Fundamental Domains for Shimura curves

James Rickards

CU Boulder

james.rickards@colorado.edu

11 January 2022

Introduction

- Let Γ be a discrete subgroup of $\mathrm{PSL}(2, \mathbb{R})$, which acts on the hyperbolic upper half plane $\mathbb{H}$.

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.

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.

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, which come paired.

Example 1

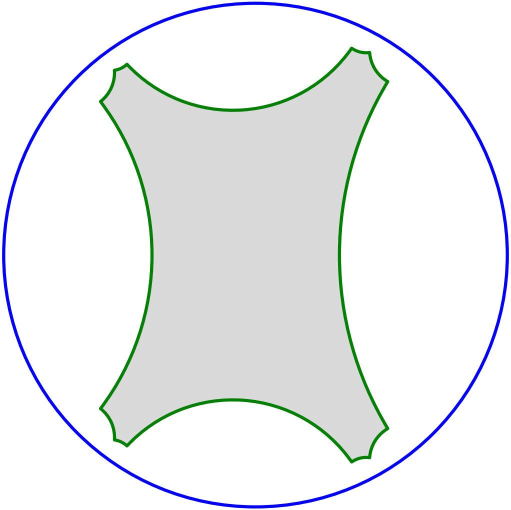

Figure 1: $F = \mathbb{Q}$, $\mathfrak{D} = 21$.

Example 2

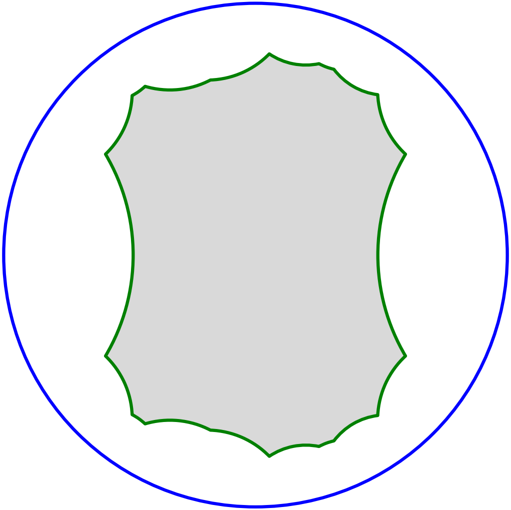

Figure 2: $F = \mathbb{Q}(\sqrt{5})$, $\mathrm{Nm}_{F/\mathbb{Q}}(\mathfrak{D}) = 61$.

Applications

- Computing a presentation for Γ with a minimal set of generators;

Applications

- Computing a presentation for Γ with a minimal set of generators;
- Solving the word problem with respect to this set of generators;

Applications

- Computing a presentation for Γ with a minimal set of generators;
- Solving the word problem with respect to this set of generators;
- Computing the cohomology of the Shimura curve, and the action of Hecke operators;

Applications

- Computing a presentation for Γ with a minimal set of generators;
- Solving the word problem with respect to this set of generators;
- Computing the cohomology of the Shimura curve, and the action of Hecke operators;
- Computing Hilbert modular forms;

Applications

- Computing a presentation for Γ with a minimal set of generators;
- Solving the word problem with respect to this set of generators;
- Computing the cohomology of the Shimura curve, and the action of Hecke operators;
- Computing Hilbert modular forms;
- Efficiently computing the intersection number of pairs of closed geodesics;

Applications

- Computing a presentation for Γ with a minimal set of generators;
- Solving the word problem with respect to this set of generators;
- Computing the cohomology of the Shimura curve, and the action of Hecke operators;
- Computing Hilbert modular forms;
- Efficiently computing the intersection number of pairs of closed geodesics;
- And many more!

Arithmetic Fuchsian groups

- Let F be a totally real number field, and B a quaternion algebra over F of discriminant $\mathfrak{D}$ that is ramified at all but one infinite place.

Arithmetic Fuchsian groups

- Let F be a totally real number field, and B a quaternion algebra over F of discriminant $\mathfrak{D}$ that is ramified at all but one infinite place.
- Take ι to be a corresponding embedding $\iota : B \to \text{Mat}(2, \mathbb{R})$.

Arithmetic Fuchsian groups

- Let F be a totally real number field, and B a quaternion algebra over F of discriminant $\mathfrak{D}$ that is ramified at all but one infinite place.
- Take ι to be a corresponding embedding $\iota : B \to \mathrm{Mat}(2, \mathbb{R})$.
- Let O be a maximal order in B, and let $\mathrm{O}_{N=1}$ be the group of elements of reduced norm 1 in O.

