Demo for Package DrawFunDoms.m

Version 1.1 (June 2022)

In[6]:=
```
<< DrawFunDoms.m
```

About

This is a demonstration for the Mathematica package *DrawFunDoms.m*.
The package was written in Mathematica version 12.1.

It was carried out in 2020/2021 by Paul Kainberger as part of his master's thesis under supervision of Univ.-Prof. Dr. Peter Paule (RISC). The thesis, which describes the program and provides mathematical theory behind it, is available here.

License

Version History

Version 1.1 (June 2022)

Bug fix for cusp width: The width of the cusps for congruence subgroups which do not contain the negative identity matrix is now calculated correctly. Thank you Univ.-Prof. Dr. Peter Paule (RISC) for pointing that out.

Version 1.0 (August 2021)

First release.

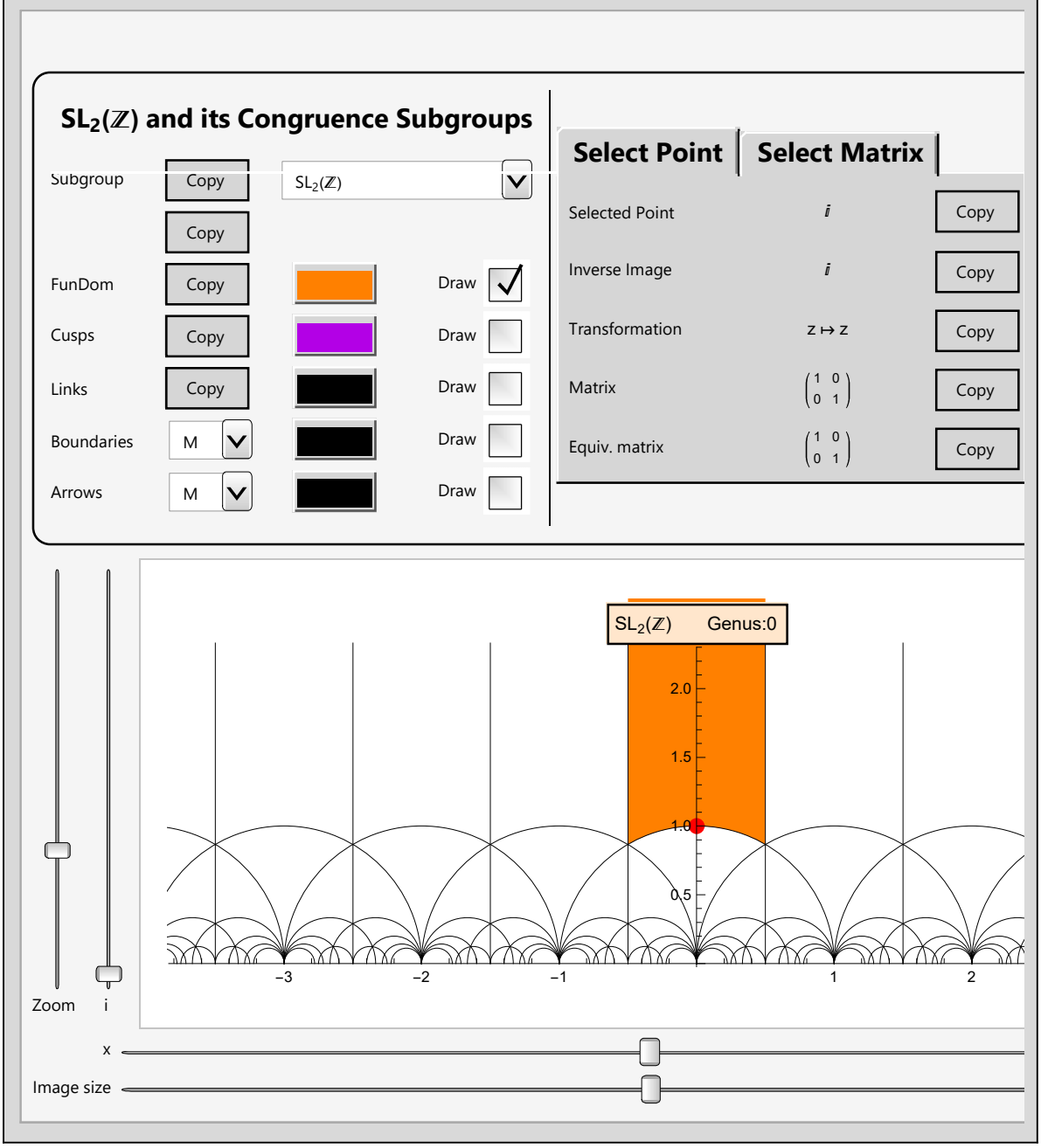
Main program

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 of this package gets loaded by

In[7]:=
```
DrawFundamentalDomains[]
```

Out[7]=



In[ ]:=
```
FindMatrix[I]
```

Out[ ]=
```
{{1, 0}, {0, 1}}
```

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

In[ ]:=
```
SetOptions[EvaluationNotebook[], Magnification → 0.6]
```

The above example sets the magnification for the current notebook 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.

Constant matrices

For a convenient use of the package's functions, one may utilise the following frequently needed constant matrices:

```
In[ ]:= MatrixForm /@ {$Id, $S, $SInv, $T, $TInv}
```

$$\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}\right\}$$

```
In[ ]:= $S.$SInv == $Id
Out[ ]= True
```

```
In[ ]:= $TInv.$T == $Id
Out[ ]= True
```

Other useful functions

Most functions require a congruence subgroup's name in form of a string as argument. The following subgroups are available:

