Algorithm for Drawing Fundamental Domains.

H. A. Verrill
http://hverrill.net/
verrill@math.ku.dk

30th January 2001

1 Introduction

The "FunDomain" Java program was mostly written in January 2000. It is a program which computes and displays the fundamental domain of certain congruence subgroups of $\mathrm{SL}_2(\mathbf{Z})$ acting on the upper half plane, and allows manipulation of the retulting figures. It consists of 8 java files, giving 12 classes. The java files can be downloaded from http://hverrill.net/fundomain/java-source/. The algorithm described below is implemented in the file RepList.java. There are two main problems to deal with for writing this program. One is how to find the coset representatives of the given group in $\mathrm{SL}_2(\mathbf{Z})$. The other is how to actually draw the picture. In fact more of the work went into getting the latter part of the program to work reasonably. But since the drawing part is not so interesting mathematically, I will not describe that here. The method of finding the coset representatives, described below is very simple, though currently I do not know of other methods that are significantly different. One can use a different fundamental domain for $\mathrm{SL}_2(\mathbf{Z})$, as is done by [3], though an algorithm for constructing the fundamental domain in that case, as described in [1] § 3.1 is essentially the same as the algorithm described below, though cleverly uses a certain union of domains for $\mathrm{SL}_2(\mathbf{Z})$ to improve matters.

One application of finding the fundamental domain of a congruence subgroup is to give generators for the group. This feature may be included in a future version of the java package, though currently I have only implemented this in magma. Notes on this application are also included below. All of the mathematics is standard. However, I recall some of the general theory below. For more details see [5].

2 Notation

Denote by Γ a congruence subgroup of $\mathrm{SL}_2(\mathbf{Z})$. In the current program on the web page, Γ is one of the groups $\Gamma_0(N)$, $\Gamma^0(N)$, $\Gamma_1(N)$, $\Gamma^1(N)$ or $\Gamma(N)$, for some positive integer N. In the version which can be downloaded from the source code part of the page, intersections of two such groups is also possible. It's easy to write versions for other subgroups; what changes is the definition of equivalence (denoted $\sim$) of coset representatives.

I use the following notation for certain elements of $\mathrm{SL}_2(\mathbf{Z})$:

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad S = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}$$

Table of other notation:

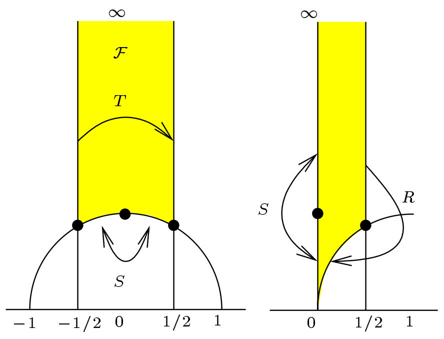
Γ	congruence subgroup in $\mathrm{SL}_2(\mathbf{Z})$
$\mathcal{R}$	A set of coset representatives of Γ in $\mathrm{SL}_2(\mathbf{Z})$
$\mathfrak{h}$	The upper half complex plane
$X(\Gamma)$	The completion of the quotient $\Gamma \setminus \mathfrak{h}$
$\mathcal{F}$	Certain fundamental domain for $\mathrm{SL}_2(\mathbf{Z})$ acting on $\mathfrak{h}$
$A \sim B$	For $A, B \in \mathrm{SL}_2(\mathbf{Z})$ we write $A \sim B$ to mean $AB^{-1} \in \Gamma$.

3 Some Theory

The group $\mathrm{SL}_2(\mathbf{Z})$, and any subgroup Γ, acts on the left on $\mathfrak{h}$, by:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} z = \frac{az+b}{cz+d}$$

A fundamental domain F for Γ is a region in the upper half complex plane such that for all $z \in \mathfrak{h}$ there is exactly one element $\gamma \in \Gamma$ with $\gamma z \in F$. A fundemental domain should be connected. There are several standard choices for a fundamental domain for $\mathrm{SL}_2(\mathbf{Z})$ here are two, pictured together with matrices giving identifications of the edges:



In the diagram elliptic points are marked. The fundamental domain on the right is used in the program. This is the region

$$\mathcal{F} = \{z \in \mathfrak{h} | -1/2 < z \leq 1/2, \text{ and } |z| \geq 1, \text{ and } |z| > 1 \text{ if } \Re(z) < 0\}.$$

The domain on the right is a better choice for certain theoretical purposes, and this domain, and unions of this domain, were used in an algorithm described in [3] and [1].

