O.

To demonstrate Heuristic 4.10, we computed the enumeration time for 1000 C's in two quaternion algebras. Figure 9 demonstrates the results for $n = 1$, along with the associated best fit curve $t = 1.0560652 \cdot 10^{-3} + 1.1777653 \cdot 10^{-8}C^2$, which gives an R^2 value of 0.99559127 (the horizontal lines are due to PARI timings being integer multiples of one millisecond). Figure 10 demonstrates the results for $n = 4$, along with the associated best fit curve $t = 1.1927396 \cdot 10^{-2} + 6.3362747 \cdot 10^{-14}C^8$, which gives an R^2 value of 0.99761548.

In particular, combining Heuristics 4.7 and 4.10, the expected time taken before a success is given by

$$\left(\frac{\mu(\Gamma_{\mathrm{O}})B}{2\pi}\right)\frac{C^{2n} + A/B}{C - n}. \tag{4.1}$$

Basic calculus implies that this is minimized for the unique real solution $C > n$ of the polynomial

$$(2n - 1)C^{2n} - 2n^2C^{2n-1} = A/B. \tag{4.2}$$

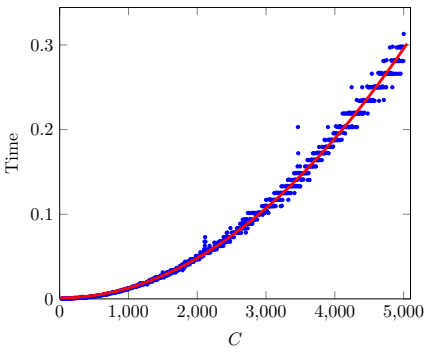


FIGURE 9. Computation time of $Q_{0,z}(g') \leq C$ for curve $(1, 1, 2021)$.

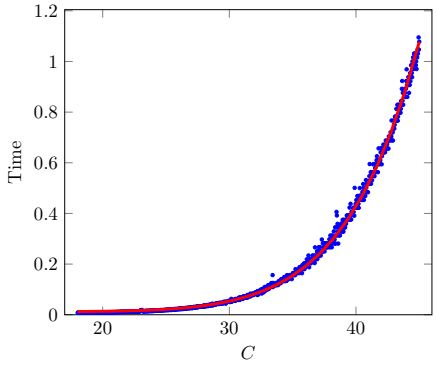


FIGURE 10. Computation time of $Q_{0,z}(g') \leq C$ for curve $(4, 14013, 109)$.

Remark 4.11. The values of A and B are highly dependent on factors like precision, processor speed, etc. However, these factors should affect A and B to approximately the same factor, leaving A/B constant. On the other hand, A/B is highly dependent on the implementation of Fincke-Pohst, the setup to the enumeration, or even the computation of the norm of an element.

For a fixed O, we can run similar computations to Figures 9 and 10, and obtain approximate values for A, B with a least squares regression. Solving Equation (4.2) gives the optimal value of C for this quaternion order. By performing this computation across a large selection of arithmetic Fuchsian groups, we can deduce a heuristic for the optimal value of C. This is presented in Heuristic 4.12, with the rest of this section dedicated to providing experimental evidence justifying the heuristic. It is important that the heuristic is run across a large range of C's, algebra discriminants, and field discriminants, as otherwise the final constants may be significantly off.

Heuristic 4.12. Let O have reduced discriminant $\mathfrak{D}$. The optimal choice of C is given by

$$C_{\mathrm{O}} := C_n \operatorname{disc}(F)^{1/n} \operatorname{Nm}_{F/\mathbb{Q}}(\mathfrak{D})^{1/2n},$$

for constants C_n. Approximate values for C_n for $n \leq 8$ to 6 decimal places are:

n	C_n	n	C_n
1	2.830484	5	1.019539
2	0.933176	6	1.018481
3	0.909751	7	0.994256
4	0.973456	8	0.964400

Remark 4.13. The non-constant part of C_{O} differs to the heuristic in [Pag15]. However, this was a typo! The Magma implementation of the algorithm uses the correct heuristic.

