JOHANNES KEPLER
UNIVERSITY LINZ

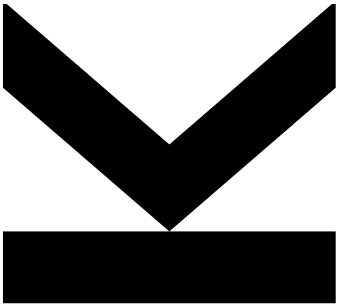
Submitted by
Paul Kainberger

Submitted at
Research Institute for Symbolic Computation (RISC)

Supervisor
Univ.-Prof. Dr.
Peter Paule

August 2021

Drawing Fundamental Domains for Congruence Subgroups using Mathematica



Master's Thesis

to obtain the academic degree of

Diplom-Ingenieur

in the Master's Program

Computer Mathematics

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Statutory Declaration

I hereby declare under oath that the submitted Master's Thesis has been written solely by me without any third-party assistance, information other than provided sources or aids have not been used and those used have been fully documented. Sources for literal, paraphrased and cited quotes have been accurately credited.
The submitted document here present is identical to the electronically submitted text document.

Linz/Date

Paul Kainberger

Acknowledgement

First and foremost, I am very thankful for my supervisor Univ.-Prof. Dr. Paule always helping me during the whole process of writing this thesis and the Mathematica package behind it. His knowledge, engagement and enthusiasm about the subject motivated me a lot. The program was his idea and influenced by Dr.in Helena A. Verrill's Java program [19].

Secondly, I appreciate every participant of the Algorithmic Combinatorics Seminar at the Research Institute for Symbolic Computation (RISC) reviewing the presentation of the Mathematica package and providing me with suggestions for improvement. Special thanks go to Dr. Ralf Hemmecke who moreover helped me set up a web page for the program.

Furthermore, I would like to thank and emphasise the help of my colleagues and dear friends Dipl.-Ing. Philipp Nuspl, who not only assisted me on my thesis but throughout my whole studies, as well as Louisa Hofmann for spending many productive hours in the library together and helping me.

Lastly, I must express my profound gratitude to my parents and sisters. Without their support and encouragement, these years of studies would not have been possible. Thank you.

Abstract

The goal of this thesis is the visualisation of fundamental domains for congruence subgroups in the modular group $\mathrm{SL}_2(\mathbb{Z})$, also denoted by $\mathrm{SL}(2,\mathbb{Z})$, using the software *Wolfram Mathematica* [7].

The first part of the thesis is dedicated to basic theory and propositions needed in order to study fundamental domains. We begin by defining the special linear group $\mathrm{SL}_2(\mathbb{Z})$ and a group action on the upper half complex plane. After the examination of congruence subgroups and fundamental domains, we study an algorithm of how to construct a list of right coset representatives, which is our basis for visualising fundamental domains.

The second part centres on the Mathematica package *DrawFunDoms.m* which was carried out as part of this master's thesis. It contains a graphical user interface that allows one to draw fundamental domains for different congruence subgroups. In addition, one becomes able to comprehend the way matrices in the special linear group $\mathrm{SL}_2(\mathbb{Z})$ act on the upper half complex plane and how equivalence relations between these matrices as well as between complex points behave. Afterwards, further functions available in the Mathematica package are discussed based on examples. Finally, a tutorial on how to define and add congruence subgroups which are not included in the package by default like $\Gamma(N)$, $\Gamma_0(N)$, $\Gamma_1(N)$, $\Gamma^0(N)$, $\Gamma^1(N)$, $\Gamma(N,M,P)$ and Γ_θ concludes the thesis.

Kurzfassung

Ziel dieser Arbeit ist die Visualisierung von Fundamentalbereichen bekannter Kongruenzuntergruppen zur Modulgruppe $\mathrm{SL}_2(\mathbb{Z})$, auch $\mathrm{SL}(2,\mathbb{Z})$ genannt, mithilfe des Softwarepakets *Wolfram Mathematica* [7].
Im ersten Teil der Masterarbeit werden grundlegende Definitionen und Aussagen, welche essenziell für die Auseinandersetzung mit Fundamentalbereichen sind, besprochen. Wir beginnen mit den Definitionen der speziellen linearen Gruppe $\mathrm{SL}_2(\mathbb{Z})$ und einer Gruppenwirkung auf der oberen Halbebene. Nach der Untersuchung von Kongruenzuntergruppen sowie Fundamentalbereichen, analysieren wir einen Algorithmus zur Konstruktion einer Liste von Rechtsnebenklassen, welche die Basis für die Visualisierung von Fundamentalbereichen darstellt.
Der zweite Teil widmet sich dem Mathematica Paket *DrawFunDoms.m*, welches im Rahmen dieser Masterarbeit entstand. Dieses beinhaltet eine grafische Benutzeroberfläche, mithilfe derer man interaktiv Fundamentalbereiche für verschiedene Kongruenzuntergruppen zeichnen kann. Zudem lernt man zu verstehen, wie Matrizen der speziellen linearen Gruppe $\mathrm{SL}_2(\mathbb{Z})$ auf der oberen Halbebene wirken und wie sich dabei Aquivalenzrelationen zwischen diesen Matrizen oder auch zwischen komplexen Punkten verhalten. Schließlich werden weitere verfügbare Funktionen des Mathematica Pakets anhand von Beispielen besprochen. Eine Anleitung zum Definieren und Hinzufügen der nicht standardmäßig im Package enthaltenen Kongruenzuntergruppen $\Gamma(N)$, $\Gamma_0(N)$, $\Gamma_1(N)$, $\Gamma^0(N)$, $\Gamma^1(N)$, $\Gamma(N,M,P)$ und Γ_θ schließt die Arbeit ab.

Contents

1 Theory **3**

1.1 Special Linear Group $SL_2(\mathbb{Z})$ . . . 3
1.2 Group Action on $\mathbb{H}$ . . . 6
1.3 Congruence Subgroups . . . 7
1.4 Fundamental Domains . . . 9
1.5 An Algorithm for finding Fundamental Domains for Congruence Subgroups . . . 11
1.5.1 Visualisation . . . 14

2 A Mathematica Program for drawing Fundamental Domains **19**

2.1 Motivation . . . 19
2.2 The Main Program . . . 20
2.2.1 Control panel "$SL_2(\mathbb{Z})$ and its Congruence Subgroups" . . . 21
2.2.2 Control panel "Select Point" . . . 23
2.2.3 Control panel "Select Matrix" . . . 25
2.2.4 Graphic . . . 26
2.3 Other useful functions . . . 26
2.4 How to manually add congruence subgroups . . . 39
2.5 Examples . . . 43

Bibliography **49**

1 Theory

This chapter is mainly based on Joseph Lehner's books *Lectures on Modular Forms* [10, Chapter 1], *A Short Course in Automorphic Functions* [9, Chapter 1] as well as Bruno Schoeneberg's *Elliptic Modular Forms* [16, Chapter], unless explicitly stated otherwise. The reader is expected to bring basic knowledge about algebra, group theory and number theory, which can be found in many books, such as Thomas W. Hungerford's *Algebra* [6].

1.1 Special Linear Group $\mathbf{SL}_2(\mathbb{Z})$

The whole concept of fundamental domains is mapping points on the upper half complex plane $\{x + iy : y > 0\} =: \mathbb{H}$. Such mappings can be performed by so called *linear fractional transformations*.

Definition 1.1.1 (Linear Fractional Transformation)**.** Let $a, b, c, d \in \mathbb{Z}$ with $ad - bc = 1$. The function $\mathbb{C} \to \mathbb{C}$,

$$z \mapsto \frac{az + b}{cz + d}$$

is called a *linear fractional transformation*.

The name derives from the fact that we have a fractional transformation and both nominator and denominator are linear transformations. A linear fractional transformation is a special case of a Möbius transformation, which solely requires from a, b, c, d to be complex numbers with $ad - bc \neq 0$. The condition $ad - bc \neq 0$ guarantees that the function is non-linear. [13]
It is worth noting that these transformations' parameters are not unique. In fact, for values $a, b, c, d \in \mathbb{Z}$, the according linear fractional transformation is equivalent to the one defined by $-a, -b, -c, -d$.

Example 1.1.2. Consider the linear fractional transformation $\mathbb{H} \to \mathbb{H}$ with $z \mapsto \frac{-z-1}{z}$. Following the notation of the above definition, this transformation could have the parameters $a = 1, b = 1, c = -1, d = 0$ as well as $a = -1, b = -1, c = 1, d = 0$.

As mentioned, we aim to map points on the upper half complex plane. If we want to use linear fractional transformations, we need to make sure that $\mathbb{H}$ is invariant under them. Let $z = x + yi \in \mathbb{H}$ with $|z| := \sqrt{x^2 + y^2}$ and $a, b, c, d \in \mathbb{Z}$ with $ad - bc = 1$. Then

$$\frac{az + b}{cz + d} = \frac{ax + b + ayi}{cx + d + cyi} = \frac{z + (ad - bc)yi}{|cx + d + cyi|^2},$$

so $\mathsf{Im}\left(\frac{az+b}{cz+d}\right) = \frac{(ad-bc)\mathsf{Im}(z)}{|cz+d|^2} > 0$.

Another, more convenient way of defining such transformations on the upper half complex plane is to let specific matrices act on $\mathbb{H}$. These matrices form a *group*.

Definition 1.1.3 (Group, [6])**.** A set G, together with a binary operation $\cdot : G \times G \to G$, forms a *group* if and only if

- $\forall a, b, c \in G : (a \cdot b) \cdot c = a \cdot (b \cdot c)$ (Associativity),
- $\exists e \in G \; \forall a \in G : a \cdot e = e \cdot a = a$ (Identity element),
- $\forall a \in G \; \exists a^{-1} \in G : a \cdot a^{-1} = a^{-1} \cdot a = e$ (Inverse element).

From now on, we will call a set G a group, if $(G, \cdot)$ is a group and the operation $\cdot$ is clear from the context. In our case that is the usual matrix multiplication:

Definition 1.1.4 (Special linear group)**.** The *special linear group* of degree 2 consists of all 2×2 integer matrices having determinant 1 and is denoted by $\mathrm{SL}_2(\mathbb{Z})$ (or sometimes by $\mathrm{SL}(2, \mathbb{Z})$). I.e.

$$\mathrm{SL}_2(\mathbb{Z}) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} : a, b, c, d \in \mathbb{Z}, \det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = 1 \right\}$$

Remark 1.1.5. These matrices have integer entries and $\det\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) = ad - bc$, so this coincides with the conditions for the parameters of a linear fractional transformation. The requirement $\det\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) = 1 \neq 0$ guarantees the existence of inverses in the group.

Definition 1.1.6 (Subgroup, [6])**.** Let G be a group. We say H is a *subgroup* of G, if and only if $H \subseteq G$ and H forms a group. We say H is a *proper subgroup* of G, if and only if $H \subset G$ and H forms a group.

Remark 1.1.7. The special linear group is a subgroup of the *general linear group* of degree 2, $\mathrm{GL}_2(\mathbb{Z})$ or also called $\mathrm{GL}(2, \mathbb{Z})$, which requires the matrices to be regular, i.e. the determinant must be unequal zero. Compare this relation between $\mathrm{SL}_2(\mathbb{Z})$ and $\mathrm{GL}_2(\mathbb{Z})$ with the one between linear fractional transformations and Möbius transformations.

Knowing that $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)^{-1} = \left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right)$, it is an easy exercise to check that $\mathsf{SL}_2(\mathbb{Z})$ is indeed a group. Furthermore, $\mathsf{SL}_2(\mathbb{Z})$ is finitely generated by two matrices, which will be of great importance for the Mathematica package described in Chapter 2.

Lemma 1.1.8. *$SL_2(\mathbb{Z})$ is generated by the matrices S and T with*

$$S \coloneqq \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \qquad T \coloneqq \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Proof, [2]. Obviously, $\langle S, T\rangle \subseteq \mathrm{SL}_2(\mathbb{Z})$. To prove $\langle S, T\rangle \supseteq \mathrm{SL}_2(\mathbb{Z})$, let $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$. We want to show that A can be expressed by a multiplicative composition of S and T in a constructive way and distinguish between three cases:

Case 1: $c = 0$.
The determinant of A is 1, so $A = \left(\begin{smallmatrix} \pm 1 & b \\ 0 & \pm 1 \end{smallmatrix}\right)$ and $A \in \{T^b, -T^{-b}\}$.

Case 2: $c \neq 0$ and $|a| < |c|$.
Then $SA = \left(\begin{smallmatrix} -c & -d \\ a & b \end{smallmatrix}\right) =: \left(\begin{smallmatrix} \tilde{a} & \tilde{b} \\ \tilde{c} & \tilde{d} \end{smallmatrix}\right)$ and $|\tilde{a}| > |\tilde{c}|$ which leads to Case 1 or Case 3.

Case 3: $c \neq 0$ and $|a| \geq |c|$.
We divide a by c, so $a = cq + r$ with $r, q \in \mathbb{Z}$ and $0 \leq r < |c|$. Then $T^{-q}A = \left(\begin{smallmatrix} r & b-qd \\ c & d \end{smallmatrix}\right)$ leads to Case 2.

Since $|a|$ and $|c|$ keep decreasing, this algorithm terminates and $\langle S, T\rangle \supseteq \mathrm{SL}_2(\mathbb{Z})$. □

Example 1.1.9. The above proof provides an algorithm to express any matrix in $\mathrm{SL}_2(\mathbb{Z})$ as a multiplicative composition of the generators S and T. Let us do that for $A = \left(\begin{smallmatrix} -5 & -7 \\ 3 & 4 \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$.
First, Case 3 and $-5 = (-2) \cdot 3 + 1$ lead to

$$T^2 A = \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix}.$$

Next, Case 2 gives

$$ST^2 A = \begin{pmatrix} -3 & -4 \\ 1 & 1 \end{pmatrix}.$$

Then, Case 3 and $-3 = (-3) \cdot 1 + 0$ give

$$T^3 ST^2 A = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix}.$$

Finally, Case 2 and Case 1 result in

$$ST^3 ST^2 A = \begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix} = -T = S^2 T.$$

Therefore, $A = T^{-2} S^{-1} T^{-3} S^{-1} (S^2 T) = T^{-2} S^{-1} T^{-3} ST$.

Remark 1.1.10 ([2])**.** Since $T^n = \left(\begin{smallmatrix} 1 & n \\ 0 & 1 \end{smallmatrix}\right)$, the element T has infinite order in $\mathrm{SL}_2(\mathbb{Z})$. However, S has order 4 and ST has order 6. Apparently, $\mathrm{SL}_2(\mathbb{Z}) = \langle S, T\rangle = \langle S, ST\rangle$, so the group $\mathrm{SL}_2(\mathbb{Z})$ can be generated by two matrices of finite order.

Remark 1.1.11 ([2])**.** Some authors use the matrices T and $U \coloneqq \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$ as the generators of $\mathrm{SL}_2(\mathbb{Z})$. This is equivalent to our generating set: Both T and U are in $\mathrm{SL}_2(\mathbb{Z})$, so $\langle T, U\rangle \subseteq \mathrm{SL}_2(\mathbb{Z})$. On the other hand, since $S = T^{-1}UT^{-1}$, also $\langle T, U\rangle \supseteq \mathrm{SL}_2(\mathbb{Z})$ holds.

1.2 Group Action on $\mathbb{H}$

We start off this section with the notion of a group action.

Definition 1.2.1 (Group action, [6])**.** If G is a group and X is a set, then a *group action* φ of G on X is a function

$$\varphi : G \times X \to X, (g, x) \mapsto \varphi(g, x)$$

that satisfies

- $\forall x \in X : \varphi(e, x) = x$ (Identity),
- $\forall g, h \in G, \forall x \in X : \varphi(gh, x) = \varphi(g, \varphi(h, x))$ (Compatibility).

