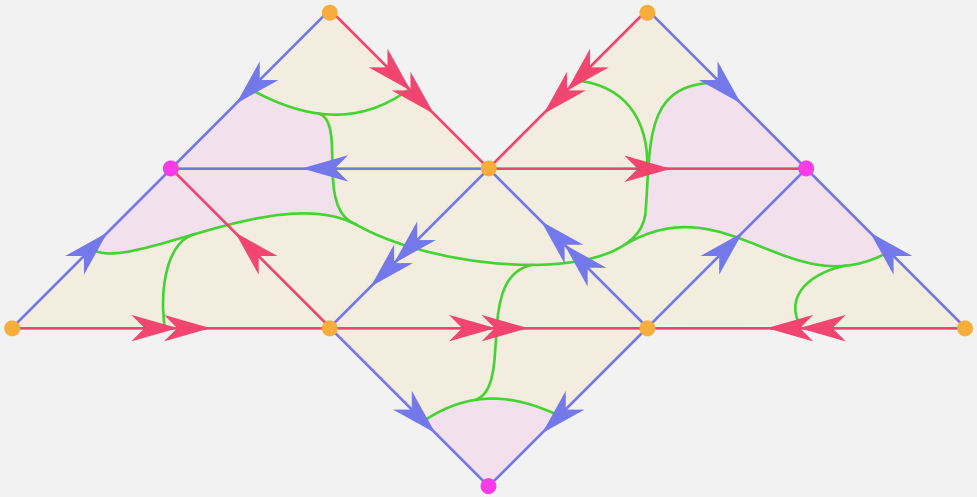
Pseudo-Anosov homeomorphisms via veering triangulations



Anna Parlak

University of Oxford

Institut Henri Poincaré

Mapping class groups and $\mathrm{Out}(F_n)$

25th of April 2022

Veering triangulations of pseudo-Anosov mapping tori

S — closed orientable surface with finite genus

$\varphi : S \to S$ — pseudo-Anosov homeomorphism

Veering triangulations of pseudo-Anosov mapping tori

S — closed orientable surface with finite genus

$\varphi : S \to S$ — pseudo-Anosov homeomorphism

Consider the homeomorphism $\mathring{\varphi} : \mathring{S} \to \mathring{S}$ on S punctured at the singularities of the invariant foliations of φ

Veering triangulations of pseudo-Anosov mapping tori

S — closed orientable surface with finite genus

$\varphi : S \to S$ — pseudo-Anosov homeomorphism

Consider the homeomorphism $\mathring{\varphi} : \mathring{S} \to \mathring{S}$ on S punctured at the singularities of the invariant foliations of φ

$$M = \left(\mathring{S} \times [0,1] \right) / \{(x,1) \sim (\mathring{\varphi}(x), 0)\}$$

Veering triangulations of pseudo-Anosov mapping tori

S — closed orientable surface with finite genus

$\varphi : S \to S$ — pseudo-Anosov homeomorphism

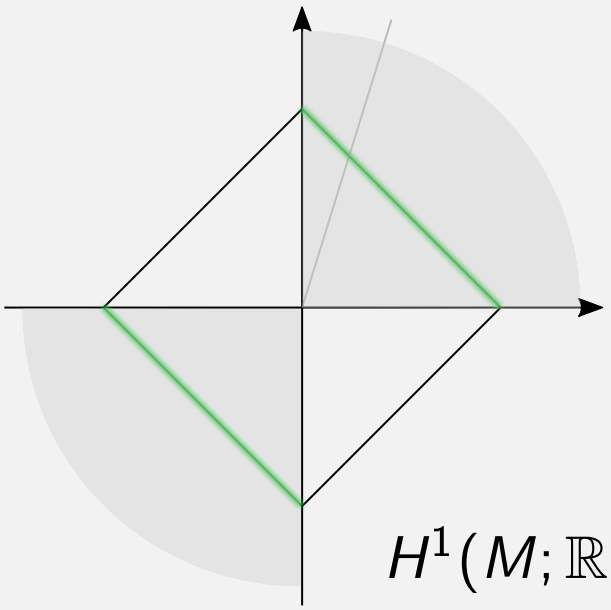
Consider the homeomorphism $\mathring{\varphi} : \mathring{S} \to \mathring{S}$ on S punctured at the singularities of the invariant foliations of φ

$$M = \left(\mathring{S} \times [0,1] \right) / \{(x,1) \sim (\mathring{\varphi}(x), 0)\}$$

Agol '10:

M admits a veering triangulation which combinatorially encodes information about $\mathring{S}$ and $\mathring{\varphi}$.

