`flipper` 0.12.0 Mark Bell

`flipper` is a program for computing the action of mapping classes on laminations on a punctured surface using ideal triangulation coordinates. These are similar to, but subtly different from, the foliations originally used by Thurston. Among other things, `flipper` can decide the Nielsen–Thurston type of a mapping class and construct triangulations of mapping tori.

Note: This guide is written for UNIX / OS X users and so refers to running commands in a terminal. Windows users will need to use the command prompt instead.

0 Getting set up

`flipper` is a Python package, so you should start by making sure that you have a modern version of Python installed on your system (Python 3.4+ or Python 2.7.9+). It can be run using Python 2 but prefers to be run using Python 3.

Exercise 0.1 Install `flipper` by opening a terminal and running the command:

```
> python -m pip install flipper --user --upgrade
```

Warning: The packages used by `flipper` require an updated version of the `six` package. Since this is included as an 'Extra' package in the included system Python on OS X, Mac users may need to:

- install Python manually,
- modify their `PYTHONPATH` environment variable, or
- install `flipper` within `virtualenv`

as described here: `http://stackoverflow.com/questions/29485741/`

You can check that `flipper` was installed correctly by running the command: `python -m flipper.test`

Exercise 0.2 Start the `flipper` GUI by opening a terminal and running the command:

```
> python -m flipper.app
```

Warning: In order to use the GUI on OS X, users must first update their copy of Tk/Tcl as described here: `https://www.python.org/download/mac/tcltk/`. `flipper` has been tested with ActiveTcl 8.5.18 available from `http://www.activestate.com/activetcl/downloads`.

1 The once-punctured torus

Now that we can start `flipper`, we can use its GUI to manipulate some curves on surfaces. We will start with a once-punctured torus (denoted $S_{1,1}$).

Exercise 1.1 Open the $S_{1,1}$ example from the menu `File > Open example...` and choose `S_{1,1}` from the drop-down menu. Click on the different *laminations* in the *object list* on the left hand side to find a vertical curve on this torus.

This example also comes with two mapping classes. These are the left[1] *Dehn twists* about the curves a and b, denoted T_a and T_b, and are listed as `a` and `b` under the `Mapping Classes:` section of the object list. (To avoid confusion we will use maths characters, like a, to refer to curves and typewriter characters, like `a`, to refer to mapping classes.) Expand the mapping class `b` by clicking the `+` to the left of its name.

[1] Internally these are actually right Dehn twists but the GUI reverses things vertically.

Exercise 1.2 Apply several Dehn twists about b to the vertical curve by repeatedly clicking on `Apply`. What Dehn twist does `Apply inverse` perform?

Reload your vertical curve and perform a right Dehn twist about b to obtain a horizontal curve. We can add this new curve to the object list. Use the menu `Create > Lamination` and name this lamination c.

Exercise 1.3 The *unsigned-slope* of a curve is

$$\#\{\text{times it crosses the vertical yellow side}\} \Big/ \#\{\text{times it crosses the horizontal blue side}\}.$$

So the unsigned-slopes of a and b are ∞ and 1 respectively.

(a) Find a sequence of Dehn twists about a and b to map c to a curve with unsigned-slope: $\frac{1}{2}$, $\frac{7}{3}$ and $\frac{13}{8}$.

(b) Prove that for every rational $p/q \geq 0$ you can use these Dehn twists to reach a curve whose unsigned-slope is p/q.

`flipper` can also create a new Dehn twists about the currently drawn curve. Reload the horizontal curve c and use the menu `Create > Mapping class > Dehn twist` to create a new Dehn twist about this named `c`.

Exercise 1.4 A Dehn twist about a curve fixes that curve. Check that $T_x(x) = x$ for several curves.

We can also build new mapping classes by composing together old ones. The composition $T_a T_b T_a^{-1}$ is specified by the string `abA`, since upper case names refer to the inverse of the mapping class. Use the menu `Create > Mapping class > Composition` and this string to create this new mapping class. You can give this mapping class a different name but for now use the default (its composition string). The default name is often very useful since it tells you how the mapping class was constructed.