```
In[ ]:= $listOfSubgroups
Out[ ]= {SL2Z, Gamma, Gamma0, Gamma1, GammaUpper0,
  GammaUpper1, Gamma0Upper0, GammaNMP, Lambda}
```

```
In[ ]:= SubgroupNotation /@ $listOfSubgroups
```

$$\left\{\mathrm{SL}_2(\mathbb{Z}),\ \Gamma,\ \Gamma_0,\ \Gamma_1,\ \Gamma^0,\ \Gamma^1,\ \Gamma_0{}^0,\ \Gamma,\ \Lambda\right\}$$

The subgroup's arguments are given as list of positive integers where its length depends on the number of needed arguments.

```
In[ ]:= SubgroupNotation[#] ~~ "(" ~~ ArgumentsNotation[#] ~~ ")" & /@ $listOfSubgroups
```

$$\left\{\mathrm{SL}_2(\mathbb{Z})(),\ \Gamma(\mathrm{N}),\ \Gamma_0(\mathrm{N}),\ \Gamma_1(\mathrm{N}),\ \Gamma^0(\mathrm{N}),\ \Gamma^1(\mathrm{N}),\ \Gamma_0{}^0(\mathrm{N}),\ \Gamma(\mathrm{N},\mathrm{M},\mathrm{P}),\ \Lambda()\right\}$$

$SL_2(\mathbb{Z})$ and Congruence Subgroups

GroupAction

Map points from the upper half complex plane using a matrix in $SL_2(\mathbb{Z})$.

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

```
Out[ ]= -1/2 + I/2
```

Map one point using several matrices:

```
In[ ]:= GroupAction[#, "z"] & /@ {$Id, $T}
```

```
Out[ ]= {z, 1 + z}
```

Map several points using one matrix:

```
In[ ]:= GroupAction[$T, #] & /@ {0, 1}
```

```
Out[ ]= {1, 2}
```

Map several points using several matrices:

```
In[ ]:= Outer[GroupAction, {$Id, $T}, {0, -1, 1}, 1]
```

```
Out[ ]= {{0, -1, 1}, {1, 0, 2}}
```

SubgroupMemberQ

Given a matrix, test if it is element of a specific congruence subgroup.

```
In[ ]:= SubgroupMemberQ["Gamma", {2}, $T.$SInv]
```

```
Out[ ]= False
```

```
In[ ]:= SubgroupMemberQ["GammaNMP", {3, 2, 2}, $T.$T]
```

```
Out[ ]= True
```

Equivalent Matrices

Check whether two given matrices are equivalent in a given congruence subgroup.