Remark 4.14. We can also justify the heuristic theoretically (thanks to Aurel Page for the argument). Fix n, and consider Heuristic 4.10. The term BC^{2n} should be proportional to the number of elements enumerated, and thus B is inversely proportional to the covolume of O. Thus, $1/B$ should be proportional to

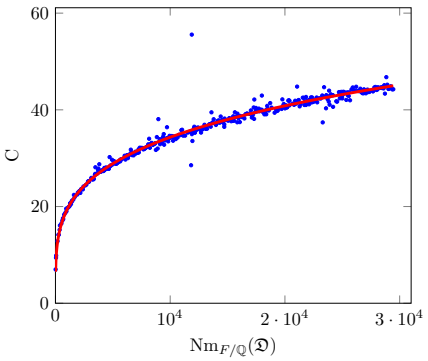


FIGURE 11. C for 400 quaternion algebras over $\mathbb{Q}(y)$, with $y^2 - 13 = 0$.

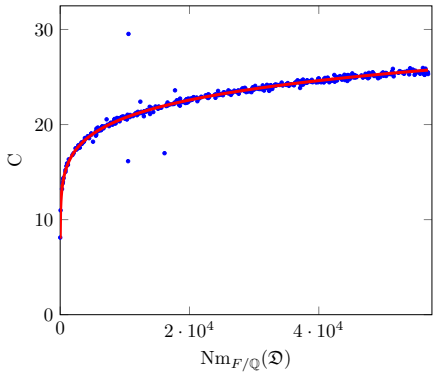


FIGURE 12. C for 400 quaternion algebras over $\mathbb{Q}(y)$, with $y^4 - 5y^2 + 5 = 0$.

$\mathrm{disc}(F)^2 \,\mathrm{Nm}_{F/\mathbb{Q}}(\mathfrak{D})$. Assume A is constant and A/B is large, so that C is also large (say $C \gg n$). Then the secondary term of Equation (4.2) can be ignored, and Heuristic 4.12 follows.

The value of C_n is dependent on implementation. If there is a future improvement in part of the implementation, then the new optimal algorithm would necessarily have larger optimal values for C_n. However, unless it was a very significant improvement, the current optimal values of C_n would be still perform well.

The behaviour of C_n is hard to explain. When $n \geq 2$, we have to deal with number theory arithmetic, which is more costly than integer arithmetic. This may explain the big drop from C_1 to C_2, but does not explain the further oscillation. For $n \geq 9$, we suggest taking $C_n = 1$ as a baseline, since the data does hover around this value. When running a large computation with $n \geq 9$, it would also be worth running these experiments again for this n, or at least experimenting with the value of C_n a bit to get a more accurate value, and thus better performance.

4.3. **Computational evidence.** To demonstrate Heuristic 4.12, start by fixing F and varying $\mathfrak{D}$. By computing A and B (with a regression) as in Heuristic 4.10 and solving for C with Equation (4.2), we can determine the optimal C for each case. In Figure 11, we carry this out for 400 quaternion algebras over a quadratic field. The curve of best fit, $C = 3.43356822 \left(\mathrm{Nm}_{F/\mathbb{Q}}(\mathfrak{D})\right)^{1/4}$, is given in red. In Figure 12, this is done for 400 quaternion algebras in a quartic setting. The curve of best fit is $C = 6.54768871 \left(\mathrm{Nm}_{F/\mathbb{Q}}(\mathfrak{D})\right)^{1/8}$.

Next, fix n, and vary F. For each value of n we take 400 quaternion algebras over totally real number fields of degree n, and compute $C' = C/\,\mathrm{Nm}_{F/\mathbb{Q}}(\mathfrak{D})^{1/2n}$. The curves of best fit are all of the form $C' = C_n \,\mathrm{disc}(F)^{1/n}$. The data for $n = 2$ and $n = 3$ is displayed in Figures 13 and 14 respectively.