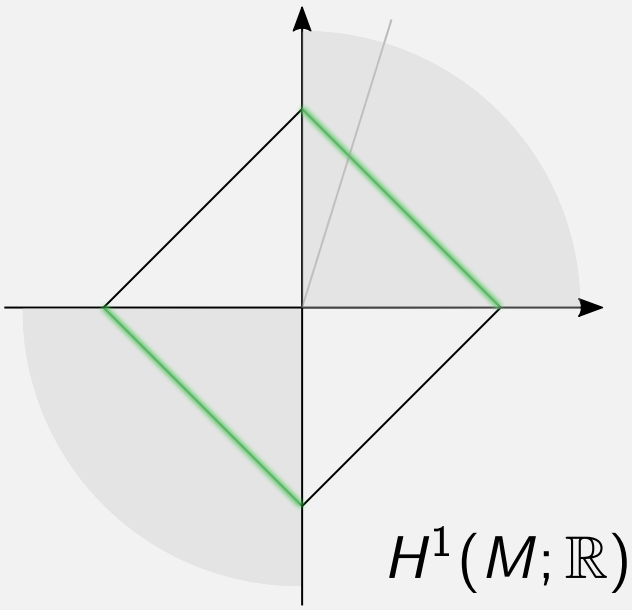
Veering triangulations and fibered faces



Minsky-Taylor '16:

If $b_1(M) > 1$ then the constructed veering triangulation encodes information about all fibrations lying over the same fibered face of the Thurston norm ball.

Veering triangulations and fibered faces



Minsky-Taylor '16:

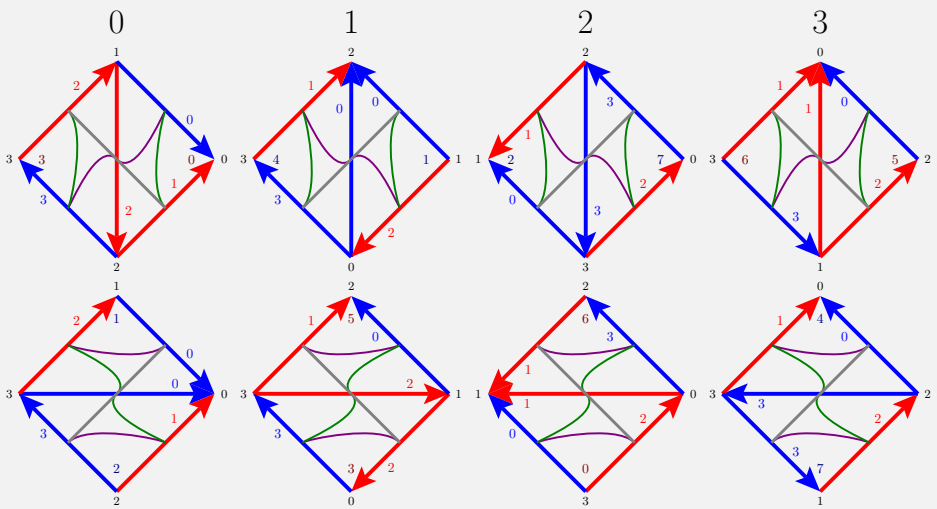
If $b_1(M) > 1$ then the constructed veering triangulation encodes information about all fibrations lying over the same fibered face of the Thurston norm ball.

Hence we can use (layered) veering triangulations to study different ways in which the same 3-manifold fibers over the circle