In[•]:=
```
EquivalentMatricesQ["Gamma", {3}, MatrixPower[$T, 6], $Id]
```
Out[•]=
```
True
```

In[•]:=
```
EquivalentMatricesQ["Gamma", {4}, MatrixPower[$T, 6], $Id]
```
Out[•]=
```
False
```

FindMatrix

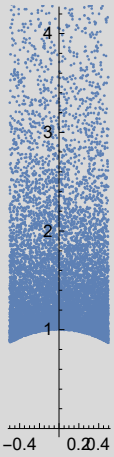
For a given point z in the upper half complex plane, find a matrix such that its inverse maps z into the standard fundamental domain.

In[•]:=
```
FindMatrix[1.5 + 0.22 ⅈ]
```
Out[•]=
```
{{3, 1}, {2, 1}}
```

In[•]:=
```
FindMatrix[0.0000001 + 0.0000001 I]
```
Out[•]=
```
{{0, -1}, {1, -5 000 000}}
```

Map random points from the upper half complex plane into the standard fundamental domain:

In[•]:=
```
ComplexListPlot[
 GroupAction[Inverse[FindMatrix[#]], #] & /@ RandomComplex[{-100, 100 + I}, 10 000]]
```
Out[•]=



The standard fundamental domain has closed and open boundaries:

In[•]:=
```
FindMatrix[-0.5 + I]
```
Out[•]=
```
{{1, 0}, {0, 1}}
```

In[•]:=
```
FindMatrix[0.5 + I]
```
Out[•]=
```
{{1, 1}, {0, 1}}
```

Random Matrix

RandomSL2Z[ ] gives a pseudorandom matrix in $SL_2(\mathbb{Z})$.

In[ ]:=
```
RandomSL2Z[]
```

Out[ ]=
```
{{-961, -853}, {436, 387}}
```

An optional positive integer argument (default is 1000) defines an upper bound for the absolute value of each matrix entry.

In[ ]:=
```
RandomSL2Z[5]
```

Out[ ]=
```
{{-5, 1}, {-1, 0}}
```

All these matrices are elements of $SL_2(\mathbb{Z})$:

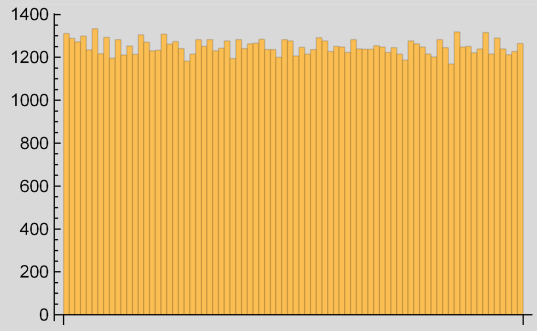
In[ ]:=
```
AllTrue[Table[RandomSL2Z[100], {10000}], SubgroupMemberQ["SL2Z", {}, #] &]
```

Out[ ]=
```
True
```

The matrices are distributed evenly:

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

Out[ ]=



Generate Matrices

We generate matrices with MaxNorm ≤ *n*, where *n* is a given positive integer. For a matrix *M*, MaxNorm[*M*] is defined as the maximum of all entries' absolute values. This way, the generated matrices will be all possible matrices in $SL_2(\mathbb{Z})$ cumulated around the origin. The positive integer n can be seen as radius around the origin and all matrices inside this radius will be found

In[ ]:=
```
MatrixForm /@ GenerateMatrices[1]
```

Out[ ]=

$$\left\{\begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right\}$$

In[ ]:=
```
Length[GenerateMatrices[100]]
```
Out[ ]=
```
48 698
```

In[ ]:=
```
AllTrue[GenerateMatrices[100], SubgroupMemberQ["SL2Z", {}, #] &]
```
Out[ ]=
```
True
```

We neglect negative matrices (for a matrix **{{a,b},{c,d}}** we always have **a≥0** and if **a==0**, then **d≥0** and if also **d=0,** then **b>0**), because for a matrix *M*, the matrices *M* and *-M* have the same image of the fundamental domain. Therefore, these matrices' triangular areas would simply overlap each other.