These computations also give the values of C_n, as found in Heuristic 4.12. Since they are experimentally found, running the computations again will produce slightly different values; the values given are only intended as approximations.

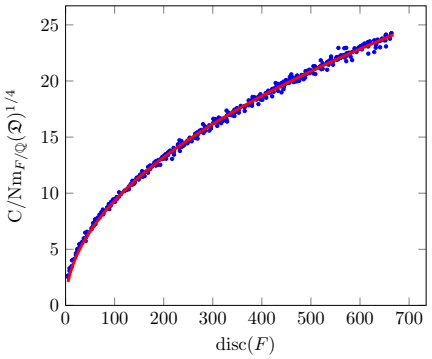


FIGURE 13. C' for $n = 2$.

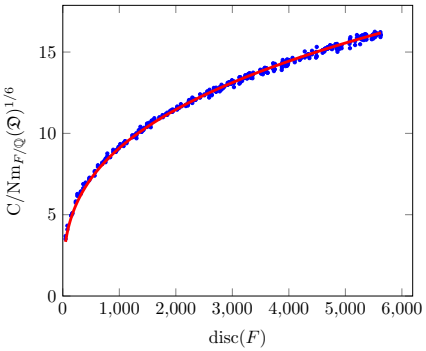


FIGURE 14. C' for $n = 3$.

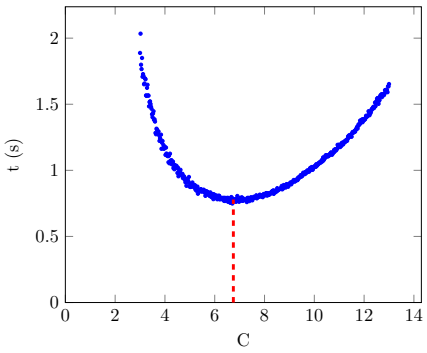


FIGURE 15. Time to obtain 1000 elements for $(2, 5, 101)$.

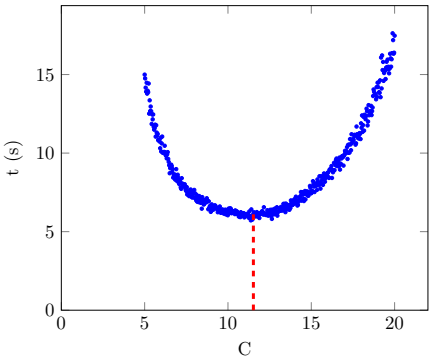


FIGURE 16. Time to obtain 1000 elements for $(3, 316, 46)$.

To demonstrate that this theory actually works, we can fix an algebra, compute the time taken to find N non-trivial elements over a range of C's, and verify that Heuristic 4.12 is close to the observed minimum. In Figures 15 and 16, we take curves with data $(2, 5, 101)$ and $(3, 316, 46)$, and compute the time to generate 1000 non-trivial elements of each for 500 values of C (outputting at most one value for each z, due to Remark 4.8). A dotted red line is drawn at the value of C predicted by Heuristic 4.12 to be minimal. In each example this value is close to the absolute minimum.

4.4. **Improved Fincke-Pohst.** Let $Q(x_1, x_2, \ldots, x_N)$ be a positive definite quadratic form. The general idea of the Fincke-Pohst enumeration of $Q(x) \leq C$ is to make a change of basis to variables $x'_1, x'_2, \ldots, x'_N$, write Q as a sum of squares in this basis, and incrementally bound $x'_N, x'_{N-1}, \ldots, x'_1$. The change of basis was chosen in a way to minimize the number of "dead ends" in the incremental enumeration, and is the key part of the algorithm (for details, see Section 2 of [FP85]). In our situation, we also have an (indefinite) quadratic form Q' for which we require $Q'(x_1, x_2, \ldots, x_N) = 1$. If we compute the change of basis for Q' to the variables

