A users guide to Flipper

Mark Bell

February 10, 2014

Flipper is a program for computing the action of mapping classes on curves and laminations on a punctured surface using ideal triangulation coordinates. Flipper is currently under development and this users guide will be based on Flipper 0.1.0. Eventually it will also be able to construct the canonical triangulation of the surface bundle associated to pseudo-Anosov mapping classes.

1 Getting and starting Flipper

Remark 1.1. To see just the commands needed to setup and start Flipper, see Appendix A.

Flipper has been tested on Python 2.7 and Python 3.2 on Windows 7 and Ubuntu 12.04. Some of its features require exact arithmetic. In Python this is done using SympPy (tested with SympPy 0.7.3.rc1) but you can also create an interface to your own exact arithmetic library. Alternatively, you can install Flipper as a Sage module (tested with Sage 5.12) and use Sages exact arithmetic libraries. These appear to be significantly faster.

1.1 Getting Flipper

You can get the latest copy of flipper from `https://bitbucket.org/Mark_Bell/Flipper` or straight from the mercurial repository with the command:

```
> hg clone https://bitbucket.org/Mark_Bell/Flipper
```

1.2 Installing and starting Flipper under Python

To install Flipper, in a terminal run the following command inside of its folder:

```
> python setup.py install
```

If that fails because you do not have the necessary permission, you can install it locally instead using the command:

```
> python setup.py install --user
```

If you want, you can test your install by using the command:

```
> python setup.py test
```

This will list out varous tests that it is running and if they passed.

Finally, you can now start Flipper by running the command:

```
> python -m Flipper.app
```

1.3 Installing and starting Flipper under Python

To install Flipper as a Sage, in a terminal run the following command inside of its folder:

```
> sage -python setup.py install
```

If that fails because you do not have the necessary permission, you can install it locally instead using the command:

```
> sage -python setup.py install --user
```

Again, if you want, you can test your install by using the command:

```
> sage -python setup.py test
```

You can then start Flipper from within Sage by using the command

```
> sage -python -m Flipper.app
```

Note that Sage if does not recognise your Tkinter install you may initally see an error such as `Error: no module named _tkinter`. You can fix this by installing the tcl/tk development library and then rebuilding Sage's Python. On Ubuntu you can do this using the commands:

```
> sudo apt-get install tk8.5-dev
> sage -f python
```

1.4 Creating executables

Flipper also includes a freeze file. So if you have cx_Freeze installed as a Python module you can use the command:

```
> python freeze.py build
```

to create an executable within the build directory. As with most cx_Freeze distutils scripts, you can also build a Windows installer by doing:

```
> python freeze.py bdist_msi
```

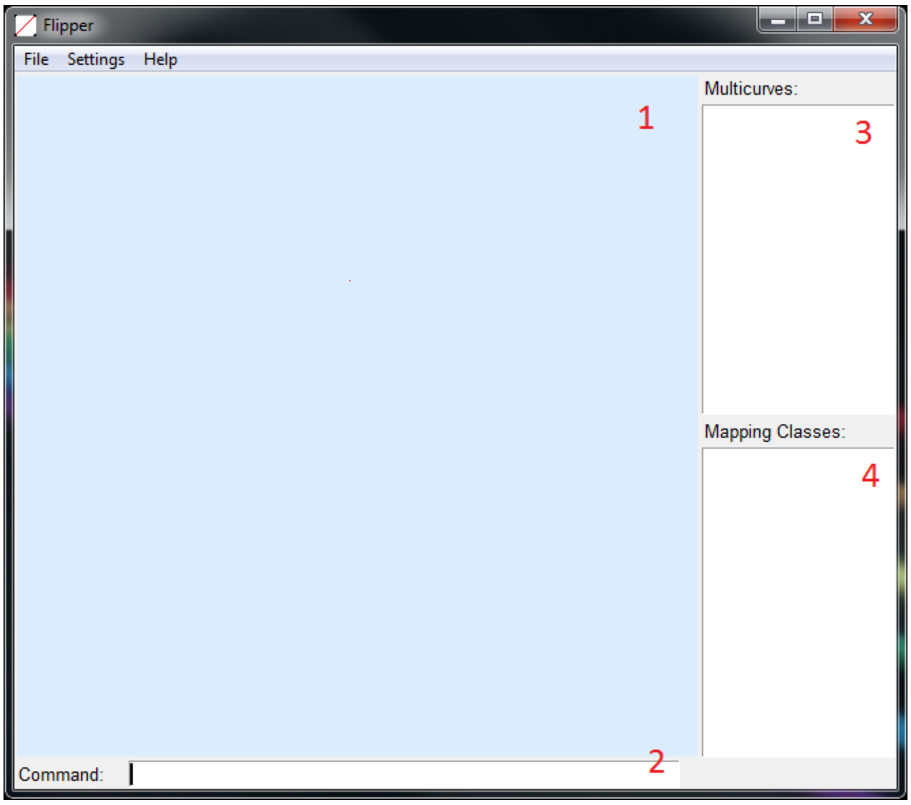
and a Mac disk image by doing:

```
> python freeze.py bdist_dmg
```