Arithmetic Fuchsian groups

- Let F be a totally real number field, and B a quaternion algebra over F of discriminant $\mathfrak{D}$ that is ramified at all but one infinite place.
- Take ι to be a corresponding embedding $\iota : B \to \mathrm{Mat}(2, \mathbb{R})$.
- Let O be a maximal order in B, and let $\mathrm{O}_{N=1}$ be the group of elements of reduced norm 1 in O.
- Then $\Gamma_{\mathrm{O}} := \iota(\mathrm{O}_{N=1})/\{\pm 1\} \subseteq \mathrm{PSL}(2, \mathbb{R})$ is a discrete subgroup.

Arithmetic Fuchsian groups

- Let F be a totally real number field, and B a quaternion algebra over F of discriminant $\mathfrak{D}$ that is ramified at all but one infinite place.
- Take ι to be a corresponding embedding $\iota : B \to \mathrm{Mat}(2, \mathbb{R})$.
- Let $\mathcal{O}$ be a maximal order in B, and let $\mathcal{O}_{N=1}$ be the group of elements of reduced norm 1 in $\mathcal{O}$.
- Then $\Gamma_{\mathcal{O}} := \iota(\mathcal{O}_{N=1})/\{\pm 1\} \subseteq \mathrm{PSL}(2, \mathbb{R})$ is a discrete subgroup.
- We will be focusing on computing Dirichlet domains for $\Gamma_{\mathcal{O}}$.

Arithmetic Fuchsian group initialization

- John Voight has some lists of totally real number fields on his website (up to degree 10).

Arithmetic Fuchsian group initialization

- John Voight has some lists of totally real number fields on his website (up to degree 10).
- The algebras package in PARI allows us to initialize quaternion algebras with a maximal order.

Arithmetic Fuchsian group initialization

- John Voight has some lists of totally real number fields on his website (up to degree 10).
- The algebras package in PARI allows us to initialize quaternion algebras with a maximal order.
- Quaternion algebras can be initialized by specifying the ramification.

Arithmetic Fuchsian group initialization

- John Voight has some lists of totally real number fields on his website (up to degree 10).
- The algebras package in PARI allows us to initialize quaternion algebras with a maximal order.
- Quaternion algebras can be initialized by specifying the ramification.

```
?  F=nfinit(y^3-5*y+1);
?  A1=alginit(F, [y-1, -5]);
```

Arithmetic Fuchsian group initialization

- John Voight has some lists of totally real number fields on his website (up to degree 10).
- The algebras package in PARI allows us to initialize quaternion algebras with a maximal order.
- Quaternion algebras can be initialized by specifying the ramification.

```
?  F=nfinit(y^3-5*y+1);
?  A1=alginit(F, [y-1, -5]);
?  I1=idealprimedec(F, 5)[1];
?  I2=idealprimedec(F, 17)[1]);
?  A2=alginit(F, [2, [[I1, I2], [1, 1]], [1, 1, 0]]);
```

General algorithm I

- In 2009, John Voight published an algorithm to compute the fundamental domain ([Voi09]), which was implemented in Magma.

General algorithm I

- In 2009, John Voight published an algorithm to compute the fundamental domain ([Voi09]), which was implemented in Magma.
- The outline of the algorithm is:
  1. Compute $\mu = \mu(\Gamma_O)$ via theoretical means;

General algorithm I

- In 2009, John Voight published an algorithm to compute the fundamental domain ([Voi09]), which was implemented in Magma.
- The outline of the algorithm is:
  1. Compute $\mu = \mu(\Gamma_{\mathcal{O}})$ via theoretical means;
  2. Enumerate some elements of $\Gamma_{\mathcal{O}}$, and store them in a (finite) set G (algebraic part);

General algorithm I

- In 2009, John Voight published an algorithm to compute the fundamental domain ([Voi09]), which was implemented in Magma.
- The outline of the algorithm is:
  1. Compute $\mu = \mu(\Gamma_{\mathcal{O}})$ via theoretical means;
  2. Enumerate some elements of $\Gamma_{\mathcal{O}}$, and store them in a (finite) set G (algebraic part);
  3. Compute the normalized basis of G, i.e. the fundamental domain for $\langle G \rangle$ (geometric part);

General algorithm I

- In 2009, John Voight published an algorithm to compute the fundamental domain ([Voi09]), which was implemented in Magma.
- The outline of the algorithm is:
  1. Compute $\mu = \mu(\Gamma_{\mathcal{O}})$ via theoretical means;
  2. Enumerate some elements of $\Gamma_{\mathcal{O}}$, and store them in a (finite) set G (algebraic part);
  3. Compute the normalized basis of G, i.e. the fundamental domain for $\langle G \rangle$ (geometric part);
  4. If the area of the domain is $\mu(\Gamma_{\mathcal{O}})$, stop. Otherwise, go back to step 2.