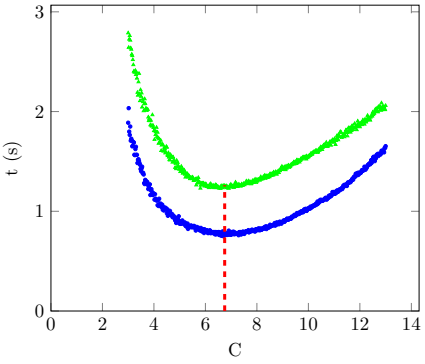


FIGURE 17. Time to obtain 1000 elements for $n = 2$, FP and IFP.

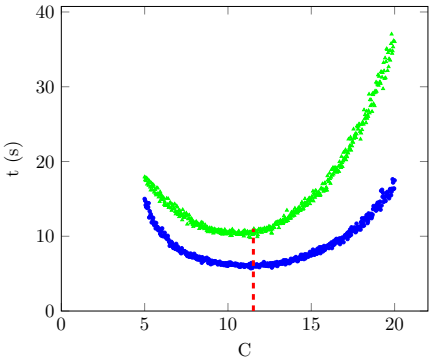


FIGURE 18. Time to obtain 1000 elements for $n = 3$, FP and IFP.

x_i', then when we get to x_1', we see that it must satisfy a quadratic equation! In particular, there are at most two possibilities for it. By solving this quadratic over the integers, we can determine if we have a solution or not. Note that we may find solutions with $Q(x) > C$, but this does not concern us since they still give an element of O^1.

Label the classical Fincke-Pohst by "FP", and this new approach by "IFP". If we have many choices for x_1', then IFP should be a faster algorithm, since we solve one quadratic equation over $\mathbb{Z}$ as opposed to checking the norms of a large set of elements. On the other hand, this requires a lot more (number field) arithmetic, some of which will be "useless" when we reach dead ends. To determine the efficacy, we compute the time required to find 1000 non-trivial elements in a given quaternion algebra with each enumeration method (as before, stopping each trial after finding an element). We use a range of C's, as the optimal value of C for this approach may not be C_{O} (in particular, Heuristic 4.10 is no longer valid). Considering the already computed examples in Figures 15 and 16, we re-compute the time taken with IFP. The output (with IFP shown in green with triangle markers) is Figures 17 and 18.

In particular, IFP is slower for both examples. This fact continues to hold for all other examples attempted with $n \geq 2$. On the other hand, if $n = 1$ (i.e. $F = \mathbb{Q}$), then IFP appears to be faster. For example, see Figures 19 and 20. The value of C_{O} is no longer optimal for IFP, but it is still quite reasonable and better than before.

Observation 4.15. For the enumeration, we should use IFP when $n = 1$, and FP when $n > 1$.

The difference in behaviour between $n = 1$ and $n > 1$ likely comes from the limiting of trials to one element. IFP thrives on cutting down large ranges for x_1', and when $n > 1$, we don't get these large ranges. If we instead find all norm one elements in a trial, then IFP will become more efficient than FP for large enough C. While this is irrelevant with regards to the current enumeration, it is useful for Voight's enumeration of elements in [Voi09], which is to solve $Q_{0,0}(g) \leq C$ for increasingly large C.

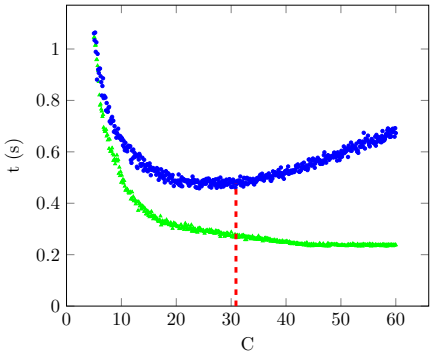


FIGURE 19. Time to obtain 1000 elements for curve $(1, 1, 119)$, FP and IFP.