We make $\mathrm{SL}_2(\mathbb{Z})$ act on the upper half complex plane via the following group action: For $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$ and $z \in \mathbb{H}$, define

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} z \coloneqq \frac{az + b}{cz + d}. \tag{1.1}$$

Additionally, we have

$$\lim_{z\to\infty} \begin{pmatrix} a & b \\ c & d \end{pmatrix} z = \lim_{z\to\infty} \frac{a + \frac{b}{z}}{c + \frac{d}{z}} = \frac{a}{c}.$$

From now on, for a matrix $A \in \mathrm{SL}_2(\mathbb{Z})$ and a point $z \in \mathbb{C} \cup \{\infty\}$, we refer to Az as A acting on z as specified above. This mapping defines indeed a group action: In Section 1.1 we already showed that $\mathbb{H}$ is invariant under linear fractional transformations, which adapts to this mapping too: The imaginary part of a complex number is invariant under a matrix $\in \mathrm{SL}_2(\mathbb{Z})$ acting on it. The identity property is clearly fulfilled: $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) z = z$. Concerning the compatibility, let $A \coloneqq \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right), \tilde{A} \coloneqq \left(\begin{smallmatrix} \tilde{a} & \tilde{b} \\ \tilde{c} & \tilde{d} \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$ and $z \in \mathbb{H}$. Then

$$(A\tilde{A})z = \begin{pmatrix} a\tilde{a} + b\tilde{c} & a\tilde{b} + b\tilde{d} \\ c\tilde{a} + d\tilde{c} & c\tilde{b} + d\tilde{d} \end{pmatrix} z = \frac{a(\tilde{a}z + \tilde{b}) + b(\tilde{c}z + \tilde{d})}{c(\tilde{a}z + \tilde{b}) + d(\tilde{c}z + \tilde{d})} = \frac{\frac{a(\tilde{a}z+\tilde{b})}{\tilde{c}z+\tilde{d}} + b}{\frac{a(\tilde{a}z+\tilde{b})}{\tilde{c}z+\tilde{d}} + d} = A(\tilde{A}z).$$

Let us investigate, in what way the special linear group's generators S and T act on $z \in \mathbb{H}$:

$$Sz = \frac{-1}{z},$$
$$Tz = z + 1.$$

If we compare the group action with linear fractional transformations, we notice that the group action also does not distinguish between matrices that differ by a sign (i.e. $Az = (-A)z$). In this sense, one could regard the matrices A and $-A$ as equivalent in $\mathsf{SL}_2(\mathbb{Z})$, even though they are different group elements. This fact will not become a problem in our further calculations in $\mathsf{SL}_2(\mathbb{Z})$, since $-I = S^2$ is a group element. However later on, when considering subgroups which do not contain the negative identity matrix, we need to keep that in mind.

1.3 Congruence Subgroups

In this section, we will define congruence subgroups of $\mathsf{SL}_2(\mathbb{Z})$, which will be of special interest when studying fundamental domains later. The most important subgroup of $\mathsf{SL}_2(\mathbb{Z})$ in this thesis is the so called *principal congruence subgroup*:

Definition 1.3.1 (Principal congruence subgroup)**.** We define the *principal congruence subgroup of level* $N \in \mathbb{N}$ as follows:

$$\Gamma(N) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z}) : \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mod N \right\}$$

The reduction modulo N is regarded entry-wise. Obviously, $\Gamma(N)$ is a subgroup of $\mathsf{SL}_2(\mathbb{Z})$ and $\Gamma(1) = \mathsf{SL}_2(\mathbb{Z})$. All subgroups of $\mathsf{SL}_2(\mathbb{Z})$ in this thesis are closely related to the principal congruence subgroup.

Definition 1.3.2 (Congruence subgroup)**.** Every subgroup of $\mathrm{SL}_2(\mathbb{Z})$ that contains $\Gamma(N)$ for some $N \in \mathbb{N}$, is called a *congruence subgroup*. The *level* N of such a congruence subgroup is the largest N such that it contains $\Gamma(N)$.

Let us list some congruence subgroups with $N \in \mathbb{N}$:

$$\Gamma_0(N) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : c \equiv 0 \mod N \right\},$$

$$\Gamma_1(N) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : c \equiv 0 \mod N \text{ and } a \equiv 1 \mod N \right\},$$

$$\Gamma^0(N) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : b \equiv 0 \mod N \right\},$$

$$\Gamma^1(N) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : b \equiv 0 \mod N \text{ and } d \equiv 1 \mod N \right\},$$

$$\Gamma_0^0(N) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : b \equiv c \equiv 0 \mod N \right\}.$$

All these subgroups are special cases of the more general group $\Gamma(N, M, P)$ with $N, M, P \in \mathbb{N}$:

$$\Gamma(N, M, P) \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : M | NP \wedge \right.$$
$$a \equiv d \equiv 1 \mod M \wedge$$
$$c \equiv 0 \mod N \wedge$$
$$\left. b \equiv 0 \mod P \right\}.$$

For instance, we have $\Gamma(N) = \Gamma(N, N, N)$, $\Gamma_0(N) = \Gamma(N, 1, 1)$ and $\Gamma_1(N) = \Gamma(N, N, 1)$ [3]. Since $\Gamma(N, M, P)$ is a congruence subgroup of level $\mathsf{lcm}(N, M, P)$, all the subgroups mentioned above are congruence subgroups.

Remark 1.3.3. One might wonder why the condition $M|NP$ is necessary here. For two matrices $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right), \left(\begin{smallmatrix} \tilde{a} & \tilde{b} \\ \tilde{c} & \tilde{d} \end{smallmatrix}\right) \in \Gamma(N, M, P)$, we expect the product to be a group element as well [6, Theorem 2.5]. The matrix product is given by

$$\begin{pmatrix} a\tilde{a} + b\tilde{c} & a\tilde{b} + b\tilde{d} \\ c\tilde{a} + d\tilde{c} & c\tilde{b} + d\tilde{d} \end{pmatrix}.$$

In order to make sure that $b\tilde{c} \equiv 0 \mod M$ and $c\tilde{b} \equiv 0 \mod M$, it requires the constraint $M|NP$.

These groups, in particular the so called *Hecke congruence subgroup* Γ_0, have many applications, like in number theory. We consider as well the *Theta-Subgroup* (also known as the *Modular Group Lambda*), which is defined by Martin Eichler [5, pages 36-39] and Joseph Lehner [9, page 61], [10, page 14] as follows:

$$\Gamma_\theta \coloneqq \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathsf{SL}_2(\mathbb{Z}) : ac \equiv bd \equiv 0 \mod 2 \right\}.$$

Lemma 1.3.4. *The Theta-Subgroup Γ_θ is a congruence subgroup.*

Proof. First, the Theta-Subgroup is obviously a subset of $\mathrm{SL}_2(\mathbb{Z})$. Therefore, the inverse of a matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma_\theta$ is given by $\left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right)$. We know ac is even, so either $a \equiv 0 \mod 2$ or $c \equiv 0 \mod 2$. Assume $a \equiv 0 \mod 2$. The latter case works equivalently. The condition $ad - bc = 1$ yields that b is an odd number. In order to have $bd \equiv 0 \mod 2$, we claim d to be even, so $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)^{-1} \in \Gamma_\theta$. Further, for $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right), \left(\begin{smallmatrix} \tilde{a} & \tilde{b} \\ \tilde{c} & \tilde{d} \end{smallmatrix}\right) \in \Gamma$, we need to show that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} \tilde{a} & \tilde{b} \\ \tilde{c} & \tilde{d} \end{pmatrix} = \begin{pmatrix} a\tilde{a} + b\tilde{c} & a\tilde{b} + b\tilde{d} \\ c\tilde{a} + d\tilde{c} & c\tilde{b} + d\tilde{d} \end{pmatrix} \in \Gamma.$$

If $ac \equiv bd \equiv \tilde{a}\tilde{c} \equiv \tilde{b}\tilde{d} \equiv 0 \mod 2$, then obviously

$$(a\tilde{a} + b\tilde{c})(c\tilde{a} + d\tilde{c}) = (ac)\tilde{a}^2 + (\tilde{a}\tilde{c})ad + (\tilde{a}\tilde{c})bc + (bd)\tilde{c}^2,$$
$$(a\tilde{b} + b\tilde{d})(c\tilde{b} + d\tilde{d}) = (ac)\tilde{b}^2 + (\tilde{b}\tilde{d})ad + (\tilde{b}\tilde{d})bc + (bd)\tilde{d}^2$$

are even numbers as well. Lastly, since $\Gamma(2) \subset \Gamma_\theta$, the Theta-Subgroup is a congruence subgroup of level 2 by Definition 1.3.2. □

1.4 Fundamental Domains

Large parts of this section are taken from [15, pages 47–62]. In Section 1.1 we defined a group action on the upper half complex plane $\mathbb{H}$. This motivates us to define an equivalence relation between complex points.

Definition 1.4.1 (Equivalence relation on $\mathbb{H}$)**.** Let Γ be a subgroup of $\mathrm{SL}_2(\mathbb{Z})$. We say $z_1, z_2 \in \mathbb{H}$ are *equivalent* and write $z_1 \sim z_2$, if and only if there exists a matrix $A \in \Gamma$, such that $Az_1 = z_2$.

This equivalence relation partitions $\mathbb{H}$ into equivalence classes, which are called *orbits*. For a congruence subgroup Γ, the orbit of $z \in \mathbb{H}$ is denoted by $\Gamma z \coloneqq \{Az | A \in \Gamma\}$.

Definition 1.4.2 (Fundamental domain)**.** A *fundamental domain* (sometimes referred to as *proper fundamental region*) $\mathcal{F}$ of a subgroup of $\mathrm{SL}_2(\mathbb{Z})$ for the group action (1.1) is a subset of $\mathbb{H}$, such that exactly one point for each orbit is contained in $\mathcal{F}$.

Such a fundamental domain is, of course, far from unique. Given a fundamental domain, we could exchange one singular point with an equivalent point in its orbit and receive another fundamental domain. Still, one may find an area of connected points which fulfils the condition, so any congruence subgroup has a fundamental domain consisting of points connected to each other. The following theorem delivers a nice fundamental domain for the special linear group itself:

Theorem 1.4.3. *A fundamental domain of $SL_2(\mathbb{Z})$ is given by*

$$\left\{z \in \mathbb{H} : -\frac{1}{2} \leq Re(z) \leq 0 \wedge 1 \leq |z|\right\} \cup$$
$$\left\{z \in \mathbb{H} : 0 < Re(z) < \frac{1}{2} \wedge 1 < |z|\right\} =: \mathcal{F}.$$

Remark 1.4.4. Figure 1.1 shows a screenshot taken from the Mathematica package described in Chapter 2 and demonstrates the visualisation of the fundamental domain given in Theorem 1.4.3. The thick edges on the left emphasise the set's open and closed boundaries. At the top, the set expands to complex infinity. The two vertices are positioned at $\pm\frac{1}{2} + i\frac{\sqrt{3}}{2}$ of which the one having positive real part is not included in the set.

In order to prove Theorem 1.4.3, we first prepare with a lemma:

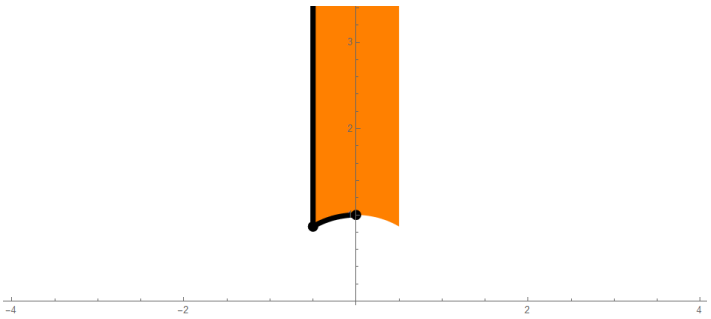


Figure 1.1: A fundamental domain for the group $\mathrm{SL}_2(\mathbb{Z})$.

Lemma 1.4.5. *For $z \in \mathbb{H}$ with $|z| < 1$ we have $Im(z) < Im(Sz)$.*

Proof. Let $z = x + yi \in \mathbb{H}$ with $|z| < 1$. Then

$$\mathrm{Im}(Sz) = \mathrm{Im}\left(\frac{-1}{x+yi}\right) = \mathrm{Im}\left(\frac{-x+yi}{x^2+y^2}\right) = \frac{y}{|z|} > y = \mathrm{Im}(z).$$

□

Proof of Theorem 1.4.3. [17, page 78f], [1, page 10], [4, page 53] Let $\mathcal{F}$ be the set described in Theorem 1.4.3. The proof that $\mathcal{F}$ is a fundamental domain of $\mathrm{SL}_2(\mathbb{Z})$ is done in 2 steps. First, we want to show that every orbit has a representative in $\mathcal{F}$. The second goal is proving that each orbit has at most one element in $\mathcal{F}$.

Step 1: Let $z = x + yi \in \mathbb{H}$. Prove that there exists an $A = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$, such that $Az \in \mathcal{F}$.

In order to find a matrix A with $Az \in \mathcal{F}$, we first want to maximise the imaginary part $\mathrm{Im}(Az)$. We have

$$\mathrm{Im}(Az) = \mathrm{Im}\left(\frac{az+b}{cz+d}\right) = \mathrm{Im}\left(\frac{(az+b)(c\bar{z}+d)}{|cz+d|^2}\right) = \frac{\mathrm{Im}(z)}{|cz+d|^2} > 0.$$

To receive a maximal $\mathrm{Im}(Az)$, let us minimise $|cz+d|$. Since c and d are integers, for a fixed $n \in \mathbb{N}$ there is only a finite number of pairs (c, d), such that $|cz+d| \leq n$ (the lattice generated by 1 and z has finitely many lattice points inside the unit circle). Therefore, a minimum exists, meaning that there exists a maximal imaginary part for Az. Next, we multiply this matrix A with an appropriate power of T such that $-\frac{1}{2} \leq \mathrm{Re}(T^m Az) \leq \frac{1}{2}$ with $m \in \mathbb{Z}$. Remember, that $\mathrm{Im}(Az) = \mathrm{Im}(T^m Az)$. We now claim that $|T^m Az| \geq 1$. If that was not the case, Lemma 1.4.5 would provoke a contradiction to the maximised $\mathrm{Im}(T^m Az)$. Now either $T^m Az \in \mathcal{F}$ or $|T^m Az| = 1$ and $0 < \mathrm{Re}(T^m Az) < \frac{1}{2}$, in which case $ST^m Az \in \mathcal{F}$.

Step 2: Let $z_1, z_2 \in \mathcal{F}$ with $z_1 \sim z_2$. Prove that $z_1 = z_2$.

Assume that there exists an $A \in \mathrm{SL}_2(\mathbb{Z})$ unequal the identity matrix such that $Az_1 = z_2$.

Without loss of generality, let $\mathrm{Im}(z_1) \leq \mathrm{Im}(z_2)$. Then

$$\mathrm{Im}(z_2) = \mathrm{Im}(Az_1) = \mathrm{Im}\left(\frac{az_1+b}{cz_1+d}\right) = \frac{\mathrm{Im}(z_1)}{|cz_1+d|^2} \geq \mathrm{Im}(z_1),$$

which gives $|cz_1 + d| \leq 1$. For $|c| \geq 2$ we get a contradiction:

$$1 \geq |cz_1+d| \geq \mathrm{Im}(cz_1+d) = \mathrm{Im}(cz_1) = c\,\mathrm{Im}(z_1) \overset{!}{>} c\frac{1}{2}.$$

This leads to the following case distinction:

Case $c = 0$.

Then $A = \pm T^m$ for some $m \in \mathbb{Z}_{\neq 0}$ and $Az_1 = z_1 + m$. This contradicts to the fact that $\mathcal{F}$ has width 1.

Case $c = 1$.

Then $|z_1 + d| \leq |z_1| + |d| \leq 1$, so z_1 lies on the unit circle and $d = 0$. Because of $ad - bc = 1$, we have $b = -1$, so $Az_1 = \frac{az_1-1}{z_1} = a - \frac{1}{z_1} = z_2$. Hence, $a = 0$ and z_2 lies on the unit circle as well with $\mathrm{Im}(z_1) = \mathrm{Im}(z_2)$ and $\mathrm{Re}(z_1) = -\mathrm{Re}(z_2)$. This contradicts to the fact that no point on the unit circle with positive real part lies in $\mathcal{F}$.

Case $c = -1$.

Considering $-A$ instead of A brings us back to the previous case.