Still, one can easily generate the negative matrices as well:

In[ ]:=
```
MatrixForm /@ Flatten[{#, -#} & /@ GenerateMatrices[1], 1]
```
Out[ ]=
$$\left\{\begin{pmatrix}0&-1\\1&1\end{pmatrix},\begin{pmatrix}0&1\\-1&-1\end{pmatrix},\begin{pmatrix}0&1\\-1&0\end{pmatrix},\begin{pmatrix}0&-1\\1&0\end{pmatrix},\begin{pmatrix}0&1\\-1&1\end{pmatrix},\begin{pmatrix}0&-1\\1&-1\end{pmatrix},\right.$$
$$\begin{pmatrix}1&-1\\1&0\end{pmatrix},\begin{pmatrix}-1&1\\-1&0\end{pmatrix},\begin{pmatrix}1&-1\\0&1\end{pmatrix},\begin{pmatrix}-1&1\\0&-1\end{pmatrix},\begin{pmatrix}1&1\\-1&0\end{pmatrix},\begin{pmatrix}-1&-1\\1&0\end{pmatrix},\begin{pmatrix}1&1\\0&1\end{pmatrix},$$
$$\left.\begin{pmatrix}-1&-1\\0&-1\end{pmatrix},\begin{pmatrix}1&0\\-1&1\end{pmatrix},\begin{pmatrix}-1&0\\1&-1\end{pmatrix},\begin{pmatrix}1&0\\1&1\end{pmatrix},\begin{pmatrix}-1&0\\-1&-1\end{pmatrix},\begin{pmatrix}1&0\\0&1\end{pmatrix},\begin{pmatrix}-1&0\\0&-1\end{pmatrix}\right\}$$

STWord, STExponents, STExponentsToMatrix

In[ ]:=
```
STExponents[$S.$S.$TInv]
```
Out[ ]=
```
{{S, 2}, {T, -1}}
```

In[ ]:=
```
STExponentsToMatrix[STExponents[$S.$S.$TInv]] == $S.$S.$TInv
```
Out[ ]=
```
True
```

In[ ]:=
```
STExponentsToMatrix[{{"S", -1}, {"T", 2}, {"S", 1}}]
```
Out[ ]=
```
{{1, 0}, {-2, 1}}
```

In[ ]:=
```
STWord[$Id]
```
Out[ ]=
```
1
```

In[ ]:=
```
STWord[$S.$S.$TInv]
```
Out[ ]=
$$S^2T^{-1}$$

In[ ]:=
```
STWord[{{1, 2}, {3, 4}}]
```

STWord: Bad argument. A 2x2 matrix in the group $SL_2(\mathbb{Z})$ is expected.

Out[ ]=
```
STWord[{{1, 2}, {3, 4}}]
```

In[ ]:=
```
STWord[RandomSL2Z[1000]]
```

Out[ ]=
```
T^3S^-1T^-5S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2
 S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2S^-1T^-2ST^-2
```

Calculations

FunDom

FunDom returns a set of coset representatives together with a list of adjacencies for each matrix. This list should not concern you, it is a crucial input for other functions. The function is quite slow, hence congruence subgroups with large index should be avoided.
However, this function's output can be used for all other functions in this section "Calculations" to avoid multiplied calculations.

In[ ]:=
```
FunDom["SL2Z", {}]
```

Out[ ]=
```
{{{{1, 0}, {0, 1}}, {{1, False}, {1, False}, {1, False}}}}
```

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

Out[ ]=
```
{{{{1, 0}, {0, 1}}, {{2, True}, {2, False}, {3, True}}},
 {{{1, 1}, {0, 1}}, {{1, False}, {1, True}, {5, True}}},
 {{{0, -1}, {1, 0}}, {{4, True}, {4, False}, {1, True}}},
 {{{0, -1}, {1, 1}}, {{3, False}, {3, True}, {6, False}}},
 {{{1, -1}, {1, 0}}, {{6, True}, {6, False}, {2, True}}},
 {{{1, 0}, {1, 1}}, {{5, False}, {5, True}, {4, False}}}}
```

The main idea for the algorithm behind the function **FunDom** was done by Helena A. Verrill [1].