General algorithm I

- In 2009, John Voight published an algorithm to compute the fundamental domain ([Voi09]), which was implemented in Magma.
- The outline of the algorithm is:
  1. Compute $\mu = \mu(\Gamma_{\mathcal{O}})$ via theoretical means;
  2. Enumerate some elements of $\Gamma_{\mathcal{O}}$, and store them in a (finite) set G (algebraic part);
  3. Compute the normalized basis of G, i.e. the fundamental domain for $\langle G \rangle$ (geometric part);
  4. If the area of the domain is $\mu(\Gamma_{\mathcal{O}})$, stop. Otherwise, go back to step 2.
- The running times were okay for small examples, but they did not scale well.

General algorithm II

- In 2015, Aurel Page generalized this algorithm to Kleinian groups ([Pag15]). His method to generate elements was probabilistic, and performed much better than Voight's method.

General algorithm II

- In 2015, Aurel Page generalized this algorithm to Kleinian groups ([Pag15]). His method to generate elements was probabilistic, and performed much better than Voight's method.
- Both the geometric and enumeration methods were running in $O(\mu^2)$ time, with the geometry generally having the larger constant.

General algorithm II

- In 2015, Aurel Page generalized this algorithm to Kleinian groups ([Pag15]). His method to generate elements was probabilistic, and performed much better than Voight's method.
- Both the geometric and enumeration methods were running in $O(\mu^2)$ time, with the geometry generally having the larger constant.
- The Magma implementation for this is available from his website.

My contributions

- Improved geometric algorithms that run in $O(\mu \log(\mu))$ time.

My contributions

- Improved geometric algorithms that run in $O(\mu \log(\mu))$ time.
- Specialized Page's probabilistic enumeration to Fuchsian groups.

My contributions

- Improved geometric algorithms that run in $O(\mu \log(\mu))$ time.
- Specialized Page's probabilistic enumeration to Fuchsian groups.
- Compiled large amounts of data to justify choices of constants.

My contributions

- Improved geometric algorithms that run in $O(\mu \log(\mu))$ time.
- Specialized Page's probabilistic enumeration to Fuchsian groups.
- Compiled large amounts of data to justify choices of constants.
- Made various code optimizations for even more speed!

My contributions

- Improved geometric algorithms that run in $O(\mu\log(\mu))$ time.
- Specialized Page's probabilistic enumeration to Fuchsian groups.
- Compiled large amounts of data to justify choices of constants.
- Made various code optimizations for even more speed!
- Code is written in PARI, and is publicly available on GitHub ([Ric21a]).

My contributions

- Improved geometric algorithms that run in $O(\mu \log(\mu))$ time.
- Specialized Page's probabilistic enumeration to Fuchsian groups.
- Compiled large amounts of data to justify choices of constants.
- Made various code optimizations for even more speed!
- Code is written in PARI, and is publicly available on GitHub ([Ric21a]).
- Python program to view and explore the computed fundamental domain (and closed geodesics).

My contributions

- Improved geometric algorithms that run in $O(\mu \log(\mu))$ time.
- Specialized Page's probabilistic enumeration to Fuchsian groups.
- Compiled large amounts of data to justify choices of constants.
- Made various code optimizations for even more speed!
- Code is written in PARI, and is publicly available on GitHub ([Ric21a]).
- Python program to view and explore the computed fundamental domain (and closed geodesics).
- See [Ric21b] for more details.

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

deg(F)	disc(F)	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

deg(F)	disc(F)	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\mathrm{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s
4	14656	17	469.145	41m 28s	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

deg(F)	disc(F)	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s
4	14656	17	469.145	41m 28s	12.107s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

deg(F)	disc(F)	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s
4	14656	17	469.145	41m 28s	12.107s
5	5763833	1	4490.383	31 days 19.9h	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

deg(F)	disc(F)	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s
4	14656	17	469.145	41m 28s	12.107s
5	5763833	1	4490.383	31 days 19.9h	20m 22.5s

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

deg(F)	disc(F)	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s
4	14656	17	469.145	41m 28s	12.107s
5	5763833	1	4490.383	31 days 19.9h	20m 22.5s
7	20134393	119	1507.964	25 days 21.4h	

Timing comparison

Computations run on the same McGill server (which is slow; my current office computer is about twice as fast).

Table 1: Running times of the PARI versus the Magma implementation.

