INSTRUCTIONS AND DESCRIPTION FOR EQMAKER

Contents

1. Software Installation

This installs the software that EQMaker relies on. There is no need to install EQMaker.py itself.

(1) Install Python 3 from https://python.org
(2) In a Python command line, enter

```
pip install numpy
```

Once that's finished, enter

```
pip install opencv-python
```

2. What EQMaker.py does, in English

EQMaker produces a bitmap AllLines.png of the diagram, sufficiently zoomed so that there is at least one pixel between each pair of parallel line segments in the diagram. It then detects and numbers the regions in the diagram. Region 0 is always the outermost unbounded region. It lists each vanishing cycle and handle foot. A vanishing cycle can have multiple components in the diagram if it runs over handles, and the union of these components is called a CompleteLine. Each handle foot is also a CompleteLine. For the foot of a handle, which is always an empty rectangle with empty description attribute, EQMaker gives that foot the description cx, where x is an integer starting at 100, unique to that foot. Now that all CompleteLines have descriptions, EQMaker creates a dictionary between these descriptions and an internal list of names, one name for each CompleteLine.

EQMaker records each crossing as a variable $c[x, y, v]$ or $c[x, y, h]$, where x is the x-coordinate of the crossing in AllLines.png, measured in pixels from the top-left corner, and y is the y-coordinate. The label v is used for a crossing if the curve with the greater integer in its description is the vertical line segment, and h is used if the greater integer comes from the horizontal segment.

The boundary of a region is a union of segments of CompleteLines. In the boundary of each region, the *first pixel* is one pixel below the leftmost of the top pixels. Starting at the first pixel in the boundary, the selected pixel proceeds counter-clockwise along the boundary of the region, keeping track of all CompleteLines that contain the selected pixel. When the number of CompleteLines containing the selected pixel increases to 2, a crossing is detected along with the identifying data of its CompleteLines. This data is recorded using internal names for the CompleteLines. Using this data, along with the description names for the lines, EQMaker writes the grading equation for each region, one equation for each labeling of the vanishing cycles. The labeling is also used to give the h or v in the crossing variables. Note that since the boundary of region 0 is supposed to be oriented clockwise, EQMaker negates the right hand side of any equation coming from region 0: Such equations will have (-1)* prepended to their right-hand sides. The iteration that generates the re-labeling of vanishing cycles comes from the following rule: Assuming the vanishing cycles had description

integers $k, k+1, \ldots, k+n$, EQMaker changes the description k to be $k+n+1$ and, essentially, repeats the process of this paragraph n times. In reality, EQMaker is much more efficient: It lists an equation for each region using its internal names, then rewrites that equation appropriately for each relabeling.

The rectangles in the diagram are supposed to come in pairs, and EQMaker assumes the user has used description attribute to pair them as follows:

$h1a$, $h1b$
$h2a$, $h2b$
...etc.

So that rectangle $h1a$ is paired with rectangle $h1b$, etc.

EQMaker records the ends of the lines at each rectangle as follows. For each pair of rectangles, say the n^{th} pair hna and hnb, EQMaker starts at the top-left corner of rectangle hna and lists the crossings going around hna in counter-clockwise order. These crossings are where lines end at rectangle hna. EQMaker lists the crossings this way:

$$c[n, a, 0], c[n, a, 2], \ldots, c[n, a, k].$$

EQMaker starts at the top-right corner of hnb (NOT the top-left) and lists the crossings going around in clockwise order (NOT counter-clockwise). EQMaker lists the crossings this way:

$$c[n, b, 0], c[n, b, 2], \ldots, c[n, b, k].$$

These crossings will already have names of the form $c[x, y, h]$ or $c[x, y, v]$ of course, so EQMaker creates an appropriate dictionary for the above names.

EQMaker creates the left hand side of the equation for each region. This expression is in a general form in the sense that the sign in front of each variable, and the h or v value in each variable, is a function that depends on the labeling of the vanishing cycles. For each n, there will be exactly one expression that has both $c[1, a, n]$ and $c[1, a, n+1]$ in it. EQMaker will also produce exactly one expression with $c[1, b, n]$ and $c[1, b, n+1]$ in it (and also for $n = k$ and $n = 0$). For each n, EQMaker replaces this pair of expressions A and B with $A + (B)$. These parentheses can be useful for when the user wants to find such combined equations in Equations.txt.

EQMaker then deletes all of the variables $c[i, a, j]$ and $c[i, b, j]$ along with their leading signs. Next, EQMaker cycles through all labelings of the vanishing cycles, evaluating the signs and h or v values for each labeling, and appending the right-hand side of each equation as a function of the signs that appear on its left hand side.

Finally, EQMaker deletes duplicate equations and outputs a comma-separated list of the remaining equations, one equation on each line, within one pair of curly braces, in the file Equations.txt.