Veering Census (Giannopolous, Schleimer, Segerman)

https://math.okstate.edu/people/segerman/veering.html

```
cPcbbbdxm_10 L 1 G 2 N 1 [2] [2,0,0] Z/5+Z ['m003','otet02_00000']
cPcbbbiht_12 L 1 G 2 E 1 [4] [2,0,0] Z ['m004','4_1','K2_1','K4a1','otet02_00001']
dLQacccjsnk_200 L 1 G 1 N 1 [2] [2,0,1] Z ['m016','K3_1','K12n242']
dLQbccchhfo_122 L 1 G 2 E 1 [4] [2,0,1] Z/2+Z ['m009']
dLQbccchhsj_122 L 1 G 2 N 1 [2] [2,0,1] Z/6+Z ['m010']
eLAkaccddjsnak_2001 L 1 G 1 N 1 [2] [2,1,1] Z ['m119']
eLAkbbcdddhwqj_2102 L 1 G 1 N 1 [2] [2,0,2] Z ['m052']
eLAkbccddhhsqs_1220 L 1 G 1 N 1 [2] [2,1,1] Z/2+Z ['m146']
eLMkbcddddedde_2100 L 2 G 1 N 1 [2,2] [4,0,0] Z+Z ['m203','6^2_2','L6a2','otet04_00001']
eLMkbcdddhhhdu_1221 L 1 G 2 N 1 [2] [2,0,2] Z/7+Z ['m022']
eLMkbcdddhhhml_1221 L 1 G 2 E 1 [4] [2,0,2] Z/3+Z ['m023']
eLMkbcdddhhqqa_1220 L 1 G 2 E 1 [4] [2,1,1] Z/2+Z/2+Z ['m136']
eLMkbcdddhhqxh_1220 L 1 G 2 N 1 [2] [2,1,1] Z/2+Z/4+Z ['m135']
```



Veering GitHub (Parlak, Schleimer, Segerman)

https://github.com/henryseg/Veering

README.md

Veering

Python code for working with transverse taut and veering ideal triangulations; implemented by Anna Parlak, Saul Schleimer, and Henry Segerman. The taut and veering polynomials are defined by Michael Landry, Yair Minsky and Sam Taylor. We thank Nathan Dunfield for many helpful comments (and for some code).

Installation

Essentially all of the veering code relies on regina; some of it relies on snappy and some on SageMath. Other parts rely on the Python vector graphics package pyx. Installation instructions for SageMath, snappy, and regina can be found at the following webpages:

https://doc.sagemath.org/html/en/installation/
https://snappy.math.uic.edu/installing.html
https://github.com/3-manifolds/regina_wheels

To install Veering via the command line type:

```
git clone https://github.com/henryseg/Veering
```

Studying fibrations using tools from the Veering GitHub

```
sage: sig = 'eLMkbcddddedde_2100'
```

Studying fibrations using tools from the Veering GitHub

```
sage: sig = 'eLMkbcddddedde_2100'
sage: import taut_polytope
sage: taut_polytope.is_layered(sig)
True                                    # carries fibrations
```

Studying fibrations using tools from the Veering GitHub

```
sage: sig = 'eLMkbcddddedde_2100'
sage: import taut_polytope
sage: taut_polytope.is_layered(sig)
True                                          # carries fibrations
sage: taut_polytope.cone_in_homology(sig)
[N(1, 0), N(1, 2)]                            # infinitely many fibrations
```

Studying fibrations using tools from the Veering GitHub

```
sage: sig = 'eLMkbcddddedde_2100'
sage: import taut_polytope
sage: taut_polytope.is_layered(sig)
True                                                    # carries fibrations
sage: taut_polytope.cone_in_homology(sig)
[N(1, 0), N(1, 2)]                                      # infinitely many fibrations
sage: taut_polytope.taut_rays(sig)                      # generators of carried surfaces
[(0, 0, 1, 0, 1, 0, 2, 0), (2, 3, 0, 0, 2, 0, 0, 1), (1, 1, 0, 0, 1, 0, 1, 0),
(1, 2, 0, 0, 0, 1, 0, 0), (2, 0, 0, 1, 2, 0, 3, 0), (3, 0, 0, 4, 0, 3, 0, 2),
(1, 0, 0, 1, 1, 0, 0, 1), (1, 0, 0, 1, 0, 1, 1, 0), (0, 0, 2, 1, 0, 2, 3, 0),
(0, 0, 3, 2, 3, 0, 0, 4), (0, 0, 1, 1, 0, 1, 0, 1), (0, 1, 1, 0, 1, 0, 0, 1),
(0, 1, 1, 0, 0, 1, 1, 0), (0, 3, 2, 0, 0, 2, 0, 1)]
```

Studying fibrations using tools from the Veering GitHub

```
sage: weights = (0, 1, 1, 0, 1, 0, 0, 1)
```