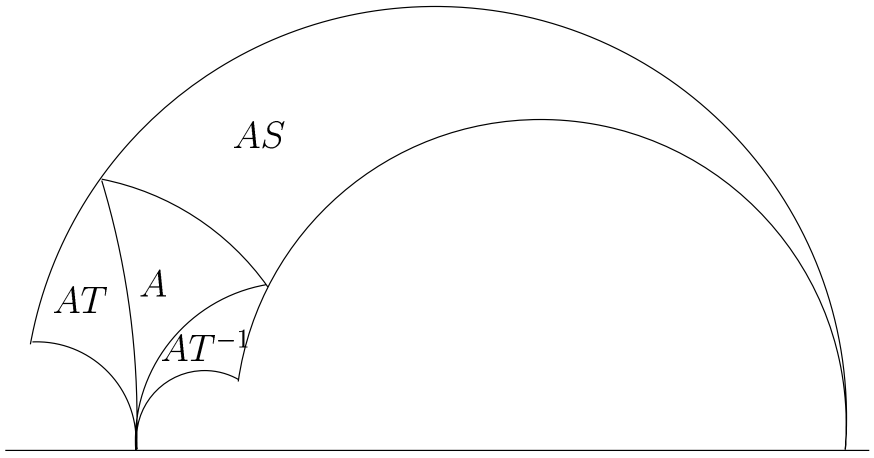
Since Γ is a subgroup of finite index in $\mathrm{SL}_2(\mathbf{Z})$, we have that $\mathrm{SL}_2(\mathbf{Z}) = \Gamma M_1 \sqcup \Gamma M_2 \dots \Gamma M_n$, for a list of right coset representatives $M_i \in \mathrm{SL}_2(\mathbf{Z})$. We have $\mathfrak{h} = \mathrm{SL}_2(\mathbf{Z})\mathcal{F}$, as a disjoint union of translates of $\mathcal{F}$ under the action of $\mathrm{SL}_2(\mathbf{Z})$ so

$$\mathfrak{h} = (\Gamma M_1 \sqcup \Gamma M_2 \dots \Gamma M_n)\mathcal{F} = \Gamma(M_1\mathcal{F} \sqcup M_2\mathcal{F} \sqcup \dots M_n\mathcal{F}).$$

This is also a disjoint union over the translates under Γ, so $\mathcal{G} = M_1\mathcal{F} \sqcup M_2\mathcal{F} \sqcup \dots M_n\mathcal{F}$ is a fundamental domain for Γ.

So, to find a fundemantal domain $\mathcal{G}$ for Γ we need a list of right coset representatives for Γ in $\mathrm{SL}_2(\mathbf{Z})$. However, we also need to choose these representatives so that $\mathcal{G}$ is connected.

For any trangle $A\mathcal{F}$, the adjacent triangles correspond to translates of $\mathcal{F}$ by AT, AT^{-1}, and AS, for example, in the diagram below, where triangle $M\mathcal{F}$ is labeled by matrix M.



So, if A and B are chosen coset represenatives $\mathcal{R}$, we also need that in $\mathcal{R}$ there is some sequence of matrices $A = M_{i_1}, M_{i_2}, \dots M_{i_k} = B$, with $M_{i_j}^{-1}M_{i_{j+1}} \in \{\pm S, T, T^{-1}\}$ for $1 \leq j < k$.

3.0.1 distance

Since the aim of the program is to give a connected fundamental domain that can be nicely drawn, we want to choose representatives so that the corresponding triangles are as "large" as possible, so they can be seen and don't just end up as small dots on the screen. In order to give a rough measure of how large something is, I define a "distance" from I. The distance is computed recursively from the list of coset representatives found. This is not most "accurate" definition of distance, in the sense that it's dependent on the list constructed, but it's easy to compute and seems to give reasonable pictures.

Definition: First, I has distance 0 from itself. Also any T^n has distance 0 from I. For any other matrix M_j, if the distance here is d, then the distance for M_jT, M_jS, and M_jT^{-1} is $d+1$, unless one of these matrices has already been enumerated and given some smaller distance.