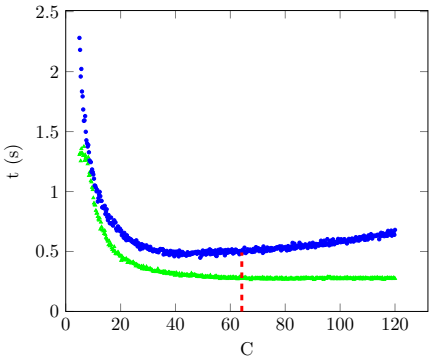


FIGURE 20. Time to obtain 1000 elements for curve $(1, 1, 514)$, FP and IFP.

4.5. **Balancing enumeration and geometry.** At this point, algorithms to efficiently generate elements of O^1 and compute normalized bases have been described. The final question is: how do we strike a balance between these two operations? If we compute the normalized basis too often, then we are wasting a lot of time with useless calls. On the other hand, if we call it less often, then we may enumerate too many elements before finishing.

One other benefit of normalized basis calls was noted in both [Voi09] and [Pag15]. If we have an infinite side of a partial fundamental domain, then an element $g \in \mathrm{O}^1$ for which $I(g)$ closes off (part) of this side can be picked up with $Q_{0,z}$ for some z near the infinite side. In particular, by searching there, we can increase the chance of finding useful elements.

On the geometric side, computing the normalized basis with Algorithm 2.9 should take $O(n \log(n))$ steps, where n is the number of elements. To estimate the number of sides of the final fundamental domain, we do this computation for 1000 domains with hyperbolic area at most 1000 (distributed among $\deg(F) \leq 4$). As seen in Figure 21, the number of sides is typically proportional to the area (the analogous statement was noted for arithmetic Kleinian groups at the end of [Pag15]). In fact, the line of best fit is

$$\text{Number of sides} \approx 0.94747172\mu(\Gamma_{\mathrm{O}}),$$

which has an R^2 value of 0.99273664. Assuming we don't compute the normalized basis *too* often, this final computation should dominate. Therefore we expect the geometric part of the algorithm to grow proportionally to $\mu(\Gamma_{\mathrm{O}}) \log(\mu(\Gamma_{\mathrm{O}}))$.

For the enumeration, as noted in [Pag15], a number of elements proportional to $\mu(\Gamma_{\mathrm{O}})$ should generate Γ_{O} (see Theorem 1.5 of [BGLS10]). Due to the probabilistic nature of the method, there is not a precise constant of proportionality for which we can guarantee success. Instead, we will get a rough idea of the proportion by computing a large number of examples. Using the same input algebras as Figure 21, we find the smallest N such that the first N random elements we computed were sufficient to generate the fundamental domain. This data is displayed in Figure

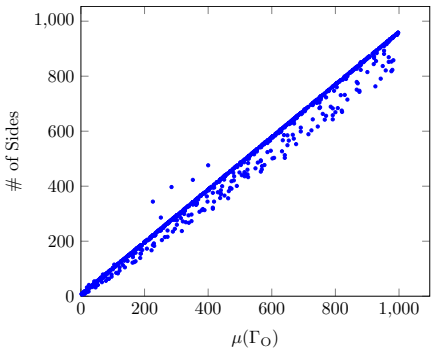


FIGURE 21. Number of sides of the fundamental domain.

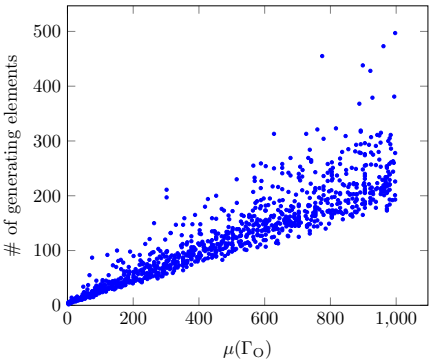


FIGURE 22. Number of random elements required to generate the fundamental domain.

22, and does appear to be approximately linear, as expected. Based on this data, around $\frac{1}{4}\mu(\Gamma_O)$ elements is a good target to generate the fundamental domain.