Exchange Matrix for Equivalent One

One can exchange a matrix in $SL_2(\mathbb{Z})$ with an equivalent one form the output of the function **FunDom** in a given congruence subgroup. The lists of adjacencies are updated accordingly.

In[ ]:=
```
FunDom["Gamma0", {3}]
```

Out[ ]=
```
{{{{1, 0}, {0, 1}}, {{1, False}, {1, False}, {2, True}}},
 {{{0, -1}, {1, 0}}, {{3, True}, {4, True}, {1, True}}},
 {{{0, -1}, {1, 1}}, {{4, False}, {2, True}, {4, False}}},
 {{{0, -1}, {1, -1}}, {{2, True}, {3, False}, {3, False}}}}
```

In[ ]:=
```
ExchangeMatrix["Gamma0", {2}, FunDom["Gamma0", {3}], {{1, 42}, {0, 1}}]
```

Out[ ]=
```
{{{{1, 42}, {0, 1}}, {{1, False}, {1, False}, {2, False}}},
 {{{0, -1}, {1, 0}}, {{3, True}, {4, True}, {1, False}}},
 {{{0, -1}, {1, 1}}, {{4, False}, {2, True}, {4, False}}},
 {{{0, -1}, {1, -1}}, {{2, True}, {3, False}, {3, False}}}}
```

Coset Representatives

Calculate all right coset representatives for a given congruence subgroup:

In[ ]:=
```
MatrixForm /@ CosetRepresentatives[FunDom["Gamma", {3}]]
```

Out[ ]=
$$\left\{\begin{pmatrix}1&0\\0&1\end{pmatrix},\begin{pmatrix}-1&0\\0&-1\end{pmatrix},\begin{pmatrix}1&1\\0&1\end{pmatrix},\begin{pmatrix}-1&-1\\0&-1\end{pmatrix},\begin{pmatrix}1&-1\\0&1\end{pmatrix},\begin{pmatrix}-1&1\\0&-1\end{pmatrix},\begin{pmatrix}0&-1\\1&0\end{pmatrix},\begin{pmatrix}0&1\\-1&0\end{pmatrix},\right.$$
$$\begin{pmatrix}0&-1\\1&1\end{pmatrix},\begin{pmatrix}0&1\\-1&-1\end{pmatrix},\begin{pmatrix}0&-1\\1&-1\end{pmatrix},\begin{pmatrix}0&1\\-1&1\end{pmatrix},\begin{pmatrix}1&-1\\1&0\end{pmatrix},\begin{pmatrix}-1&1\\-1&0\end{pmatrix},\begin{pmatrix}-1&-1\\1&0\end{pmatrix},\begin{pmatrix}1&1\\-1&0\end{pmatrix},$$
$$\left.\begin{pmatrix}1&0\\1&1\end{pmatrix},\begin{pmatrix}-1&0\\-1&-1\end{pmatrix},\begin{pmatrix}-1&-2\\1&1\end{pmatrix},\begin{pmatrix}1&2\\-1&-1\end{pmatrix},\begin{pmatrix}1&-2\\1&-1\end{pmatrix},\begin{pmatrix}-1&2\\-1&1\end{pmatrix},\begin{pmatrix}-1&0\\1&-1\end{pmatrix},\begin{pmatrix}1&0\\-1&1\end{pmatrix}\right\}$$

In case the negative identity matrix is not element of the subgroup, for all matrices **M**, also the matrix **-M** is a coset representative. One may neglect these “doubled” matrices:

In[ ]:=
```
SubgroupMemberQ["Gamma", {3}, -$Id]
```

Out[ ]=
```
False
```

In[ ]:=
```
MatrixForm /@ CosetRepresentativesWithoutNegatives[FunDom["Gamma", {3}]]
```

