FLIPPER TUTORIAL FOR SAGE DAYS 96

SAUL SCHLEIMER

Exercise 0.1. Make sure that flipper is installed into your copy of Sage. There are instructions on the wiki:

`https://wiki.sagemath.org/days96`

Mark Bell (the author of flipper) gives instructions for installing flipper into python, and much more, here:

`http://flipper.readthedocs.io/en/latest/`

Once you have installed flipper, start a Sage session and type `import flipper` at the prompt.

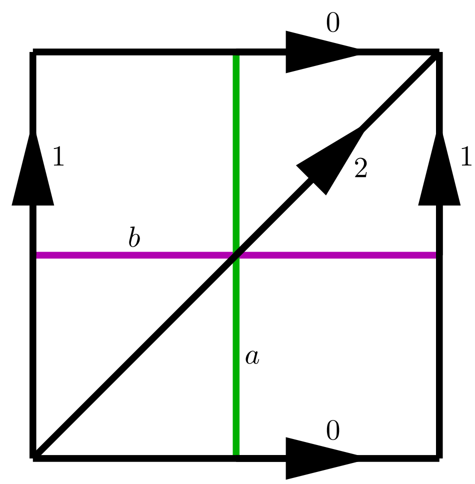

FIGURE 0.2. The standard triangulation of the (once-punctured) torus. The vertical and horizontal lines indicate the core curves for the two (left) Dehn twists built in Exercise 0.4.

Exercise 0.3. Under the hood, flipper works with triangulated surfaces and flip sequences. In this exercise we will build, inside of flipper, the once-punctured torus shown in Figure 0.2. A triangulation in flipper is a list of triangles. A triangle in flipper is a three-tuple of oriented edges, ordered counter-clockwise. So now we can create a once-punctured torus by typing:

```
T = flipper.create_triangulation([(0r, 1r, ~2r), (2r, ~0r, ~1r)])
```

Date: August 15, 2018.

(The letter `r` forces an `int` instead of an `Integer`.) Before we get started, it will be very useful to skim the methods of `T`, by typing `T`, then a period, and hitting tab.

- Check the genus, Euler characteristic, and number of unfilled vertices of `T`. (These methods are constants, not functions.)
- To be absolutely sure you have entered the surface correctly, check that its isomorphism signature is `cPbbde`.
- Check that the triangulation `T` is isomorphic to the triangulation of the "equipped triangulation" from `S = flipper.load('S_1_1')`.

Exercise 0.4. A measured lamination on `T` is represented in flipper as the list of its geometric intersection numbers with the edges of `T`. For example, typing `a_lam = T.lamination([1, 0, 1])` builds the curve a.

- Build `b_lam`, the lamination representing the curve b.
- Flipper gives `a_lam` a kind of orientation; it records the algebraic intersections of `a_lam` with the (oriented!) edges of `T`. Type `a_lam.algebraic` to find these.
- Typing `a = a_lam.encode_twist()` builds the left Dehn twist about a. Do the same thing for b. We can compose `a` and `b` by concatenating and taking powers. Define `rho = a*b`. Use flipper to check that `rho` is periodic of order six.
- Mapping classes act on laminations. Define `hyp = rho^3`. Use "==" to check that `hyp(a_lam)` is *not* the same as `(hyp^2)(a_lam)`; the latter is the same as `a_lam`.
- What happens if you type `hyp^2(a_lam)`? Why?
- For background on the "Alexander method" see [Section 2.3, Farb-Margalit]. Use this and flipper to check that `hyp` is the hyperelliptic element: that is, it acts on the torus as a 180 degree rotation.
- Set `B = b.inverse()` and `fib = a*B`. Check that `fib` is pseudo-Anosov and find its dilatation.

Exercise 0.5. We now record of our work in an *equipped triangulation*:

```
lams = {'a': a_lam, 'b': b_lam};
maps = {'a': a, 'b': b, 'rho': rho, 'hyp': hyp, 'fib': fib};
TE = flipper.kernel.EquippedTriangulation(T, lams, maps)
```

Type `TE` at the prompt to see what it contains. We can now use strings to describe mapping classes. Check that `f = TE.mapping_class('(ab)^6')` gives the identity.

Exercise 0.7. We now turn to a more complicated example.