4 The algorithm

We want to construct a list of coset represenatives $\mathcal{R} \subset \mathrm{SL}_2(\mathbf{Z})$, so that the correseponding domain is connected. We will construct a sequence of lists,

$$\mathcal{R}_1 = \{M_1\} \subset \mathcal{R}_2 \subset \ldots \mathcal{R}_n = \mathcal{R},$$

where n is the index of Γ in $SL_2(\mathbf{Z})$. So at step i we adjoin M_i to $\mathcal{R}_{i-1}$, to obtain $\mathcal{R}_i$. After n steps we're done.

As the list $\mathcal{R}$ is constucted, I also construct the information telling me about the associated graph. For each element M_i in the list, we want to know the following:

Data stored for each $M_i \in \mathcal{R}$:
Have we checked in the T direction?
Have we checked in the T^{-1} direction?
Have we checked in the S direction?

In T direction:	for which j does $M_iT \sim M_j$?
In T^{-1} direction:	for which j does $M_iT^{-1} \sim M_j$?
In S direction:	for which j does $M_iS \sim M_j$?
If $M_iT \sim M_j$:	does $M_iT = M_j$?
If $M_iT^{-1} \sim M_j$:	does $M_iT^{-1} = M_j$?
If $M_iS \sim M_j$:	does $M_iS = M_j$?

What is the distance of M_i from I?

Step 1. The first coset representative, M_1 is any choice of matrix to start with. In practice, I usually choose $M = I$, but for $\Gamma^0(N)$ and $\Gamma^1(N)$ I decided to choose $M = T^{-\lfloor N/2 \rfloor}$ so that the domain in this case ends up centered about 0.

Step i+1. Suppose we're at stage $i+1$, having just constructed $\mathcal{R}_i$.

Substep 1. Pick some $M_j \in \mathcal{R}_i$.

Substep 2. Now ask the questionns about M_j:

Questions:
Have we looked in the T direction? If not do so...
Have we looked in the T^{-1} direction? If not do so...
Have we looked in the S direction? If not do so...

Substep 3. Suppose from the previous two steps we have a matrix M_j, and we are looking in the T direction (any other direction works in the same way). First check through the list $\mathcal{R}_i$ to see if there is some M_k with $M_jT \sim M_k$. If there is, update the list of data appropriately, including answering whether $M_jT = M_k$, and the data for the T^{-1} direction of M_k. If there is no such M_k in $\mathcal{R}_i$, then we have a new coset representative, M_jT.

Now we can either define $M_{i+1} = M_jT$, or we can choose some other $M_{i+1} \sim M_jT$. In the program, to maximize the size of the picture, I look at all the neighbours of M_jT that are already listed in $\mathcal{R}_i$. Suppose these are $M_{k_1} \sim M_jT^2$, $M_{k_2} \sim M_jTS$ and M_j. (Note, there are at most these three; there might only be one or two neighbours in $\mathcal{R}_i$.) I pick the neighbour with largest distance from I (as defined above), and then choose $M_{i+1} = M_{k_1}T^{-1}, M_{k_2}S$, or M_jT, depending on which of the three this is. Then $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{M_{i+1}\}$, and the list of data is updated appropriately.

Termination. If there are no directions at substep 2 for any matrix in the list, then the algorithm terminates. A complete set of coset representatives has been found.

4.1 More details

4.1.1 At step $i+1$, substep 1, how do we decide which M_j to work with?

This depends on exactly what your aim is. Since I want to draw pictures with as large as possible triangles, so that they can be seen, I choose M_j to have the smallest "distance" from I. If there is more than one M_j with the smallest distance I just take the first one in the list.

4.1.2 Which order to ask the questions of step $i+1$, substep 2?

This depends on your aim. If you wanted to have only one representative for each cusp, you could always ask about T direction first, and in the step of which matrix to choose, you'd keep choosing the matrix you just added, and keep going in the T direction, until this is not possible. In this case you wouldn't really have to store any information about the T^{-1} direction, though this could still be useful.

If you want to use the program for finding generators for Γ, it's best to always look in the S direction first.

My aim was to try and have the best looking "picture", and so though I choose the T direction first, which is a good idea when working with $\Gamma^0(N)$ and $\Gamma^1(N)$, it doens't seem to matter too much, because the choice I make for which M_j to work with at each step has more effect.