□

How could we interpret the described fundamental domain? On the one hand, we can map any given point on the upper half complex plane into the $\mathcal{F}$ using a suitable matrix in $\mathsf{SL}_2(\mathbb{Z})$. On the other hand, we may map the whole fundamental domain by a matrix in $\mathsf{SL}_2(\mathbb{Z})$ and receive another fundamental domain which we call *triangular area*. If we did that with all elements of $\mathsf{SL}_2(\mathbb{Z})$, we would cover the whole upper half complex plane. This leads us to the following conclusion:

Corollary 1.4.6.

$$\mathbb{H} = SL_2(\mathbb{Z})\mathcal{F}.$$

1.5 An Algorithm for finding Fundamental Domains for Congruence Subgroups

Theorem 1.4.3 gives an example for a fundamental domain of $\mathsf{SL}_2(\mathbb{Z})$. The next step is to find and visualise fundamental domains for our congruence subgroups, namely for $\Gamma(N)$, $\Gamma_0(N)$, $\Gamma_1(N)$, $\Gamma^0(N)$, $\Gamma^1(N)$, $\Gamma_0^0(N)$, $\Gamma(N, M, P)$ and Γ_θ, where N, M, P are positive integers and $M|NP$.

These are naturally larger sets than $\mathcal{F}$, since these groups are proper subgroups of the special linear group. During this subsection, let Γ be a congruence subgroup.

Definition 1.5.1 (Equivalence relation on $\mathrm{SL}_2(\mathbb{Z})$)**.** Let Γ be a congruence subgroup. We say that two matrices $A, B \in \mathrm{SL}_2(\mathbb{Z})$ are *equivalent* in Γ and write $A \sim B$, if and only if $AB^{-1} \in \Gamma$.

It is an easy exercise to prove that this is indeed an equivalence relation. Furthermore, each *equivalence class* $[A] = \{B \in \mathrm{SL}_2(\mathbb{Z}) : A \sim B\}$ has infinitely many elements.

Definition 1.5.2. (Right coset, [6, page 38]). Let Γ be a congruence subgroup of $\mathrm{SL}_2(\mathbb{Z})$. For $A \in \mathrm{SL}_2(\mathbb{Z})$, the set

$$\Gamma A = \{BA : B \in \Gamma\}$$

is called *right coset* of Γ in $\mathrm{SL}_2(\mathbb{Z})$.

Definition 1.5.3. (Index of Γ in $\mathrm{SL}_2(\mathbb{Z})$, [6, page 38]). Let Γ be a congruence subgroup of $\mathrm{SL}_2(\mathbb{Z})$. The *index of* Γ *in* $SL_2(\mathbb{Z})$, denoted $[\mathrm{SL}_2(\mathbb{Z}) : \Gamma]$ is the cardinal number of the set of distinct right cosets of Γ in $\mathrm{SL}_2(\mathbb{Z})$.

Remark 1.5.4 ([2])**.** The principal congruence subgroup can be equivalently to Defintion 1.3.1 defined as the kernel of the reduction homomorphism $\mathrm{SL}_2(\mathbb{Z}) \to \mathrm{SL}_2(\mathbb{Z}/(N))$. Since $\mathrm{SL}_2(\mathbb{Z})/\Gamma(N)$ embeds into the finite group $\mathrm{SL}_2(\mathbb{Z}/(N))$, the principal congruence subgroup has finite index in $\mathrm{SL}_2(\mathbb{Z})$. Thus, each subgroup of $\mathrm{SL}_2(\mathbb{Z})$ containing some $\Gamma(N)$ has finite index in $\mathrm{SL}_2(\mathbb{Z})$.

Remark 1.5.5 (Non-congruence subgroups of $\mathrm{SL}_2(\mathbb{Z})$, [2])**.** There also exist finite–index subgroups of $\mathrm{SL}_2(\mathbb{Z})$ which are non-congruence subgroups. One can prove that for $n \in \mathbb{N}_{\geq 9}$, the kernel of the surjective homomorphism $\varphi : \mathrm{SL}_2(\mathbb{Z}) \to A_n$ is a subgroup of $\mathrm{SL}_2(\mathbb{Z})$ with finite index, which does not contain any $\Gamma(N)$, where $A_n, n \in \mathbb{N}_{\geq 2}$ is the *alternating group* (i.e. the group of even permutations of a finite set of n elements). However, this is a unique property of $\mathrm{SL}_2(\mathbb{Z})$. Bass, Lazard, Serre proved in 1964 as well as Mennicke in 1965 that for $n \geq 3$, every finite–index subgroup of $\mathrm{SL}_n(\mathbb{Z})$ is a congruence subgroup, where $\mathrm{SL}_n(\mathbb{Z})$ denotes the set of all $n \times n$ integer matrices having determinant 1.

As mentioned in Remark 1.5.4, all our mentioned subgroups have finite index in $\mathrm{SL}_2(\mathbb{Z})$. Usually, (still not always!) we have a tendency that N is somewhat directly proportional to the index. This means that the greater N, the lower the chances for an arbitrary matrix in $\mathrm{SL}_2(\mathbb{Z})$ to be member of the subgroup. We can see that behaviour in Figure 1.2. There exist formulae for calculating the index for the most important congruence subgroups, which are implemented in the Mathematica package *DrawFunDoms* (Chapter 2).

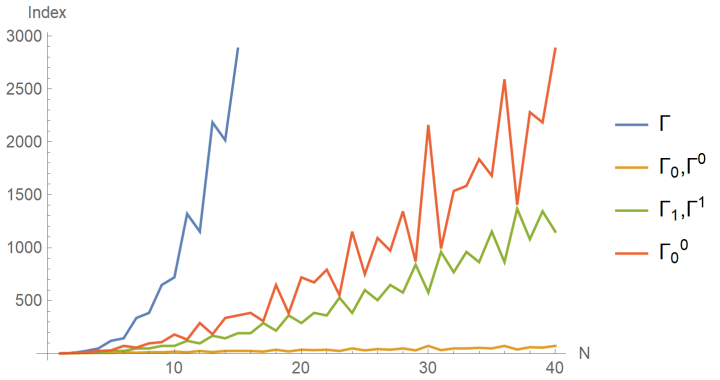


Figure 1.2: The index of our congruence subgroups tends to grow with N.

Proposition 1.5.6. *([6, page 38]). Let Γ be a congruence subgroup of $SL_2(\mathbb{Z})$. Then Γ is a disjunctive conjunction of the right cosets of Γ in $SL_2(\mathbb{Z})$.*

This proposition tells us that there exists a list of pairwise inequivalent right coset representatives $A_i \in \mathrm{SL}_2(\mathbb{Z}), i \in \{1, \ldots, n\}$, such that

$$\mathrm{SL}_2(\mathbb{Z}) = \Gamma A_1 \cup \Gamma A_2 \cup \cdots \cup \Gamma A_n.$$

Together with Corollary 1.4.6, we have

$$\mathbb{H} = (\Gamma A_1 \cup \Gamma A_2 \cup \cdots \cup \Gamma A_n)\mathcal{F} = \Gamma(A_1\mathcal{F} \cup A_2\mathcal{F} \cup \cdots \cup A_n\mathcal{F}).$$

Now the set $A_1\mathcal{F} \cup A_2\mathcal{F} \cup \cdots \cup A_n\mathcal{F}$ is a fundamental domain for Γ. As $\mathcal{F}$ is known by Theorem 1.4.3, all we have to identify is the according list of right coset representatives $A_i, i \in \{1, \ldots, n\}$. This list is certainly not unique. In fact, we may choose any of the infinitely many elements from each of the matrices' equivalence classes. Helena A. Verrill proposes in her paper [18] an algorithm for finding these right coset representatives: We start off by declaring the identity matrix I, which is always a group element of our congruence subgroup Γ by Definition 1.3.2, as the first right coset representative. In $\mathrm{SL}_2(\mathbb{Z})$, all matrices are equivalent to I, so there is just one equivalence class, which would terminate the algorithm already successfully in this simple case. Otherwise, we perform the following computation recursively: Given a possibly incomplete list of pairwise inequivalent right coset representatives L, we investigate the matrices LT, LT^{-1} and LS. As soon as we come across a matrix inequivalent to all matrices in L, we add it to L as a new right coset representative and carry on checking using the adapted L. Following that algorithm, we receive a fundamental domain which accumulates around $\mathcal{F}$ and thus can be visualised well.

This algorithm terminates due to the fact that there always exists a fundamental domain consisting of triangular areas connected to each other [16, pages 80–81] and because all congruence subgroups have finite index.

We come back to the issue when $-I$ is not a group element of a congruence subgroup Γ, addressed in Section 1.2. In that case, each right coset representative $M \in L$ is inequivalent to the matrix

$-M$, because $M(-M)^{-1} = -I \notin \Gamma$. Thus, we have $-M \in L$. However, both these matrices are visualised by the same triangular area. Hence, we consider just "positive" elements in L when it comes to drawing fundamental domains.

Example 1.5.7. One can illustrate a fundamental domain for $\Gamma(2)$ using the Mathematica package described in Chapter 2. The result in form of a screenshot is given by Figure 1.3 and may be verified by [12, pp. 19–21]. We examine the first steps the algorithm performs

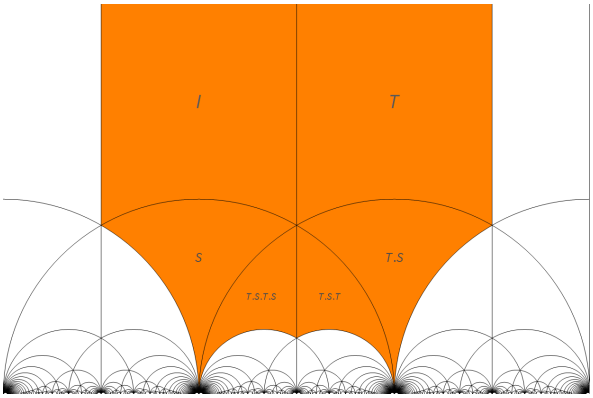


Figure 1.3: A fundamental domain for $\Gamma(2)$.

in order to create this picture. First of all, $\mathcal{F}$ is part of the fundamental domain, as it is the starting triangular area. Then, the triangular area $T\mathcal{F}$ gets added, because $T \notin \Gamma(2)$ (i.e. $T \not\sim I \in L = \{I\}$) and we cannot map any points from $T\mathcal{F}$ to $\mathcal{F}$ using a matrix in $\Gamma(2)$. The situation is different for the matrix T^2, since $T^2 \in \Gamma(2)$ (i.e. $T^2 \sim I \in \{I, T\}$). Thus, the triangular area $T^2\mathcal{F}$ is not part of the fundamental domain.

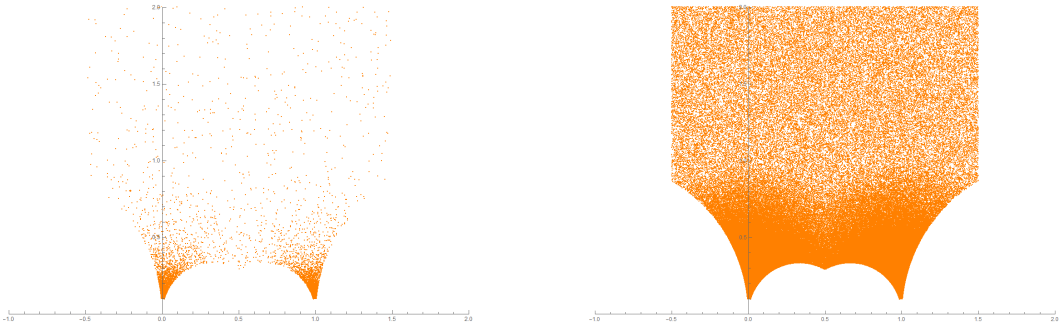
1.5.1 Visualisation

Having an algorithm to get a list of pairwise inequivalent right coset representatives, how can we visualise the associated fundamental domain using Mathematica? During the implementation, I attempted several ways of drawing a fundamental domain, given a congruence subgroup. The first three failed due to poor performance. For the following attempts, all examples show the fundamental domain for $\Gamma(3)$ given in Example 1.5.7 and Figure 1.3.

Attempt 1 (Rejected): Map single points from $\mathcal{F}$

We pseudo-randomly generate points in $\mathcal{F}$ with a certain maximal imaginary part. Using the matrices in the list of right coset representatives, one may map these points via the group action (1.1) and receive triangular areas. The problems here are pretty obvious: First, the limited imaginary part makes it impossible to draw the whole fundamental domain. For some triangular areas, one corner will be completely empty. This is visible in Figure 1.4b around the points 0 and 1, where the fundamental domain should touch the real axis. Further, we need a lot of points to make the resulting picture look smooth without holes. Increasing the permitted maximal

imaginary part as well as the number of points both decrease the performance, as presented in Figure 1.4. For the given examples, a maximal imaginary part of 10 is used which indicates that the images of points with a large imaginary part in $\mathcal{F}$ accumulate around one corner point of each triangular area.



(a) 10.000 points take ∼0.3 seconds (b) 1.000.000 points take ∼30 seconds

Figure 1.4: Increasing the amount of pseudo-randomly generated points increases the visibility but decreases the performance.

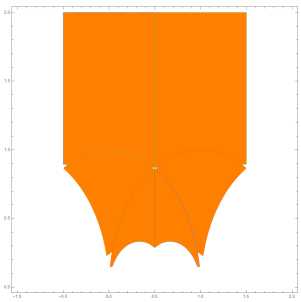
Attempt 2 (Rejected): Calculate functions for triangular areas

One can very precisely calculate the functions for the triangular areas. First, we define the functions

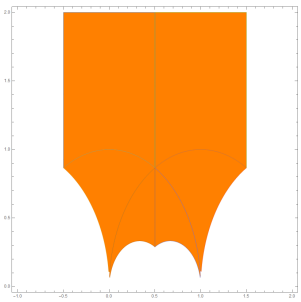
Mathematica

```
In[1]:= s[z_] := -1/z;
        t[z_] := z + 1;
        funDom[z_] := -1/2 <= Re[z] <= 1/2 && Sqrt[1 - Re[z]^2] <= Im[z];
```

Note that the definition of the fundamental domain $\mathcal{F}$ (Theorem 1.4.3) is abbreviated here to simplify the equation. All boundaries are included in the set. Then, the Mathematica inbuilt functions `InverseFunction` and `Composition` help to calculate the functions for the triangular areas which then can be visualised by `ComplexRegionPlot`. However, the fundamental domain is not printed close to the real axis. Increasing the `MaxRecursion`, which is an option for `ComplexRegionPlot`, helps but decreases the performance significantly, as shown in Figure 1.5.



(a) `MaxRecursion` of 2 takes ∼0.2 seconds



(b) `MaxRecursion` of 10 takes ∼10 seconds

Figure 1.5: Increasing the maximal recursion limit increases the visibility but decreases the performance.

Attempt 3 (Rejected): Intersect discs

All triangular areas are either a translation of $\mathcal{F}$ by a power of T or the area enclosed by three semi-circles where a parallel line to the imaginary axis is also considered a semi-circle. The idea of how to identify these semi-circles and lines can be found in Thomas Ponweiser's Mathematica package *ModularGroup* [14]. Applications for the package may be read in his diploma thesis [13]. For our purpose, this attempt is not acceptable since Mathematica requires too much time to calculate the union of several intersections, even though the resulting picture is precise (see Figure 1.6).

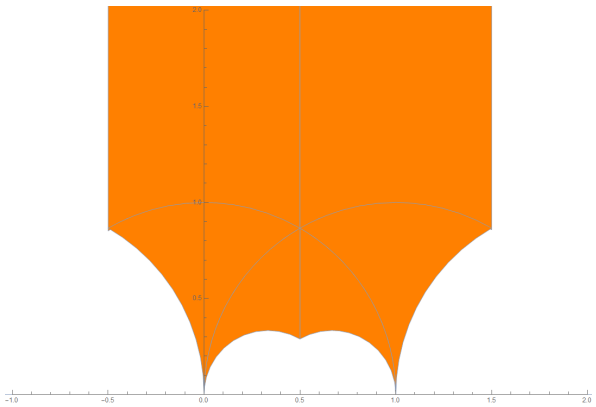


Figure 1.6: A fundamental domain for $\Gamma(2)$ using Mathematica's inbuilt function `BoundaryDiscretizeRegion` takes ∼3 seconds.