Exercise 1.5 The mapping class `abA` is in fact also a Dehn twist. Find which curve it is twisting about by finding a curve that is invariant under it.

Expand the properties of a mapping class in the object list. Dehn twists are all infinite order mapping classes and so all of the mapping classes we have built so far have `Order: Infinite`.

Exercise 1.6 Compute the orders of all of the length two mapping classes that can be made using `a`, `b` and their inverses: `aa`, `ab`, `aA`, `aB`, `ba`, ...

Since `flipper` can compute the order of a mapping class, we can use it to solve the *word problem* for $\mathrm{Mod}^+(S)$. A mapping class is the identity if and only if its order is one.

Exercise 1.7 Verify:

(a) *the braid relation*: that $T_a T_b T_a = T_b T_a T_b$.

(b) *change of coordinates*: that $T_{f(x)} = f T_x f^{-1}$ for several mapping classes f.

Create a new mapping class called `f` defined as $T_a T_c^{-1}$. Apply `f` several times to the horizontal curve c. This image winds around the surface many times and so it can be very tricky to keep track of exactly where it goes. Use the menu `Settings > Edge label > Geometric` to have each side display how many times the curve passes through it (its *geometric intersection number*).

Exercise 1.8 Investigate how the number of intersection with the vertical blue side changes as you apply different powers of `f` to c.

To deal with the rapid growth of these numbers, also turn on the `Settings > Edge label > Projectivise` option to rescale them so their sum is one. Projectively, these intersection numbers converge and `flipper` can find the lamination that they are converging to. Use the `Invariant lamination...` property of `f` to see this lamination (you will need to double-click to activate this calculation). Add it to the list of laminations in the object list under the name l. Even though l looks very similar to $\mathtt{f}^8(c)$ these are not the same since $\mathtt{f}^8(l) = \mathtt{f}^7(l) = \cdots = l$ while $\mathtt{f}^8(c) \neq c$.

Note that, although l is projectively invariant under `f`, it is not invariant. Turn off the `Projective` option and see how l is being rescaled under `f`. This rescaling factor is known as the *dilatation* of `f` and is written $\lambda^+(\mathtt{f})$. Use the `Dilatation: ?` property of `f` to compute it (again you will need to double-click to activate this calculation).

In this case, the mapping class `f` is *pseudo-Anosov* and l is its *stable lamination*.

Exercise 1.9 Compute the *unstable lamination* of `f`, that is, the stable lamination of `F`. Add it to the list of laminations with the name L. Check that L is also projectively invariant under `f`.

Pseudo-Anosov is just one of the three possible types that a mapping class can have under the Nielsen–Thurston classification and in fact we have actually now seen examples of all three types:

Theorem 1.10 (Nielsen–Thurston classification)**.** *Each mapping class is either:*

- periodic*, that is, finite order;*
- reducible*, that is, fixes a (multi-)curve; or*
- pseudo-Anosov*, that is, projectively fixes a pair of filling measured laminations.*

Moreover, a mapping class is pseudo-Anosov if and only if it is neither periodic nor reducible.

Exercise 1.11 Prove that Nielsen–Thurston type is a *conjugacy invariant*, that is, that if `g` and `h` are conjugate then they have the same Nielsen–Thurston type.

`flipper` can determine the Nielsen–Thurston type of a mapping class. Use the `Type: ?` property of `f` to verify that this mapping class is pseudo-Anosov (again you will need to double-click to activate this calculation).

Exercise 1.12 Compute the Nielsen–Thurston type of all of the mapping classes we have constructed so far. Hint: Use Exercise 1.11 to reduce the number of mapping classes you have to compute the type of.

The composition builder also supports basic powers and you can also use "`.`" separator to make it clearer what mapping classes are being composed. Create a new mapping class called `hyp` using the composition string `(a.b)^3`

Exercise 1.13 Prove that for any pseudo-Anosov mapping class `h`:

(a) `hyp.h` is pseudo-Anosov.

(b) $\lambda^+(\mathtt{hyp.h}) = \lambda^+(\mathtt{h})$.