Considering Equation (4.1), an element can be generated in expected time $O(\mu(\Gamma_O))$. Furthermore, combining the heuristics gives a computable constant w such that $w\mu(\Gamma_O)^2$ random points should be close to generating the fundamental domain.

With these heuristics in hand, we suggest choosing a constant $c < 1$, and picking $cw\mu(\Gamma_O)^2$ points in each iteration. Experiments show that $c = \frac{1}{2}$ when $n = 1$, and $c = \frac{1}{12}$ when $n > 1$ are reasonable choices.

Remark 4.16. As seen in the previous section, the "cost" of choosing a poor value of C_n was extremely high. On the other hand, the cost of a poor choice for the number of random centres in each iteration is much smaller. Any reasonable choice will perform decently well, and we thus do not delve as deep into the choice of c in each situation as we did for the choice of C.

Remark 4.17. Another solution to striking a balance between enumeration and geometry would be to use parallel computing. One processor would be enumerating group elements, with the other processor computing normalized bases (and supplying information on missing infinite sides). Furthermore, by having multiple processors enumerating elements, one can speed up the enumeration by a constant factor.

5. Sample timings

While the currently implemented algorithms correspond to the content of this paper, that may change in the future. If better approaches, constant choices, programming tricks, etc. are found, then the algorithm timings will change with it. The hardware of a computer will also affect things, and this may not be a constant factor either. In any case, these timings are a representative of the current state of affairs, and give a general guideline. All computations in this section were run on the same McGill University server as Table 1. This server is not particularly fast, so you will likely see similar or better times on your own machine.

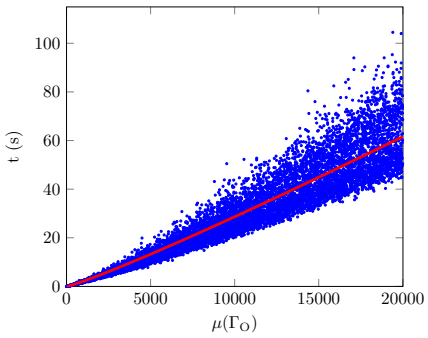


FIGURE 23. Time to compute the fundamental domain, $n = 1$.

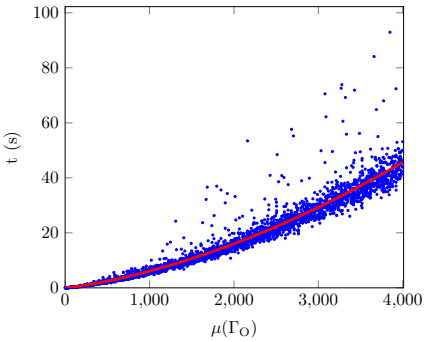


FIGURE 24. Time to compute the fundamental domain, $n = 2$.

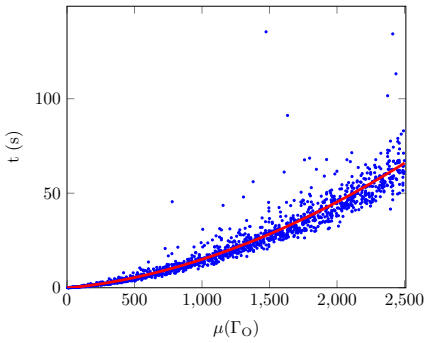


FIGURE 25. Time to compute the fundamental domain, $n = 3$.

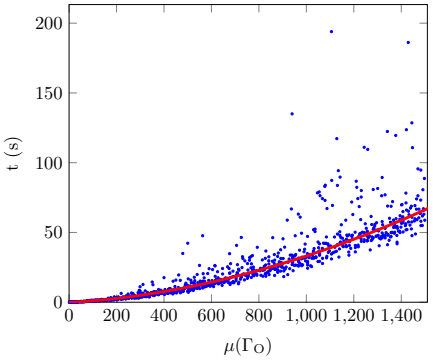


FIGURE 26. Time to compute the fundamental domain, $n = 4$.