$\deg(F)$	$\operatorname{disc}(F)$	$N(\mathfrak{D})$	Area	t(MAGMA)	t(PARI)
1	1	33	20.943	13.190s	0.022s
1	1	793	753.982	4h 22m	1.718s
2	33	37	226.195	4m 57s	0.946s
2	44	79	571.770	69m 43s	3.142s
3	473	99	418.879	28h 56m	4.382s
4	14656	17	469.145	41m 28s	12.107s
5	5763833	1	4490.383	31 days 19.9h	20m 22.5s
7	20134393	119	1507.964	25 days 21.4h	20m 14.9s

Total running time

- The expected running time is

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

where $\mu = \mu(\Gamma_{\mathcal{O}})$, and c_1 and c_2 depend on $n = \deg(F)$.

Total running time

- The expected running time is

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

  where $\mu = \mu(\Gamma_{\mathcal{O}})$, and c_1 and c_2 depend on $n = \deg(F)$.

- The constants cause the geometry to dominate for small areas, especially for n small.

Running times I

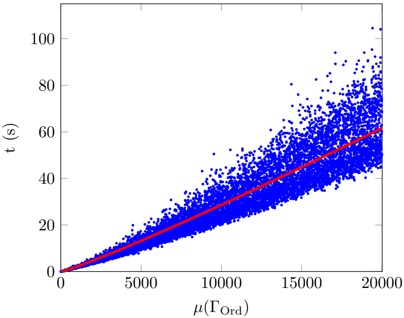

Figure 3: Time to compute the fundamental domain, $n = 1$.

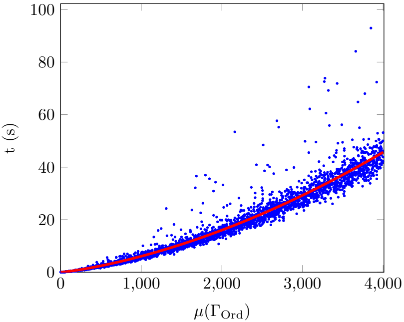

Figure 4: Time to compute the fundamental domain, $n = 2$.

Running times II

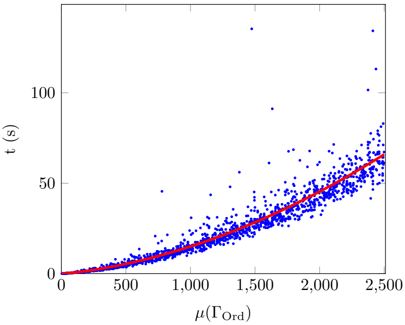

Figure 5: Time to compute the fundamental domain, $n = 3$.

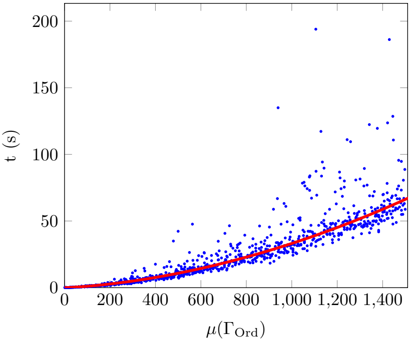

Figure 6: Time to compute the fundamental domain, $n = 4$.

Key implemented methods

- `U=algfdom(A)`: computes a Dirichlet domain for A. Can also supply an Eichler order instead of the already computed maximal order.

Key implemented methods

- `U=algfdom(A)`: computes a Dirichlet domain for A. Can also supply an Eichler order instead of the already computed maximal order.
- `P=algfdompresentation(U)`: computes a presentation for the group.

Key implemented methods

- `U=algfdom(A)`: computes a Dirichlet domain for A. Can also supply an Eichler order instead of the already computed maximal order.
- `P=algfdompresentation(U)`: computes a presentation for the group.
- `W=algfdomword(g, P, U)`: computes g as a word in terms of the presentation.

Key implemented methods

- `U=algfdom(A)`: computes a Dirichlet domain for A. Can also supply an Eichler order instead of the already computed maximal order.
- `P=algfdompresentation(U)`: computes a presentation for the group.
- `W=algfdomword(g, P, U)`: computes g as a word in terms of the presentation.
- `python_printfdom(U, "fdexample")`: prints the fundamental domain data into a file, ready to be viewed with python.

Code in action

Since I can't embed gp in LaTeX, we will switch windows.

Acknowledgments and References

This research was supported by an NSERC Vanier Scholarship.

Aurel Page.
Computing arithmetic Kleinian groups.
Math. Comp., 84(295):2361–2390, 2015.

James Rickards.
Fundamental domains for Shimura curves.
https://github.com/JamesRickards-Canada/Fundamental-Domains-for-Shimura-curves, 2021.

James Rickards.
Improved computation of fundamental domains for arithmetic Fuchsian groups.
https://arxiv.org/abs/2110.11503, 2021.

John Voight.
Computing fundamental domains for Fuchsian groups.
J. Théor. Nombres Bordeaux, 21(2):469–491, 2009.