Attempt 4 (Accepted): Draw polygons

The fourth and finally successful attempt is to calculate and dye polygons for each triangular area. To do so, we differentiate between two types of triangular areas, one of which describes areas defined as the image of $\mathcal{F}$ under a matrix T^m, $m \in \mathbb{Z}$. These areas reach complex infinity, thus are not enclosed, making it impossible to represent them as polygons. For that reason, the Mathematica package first defines $\mathcal{F}$ as a polygon capped at a maximal imaginary part of 100. We then receive a triangular area characterised by T^m by simply mapping the polygon's vertices

using the group action: $\left(\begin{smallmatrix}1 & m\\ 0 & 1\end{smallmatrix}\right)z = z + m$.
For a matrix A different from T^m, the approach is similar: We define the area characterised by the matrix S, which is enclosed by the circles $|z+1| = 1$, $|z| = 1$ and $|z-1| = 1$, as a polygon of a specified number of vertices. Then, the wanted triangular area is obtained by mapping the vertices via the group action using the matrix AS^{-1}.

The advantages of mapping vertices are evident: First, to control performance, the polygons' precision is easily adjustable by the number of vertices being used. Further, we are able to remove or add other triangular areas comfortably (see Section 2.2.2 and Section 2.2.3 for applications). The method is sufficient to achieve the goal of drawing fundamental domains for congruence subgroups considering the aim is neither delivering a mathematical equation for the area nor including complex infinity to the picture. Figure 1.7 shows the efficiency and that few vertices already give reasonable results.



(a) A precision of 9 vertices per polygon takes ∼0.002 seconds.

(b) A precision of 60 vertices per polygon takes ∼0.02 seconds.

Figure 1.7: A high and acceptable resolution still takes very little computation time.

However, there is also a disadvantage. When mapping complex points using the group action for certain matrices (e.g. ones with a large absolute value of the last entry), these points are not distributed evenly around the new area's boundary. Figure 1.8 shows the shifted vertices for the polygon representing the image of $\mathcal{F}$ under the matrix $\left(\begin{smallmatrix}0 & -1\\ 1 & 15\end{smallmatrix}\right)$. An incremented number of vertices would help, but slow down the program unnecessarily. The solution I use is shown in Figure 1.9: The initial polygon's vertices are distributed unevenly, which compensates the shift done by the mapping. The weight of adjustment is determined for each matrix individually by its lower entries' absolute values.

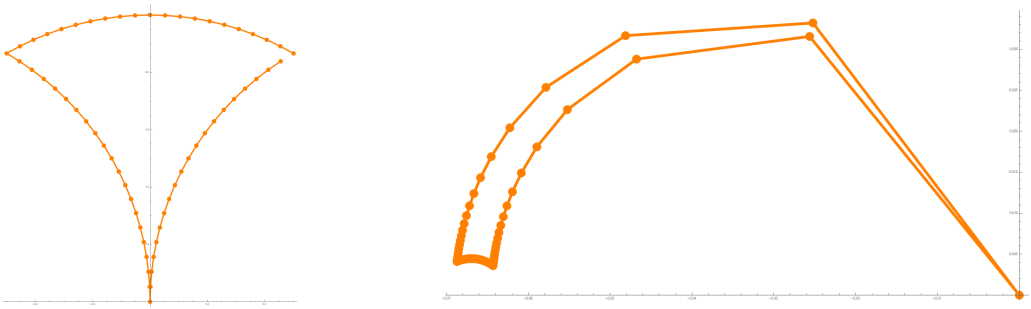


Figure 1.8: The mapped vertices are not distributed evenly around the area's boundaries.

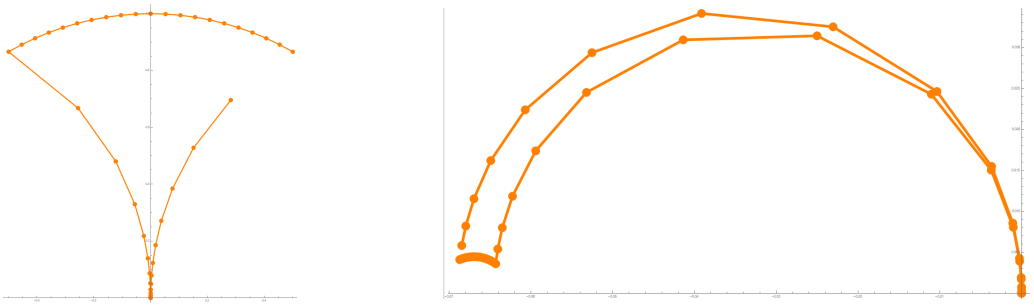


Figure 1.9: The initial vertices are chosen to be distributed unevenly, which compensates the group action's shift.

Modular Tessellation

In order to better understand the concept of mapping $\mathcal{F}$ and how the upper half complex plane gets partitioned, we want to draw the triangular areas' boundaries additionally to the fundamental domain. Jerzy Kocik [8] gives an algorithm to draw the so-called *modular tessellation*. Besides the negligible calculation time, this implementation has the neat properties that the number of drawn triangular areas is adjustable and that they accumulate around $\mathcal{F}$. Figure 1.10 shows a fundamental domain for $\Gamma(2)$ combined with the modular tessellation.

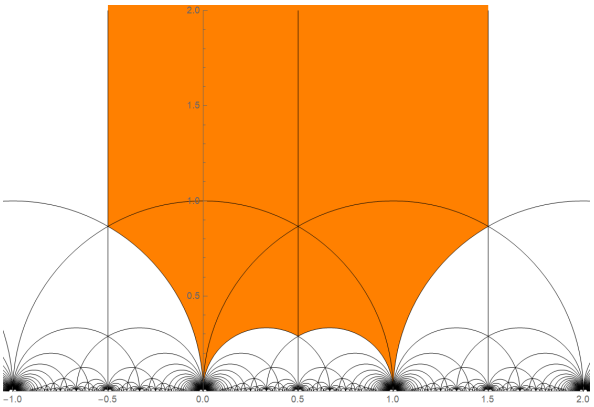


Figure 1.10: A fundamental domain for $\Gamma(2)$ combined with the modular tessellation.

2 A Mathematica Program for drawing Fundamental Domains

I wrote a Mathematica package for drawing fundamental domains for congruence subgroups. The file *DrawFunDoms.m* together with a demonstration notebook *DrawFunDoms_Demo.nb* (also as PDF) are available to download for free at https://risc.jku.at/sw/drawfundoms. The package was written in Mathematica 12.1 and is not guaranteed to work properly for older versions. Place the file inside a folder, where your Mathematica-application can find it. For me, the path on Windows is *C:\Program Files\Wolfram Research\Mathematica\12.1\AddOns\Packages*. Then open a Mathematica-notebook and load the package via

Mathematica

```
In[2]:= << DrawFunDoms.m
```

2.1 Motivation

There exists a great Java program for visualising fundamental domains for congruence subgroups by Helena A. Verrill [19] which is open source, but does not seem to be maintained any more. Figure 2.1 shows a screenshot of the program.

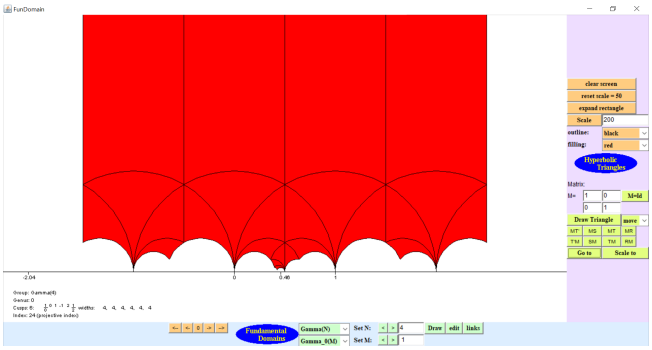


Figure 2.1: Verrill's Java program showing a fundamental domain for $\Gamma(4)$.

The Mathematica package carried out as part of this thesis provides more functionality since results may be used for further calculations (see Other useful functions). Moreover, inbuilt functions like `Manipulate` providing features to study the underlying group action (1.1) (see Section 2.2.2) motivate the usage of Mathematica. However, the Java program naturally draws the results way quicker than Mathematica which crystallises for congruence subgroups with larger index.

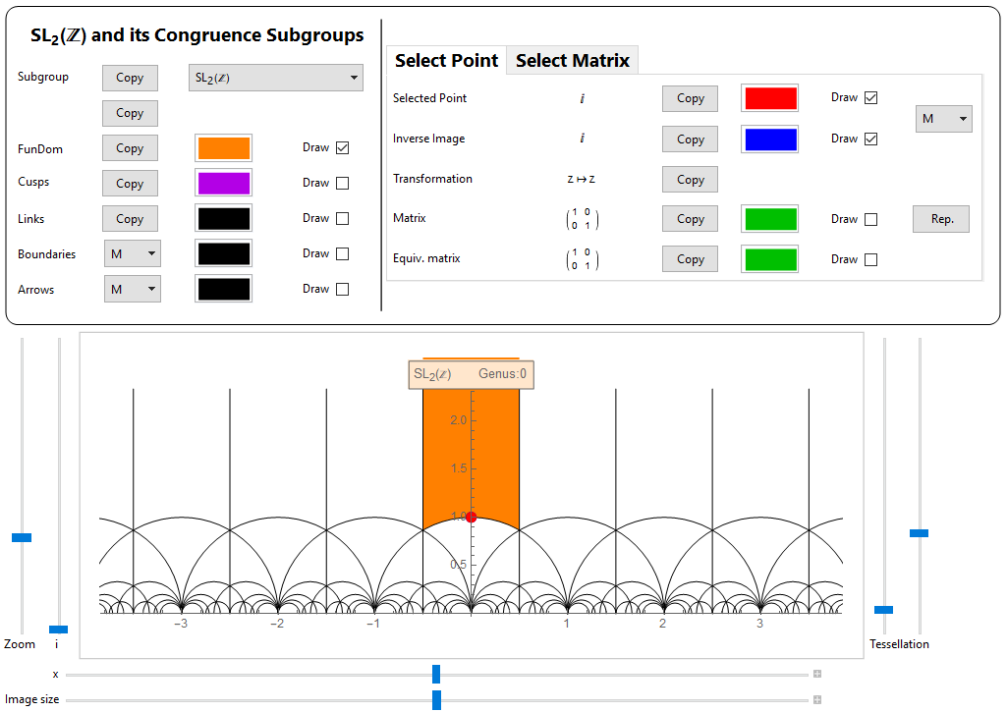
2.2 The Main Program

First, make sure that in your notebook the option *Evaluation* → *Dynamic Updating Enabled* is enabled and translations are deactivated (*Edit* → *Preferences* → *Interface* → *Language Settings* → *Show code captions*; then restart Mathematica). The main program gets then launched by executing the command `DrawFundamentalDomains[]`:

Mathematica

```
In[3]:= DrawFundamentalDomains[]
```

Out[3]=



Unfortunately, on some operating systems the buttons' labels do not fit properly in the boxes and provoke line breaks. If that is the case on your operating system, try to adapt the magnification by

Mathematica

```
In[4]:= SetOptions[EvaluationNotebook[ ], Magnification -> 0.6]
```

The above example sets the magnification to 60%. You might want to try values between 0.5 and 1, until all text is nicely formatted and the interface fits onto your screen.

The program's interface consists of a control panel and a graphic below surrounded by sliders which allow graphical configurations. The control panel itself consists of two main panels: The left one ("$\mathrm{SL}_2(\mathbb{Z})$ and its Congruence Subgroups") lets the user choose between different congruence subgroups and calculates its coset representatives, cusps and links. The right one offers two tabs

("Select Point" and "Select Matrix") for selecting and visualising either a single point in the upper half complex plane or an element in $\mathsf{SL}_2(\mathbb{Z})$.

2.2.1 Control panel "$\mathsf{SL}_2(\mathbb{Z})$ and its Congruence Subgroups"

This control panel consists of the following seven rows:

Subgroup

The first row lets one choose the wanted subgroup. That is either $\mathsf{SL}_2(\mathbb{Z})$ itself or one of the following congruence subgroups: $\Gamma(N)$, $\Gamma_0(N)$, $\Gamma_1(N)$, $\Gamma^0(N)$, $\Gamma^1(N)$, $\Gamma_0^0(N)$, $\Gamma(N, M, P)$, Γ_θ. See Section 2.4 to learn how to manually add other self-defined congruence subgroups. Pressing the button "Copy" copies the internal name of the subgroup to the computer's clipboard, which may serve for further calculations (see Other useful functions).

Arguments

The second row specifies the arguments for the selected subgroup. Note that $\mathsf{SL}_2(\mathbb{Z})$ does not have any further arguments, hence there is nothing to choose from in this case.

FunDom

This row handles the fundamental domain for the earlier chosen subgroup and arguments. The button "Copy", when pressed, copies the current list of coset representatives to the clipboard. Alongside one can modify the fundamental domain's colour via the painted button (default is orange). The check box on the right lets one decide whether the fundamental domain should be displayed in the graphic underneath the control panels.

Cusps

The cusps are the points where the fundamental domain meets the real axis or complex infinity. They are calculated as well as drawn, if desired. The button "Copy" copies the list of cusps to the clipboard. Each entry is a list itself consisting of the cusp (a complex point) and its width, which indicates the amount of triangular areas meeting that cusp. However, for a matrix $A \in \mathsf{SL}_2(\mathbb{Z})$, its according cusp is equivalent to all cusps linked to the matrices in A's equivalence class. The width cumulates the equivalent cusps. The cusps (except the point ∞i) can be displayed in the graphic by ticking "Draw" on the right-hand side in arbitrary colour which is specified by the painted button (default is purple).

Links

The fundamental domain's edges can be connected with each other. For example, the standard fundamental domain for $\mathsf{SL}_2(\mathbb{Z})$ is a set of points which is closed on one half and open on the other one. Glueing together all edges creates a compact three-dimensional surface. The subgroup's genus then indicates the number of holes this surface at least has. For instance, a fundamental domain for a subgroup with genus 1 (e.g. $\Gamma_0(11)$) forms a torus. In order to visualise these connections, one can draw links illustrated by dotted

curved lines in a specified colour (default is black). The button "Copy" copies a list to the clipboard consisting of pairs of complex numbers which represent the end points of a link.

Edges

In order to visually emphasise the closed edges of a fundamental domain, one can add them to the graphic by ticking the check box "Draw". A drop-down list lets one adjust the thickness of these lines (default is medium) and a colour can be chosen from the painted button in the middle (default is black).

Arrows

The links and thicker edges still do not provide all information about a fundamental domain's border. The end points of an edge might not just be translated to the linked edge, but also swapped. One single dotted link hides this twist, which is crucial when studying the genus. That is why I added arrows indicating the direction an edge is facing. These arrows are drawn once the check box "Draw" is ticked. A drop-down list for the size (default is medium) and a painted button for the colour (default is black) of the arrows allow further adjustments.