Out[ ]=
$$\left\{\begin{pmatrix}1&0\\0&1\end{pmatrix},\begin{pmatrix}1&1\\0&1\end{pmatrix},\begin{pmatrix}1&-1\\0&1\end{pmatrix},\begin{pmatrix}0&-1\\1&0\end{pmatrix},\begin{pmatrix}0&-1\\1&1\end{pmatrix},\begin{pmatrix}0&-1\\1&-1\end{pmatrix},\right.$$
$$\left.\begin{pmatrix}1&-1\\1&0\end{pmatrix},\begin{pmatrix}-1&-1\\1&0\end{pmatrix},\begin{pmatrix}1&0\\1&1\end{pmatrix},\begin{pmatrix}-1&-2\\1&1\end{pmatrix},\begin{pmatrix}1&-2\\1&-1\end{pmatrix},\begin{pmatrix}-1&0\\1&-1\end{pmatrix}\right\}$$

If the negative identity matrix is element of the group, both functions return the same output:

In[ ]:=
```
SubgroupMemberQ["Gamma0", {42}, -$Id]
```

Out[ ]=
```
True
```

In[ ]:=
```
Length[CosetRepresentatives[FunDom["Gamma0", {42}]]]
```

Out[ ]=
```
96
```

In[ ]:=
```
Length[CosetRepresentativesWithoutNegatives[FunDom["Gamma0", {42}]]]
```

Out[ ]=
```
96
```

Subgroup Index

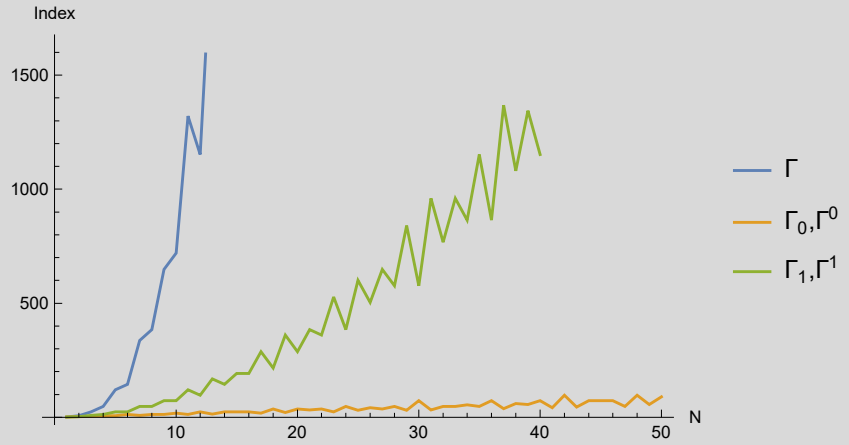
There are formulae for the index of the congruence subgroups Γ, Γ_0, Γ_1, Γ^0, Γ^1 and ${\Gamma_0}^0$.

In[ ]:=
```
SubgroupIndex["Gamma", {1000}]
```

Out[ ]=
```
720 000 000
```

In[ ]:=
```
ListLinePlot[{SubgroupIndex["Gamma", {#}] & /@ Range[15],
  SubgroupIndex["Gamma0", {#}] & /@ Range[50],
  SubgroupIndex["Gamma1", {#}] & /@ Range[40]},
 PlotLegends → {"Γ", "Γ₀,Γ⁰", "Γ₁,Γ¹"}, AxesLabel → {"N", "Index"}]
```

Out[ ]=



For other subgroups, use the function **FunDom**. Watch out however, that the index of the subgroup should is not be too large. Approach a value with smaller numbers first if you cannot approximate the index.

In[ ]:=
```
SubgroupIndex[FunDom["Gamma", {#}]] & /@ Range[5]
```

Out[ ]=
```
{1, 6, 24, 48, 120}
```

Cusps

Calculate the cusps for a congruence subgroup together with their widths.