- Build the L-shaped surface shown in Figure 0.6, and call it `L`. Remember that you need to use raw numbers, of the form `0r`, `1r`, and so on. Check your work; this triangulation has isomorphism signature `gvLQffeeaead`.
- Build the laminations a, b, c, d shown in Figure 0.6, their associated left Dehn twists, and their inverses. Check that all pairs of twists with disjoint

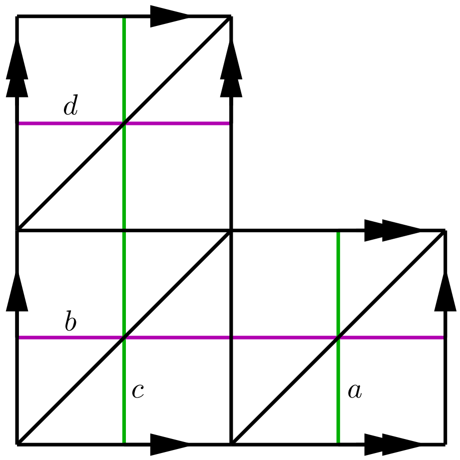

FIGURE 0.6. An L-shaped surface, with boundary edges identified by translation as indicated. The four curves of interest are labelled a, b, c, d.

support commute. Check that `rho = a*c*b*d` is periodic of order ten. Check that `hyp = rho^5` is the hyperelliptic element; you will need to compute the number of fixed points by hand.

- The Thurston-Veech construction guarantees that `h_pent = a*c*B*D` is pseudo-Anosov. Compute its dilatation.
- Store the triangulation `L`, the laminations, and all of these mapping classes in an equipped triangulation `LE`.

Exercise 0.8. We now start to extract flat structures from flipper. Type

```
fib = TE.mapping_class('fib')
```

to recover the Fibonacci mapping class from Exercise 0.4. The `FlatStructure`

```
fibflat = fib.flat_structure()
```

contains a new triangulation `T_fib = fibflat.triangulation`. It also contains a dictionary `edge_vectors`; the keys are edges of `T_fib` and the values are *holonomies*. Edges can be a bit tricky to get at, so here are two different ways:

```
e_0 = T_fib.edge_lookup[0]; # the zeroth edge
e_2 = T_fib.triangles[1].edges[1] # the second edge
```

So `fibflat.edge_vectors[e_0]` is the holonomy of `e_0`. The x and y coordinates have type:

```
flipper.kernel.numberfield.NumberFieldElement
```

To convert these to something Sage understands, you can use the code found here:

`http://wiki.sagemath.org/days96?action=AttachFile&do=get&target=flipper_nf_conversion.py`

- Add the conversion function to your Sage session and extract the complex holonomies of all three edges.
- Flipper's flat structures can deal with half-turn surfaces; thus the holonomies of the edges necessarily live in $\mathbb{C}/\{\pm 1\}$. Note that edges do have a well defined *slope*: positive, vertical, negative, or horizontal. Write a function in Sage that, given a flat structure and an edge, computes the slope.
- Write a function in Sage that, given a flat structure and a triangle, draws the corresponding euclidean triangle. The edges should be coloured red, green, blue, or purple as their slopes are positive, vertical, negative, or horizontal.
- In Sage, draw the layout of `fibflat`.

Problem 0.9. A flipper flat structure is *abelian* if and only if the holonomies can be consistently lifted to $\mathbb{C}$. Write code in Sage that detects this and, when abelian, computes a lift.

Exercise 0.10. We now consider the more difficult example `LE` from Exercise 0.7 Before we get started, recall that a rectangle R with width w and height h has *modulus* $\mathrm{Mod}(R) = w/h$. If we glue the top of R to the bottom, we obtain an *flat annulus* A. We again define $\mathrm{Mod}(A) = w/h$. The width of A is the distance between its boundaries while the height is the length of its core curve.

- In Sage, draw the layout of the flat structure

  ```
  pentflat = h_pent.flat_structure()
  ```

- The flat structure decomposes as a union of two annuli A and C with core curves a and c. Compute the moduli of A and C. Deduce that the multitwist `a*c` acts as a shear on the flat structure.
- Show that this flat structure is, up to an element of $\mathrm{SL}(2, \mathbb{R})$, the doubled pentagon.
- Show that `a*c` and `b*d` generate the Veech group.