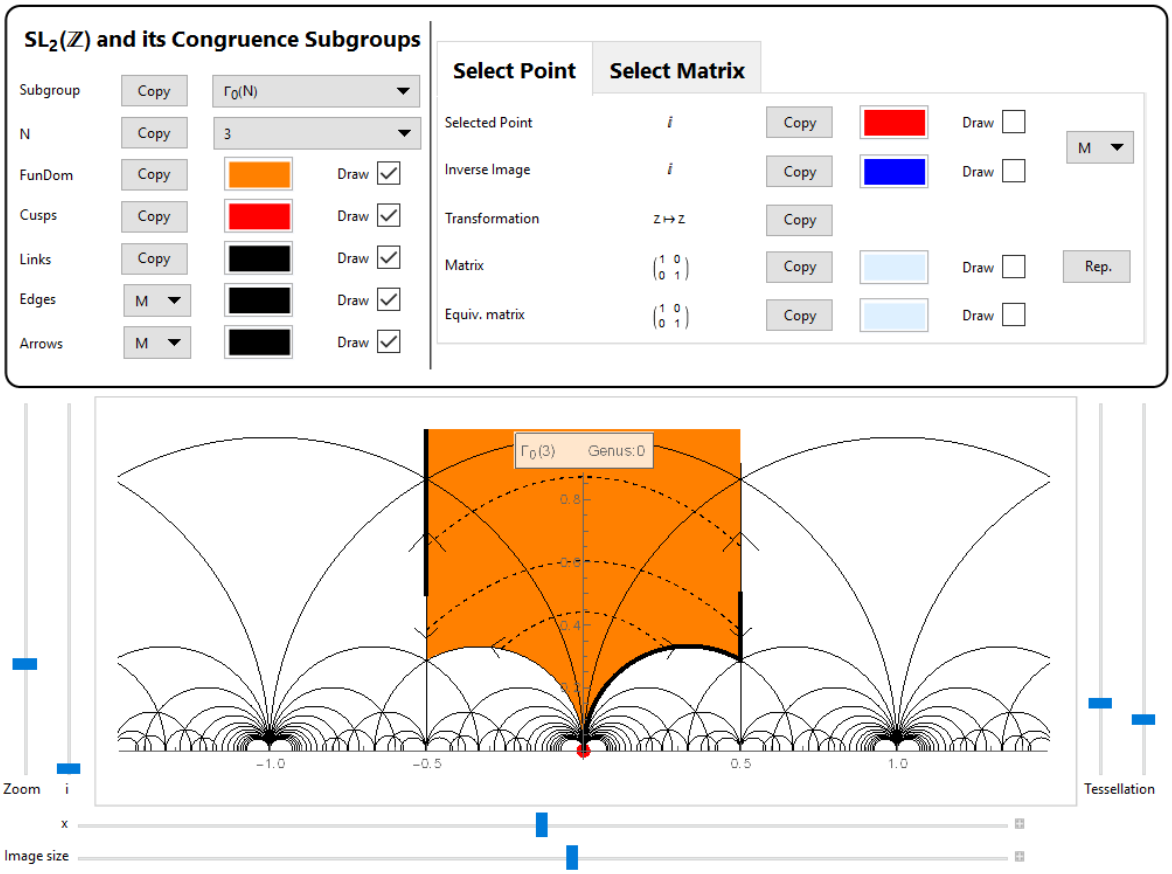


Figure 2.2: A fundamental domain for $\Gamma_0(3)$.

Figure 2.2 shows a fundamental domain for the congruence subgroup $\Gamma_0(3)$ with a cusp (red dot).

Considering the links and arrows, one can easily see that the genus must be zero.

2.2.2 Control panel "Select Point"

This control panel consists of the following five rows:

Selected Point

The user can select a point in the upper half complex plane by simply clicking in the graphic. For subgroups with small index, this point can even be dragged around the graphic. This point will then be used for the below rows' calculations. The currently selected point is displayed in the control panel and can be copied to the clipboard for further calculations with the help of the button "Copy". On the right the drawing options are: A painted button for the colour (default is red), a check box "Draw" which hides the point in the graphic if unselected, and a drop-down list for the size (default is medium) of the drawn point. Note that this also adjusts the size of the point "Inverse Image" below as well as the size of the drawn cusps. If you aim mapping an exact given point, please refer to the function GroupAction.

Inverse Image

Every point in the upper half complex plane has an inverse image inside the fundamental domain. The inverse image of the currently selected point is displayed in this row and is drawn, if "Draw" is ticked, in a given colour (default is blue) and size (default is medium).

Transformation

The transformation needed to map the inverse image to the selected point via the group action is visible in this row. The button labelled "Copy" creates a new variable z and copies this transformation to the clipboard. For instance, if the transformation is given by $z \mapsto -\frac{1}{z}$, the button first creates a fresh variable which is not initialised yet (e.g. $z1$) and copies to the clipboard $-\frac{1}{z1}$. One can then plug in for instance the present inverse image to receive the currently selected point easily:

Mathematica

```
In[5]:= -1/z1 /. z1 -> -0.25 + 1.75 I
Out[5]= 0.08 + 0.56 I
```

Matrix

This row is dedicated to the matrix needed to map the inverse image to the selected point via the group action. It may be copied to the clipboard using the button "Copy". Furthermore, one can add its triangular area to the graphic by ticking the check box "Draw" and selecting a desired colour (default is green). The button "Rep." on the very right exchanges the current matrix with the unique equivalent one in the set of right coset representatives.

The fundamental domain, the cusps, links, edges and arrows from the panel "$SL_2(\mathbb{Z})$ and its subgroups" get updated accordingly which can take a while for subgroups with large index. Also, the selected point now lies inside the fundamental domain, so the values for the inverse image, transformation and inverse image also change. This button can be helpful when trying to find "nicer" fundamental domains than the program suggests or for comparison reasons.

Equiv. Matrix

In this last row the user can find the equivalent matrix to the above matrix with respect to the set of coset representatives. Likewise, it can be copied to the clipboard and its triangular area is drawn having selected "Draw" in a specific colour (default is green).

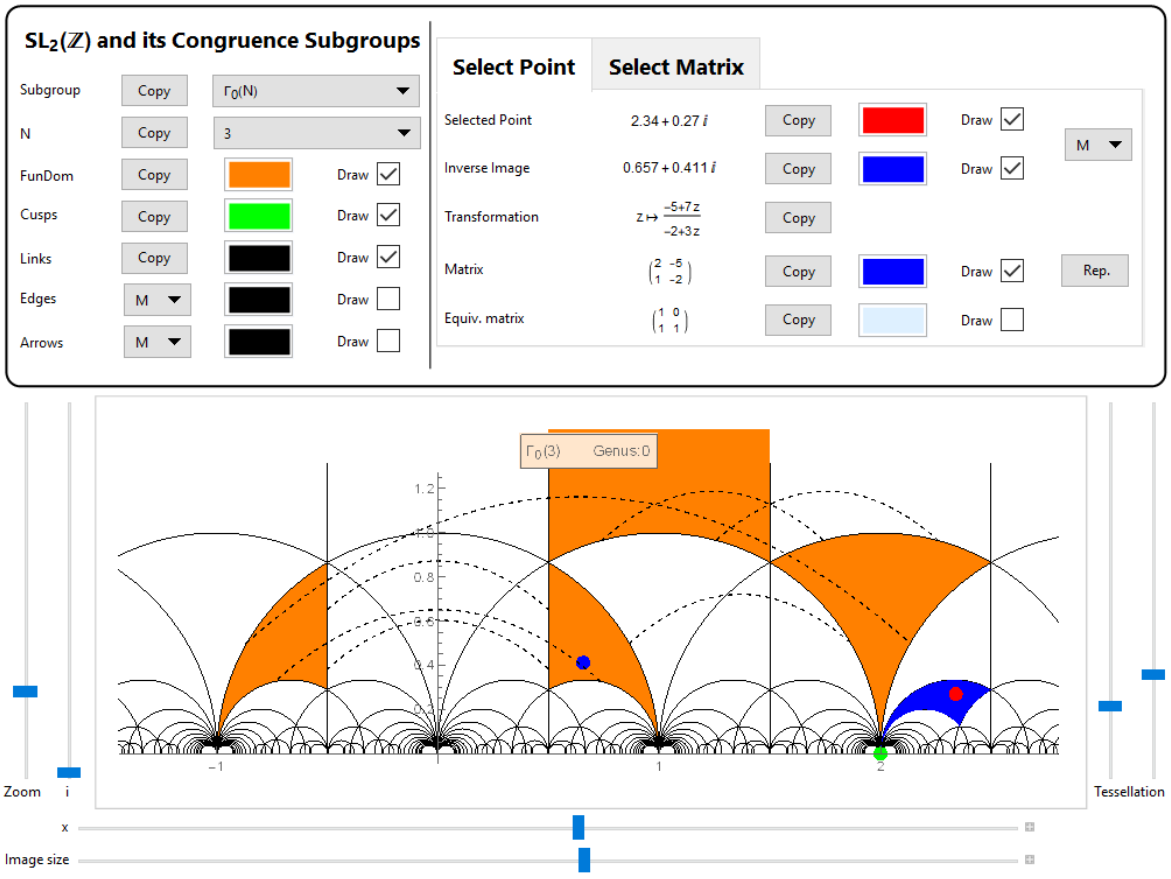


Figure 2.3: Another fundamental domain for $\Gamma_0(3)$.

Figure 2.3 shows a fundamental domain for the congruence subgroup $\Gamma_0(3)$ after pressing the button "Rep." a couple of times. If we compare this fundamental domain with the one shown in Figure 2.2, we can see that one of the cusps (besides the cusp ∞i) is now located at the point 2 (green dot) instead of the origin and the links are updated as well.

2.2.3 Control panel "Select Matrix"

Switching to the tab "Select Matrix" hides everything from the tab "Select Point" until one switches back, meaning that the selected point and other adjustments are saved in the background. In the middle of the panel one finds a matrix in `MatrixForm` which is an element of $\mathrm{SL}_2(\mathbb{Z})$. The triangular area of this matrix is added to the graphic if the check box "Draw" is ticked in a colour specified by the painted button (default is green). At the top there are additional buttons for setting the matrix to the identity ("Id"), copying the matrix to the clipboard ("Copy") and promoting the matrix to a coset representative ("Rep.") which once again updates all values at the panel "$\mathrm{SL}_2(\mathbb{Z})$ and its Congruence Subgroups". Right above the matrix there is a text field which allows the user to input an arbitrary matrix from $\mathrm{SL}_2(\mathbb{Z})$. After either pressing "Enter" on the keyboard or clicking somewhere outside the text field, the matrix gets updated. Trying to input a matrix outside $\mathrm{SL}_2(\mathbb{Z})$ resets the matrix to the identity. At the bottom the current matrix is written in form of an ST-word. Furthermore, one is able to apply left side matrix multiplication to the matrix using the $\mathrm{SL}_2(\mathbb{Z})$-generators S and T and their inverses S^{-1} and T^{-1}. Four buttons on the left are provided to do so. On the right, the four counterparts allow right side matrix multiplication to the matrix. This way one can study the changing of the matrix as well as its triangular area when multiplying it with a generator. Lastly, it is possible to select a triangular area directly from the graphic by clicking into or dragging the mouse over the upper half complex plane. Naturally, the matrix and its ST-word update during that process. Figure 2.4

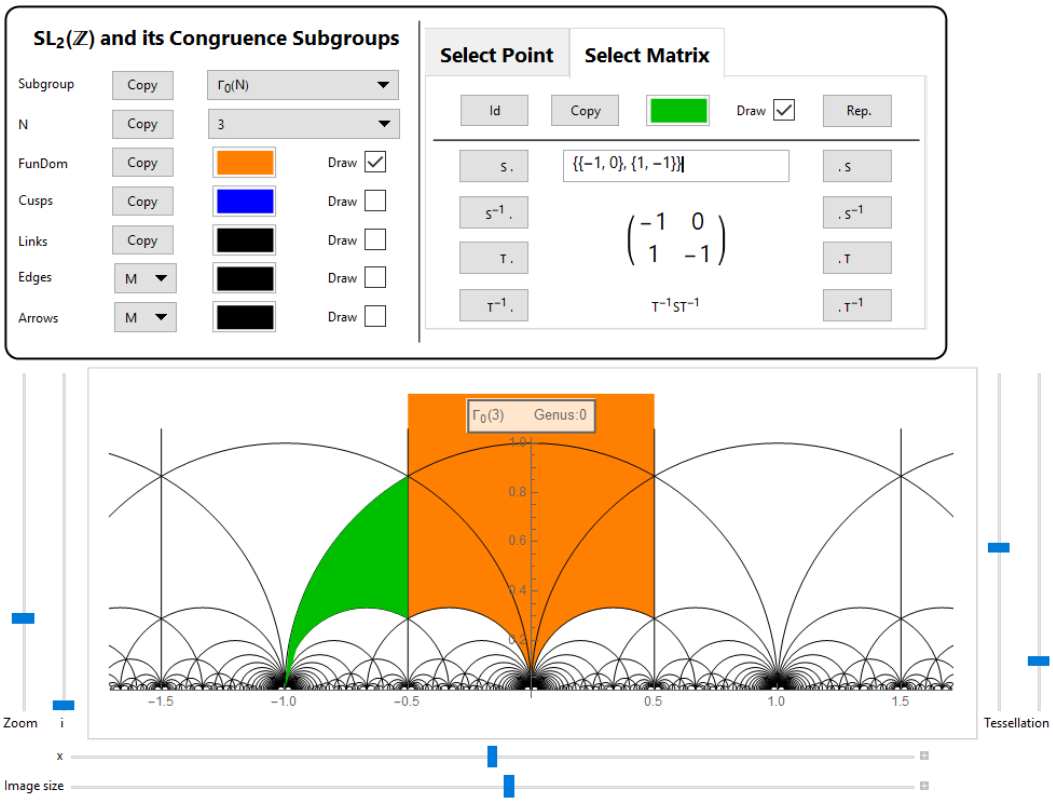


Figure 2.4: A screenshot of the active panel "Select Matrix".

presents a fundamental domain for the congruence subgroup $\Gamma_0(3)$ together with the triangular area for the selected matrix $T^{-1}ST^{-1}$.

2.2.4 Graphic

The Graphic displays all elements specified in the panels. On its very top, one can read off the current subgroup as well as its genus. The two vertical sliders on the left provide the possibility of zooming in/out and adapting the imaginary axis. Underneath the graphic the two horizontal sliders let one adjust the real axis and the image size. On the right-hand side one can tune or even remove the modular tessellation using the two vertical sliders. The program runs smoother for smaller values here. However, some subgroups' fundamental domains (e.g. Γ_0) request a greater depth, while others (e.g. Γ^0) need a wider range.

2.3 Other useful functions

The package offers some other useful functions, which the main program is partly based on, but which can be accessed independently from it. For a convenient use of these functions, one may utilise the following frequently needed constant matrices: `$Id` for $\{\{1,0\},\{0,1\}\}$, `$S` for $\{\{0,-1\},\{1,0\}\}$, `$SInv` for $\{\{0,1\},\{-1,0\}\}$, `$T` for $\{\{1,1\},\{0,1\}\}$ and `$TInv` for $\{\{1,-1\},\{0,1\}\}$.
Most functions require a congruence subgroup's name in form of a string as argument. The subgroup's arguments are given as list of positive integers where its length depends on the number of needed arguments. The following table shows the translations of the available subgroups to string-form and the lists of arguments:

Subgroup	**String**	**Arguments**
$\Gamma(N)$	"Gamma"	{n_}
$\Gamma_0(N)$	"Gamma0"	{n_}
$\Gamma_1(N)$	"Gamma1"	{n_}
$\Gamma^0(N)$	"GammaUpper0"	{n_}
$\Gamma^1(N)$	"GammaUpper1"	{n_}
$\Gamma(N,M,P)$	"GammaNMP"	{n_, m_, p_}
Γ_θ	"GammaTheta"	{ }

For each function there is a short description together with information about input/output and a basic example.

GroupAction

This function makes a matrix map a complex point via the group action.

Input

2×2-matrix

Complex point

Output

Complex point

Mathematica

In[6]:= **GroupAction[$S, 1 + I]**

Out[6]= $-\frac{1}{2} + \frac{I}{2}$

SubgroupMemberQ

This function checks whether a given matrix is element of a congruence subgroup.

Input

String (group name)

List of positive integers (arguments)

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

Output

True/False

Mathematica

In[7]:= **SubgroupMemberQ["Gamma", {1}, {{1, 3}, {0, 1}}]**

Out[7]= True

In[8]:= **SubgroupMemberQ["GammaNMP", {3, 2, 2}, {{1, 3}, {0, 1}}]**

Out[8]= False

EquivalentMatricesQ

This method checks whether two given matrices are equivalent in a given congruence subgroup.

Input

String (group name)

List of positive integers (arguments)

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

Output

True/False

Mathematica

```
In[9]:= EquivalentMatricesQ["Gamma", {1}, $T.$T.$T, $Id]
Out[9]= True
```

FindMatrix

For a given point in the upper half complex plane, this function finds the unique matrix that is necessary to map a point from the standard fundamental domain to the given point using the group action.