At some point the BitBucket repository may also include precompiled binaries.

2 Getting Started

This is the main window of Flipper. It has a canvas for drawing on (1), a command bar for entering instructions (2), a list of known curves (3) and a list of known mapping classes (4).



The main work flow for using Flipper is:

1. Create a triangulation.
2. Create curves on the surface.
3. Specify mapping classes on the surface.
4. Apply some of the known mapping classes to a curve.

We now describe how to perform each of these steps in detail. First note that in each stage the currently selected object is highlighted in red. You can cancel your current selection at any time by clicking on the object again, pressing `Escape`, right clicking, or double clicking. Additionally you can delete the currently selected object by pressing either `Delete` or `Backspace`.

2.1 Creating a triangulation

To create a triangulation, click on the canvas to create vertices. Click on two in succession to connect them via an edge. You cannot add an edge if it would meet the interior of an existing edge. Click on two edges, each of which are part of exactly one triangle, in succession to identify them. Clicking on an identified edge will destroy the identification.

Flipper automatically adds triangles between any triple of vertices each of which is pairwise connected via an edge.

The triangulation is *complete* if each edge is either contained in two triangles or is contained in one triangle and is identified with another. Once the triangulation is complete Flipper will switch to interpret clicks as drawing curves. You can force Flipper to place a vertex or select an edge, even if the triangulation is complete, by holding `Shift` while clicking.

2.2 Creating curves

Once the surface drawn is complete you can start drawing curves on it. Click on the canvas to start drawing a section of curve. Once started, click on the canvas again to extend the current section of curve through the current point. To finish drawing a section of curve press `Escape`, right click or double click.

You can remove the last point of the section of curve currently being drawn by either pressing `Delete` or `Backspace`. You should make sure to draw the curve transverse to the underlying triangulation.

The currently drawn multicurve can be named and added to the list of known curves by using the `curve <name>` command. Known multicurves can be shown by using the `show <name>` command.

2.3 Specifying mapping classes

There are currently three ways to specify a mapping class. The first is to define the mapping class to be a left Dehn twist about the currently drawn curve, see Section 2.2. This is done by using the `twist` command. This takes a `<name>` and adds this twist to the list of known mapping classes under the name `<name>`. Note that this also adds the curve to the list of known curves under the same name.

The second is, when the currently drawn curve bounds a twice marked disk, to define the mapping class to be a left half twist about the currently drawn curve, see Section 2.2. This is done by using the `half` command. This takes a `<name>` and adds this half twist to the list of known mapping classes under the name `<name>`. Note that this also adds the curve to the list of known curves under the same name.

Alternatively, a mapping class can be defined by specifying how it permutes the underlying triangles of the triangulation. This is done by using the `isometry` command. This takes a `<name>`, `<source triangle>` and `<target triangle>` and adds the mapping class induced by the the isometry which sends `<source triangle>` to `<target triangle>` to the list of known mapping classes under the name `<name>`.

In either case, the inverse of the mapping class is also added to the list of known mapping classes under the swapcase of `<name>`.

Remark. Currently Flipper is only capable of performing Dehn twists and half twists about *good curves*, where every complementary region contains at least one puncture. Hence, for example, it cannot perform a half twist on a twice marked surface.