In[3]:=
```
Cusps[FunDom["Gamma", {3}]]
```

Out[3]=
```
{{-1, 3}, {0, 3}, {1, 3}, {ComplexInfinity, 3}}
```

In[4]:=
```
Cusps[ExchangeMatrix["Gamma", {3}, FunDom["Gamma", {3}], {{51, -76}, {-2, 3}}]]
```

Out[4]=
$$\left\{\left\{-\frac{51}{2}, 3\right\}, \{-1, 3\}, \{1, 3\}, \{\text{ComplexInfinity}, 3\}\right\}$$

In[5]:=
```
Cusps[FunDom["Gamma1", {1}]]
```

Out[5]=
```
{{ComplexInfinity, 1}}
```

Number of Elliptic Points

Get the number of elliptic points of order 2:

In[ ]:=
```
NumberOfEllipticPointsOrder2[FunDom["Gamma1", {#}]] & /@ Range[5]
```

Out[ ]=
```
{1, 1, 0, 0, 0}
```

Get the number of elliptic points of order 3:

In[ ]:=
```
NumberOfEllipticPointsOrder3[FunDom["Gamma1", {#}]] & /@ Range[5]
```

Out[ ]=
```
{1, 0, 1, 0, 0}
```

The ideas for the algorithms behind these functions are adapted from Helena A. Verrill [1].

Genus

The function **Genus** calculates the genus for a congruence subgroup and is based on a formula by Bruno Schoeneberg [2, pages 93-98].

In[ ]:=
```
Genus[FunDom["Gamma0", {11}]]
```

Out[ ]=
```
1
```

In[ ]:=
```
Genus /@ Table[FunDom["GammaUpper0", {n}], {n, 1, 25}]
```

Out[ ]=
```
{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}
```

Drawing

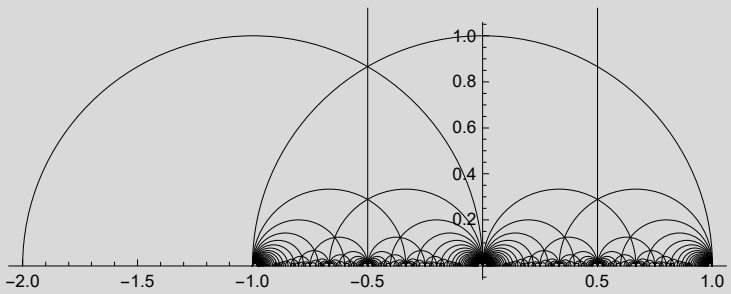
Modular Tessellation

Draw a modular tessellation using **Tessellation[d, r],** where d defines the depth and r specifies the range.

An example for high depth and low range:

In[ ]:=
```
Graphics[Tessellation[100, 0], Axes → True]
```

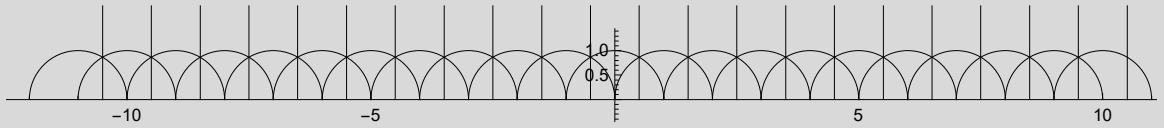
Out[ ]=



An example for low depth and high range:

In[ ]:=
```
Graphics[Tessellation[2, 10], Axes → True]
```

Out[ ]=



The algorithm behind this function was created by Jerzy Kocik [3].

Draw Polygons

One can draw the image of the standard fundamental domain under a matrix in $SL_2(\mathbb{Z})$. The arguments for the function **DrawMatrixImage** are a matrix in $SL_2(\mathbb{Z})$ and a positive integer (default is 10) which specifies the number of vertices of the resulting object of type *Polygon*.

In[ ]:=
```
DrawMatrixImage[$S, 2]
```

Out[ ]=

Polygon [Number of points: 6 Embedding dimension: 2]

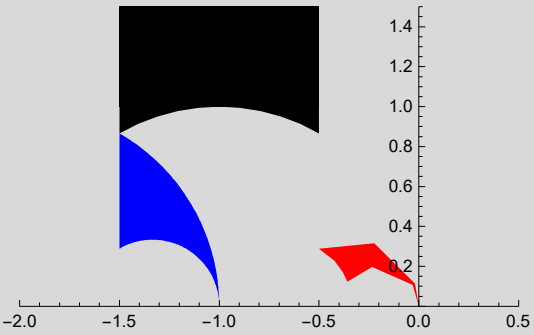
In[ ]:=
```
DrawMatrixImage[$S, 100]
```