Input

Point in the upper half complex plane

Output

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

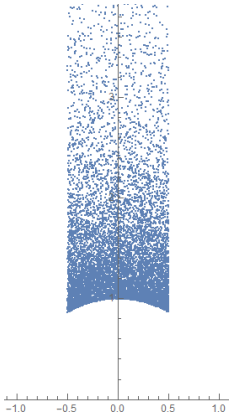
Mathematica

```
In[10]:= FindMatrix[1.5 + 0.22I]
Out[10]= {{3, 1}, {2, 1}}
In[11]:= FindMatrix[0.0000001 + 0.0000001I]]
Out[11]= {{0, -1}, {1, -5000000}}
```

In other words, the inverse of the found matrix maps the given point into the standard fundamental domain. Let us draw the standard fundamental domain by mapping random points from the upper half complex plane into it:

Mathematica

```
In[12]:= ComplexListPlot[GroupAction[Inverse[FindMatrix[#]], #]& /@
           RandomComplex[{-100, 100 + I}, 10000]]
Out[12]=
```



RandomSL2Z

This function creates a pseudorandom element in $\mathrm{SL}_2(\mathbb{Z})$ such that $||\texttt{RandomSL2Z}[n]||_{\max} \leq n$ with $\left\|\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)\right\|_{\max} := \max\{|a|, |b|, |c|, |d|\}$.

Input

Positive integer (default is 1000)

Output

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

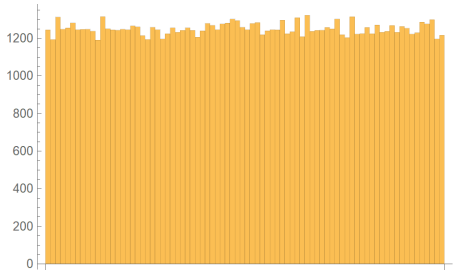
Mathematica

```
In[13]:= RandomSL2Z[5]
Out[13]= {{-4, 5}, {-1, 1}}
In[14]:= RandomSL2Z[]
Out[14]= {{452, -719}, {215, -342}}
```

These matrices are distributed evenly:

Mathematica

```
In[15]:= BarChart[Values[Counts[Table[RandomSL2Z[5], {100000}]]]]
Out[15]=
```



GenerateMatrices

The method `GenerateMatrices[`n`]` generates all "positive" matrices in $\mathrm{SL}_2(\mathbb{Z})$ with maximum norm $\leq n$, which is defined as above (see RandomSL2Z).

Input

Positive integer (default is 5)

Output

List of matrices

Mathematica

```
In[16]:= MatrixForm /@ GenerateMatrices[1]
Out[16]= {{{0, -1}, {1, 1}}, {{0, 1}, {-1, 0}}, {{0, 1}, {-1, 1}}, {{1, -1}, {1, 0}}, {{1, -1}, {0, 1}}, {{1, 1}, {-1, 0}}, {{1, 1}, {0, 1}}, {{1, 0}, {-1, 1}}, {{1, 0}, {1, 1}}, {{1, 0}, {0, 1}}}
```

The max norm is defined above for the function RandomSL2Z. Furthermore, let me clarify the term "positive" matrix. A matrix $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ in $\mathrm{SL}_2(\mathbb{Z})$ maps the standard fundamental domain into a triangular area. Its counterpart $-M$ yields the exact same image. That is why I define a matrix to be positive if and only if $a \geq 0$ and if $a = 0$, then $d \geq 0$ and if also $d = 0$, then $b > 0$. The function `GenerateMatrices` returns only "positive" matrices. Still, one can easily generate the "negative" matrices as well with the command `MatrixForm/@Flatten[{#, -#} & /@GenerateMatrices[1],1]]`.

STExponents

Given a matrix in $\mathrm{SL}_2(\mathbb{Z})$, this function creates a list of pairs. Each pair has two entries: The first one is either S or T, and the second one is an integer $\neq 0$ indicating the power of the respective matrix. Multiplying all these powers of matrices in the given order results in the input matrix.

Input

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

Output

List of pairs

Mathematica

In[17]:= **STExponents[{{1, 0}, {−2, 1}}]**

Out[17]= $\{\{S, -1\}, \{T, 2\}, \{S, 1\}\}$

In[18]:= $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

Out[18]= $\{\{1, 0\}, \{-2, 1\}\}$

STExponentsToMatrix

This function is the inverse of the method `STExponents`.

Input

List of pairs of the form {"S",Integer $\neq 0$} or {"T",Integer $\neq 0$}

Output

Matrix in $SL_2(\mathbb{Z})$

Mathematica

In[19]:= **STExponentsToMatrix[{{"S", −1}, {"T", 2}, {"S", 1}}]**

Out[19]= $\{\{1, 0\}, \{-2, 1\}\}$

STWord

The method `STWord` returns a string of concatenated powers of the matrices S and T such that the product of these powers of matrices results in the input matrix.

Input

Matrix in $SL_2(\mathbb{Z})$

Output

String

Mathematica

In[20]:= **STWord[{{1, 0}, {−2, 1}}]**

Out[20]= $S^{-1}T^2S$

FunDom

The main idea for the algorithm behind this function was done by Helena A. Verrill [18, pages 4-5] and is described in Section 1.5. The method `FunDom` calculates a set of right coset representatives together with an adjacency list for each matrix representing a coset of the given congruence subgroup. This adjacency list consists of three lists itself: The first one for the direction T, the second one for the direction T^{-1} and the third one for the direction S. For such a direction-list, the first entry tells the position of the matrix in the set of right coset representatives which is equivalent to the one we are looking at in the particular direction. The second entry is a boolean. `True` means that the neighbouring matrix is a right coset representative and `False` tells us that we have an outer edge of the fundamental domain in the current direction. In case the negative identity matrix is not member of the subgroup, every matrix also appears in the list in negated form. For one matrix of each such pair, the list of adjacencies is $\{0, \texttt{False}\}$.
`FunDom` is the most time consuming method in the whole package as for each found matrix that gets added to the set of coset representatives, two more matrices need to be checked and might be added as well. Hence, the algorithm will take some time or even reach a recursion limit for subgroups with large index. But once one gets a result, the output may be assigned to a variable and used for the methods

- `ExchangeMatrix`,
- `CosetRepresentatives`,
- `CosetRepresentativesWithoutNegatives`,
- `SubgroupIndex`,
- `NumberOfEllipticPointsOrder2`,
- `NumberOfEllipticPointsOrder3`,
- `Genus`,
- `Cusps` and
- `DrawFunDom`.

This way one prevents a restart of calculations for each of these methods.

Input
: String (group name)
: List of positive integers (arguments)

Output
: List

Mathematica

```
In[21]:= FunDom["Gamma0", {2}]

Out[21]= {{{{1, 0}, {0, 1}}, {{1, False}, {1, False}, {2, True}}},
          {{{0, -1}, {1, 0}}, {{3, True}, {3, False}, {1, True}}},
          {{{0, -1}, {1, 1}}, {{2, False}, {2, True}, {3, False}}}}
```

ExchangeMatrix

This function exchanges a matrix from a list in form of the output of the function `FunDom` with the equivalent coset representative and updates the list of adjacencies for all other matrices as well.

Input

String (group name)

List of positive integers (arguments)

List in form of the output of `FunDom`

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

Output

List in form of the output of `FunDom`

Mathematica

```
In[22]:= ExchangeMatrix["Gamma0", {2},
             FunDom["Gamma0", {2}], {{1, 42}, {0, 1}}]

Out[22]= {{{{1, 42}, {0, 1}}, {{1, False}, {1, False}, {2, False}}},
          {{{0, -1}, {1, 0}}, {{3, True}, {3, False}, {1, False}}},
          {{{0, -1}, {1, 1}}, {{2, False}, {2, True}, {3, False}}}}
```

CosetRepresentatives

This method gives a list of coset representatives for a given congruence subgroup.

Input

List in form of the output of `FunDom`

Output

List of matrices in $\mathrm{SL}_2(\mathbb{Z})$

Mathematica

```
In[23]:= MatrixForm /@ CosetRepresentatives[FunDom["Gamma1", {3}]]
Out[23]= (1 0; 0 1), (-1 0; 0 -1), (0 -1; 1 0), (0 1; -1 0), (0 -1; 1 1), (0 1; -1 -1), (0 -1; 1 -1), (0 1; -1 1)
```

CosetRepresentativesWithoutNegatives

If the negative identity matrix is not element of a given congruence subgroup, for each coset representative the negative form is also in the list of coset representatives. This might be unnecessary, for instance if we want to draw the triangular areas, always two would just overlap each other. `CosetRepresentativesWithoutNegatives` returns a list of coset representatives without "negative" forms of the matrices. In this case, the length of the resulting list here is half the length of the output of the function `CosetRepresentatives`.

Input

List in form of the output of `FunDom`

Output

List of matrices in $\mathrm{SL}_2(\mathbb{Z})$

Mathematica

```
In[24]:= MatrixForm /@ CosetRepresentativesWithoutNegatives[
           FunDom["Gamma1", {3}]]
Out[24]= (1 0; 0 1), (0 -1; 1 0), (0 -1; 1 1), (0 -1; 1 -1)
```

SubgroupIndex

This function returns the index for a given subgroup. For the subgroups Γ, Γ_0, Γ_1, Γ^0, Γ^1, Γ_0^0 and Γ_θ standard formulae are used. For other congruence subgroups the number of coset representatives must be calculated first, which might take some time.

Input

[String (group name) and
List of positive integers (arguments)]
OR List in form of the output of `FunDom`

Output

Positive integer

Mathematica

```
In[25]:= Timing[SubgroupIndex["Gamma", {1000}]]
Out[25]= {0., 720000000}
In[26]:= Timing[SubgroupIndex[FunDom["Gamma", {5}]]]
Out[26]= {0.890625, 120}
```

Cusps

`Cusps` gives the cusps together with their widths for a given subgroup.

Input

List in form of the output of `FunDom`

Output

List of cusps

Mathematica

```
In[27]:= Cusps[FunDom["Gamma", {3}]]
Out[27]= {{-1, 6}, {0, 6}, {1, 6}, {∞I, 6}}
In[28]:= Cusps[ExchangeMatrix["Gamma", {3},
           FunDom["Gamma", {3}], {{51, -76}, {-2, 3}}]]
Out[28]= {{-51/2, 6}, {-1, 6}, {1, 6}, {∞I, 6}}
```

NumberOfEllipticPointsOrder2

This function calculates the number of elliptic points of order 2. The method is based on Verrill's description of her Java program [18, page 6].

Input

List in form of the output of `FunDom`

Output

Non-negative integer

Mathematica

```
In[29]:= Table[NumberOfEllipticPointsOrder2[
```

```
        FunDom["Gamma1", {n}]], {n, 1, 5}]
Out[29]= {1, 1, 0, 0, 0}
```

NumberOfEllipticPointsOrder3

This method calculates the number of elliptic points of order 3. The idea for the algorithm behind this function is adapted from Helena A. Verrill [18, page 6].

Input

List in form of the output of `FunDom`

Output

Non-negative integer

Mathematica

```
In[30]:= Table[NumberOfEllipticPointsOrder3[
        FunDom["Gamma1", {n}]], {n, 1, 5}]
Out[30]= {1, 0, 1, 0, 0}
```

Genus

The purpose of this function is to calculate the subgroup's genus using Bruno Schoeneberg's formula [16, pages 93-98].

Input

List in form of the output of `FunDom`

Output

Non-negative integer

Mathematica

```
In[31]:= Genus /@ Table[FunDom["GammaUpper0", {n}], {n, 1, 35}]
Out[31]= {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 2, 2, 1, 0, 2, 1, 2, 2, 3, 2, 1, 3, 3, 3}
```

Tessellation

The algorithm used here is adapted from Jerzy Kocik [8]. With the help of this method, one can draw modular tessellations to visualise the boundaries for each triangular area. The first non-negative integer argument indicates the depth of the circles, whereas the second one specifies the range.

Input

Non-negative integer

Non-negative integer

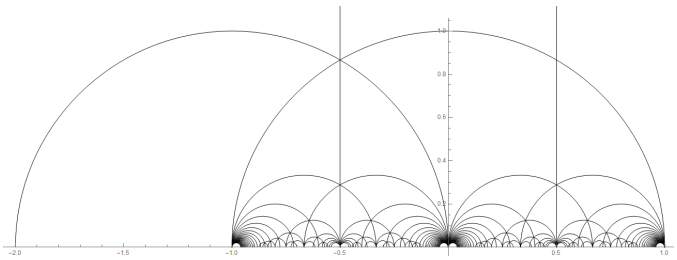
Output

A set of `Circles` and `Lines`

Mathematica

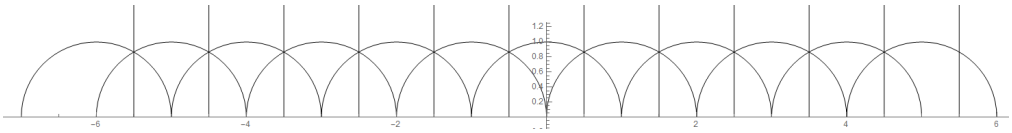
```
In[32]:= Graphics[Tessellation[50, 0], Axes -> True]
```

Out[32]=



```
In[33]:= Graphics[Tessellation[2, 5], Axes -> True]
```

Out[33]=



The first of the above two examples demonstrates the function `Tessellation` for high depth and low range, while for the second one a low depth but high range is used.

DrawMatrixImage

This function lets one draw a triangular area given a matrix in $\mathrm{SL}_2(\mathbb{Z})$. That is in other words the image of the standard fundamental under the matrix, whereas the group action serves as the operation. A higher precision makes the edges smoother.

Input

Matrix in $\mathrm{SL}_2(\mathbb{Z})$

Positive integer (precision; default is 10)

Output

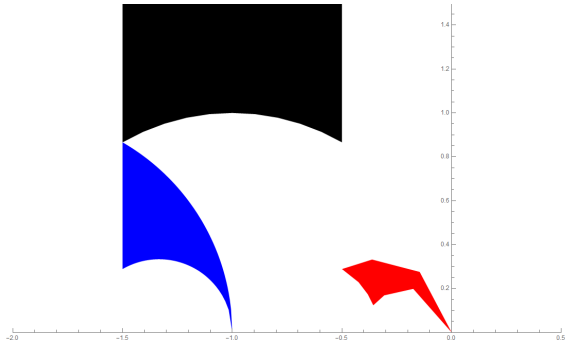
Polygon

Mathematica

```
In[34]:= Graphics[{DrawMatrixImage[{{1, -1}, {0, 1}}],
           Red, DrawMatrixImage[{{0, -1}, {1, 2}}, 3],
```

```
Blue, DrawMatrixImage[{{−1, −2}, {1, 1}}, 100]},
Axes −> True, PlotRange −> {{−2, 0.5}, {0, 1.5}}]
```

Out[34]=



DrawFunDom

The method `DrawFunDom` calculates a set of polygons which together represent a fundamental domain for a given congruence subgroup.

Input

List in form of the output of `FunDom`

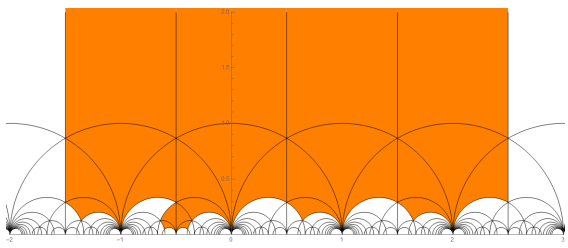
Output

Set of polygons

Mathematica

```
In[35]:= Graphics[{Orange, DrawFunDom[FunDom["Gamma", {4}]],
Black, Tessellation[20, 3]},
Axes −> True, PlotRange −> {{−2, 3}, {0, 2}}]
```

Out[35]=



If we compare the above fundamental domain with the one created by Verrill's Java program (Figure 2.1), we can spot a few differences. One could use the function ExchangeMatrix though or more simple the button "Rep." at the main program to replicate the picture.

2.4 How to manually add congruence subgroups

One may fairly easily add further congruence subgroups to the main program by editing the package file *DrawFunDoms.m*. This subsection gives a tutorial consisting of 8 steps on how to do so. Each step can be found in the package under the chapter **"Private"** → section **"Subgroups"** → subsection **"Add Congruence Subgroups"**.

Note that since the subgroup's arguments can be chosen from drop-down lists and there is only limited space for these lists inside the control panel, I decided to provide lists for only up to 3 arguments. One is still able to draw fundamental domains for other subgroups by defining the subgroup for fixed arguments but 3.

Verrill's Java-program is capable of drawing fundamental domains for the intersection of two congruence subgroups. In order to not overload the program, I decided to not add this functionality. However, one can manually add such a subgroup. I will give an example of how to do so for the congruence subgroup $\Gamma_0(N_1) \cap \Gamma_1(N_2)$. Since we only have $2 \leq 3$ arguments N_1 and N_2, we do not need to fix any variables. All the coding lines one has to add to the file *DrawFunDoms.m* are highlighted in orange colour.