Studying fibrations using tools from the Veering GitHub

```
sage: weights = (0, 1, 1, 0, 1, 0, 0, 1)
sage: import carried_surface
sage: surface = carried_surface.build_surface(sig, weights)
sage: carried_surface.stratum_from_weights_surface(weights, surface)
((0, 4), [1, 1, 1, 1])    # 4-times punctured sphere, after filling four 1-prongs
```

Studying fibrations using tools from the Veering GitHub

```
sage: weights = (0, 1, 1, 0, 1, 0, 0, 1)
sage: import carried_surface
sage: surface = carried_surface.build_surface(sig, weights)
sage: carried_surface.stratum_from_weights_surface(weights, surface)
((0, 4), [1, 1, 1, 1])    # 4-times punctured sphere, after filling four 1-prongs
sage: import stretch_factors
```

Studying fibrations using tools from the Veering GitHub

```
sage: weights = (0, 1, 1, 0, 1, 0, 0, 1)
sage: import carried_surface
sage: surface = carried_surface.build_surface(sig, weights)
sage: carried_surface.stratum_from_weights_surface(weights, surface)
((0, 4), [1, 1, 1, 1])    # 4-times punctured sphere, after filling four 1-prongs
sage: import stretch_factors
sage: P = taut_polytope.projection_to_homology(sig)
sage: P*vector(weights)
(1, 1)                    # the homology class of the surface in H_2(M, ∂M)
```

Studying fibrations using tools from the Veering GitHub

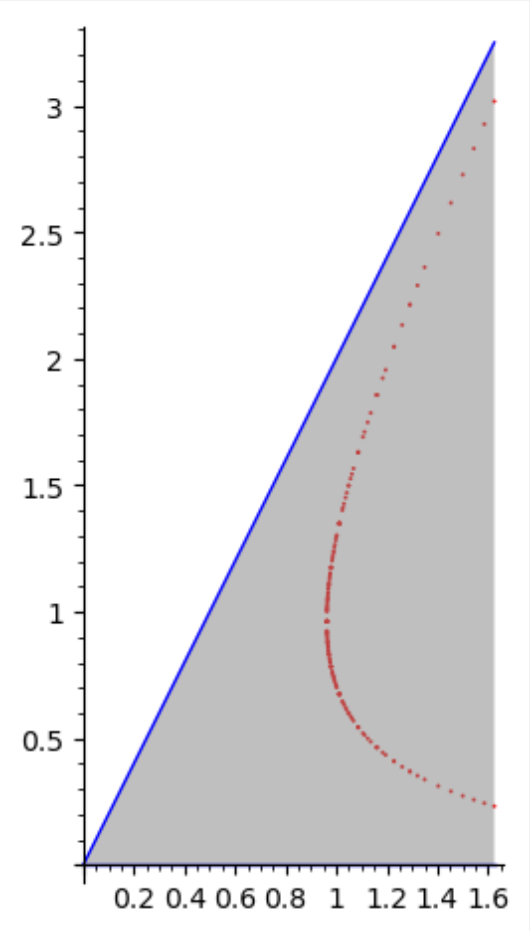
```
sage: weights = (0, 1, 1, 0, 1, 0, 0, 1)
sage: import carried_surface
sage: surface = carried_surface.build_surface(sig, weights)
sage: carried_surface.stratum_from_weights_surface(weights, surface)
((0, 4), [1, 1, 1, 1])    # 4-times punctured sphere, after filling four 1-prongs
sage: import stretch_factors
sage: P = taut_polytope.projection_to_homology(sig)
sage: P*vector(weights)
(1, 1)                    # the homology class of the surface in H_2(M, ∂M)
sage: stretch_factors.stretch_factor(sig, (1,1))
2.61803398874989          # stretch factor of the monodromy of the fibration
```

Many stretch factors in one picture

```
sage: stretch_factors.level_set_entropy_graph(sig)
```

Many stretch factors in one picture

sage: stretch_factors.level_set_entropy_graph(sig)



fibered class $\alpha \rightsquigarrow$ stretch factor λ

$\rightsquigarrow$ entropy $\log(\lambda)$

entropy for $k \cdot \alpha$ is $\frac{\log(\lambda)}{k}$

plotted:

unique point on the ray through α with

entropy $= 1$

(i.e. $\log(\lambda) \cdot \alpha$)