Out[ ]=

Polygon [Number of points: 300 Embedding dimension: 2]

In[ ]:=
```
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}}]
```

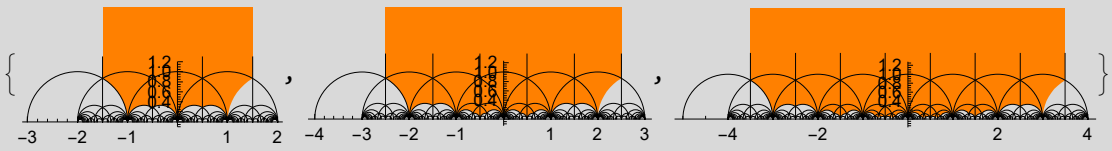
Out[ ]=



Draw the images for all matrices around the standard fundamental domain given a certain radius:

In[ ]:=
```
Table[
 Graphics[{Orange, DrawMatrixImage /@ GenerateMatrices[n],
   Black, Tessellation[20, n]},
  Axes → True],
 {n, {1, 2, 3}}]
```

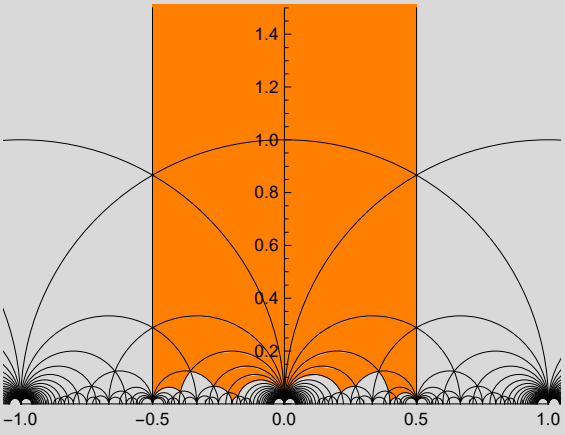
Out[ ]=



Draw the fundamental domain for a congruence subgroup:

In[ ]:=
```
Graphics[{Orange,
   DrawMatrixImage /@ CosetRepresentativesWithoutNegatives[FunDom["Gamma0", {10}]],
   Black, Tessellation[50, 1]},
 Axes → True,
 PlotRange → {{-1, 1}, {0, 1.5}}]
```

Out[ ]=

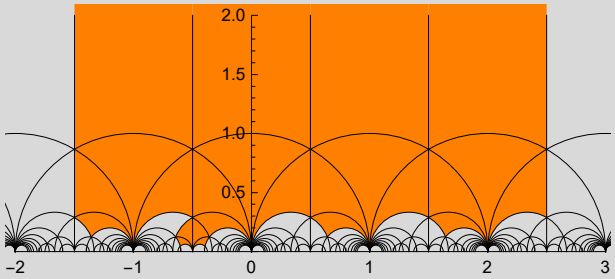


Draw Fundamental Domain

DrawFunDom[FunDom[group,{args}]] returns a set of objects of type Polygon which together represent a fundamental domain for the congruence subgroup group(args).

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

Out[ ]=



How to manually add congruence subgroups

One can manually add congruence subgroups to the package by modifying the package file *Draw-FunDoms.m*.
7 steps on how to do so can be found in the package under the chapter **Private →** section **SL2(Z) and Congruence Subgroups →** subsection **Add Congruence Subgroups**.
Please read the thesis for more details.

References

[1] Helena A. Verrill. *Algorithm for Drawing Fundamental Domains.* Jan. 2001.
[2] Bruno Schoeneberg, *Elliptic Modular Functions. An Introduction.* Trans. by J.R. Smart and E.A. Schwandt from German. 1st ed. Springer-Verlag Berlin Heidelberg, 1974.
[3] Jerzy Kocik. *On the Dedekind tessellation*. Dec. 2019. arXiv: 1912.05768 [math.HO].