Step 1: Add the subgroup to the list of groups in form of strings. Let us call the group `Gamma0CapGamma1`. This label is going to be used for the steps 2-8. Modify the already existing variable `$listOfSubgroups` such that the new group is added.

Mathematica

```
In[36]:= $listOfSubgroups = {"SL2Z", "Gamma", "Gamma0", "Gamma1",
          "GammaUpper0", "GammaUpper1", "Gamma0Upper0",
          "GammaNMP", "GammaTheta", "Gamma0CapGamma1"};
```

Step 2: Define the subgroup itself. This step is the most important and error-prone one. The program needs to know whether a given matrix is element of our group. In our case the mathematical definition is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_0(N_1) \cap \Gamma_1(N_2) :\Leftrightarrow \underbrace{a,b,c,d \in \mathbb{Z} \wedge ad - bc = 1}_{\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathsf{SL}_2(\mathbb{Z})} \wedge$$
$$\underbrace{c \equiv 0 \mod N_1}_{\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma_0(N_1)} \wedge$$
$$\underbrace{c \equiv 0 \mod N_2 \wedge a \equiv 1 \mod N_2}_{\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma_1(N_2)}.$$

In Mathematica-language that is

Mathematica

```
In[37]:= (* other definitions *)
         SubgroupMemberQ["Gamma0CapGamma1",{N1_, N2_},
         {{a_, b_}, {c_, d_}}] :=
         AllTrue[{a, b, c, d}, IntegerQ] && ad - bc == 1 &&
         Mod[c, N1] == 0 &&
         Mod[c, N2] == 0 && Mod[a, N2] == Mod[1, N2]
```

The string-argument has to be the one defined at step 1. All other definitions for the function `SubgroupMemberQ` are abbreviated in the above code by `(* other definitions *)`. Also, note that the condition $a \equiv 1 \mod N_2$ is in Mathematica-language `Mod[`a, N_2`]==Mod[1,`N_2`]` instead of `Mod[`a, N_2`]==1`. Otherwise, for instance the congruence subgroup $\Gamma_1(1)$ would be empty.

Step 3: Specify conditions for which the subgroup is empty via the function `SubgroupEmptyQ`. There may be argument values for which the subgroup contains no matrices. Consequently, the program will never stop searching for and finding coset representatives. Hence, the definition of arguments which prevent termination is crucial. For example, the congruence subgroup $\Gamma(N, M, P)$ is empty if and only if $M \nmid NP$. For these arguments, the set of coset representatives is given by $\mathrm{SL}_2(\mathbb{Z})$, the fundamental domain equals the whole upper half complex plane and the cups are $\mathbb{Q} \cup \{\infty i\}$, each point having width ∞. Our group $\Gamma_0(N_1) \cap \Gamma_1(N_2)$ has a finite set of coset representatives for all positive integer values N_1, N_2, thus no further adaption of the function `SubgroupEmptyQ` is necessary. Still, I will demonstrate this modification by defining that our subgroup is never empty:

Mathematica

```
In[38]:= SubgroupEmptyQ["GammaNMP", {n_, m_, p_}] /; Mod[np, m]! = 0
         := True;
         SubgroupEmptyQ["Gamma0CapGamma1", _] := False;
         SubgroupEmptyQ[ _, _ ] := False;
```

Step 4: Define a notation for the group which will be displayed in the drop-down list for selecting a subgroup without parentheses for its arguments. In order to do that, add a definition for the function `GroupNotation` like the following:

Mathematica

```
In[39]:= GroupNotation["SL2Z"] := "SL_2(ℤ)";
         GroupNotation["Gamma"] := "Γ";
         GroupNotation["Gamma0"] := "Γ_0";
         GroupNotation["Gamma1"] := "Γ_1";
         GroupNotation["GammaUpper0"] := "Γ^0";
         GroupNotation["GammaUpper1"] := "Γ^1";
         GroupNotation["Gamma0Upper0"] := "Γ_0^0";
         GroupNotation["GammaNMP"] := "Γ";
         GroupNotation["GammaTheta"] := "Γ_θ";
         GroupNotation["Gamma0CapGamma1"] := "Γ_0 ∩ Γ_1"
         GroupNotation[group_] := group;
```

You can find these and many more symbols in the Mathematica menu bar at *Palettes* → *Special Characters* and/or *Palettes* → *Others* → *Basic Typesetting*.

Step 5: Particularise notations for the subgroup's arguments. This definition will be displayed in the control panel as well as in the graphic. The input variables for our group are N_1 and N_2, hence

Mathematica

```
In[40]:= ArgumentsNotation["SL2Z"] := "";
         ArgumentsNotation["Gamma"] := "N";
         ArgumentsNotation["Gamma0"] := "N";
         ArgumentsNotation["Gamma1"] := "N";
         ArgumentsNotation["GammaUpper0"] := "N";
         ArgumentsNotation["GammaUpper1"] := "N";
         ArgumentsNotation["Gamma0Upper0"] := "N";
         ArgumentsNotation["GammaNMP"] := "N, M, P";
         ArgumentsNotation["GammaTheta"] := "";
         ArgumentsNotation["Gamma0CapGamma1"] := "N_1, N_2";
         ArgumentsNotation[ _ ] := "";
```

Step 6: Modify the function `ArgumentsNumber`. We have to specify whether our group has 0,1,2 or 3 arguments. In our case, that is 2: N_1 and N_2.

Mathematica

```
In[41]:= ArgumentsNumber["SL2Z"] := 0;
         ArgumentsNumber["Gamma"] := 1;
         ArgumentsNumber["Gamma0"] := 1;
         ArgumentsNumber["Gamma1"] := 1;
         ArgumentsNumber["GammaUpper0"] := 1;
         ArgumentsNumber["GammaUpper1"] := 1;
         ArgumentsNumber["Gamma0Upper0"] := 1;
```

```
ArgumentsNumber["GammaNMP"] := 3;
ArgumentsNumber["GammaTheta"] := 0;
ArgumentsNumber["Gamma0CapGamma1"] := 2;
ArgumentsNumber[ _ ] := 0;
```

As mentioned above, if you need more than 3 arguments, fix all arguments but 3 and then define the subgroup for just 3 arguments.

Step 7: Specify reasonable maximal integer values for the group arguments. We might overcharge the program by accidentally trying to draw fundamental domains for a subgroup with very large index. Therefore, we restrict the arguments by upper bounds. For our subgroup $\Gamma_0(N_1) \cap \Gamma_1(N_2)$, let us define $N_1 \in \{1, 2, \ldots, 10\}$ and $N_2 \in \{1, 2, \ldots, 5\}$. We extend the function `ArgumentsMaxValues` as follows:

Mathematica

```
In[42]:= ArgumentsMaxValues["Gamma"] := {8};
         ArgumentsMaxValues["Gamma0"] := {99};
         ArgumentsMaxValues["Gamma1"] := {22};
         ArgumentsMaxValues["GammaUpper0"] := {99};
         ArgumentsMaxValues["GammaUpper1"] := {22};
         ArgumentsMaxValues["Gamma0Upper0"] := {10};
         ArgumentsMaxValues["GammaNMP"] := {5, 5, 5};
         ArgumentsMaxValues["Gamma0CapGamma1"] := {10, 5};
         ArgumentsMaxValues[group_] :=
             Table[5, ArgumentsNumber[group]];
```

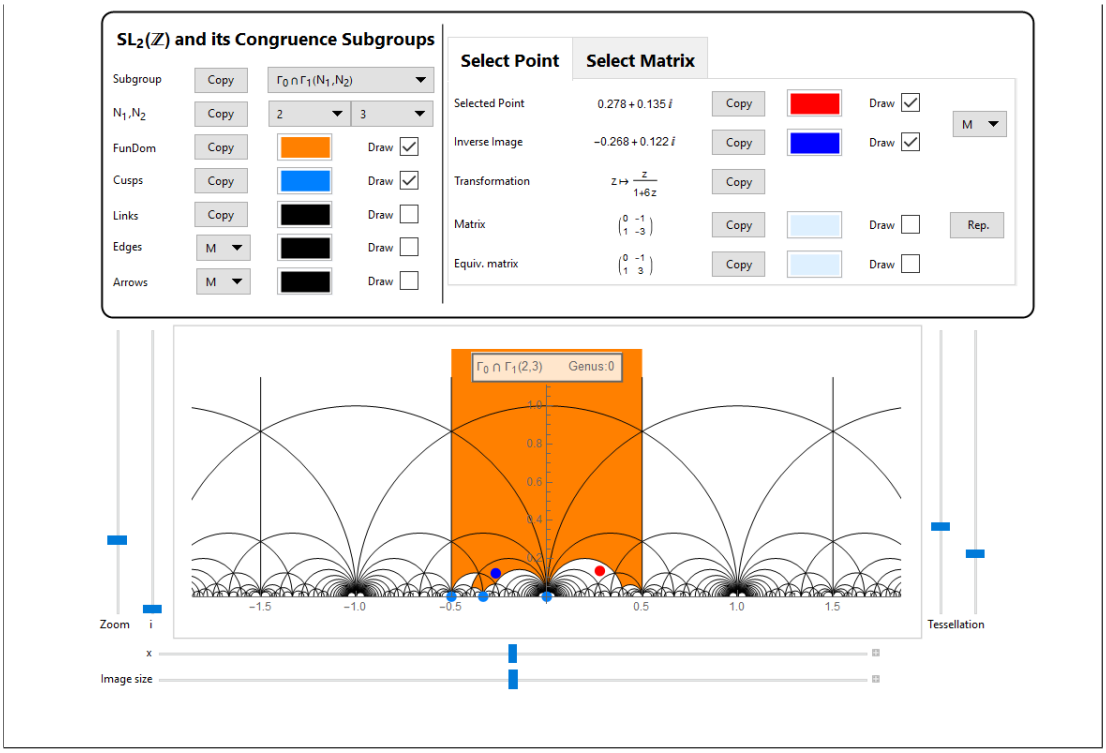
When switching between subgroups, the arguments will remain the same unless at least one of them is greater than the maximal value. In that case, all variables will be reset to 1. For example, if the program is currently showing a fundamental domain for $\Gamma_0(50)$ (index 90) and we then select the subgroup Γ, the program evaluates $\Gamma(1)$ (index 1) instead of $\Gamma(50)$ (index 90000).

Step 8: Save the file. In your current Mathematica-notebook, reload the package by evaluating the command « `DrawFunDoms.m`. After restarting the main program, the new subgroup is listed in the drop-down menu on the left control panel *$SL_2(\mathbb{Z})$ and its Congruence Subgroups*. Once selected, we may choose values for the arguments N_1 and N_2 below. All features (coset representatives, genus, cusps, links, edges, arrows, tab *Select Point*, button "Rep.") are now available for this subgroup.

Mathematica

```
In[43]:= DrawFundamentalDomains[ ]

Out[43]=
```



Besides that, the functions `SubgroupMemberQ`, `EquivalentMatricesQ`, `FunDom`, `ExchangeMatrix`, `CosetRepresentatives`, `CosetRepresentativesWithoutNegatives`, `SubgroupIndex`, `Cusps`, `NumberOfEllipticPointsOrder2`, `NumberOfEllipticPointsOrder3`, `Genus` and `DrawFunDom` may be utilised for the newly defined subgroup as well. In case you still need help or do not know how to specify a subgroup in Mathematica-language, feel free to contact me via e-mail: kainberger.paul@gmail.com.

2.5 Examples

We conclude with some screenshots of fundamental domains for the congruence subgroups $\mathrm{SL}_2(\mathbb{Z})$, Γ, Γ_0, Γ_1, Γ^0, Γ^1, Γ_0^0, $\Gamma(N, M, P)$ and Γ_θ, which are included in the Mathematica package by default:

$\mathbf{SL}_2(\mathbb{Z})$

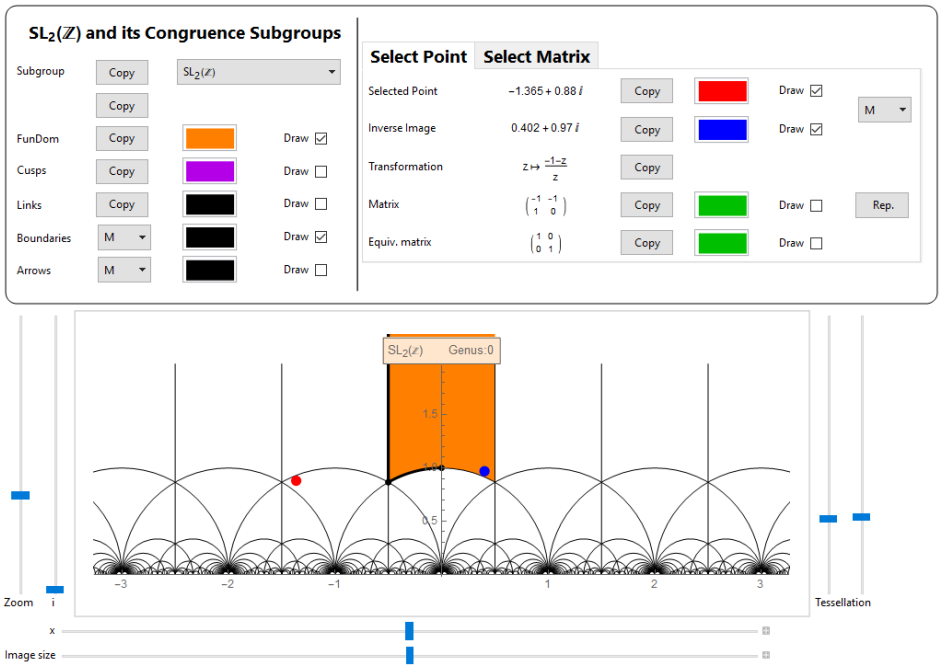


Figure 2.5: A fundamental domain for $\mathrm{SL}_2(\mathbb{Z})$.

$\Gamma(N)$

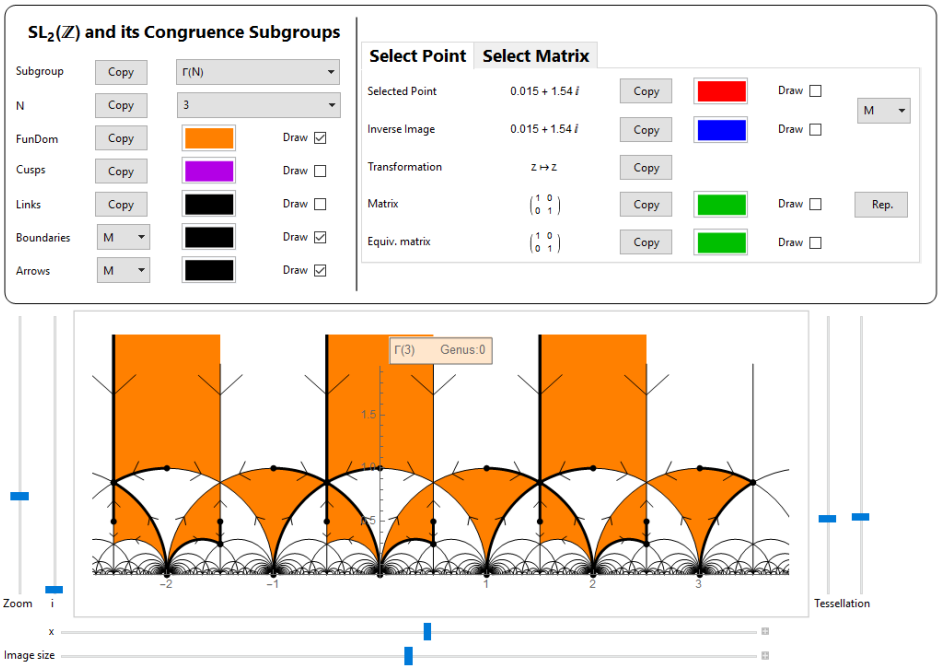


Figure 2.6: A fundamental domain for $\Gamma(3)$.