EQMaker also produces a number of files that document the above process. For example, EQMaker creates:

- An image of each CompleteLine
- An image of each region
- An image of each boundary polygon with its detected corners
- Spreadsheets detailing the relabeling and corresponding corner variable assignments
- An image of the full diagram showing all of the regions
- An image of each pair of combined regions at either end of each handle
- An image of the full diagram showing all of the CompleteLines (this is the file AllLines.png mentioned in the first sentence of this section).

3. Making a diagram

These are requirements for making a diagram that EQMaker can properly parse. It is not instructions for making an allowable crown diagram of a 4-manifold: EQMaker will happily output a list of equations for any picture that satisfies the following requirements, even if it has nothing to do with an allowable crown diagram. The *regions* of the diagram are the connected components of the complement of the union of all bezier curves. This includes the inside of each rectangular handle foot.

3.1. **Overall requirements.** In Inkscape, a free app available at inkscape.org, create your diagram entirely out of 1-pixel-thick bezier curves consisting of perfectly horizontal and perfectly vertical line segments. Do not create an outer rectangle border for the diagram; this boundary is specified by the dimensions of the svg file you are creating. The bezier curves must not intersect the image boundary. All bezier curves must be pairwise transverse, with visible space between parallel segments. All crossings should occur as a crossing between exactly two bezier curves with different *description* attributes (explained below). This last requirement will be satisfied for any diagram satisfying the other requirements in this section, but it is a good idea to click around a few crossings to check.

3.2. **Feet of handles.**

(1) The bezier curve for each foot of each handle must be a closed rectangle.
(2) The outermost region cannot touch any handle foot.
(3) Record the names of the two feet of each handle in their *description* attribute in the format *hna* and *hnb*, where n is any integer 0–9 (the description can be viewed by right-clicking the curve and choosing *Object Properties*). Thus, for example, the two feet of handle number 1 have the descriptions $h1a$ and $h1b$. This limits the number of handles to no more than 10.
(4) The intersection of each region with the union of all the handle feet must be connected. That is, each region is allowed to touch no more than one handle foot along no more than one connected segment.

It may be helpful to know that EQMaker assumes the top-left corner of handle *hna* is identified with the top-right corner of *hnb*. A pixel travels around *hna* counter-clockwise starting at the top-left, enumerating the ends of vanishing cycles, and a pixel starts at the top-right corner of *hnb*, enumerating ends clockwise. The k^{th} entry in the sequence of ends for *hna* is identified with the k^{th} entry in the sequence for *hnb*.

3.3. **Vanishing cycles.**

(1) Choose a labeling for the vanishing cycles and record each vanishing cycle's label as its *description* attribute in the format $c01$, $c02$, and so forth. EQMaker will use this to generate all other labelings.
(2) Each vanishing cycle must have no self-intersections.
(3) Some vanishing cycles will run over multiple handles, so be sure every segment that composes a vanishing cycle has the same description (consider combining the segments of that vanishing cycle into a single object by selecting them all and choosing *Combine* from the *Path* menu).
(4) The ends of each vanishing cycle must connect with the feet of whatever handles they run over without visibly passing into the interior of the rectangle. This can be achieved by choosing an appropriate style for the ends of Bezier curves, and using the alignment tool on the end of the vanishing cycle and a corner of the handle foot.

4. Producing the equations for your diagram

(1) Move the output of any previous EQMaker sessions into a different folder than the one containing EQMaker.py. Otherwise, things will be overwritten.
(2) Name your diagram image.svg, and put it in the same folder as EQMaker.py
(3) In a python command line, cd to the folder containing EQMaker.py and enter

```
python EQMaker.py
```

The command line window will display the progress of EQMaker, counting off the regions as they are processed. When this is complete, EQMaker will create a folder called image.svg_Data containing all of the output. The equations will be listed in Equations.txt.

5. When things go wrong

Here are a few common sources of EQMaker errors:

(1) A CompleteLine may coincide with a copy of itself because the user forgot about a copy/paste they did.
(2) A line segment may be almost horizontal or vertical, but not perfectly so.
(3) Two line segments may coincide for a bit in a non-transverse intersection.

(4) A segment of the diagram may have an empty or malformed description attribute.
(5) In the process of making the diagram, there may be an extra invisible forgotten Bezier curve whose width attribute was cleared somehow. You might select all, then open the Fill and Stroke menu to see if the width attribute is 1 px. If it is blank, or a percentage, then there is probably an invisible Bezier curve somewhere.

If EQMaker stops with an error, one way to look for issues in your diagram is to examine the pictures in the RegionImages folder. For example, none of those regions should have any lines going through them. It can also be helpful to look at the last image in the folder in case EQMaker stops before processing all the regions, because it probably stopped at an adjacent region.