Although `hyp` fixes all curves on the once-punctured torus, is not the identity since it reverses their orientation. To see, this label each side with its *algebraic intersection number* with the curve by using the menu `Settings > Edge label > Algebraic`. Notice how the algebraic intersection numbers of c changes as `hyp` is applied. Hence `hyp` is a periodic mapping class with order 2.

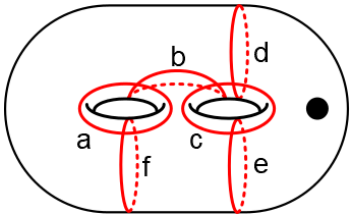
Exercise 1.14 What are the different possibilities for the orders of periodic mapping classes?

You can also see the edge orientations by using the menu `Settings > Show edge orientation`. Reset the edges to be unlabelled and unoriented before continuing.

2 A higher genus example

Load the example surface `S_{2,1}`.

Exercise 2.1 Check that the curves on this surface are arranged like so:



There are even more relations in the mapping class group of this surface.

Exercise 2.2 Verify:

(a) *far commutativity*: that for any pair of disjoint curves their Dehn twists commute.

(b) *the chain relation*: that $(T_f T_a T_b)^4 = T_e T_d$.

Exercise 2.3 Find a projectively invariant lamination for `DCaBcd`. Why is this mapping class *not* pseudo-Anosov?

Exercise 2.4 (a) Construct a pseudo-Anosov mapping class on $S_{2,1}$.

(b) Prove that if you make a mapping class from a composition string of length three or less then it is not pseudo-Anosov.

(c) Use a chain relation to construct a periodic mapping class.

`flipper` also includes a solution to the *conjugacy problem* for pseudo-Anosovs. Build a new mapping class `aBcfcDDEF` called `f` and use the `Conjugate to...` property of `f` (again you will need to double-click to activate this calculation) to check that this is conjugate to itself. Check that `f` is also conjugate to `abDABEccD` but not conjugate to `accABcDDE`.

Exercise 2.5 Prove that, for pseudo-Anosovs, dilatation is a conjugacy invariant. Is it a *total* conjugacy invariant? That is, does $\lambda^+(\mathtt{g}) = \lambda^+(\mathtt{h})$ imply that `g` and `h` are conjugate?

Exercise 2.6 Prove that all Dehn twists on $S_{1,1}$ are conjugate. Are all Dehn twists on $S_{2,1}$ conjugate?

3 Symmetries

Open a `Circular n-gon` example and enter `abABcdCDefEF` as the boundary pattern. The capitalisation of the boundary pattern is very important.

Exercise 3.1 Check that you get the same surface if you build a `Circular n-gon` by using the boundary pattern `abcdefABCDEF`. That is, find a scissors-congruence from one identification diagram to the other.

This surface comes with some symmetries. You can ask `flipper` to create the mapping class associated to these by using the menu `Create > Mapping class > Isometry`. Choose the one that begins "`2, 3, 4 ...`" and name it `p`.

To see how this new mapping class acts we will draw a lamination on this surface. Click outside of the polygon to start drawing a lamination, click to place segments and then click outside again to stop drawing. If you make a mistake you can use `Esc` or `Backspace` to remove the last segment you placed. You can start again by using the menu `Edit > Erase lamination`.

Draw a lamination that enters through the pink side at 12 o'clock and exits through the pink side at 2 o'clock and let `a` be the Dehn twist about this curve. You can use the menu `Edit > Tighten lamination` to ask `flipper` to redraw this using its internal algorithm.

Use this lamination to discover how the symmetry `p` acts on the surface.

Exercise 3.2 Determine the Nielsen–Thurston type of all of the mapping classes that can be made using compositions of `p`, `P`, `a` and `A`.

Make a second curve b that connect from the 1 o'clock & 8 o'clock sides and the 10 o'clock and 11 o'clock sides (the turquoise and orange sides). Create `b`, the Dehn twist about b.

Exercise 3.3 Verify that `aBp` and `bAP` are both pseudo-Anosov and have the same dilatation. What other pairs of three letter mapping classes ending with `p` or `P` have the same dilatation? How does changing the case of the `p` on the end affect the dilatation?