2.4 Applying a mapping classes

To apply one of the known mapping classes to the currently drawn curve click on its name in the list of known mapping classes. The result will be instantly displayed on the triangulation. To apply the inverse of a known mapping class, right click on its name in the list of known mapping classes. Alternatively use the command `apply <mapping class>` where `<mapping class>` is a string of mapping class names and inverse names each separated by a '.'.

A Quick start

These are the quickest ways to get, install, test and start Flipper under various systems.

A.1 Under Python

```
> hg clone https://bitbucket.org/Mark_Bell/Flipper
> cd ./Flipper
> python setup.py install --user
> python setup.py test
> python -m Flipper.app
```

A.2 Under Sage

```
> hg clone https://bitbucket.org/Mark_Bell/Flipper
> cd ./Flipper
> sage -python setup.py install
> sage -python setup.py test
> sage -python -m Flipper.app
```

B All commands

The main Flipper commands are:

- `curve <name>` - Stores the currently drawn multicurve as `<name>`.
- `show <name>` - Shows the multicuve with name `<name>`.
- `twist <name>` - Stores a left Dehn twist about the currently drawn curve as `<name>`.
- `half <name>` - Stores a left half twist about the currently drawn curve as `<name>`. This curve must bound a twice marked disk.
- `isometry <name> <source triangle> <target triangle>` - Stores the mapping class induced by the isometry of the underlying triangulaton as `<name>`.
- `apply <mapping class>` - Applies `<mapping class>` to the currently drawn multicurve.
- `order <mapping class>` - Computes the order of `<mapping class>`.
- `periodic <mapping class>` - Determines if `<mapping class>` is periodic.
- `reducible <mapping class>` - Determines if `<mapping class>` is reducible.
- `pA <mapping class>` - Determines if `<mapping class>` is pseudo-Anosov.
- `lamination <mapping class>` - Computes a lamination which is projectively invariant under `<mapping class>` along with its dilatation. This functionality requires a symbolic computation library.
- `lamination_estimate <mapping class>` - Computes a lamination which is approximately projectively invariant under `<mapping class>` along with its dilatation.
- `split <mapping class>` - Computes the maximal splitting sequence of `<mapping class>`. This functionality requires a symbolic computation library.
- `bundle <mapping class>` - Build the canonical bundle associated to the maximal splitting sequence of `<mapping class>`. This functionality requires a symbolic computation library.

Flipper also has commands for quickly creating standard triangulations:

- `ngon <specification>` - Creates a regular ngon with either the number of sides or side gluings given by `<specification>`.
- `rngon <specification>` - Creates a regular ngon with either the number of sides or side gluings given by `<specification>` where the underlying triangulation has rotational symmetry.

The following commands are also available:

- `information` - Reports information about the underlying surface such as genus and Euler characteristic.
- `tighten` - Tightens the currently draw curve to the underlying triangulation.
- `vectorise` - Shows the edge vector of the currently drawn curve.
- `render <curve vector>` - Draws the curve with edge vector `<curve vector>`.
- `erase` - Erases the current curve.
- `zoom` - Scales the current drawing to fill the canvas.

Finally, there are Flipper commands which provide perform the actions available in the various menus:

- `new` - Resets Flipper. Equivalent to `File > New`.
- `save <filename>` - Saves the current configuration to `<filename>`. Equivalent to `File > Save`.
- `open <filename>` - Loads the configuration stored in `<filename>`. Equivalent to `File > Open`.
- `export_script <filename>` - Exports the current surface and mapping classes as a Python script to `<filename>`. Equivalent to `File > Export > Export script`.
- `export_image <filename>` - Exports the current canvas as a postscript file to `<filename>`. Equivalent to `File > Export > Export image`.
- `help` - Shows the help information. Equivalent to `Help > Help`.
- `about` - Shows the about information. Equivalent to `Help > About`.
- `exit` - Exits Flipper. Equivalent to `File > Exit`.

Arguments given to Flipper commands must meet the Conventions:

- `<filename>` must be a valid file name or path.
- `<name>` must be an string of letters, numbers and underscores. It must contain at least one letter.
- `<mapping class>` must be a string of mapping class names and inverse names, given by their swapcase, each separated by a '.'. This is read left to right to agree with the order of composition.
- `<source triangle>` and `<target triangle>` must be a string of three edge indices, each separated by a '.'.
- `<curve vector>` must be a list of edge weights each separated by a '.'.