4.2 Cusps and Elliptic points

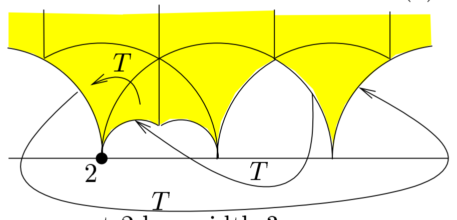
These can be found from the table of adjacencies. I now give a brief description of each case and an illustrative picture of examples.

4.2.1 Cusps

For a matrix $\gamma = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, the corresponding cusp is $\gamma\infty = \frac{a}{c}$. Starting from any $M_i = \gamma$ in $\mathcal{R}$, we can use the adjacency table to find a list $M_{i_1} = M_i, M_{i_2}, \ldots M_{i_m}$ with $M_{i_j}T \sim Mi_{j+1}$ for $1 \leq j < m$, and $M_{i_m}T \sim M_i$. Then the cusp a/c has width m.

Cusp example:

Part of a fundamental domain for $\Gamma^0(6)$



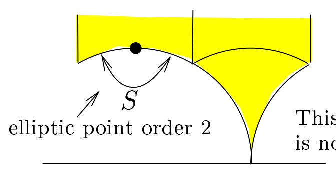
The cusp at 2 has width 3

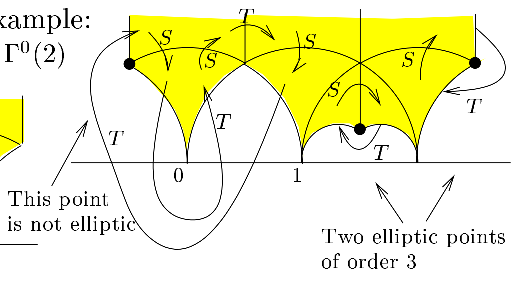
Order 3 elliptic point example:

Fundamental domain for $\Gamma^0(7)$

Order 2 elliptic point example:

Fundamental domain for $\Gamma^0(2)$





4.2.2 Elliptic points of order 2

The number of elliptic points of order 2 is just the number of M_j in $\mathcal{R}$ with $M_jS \sim M_j$.

4.2.3 Elliptic points of order 3

To find elliptic points of order 3, start at any matrix M_j in $\mathcal{R}$, and use the adjacency table to find the representatives equivalent to

$$M_j, M_jT, M_jTS, M_jTST, M_jTSTS, M_jTSTST$$

if this set consists of at most 2 matrices, then we have an order 3 elliptic point.

4.3 Note on Formulae

There are simple formulae for the index of these subgroups in $\mathrm{SL}_2(\mathbf{Z})$, e.g.:

$$\begin{aligned}
[\mathrm{SL}_2(\mathbf{Z}) : \Gamma_0(N)] &= N\prod_{p|N}\left(1+\frac{1}{p}\right)\\
[\Gamma_0(N) : \Gamma_1(N)] &= N\prod_{p|N}\left(1-\frac{1}{p}\right)\\
[\Gamma_1(N) : \Gamma(N)] &= N,
\end{aligned}$$

where the products run over the prime divisors of N.

There are also formulae for finding the numbers of cusps and elliptic points. However, I did not use these. For the index, there seemed little point, since the algorithm described above terminates when a full set of coset representatives is found. If I was not interested in the way the cosets are related then it would probably be necessary to use these formualae. Also, using the formulae would be a good way to specify the length of the array used in advance. So perhaps this could be incorperated in a future version. Finding the number of cusps and their widths, and the elliptic points from the diagram as above seemed like a good test that the program was doing the correct thing. The number of cusps and elliptic points was used to find the genus using the following formula:

$$g = 1 + \frac{1}{12}(\text{index} - 6\#\text{cusps} - 4e_2 - 3e_3),$$

where e_2 is the number of elliptic points of order 2, and e_3 is the number of elliptic points of order 3. All these formulae can be found in [5].

5 Subgroups

There are various ways of reducing the amount of work done to find a fundmental domain, by working in stages, e.g., using the computations for Γ to help in the computations of the fundamental domain for a subgroup of Γ.