4 Within Python

Note: For these exercises you should have some basic experience with Python. Things like `if`, `else`, `for ... in ...`, `list(...)`, `print(...)`, `assert(...)` and so on. If you don't then have a look at Sections 3, 4 and 5 of `https://docs.python.org/3/tutorial/`

We can use a lot more features of `flipper` by running it interactively. Start the Python interpreter by opening a terminal and running the command:

```
> python
```

Within Python we can import `flipper` and load one of our standard examples:

```
>>> import flipper
>>> S = flipper.load('S_2_1')
```

As before, we can construct mapping classes by composing together generators.

```
>>> h = S.mapping_class('abCCdEa')
>>> g = S.mapping_class('aC')
>>> assert(g * h != h * g)
>>> assert(h * h * h == h**3) # Python uses ** for powers.
```

We can compute all of the results available in the GUI by using the corresponding method, for example:

```
>>> h.dilatation()
4.611581?
```

Other properties you have seen so far can be computed by using: `h.nielsen_thurston_type()`, `h.dilatation()`, `h.invariant_lamination()` and `h.is_conjugate_to(g)`

Note: In Python we can get more information about an object by using `help(<object>)`.

Exercise 4.1 Use `help(h)` and `help(h.<method>)` to learn more about each of the methods of a mapping class. (Skip all the boring ones that start with "`__`".)

Many of the values we can compute also have additional methods:

```
>>> d = h.dilatation()
>>> d.minimal_polynomial() # This is an algebraic number.
1 - 4*x - 2*x^2 - 4*x^3 + x^4
```

Some functions can be used in different ways. For example, `help(S.mapping_class)` also describes how, if given an integer, this method builds a random mapping class of that length. This can be very useful for running experiments on typical mapping classes.

Exercise 4.2 Make a list containing 100 random mapping classes of length 20. What percentage of these are pseudo-Anosov? Investigate how the percentage depends on the length.

However sometimes you want to look for an atypical mapping class with an unusual property. Fortunately, just like SnapPy, `flipper` can systematically build mapping classes on S:

```
>>> MC = list(S.all_mapping_classes(length=3))
>>> len(MC)
208
```

Be careful as this can result in a lot of mapping classes. To generate longer length words you should probably set some addition arguments to try to filter out typical mapping classes.

Exercise 4.3 Use a *list comprehension* and `h.is_pseudo_anosov()` to create a new list called `PA` containing all the pseudo-Anosov mapping classes in `MC`. Of these, find the one(s) with smallest dilatation.

Exercise 4.4 Divide the elements of `PA` into conjugacy classes. How does the number of conjugacy classes grow with `length`?

5 Extensions and projects

Exercise 5.1 Find a relationship between $\lambda^+(\mathtt{h})$ and $\lambda^+(\mathtt{h}^n)$.

Exercise 5.2 Use `flipper` to verify the *lantern relation* by choosing suitable Dehn twists on $S_{2,1}$.

Exercise 5.3 Any pseudo-Anosov on $S_{2,1}$ has dilatation at least $1.72208\ldots$. Find a pseudo-Anosov with this dilatation.

Exercise 5.4 Investigate when a mapping classes is conjugate to its inverse.

Exercise 5.5 The *degree* of a mapping class `h` is $\deg(\mu_{\lambda_\mathtt{h}})$: the degree of the minimal polynomial of the dilatation of `h`. For different $S_{g,1}$ and $1 \leq k \leq 3g-3$, find a mapping class $\mathtt{h} \in \mathrm{Mod}^+(S_{g,1})$ such that $\deg(\mathtt{h}) = 2k$.

Exercise 5.6 Use the `Bundle...` property of a mapping class to build a SnapPy triangulation of its *mapping torus*. You can also use `snappy.Manifold(h.bundle())` within Python. Use `flipper` and SnapPy together to investigate the relationship between dilatation and hyperbolic volume.

Exercise 5.7 Write down all of the typos in this exercise sheet and email them to:
`mcbell@illinois.edu`