$\Gamma_0(N)$

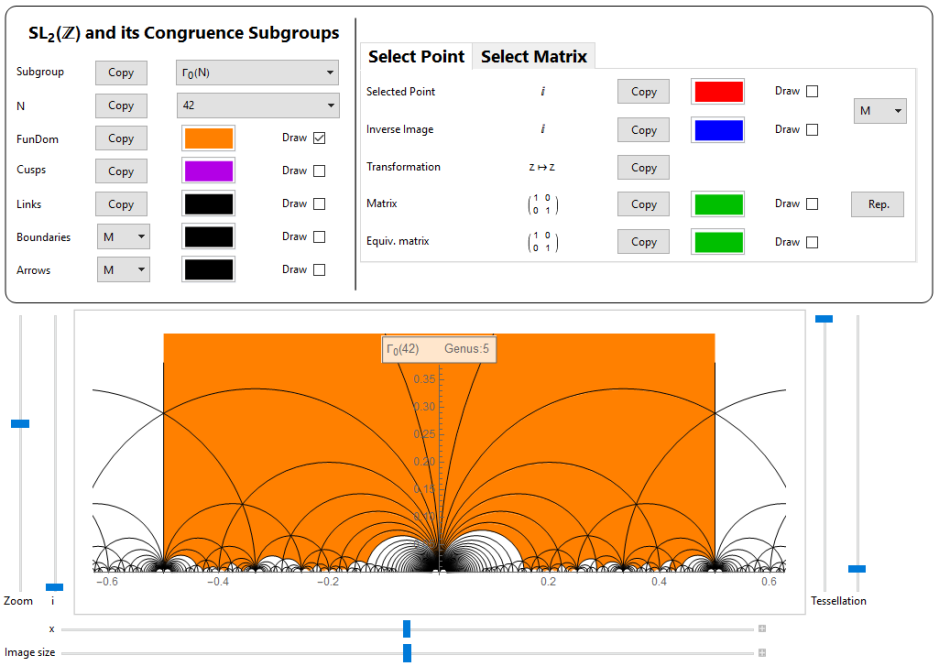


Figure 2.7: A fundamental domain for $\Gamma_0(42)$.

$\Gamma_1(N)$

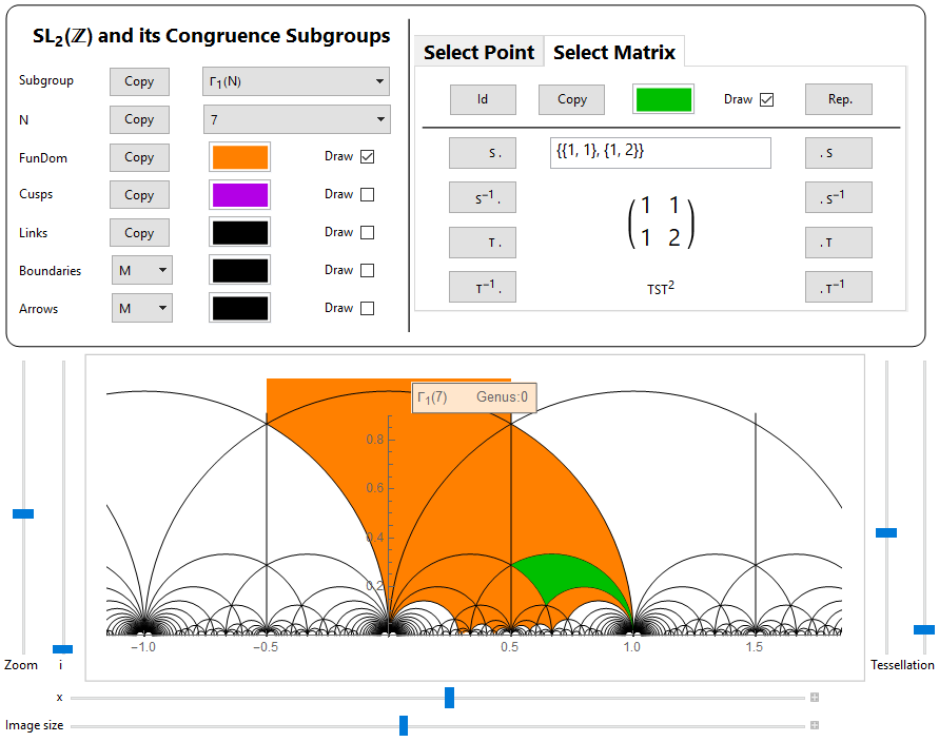


Figure 2.8: A fundamental domain for $\Gamma_1(7)$.

$\Gamma^0(N)$



Figure 2.9: A fundamental domain for $\Gamma^0(100)$.

$\Gamma^1(N)$

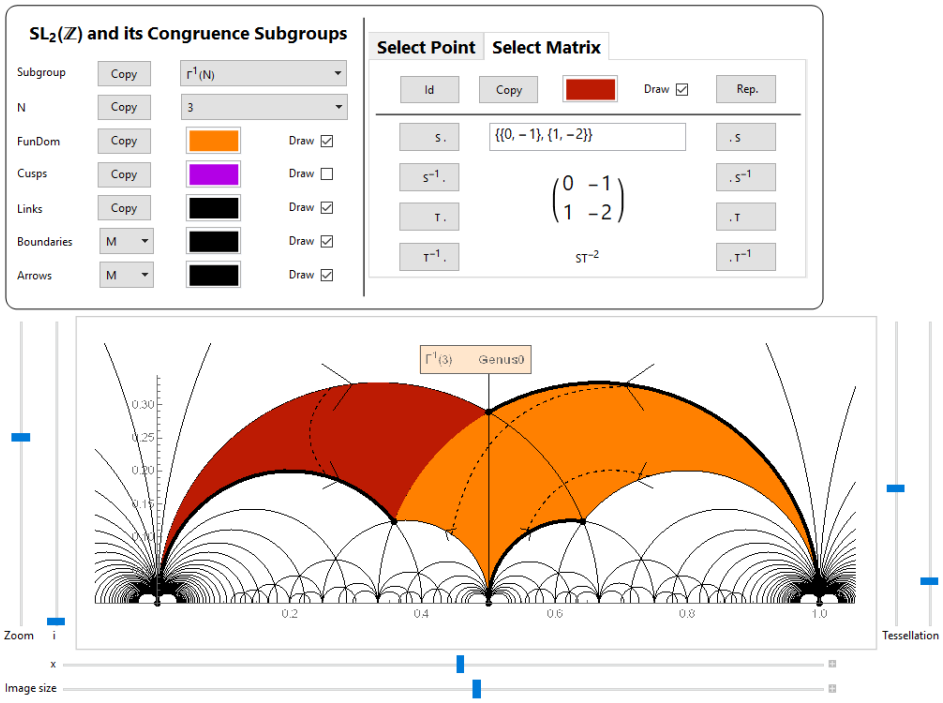


Figure 2.10: A fundamental domain for $\Gamma^1(3)$.

$\Gamma_0^0(N)$

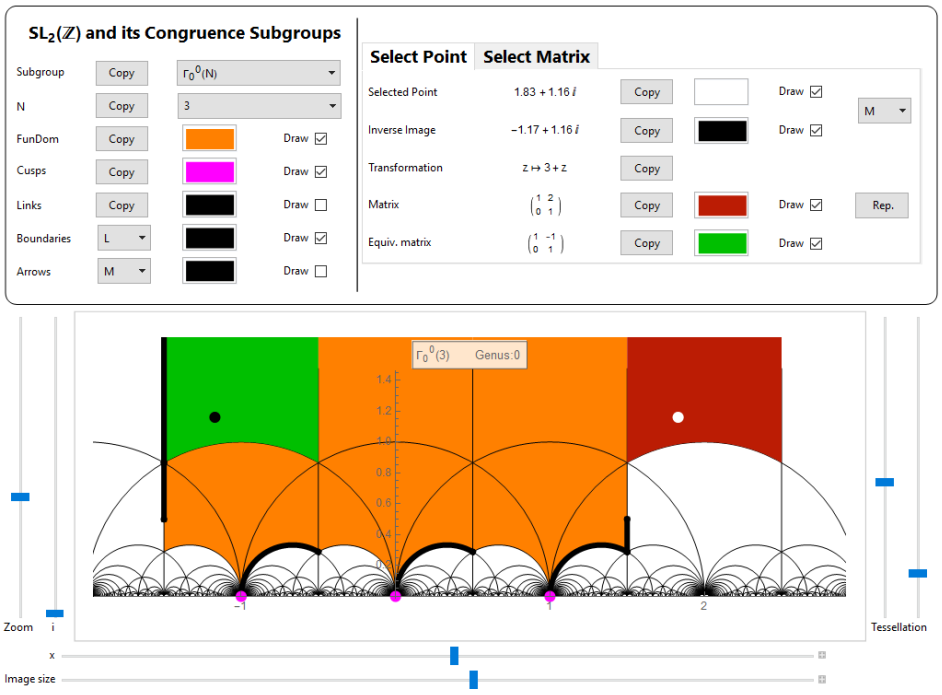


Figure 2.11: A fundamental domain for $\Gamma_0^0(3)$.

$\Gamma(N, M, P)$

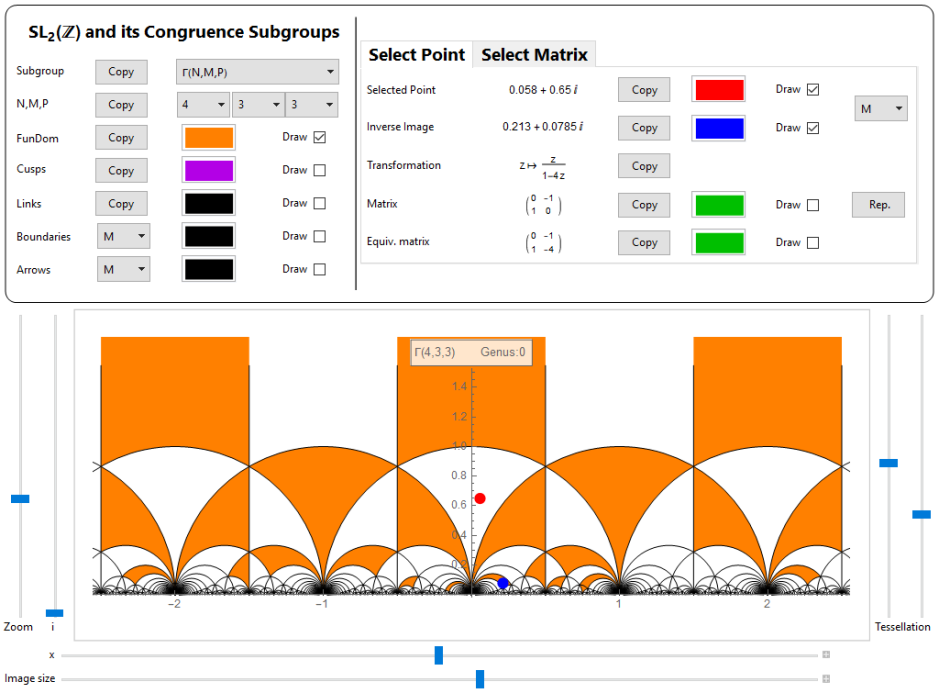


Figure 2.12: A fundamental domain for $\Gamma(4, 3, 3)$.

Γ_θ

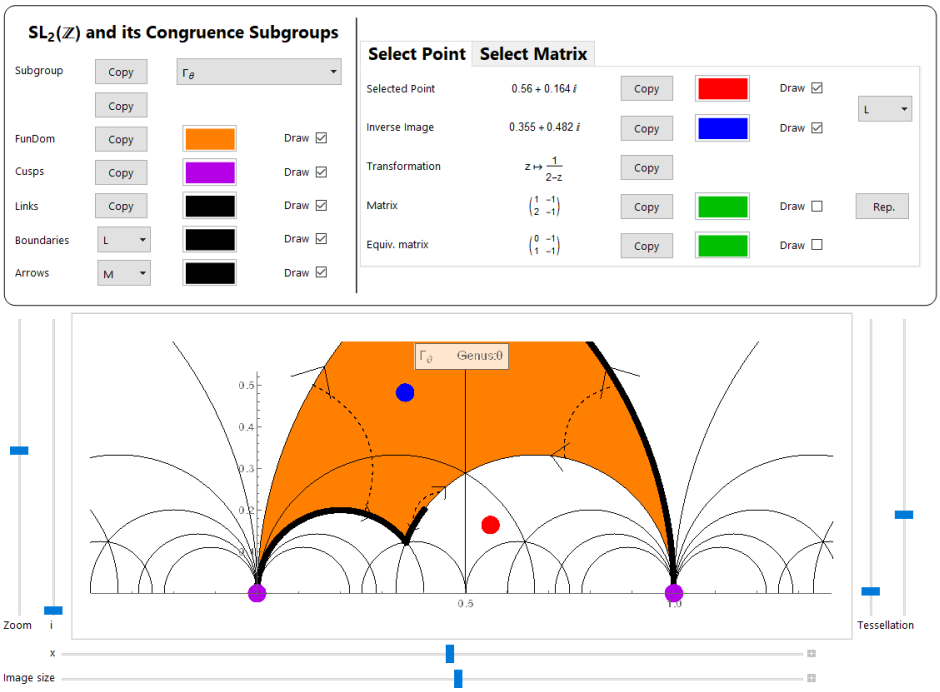


Figure 2.13: A fundamental domain for Γ_θ.

Bibliography

[1] Constantin Cristian Caranica. "Algorithms related to subgroups of the modular group". PhD thesis. LSU Doctoral Dissertations, 2009. URL: https://digitalcommons.lsu.edu/gradschool_dissertations/854.

[2] Keith Conrad. *SL(2,Z)*. 2001. URL: https://kconrad.math.uconn.edu/blurbs/grouptheory/SL(2,Z).pdf.

[3] C. J. Cummins. "Congruence Subgroups of Groups Commensurable with PSL$(2, \mathbb{Z})$ of Genus 0 and 1". In: *Experimental Mathematics* 13.3 (2004), pp. 361 –382.

[4] Fred Diamond and Jerry Shurman. *A First Course in Modular Forms*. Springer Science+Business Media, 2005.

[5] Martin Eichler. *Introduction to the Theory of Algebraic Numbers and Functions*. Trans. by George Striker. Vol. 23. Pure and Applied Mathematics. Academic Press, 1966.

[6] Thomas W. Hungerford. *Algebra*. Vol. 74. Graduate Texts in Mathematics. Springer New York, 1974.

[7] Wolfram Research, Inc. *Mathematica, Version 12.1*. Champaign, IL, 2020. URL: https://www.wolfram.com/mathematica.

[8] Jerzy Kocik. *On the Dedekind tessellation*. Dec. 2019. arXiv: 1912.05768 [math.HO].

[9] Joseph Lehner. *A Short Course in Automorphic Functions*. Holt, Rinehart and Winston, 1966.

[10] Joseph Lehner. *Lectures on Modular Forms*. Applied Mathematics Series, 61. National Bureau of Standards, 1969.

[11] James S. Milne. *Modular Functions and Modular Forms (v1.31)*. Available at www.jmilne.org/math/. 2017.

[12] Antonio Lascurain Orive and Rubén Molina Hernández. "On Fundamental Domains for Subgroups of Isometries Acting in $\mathbb{H}^n$". In: *International Scholarly Research Notices* 2012 (2012).

[13] Thomas Ponweiser. "Computer Algebra and Analysis: Complex Variables Visualized". Diploma thesis. Research Institute for Symbolic Computation (RISC), 2014.

[14] Thomas Ponweiser. *Modular Group*. Mathematica package. URL: https://www3.risc.jku.at/research/combinat/software/ModularGroup/index.html.

[15] Robert A. Rankin. *Modular Forms and Functions*. Cambridge University Press, 1977.

[16] Bruno Schoeneberg. *Elliptic Modular Functions. An Introduction.* Trans. by J.R. Smart and E.A. Schwandt. 1st ed. Springer-Verlag Berlin Heidelberg, 1974.

[17] Jean-Pierre Serre. *A Course in Arithmetic.* Graduate Texts in Mathematics. Springer, 1973.

[18] Helena A. Verrill. *Algorithm for Drawing Fundamental Domains.* 2001.

[19] Helena A. Verrill. *Fundamental Domain drawer.* 2001. URL: `https://wstein.org/Tables/fundomain/index2.html`.