This has not yet been done to a great extent in the program, but is used for the computation of $\Gamma(N)$ from $\Gamma_1(N)$ as follows.

5.1 $\Gamma(N)$ from $\Gamma_1(N)$

Note that $\Gamma(N)$ is a normal subgroup of $\Gamma_1(N)$, and that

$$\Gamma(N) \setminus \Gamma_1(N) = \langle \Gamma(N)T \rangle \cong \mathbf{Z}/N\mathbf{Z}.$$

So we have that if $\mathcal{R}$ is the set of coset representatives of $\Gamma_1(N)$, then the following set is a set of coset representatives of $\Gamma(N)$:

$$\{T^i x \in \mathrm{SL}_2(\mathbf{Z}) | x \in \mathcal{R}, 0 \le i < N\}.$$

So, it is very easy to write a new list of coset representatives for $\Gamma(N)$ from the list for $\Gamma_1(N)$.

Next the table of adjacencies must be updated. This is necessary because the program has the "edit" mode to allow the user to change the domain found, and also this is needed for the application of finding a set of generators for $\Gamma(N)$.

Suppose $r_i, r_j \in \Gamma_1(N)$ and $r_iM \sim r_j$, for $M = T, T^{-1}$ or S. This means $r_iMr_j^{-1} \in \Gamma_1(N)$. So for some integer m, we have $r_iTr_j^{-1} \in \Gamma(N)T^m$. So $r_iTr_j^{-1}T^{-m} \in \Gamma(N)$. Since $\Gamma(N)$ is normal in $\Gamma_1(N)$, we have $T^kr_iMr_j^{-1}T^{-m-k} \in \Gamma(N)$ for any integer k. So $T^kr_iM \sim T^{k+m}r_j$. With the above set of reprsentatives, we have $T^{\mod(k,N)}r_iM \sim T^{\mod(k+m,N)}r_j$, where $\mod(k,N)$ is an integer such that $0 \leq \mod(k,N) < N$ and $N|(k - \mod(k,N))$. Write $A \overset{M}{\mapsto} B$ to mean $AM \sim B$, then to summarize, we have:

$$\begin{array}{ccc} \text{for each relation} & & \text{we get N relations} \\ r_i \overset{M}{\mapsto} r_j & \longrightarrow & T^kr_i \overset{M}{\mapsto} T^{\mod(k+m)}r_j \\ \text{in } \Gamma_1(N) & & \text{in } \Gamma(N), \text{ for } 0 \leq k < N. \end{array}$$

If $r_iM = r_j$, then we get $T^kr_iM = T^kr_j$, and also we have some other equalities, which happen when $r_iMr_j^{-1} = T$, which gives $T^kr_iM = T^{k+1}r_j$ for $0 \leq k < N$.

6 Generators for Γ

Given the fundamental domain, it is then a simple matter to find generators for Γ. The easiest way to get a set of generators (which usually is too big by at least factor of two) is just to take the set

$$\{r_iMr_j^{-1} | r_iM \sim r_j, M \in \{T, S\}\}.$$

Better sets of generators can be found from the theory of groups acting on trees, and extracting the tree from the adjacency table. The current java program does not make this computation, though I have written a magma program to do this. If you are intersted in finding generators in this way see the first chapter of [2]. Another good reference is [4]. See also [3] and [1] for an approach which gives the generators much more effectively from a certain fundamental domain.

References

[1] Chan, S.-P., Lang, M.-L., Lim, C.-H., and Tan, S.-P. *Special polygons for subgroups of the modular group and applications.* Internat. J. Math. 4 (1993), no. 1, 11–34.

[2] Dicks, Warren and Dunwoody, M. J. *Groups acting on graphs*, Cambridge Studies in Advanced Mathematics, 17, Cambridge University Press, Cambridge-New York, 1989.

[3] Kulkarni, R. S., *An arithmetic-geometric method in the study of the subgroups of the modular group.* Amer. J. Math. 113 (1991), no. 6, 1053–1133.

[4] Serre, J. P., *Trees*, Springer-Verlag, Berlin-New York, 1980.

[5] Shimura, G. *Introduction to the arithmetic theory of automorphic functions*, Reprint of the 1971 original, Kan Memorial Lectures, 1, Princeton University Press, Princeton, NJ, 1994.