Considering the heuristics given in Section 4.5, the expected running time is

$$c_1 \mu \log(\mu) + c_2 \mu^2,$$

where $\mu = \mu(\Gamma_O)$, and c_1, c_2 depend on n. While the enumeration time will eventually be the most costly part of the algorithm, the normalized basis will dominate for smaller areas.

For $n = 1$, we computed all fundamental domains with area at most 20000 that correspond to maximal orders (there are 9550 such examples). In this range, the normalized basis dominates, and the curve with $(c_1, c_2) = (0.88824714, 0)$ is shown in red in Figure 23.

For $n = 2$, we computed 2975 examples, all with area at most 4000, over three fields. The curve with $(c_1, c_2) = (0.00066605054, 0.0000014754184)$ is shown in red in Figure 24. The enumeration becomes the dominant part of the computation time at $\mu(\Gamma_O) \approx 3700$.

For $n = 3$, we computed 1481 examples, all with area at most 2500, over four fields. The curve with $(c_1, c_2) = (0.0011822431, 0.0000068505131)$ is shown in red in Figure 25.

Finally, for $n = 4$, we computed 912 examples, all with area at most 1500, over six fields. The curve with $(c_1, c_2) = (0.0018590790, 0.000021764718)$ is shown in red in Figure 26.

References

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993). MR MR1484478

[BGLS10] Mikhail Belolipetsky, Tsachik Gelander, Alexander Lubotzky, and Aner Shalev, *Counting arithmetic lattices and surfaces*, Ann. of Math. (2) **172** (2010), no. 3, 2197–2221. MR 2726109

[CL16] Michelle Chu and Han Li, *Small generators of cocompact arithmetic Fuchsian groups*, Proc. Amer. Math. Soc. **144** (2016), no. 12, 5121–5127. MR 3556258

[DV13] Lassina Dembélé and John Voight, *Explicit methods for Hilbert modular forms*, Elliptic curves, Hilbert modular forms and Galois deformations, Adv. Courses Math. CRM Barcelona, Birkhäuser/Springer, Basel, 2013, pp. 135–198.

[FP85] U. Fincke and M. Pohst, *Improved methods for calculating vectors of short length in a lattice, including a complexity analysis*, Math. Comp. **44** (1985), no. 170, 463–471. MR 777278

[GK19] Konstantin Golubev and Amitay Kamber, *Cutoff on hyperbolic surfaces*, Geom. Dedicata **203** (2019), 225–255. MR 4027593

[Pag15] Aurel Page, *Computing arithmetic Kleinian groups*, Math. Comp. **84** (2015), no. 295, 2361–2390.

[PAR22] The PARI Group, Univ. Bordeaux, *Pari/gp version 2.14.0*, 2022, available from `http://pari.math.u-bordeaux.fr/`.

[Ric21] James Rickards, *Counting intersection numbers of closed geodesics on Shimura curves*, `https://arxiv.org/abs/2104.01968`, 2021.

[Ric22] ———, *Fundamental domains for Shimura curves*, `https://github.com/JamesRickards-Canada/Fundamental-Domains-for-Shimura-curves`, 2022.

[Ste] Raphael S. Steiner, *Small diameters and generators for arithmetic lattices in $SL_2(\mathbb{R})$ and certain Ramanujan graphs*, `https://arxiv.org/abs/2207.12684`, 2022.

[Voi09] John Voight, *Computing fundamental domains for Fuchsian groups*, J. Théor. Nombres Bordeaux **21** (2009), no. 2, 469–491.

[Voi21] ———, *Quaternion algebras*, Graduate Texts in Mathematics, vol. 288, Springer, Cham, [2021] ©2021. MR 4279905

University of Colorado Boulder, Boulder, Colorado, USA
Email address: `james.rickards@colorado.edu`
URL: `https://math.colorado.edu/~jari2770/`