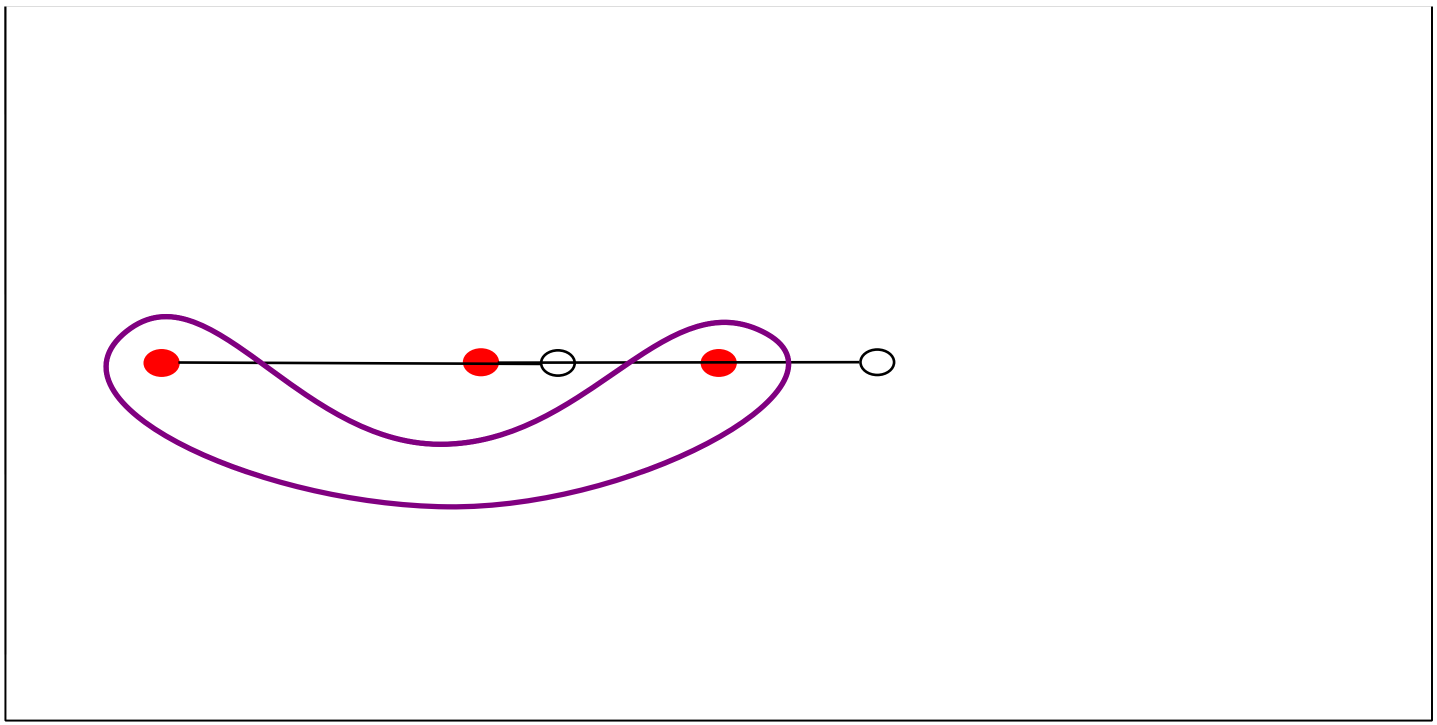
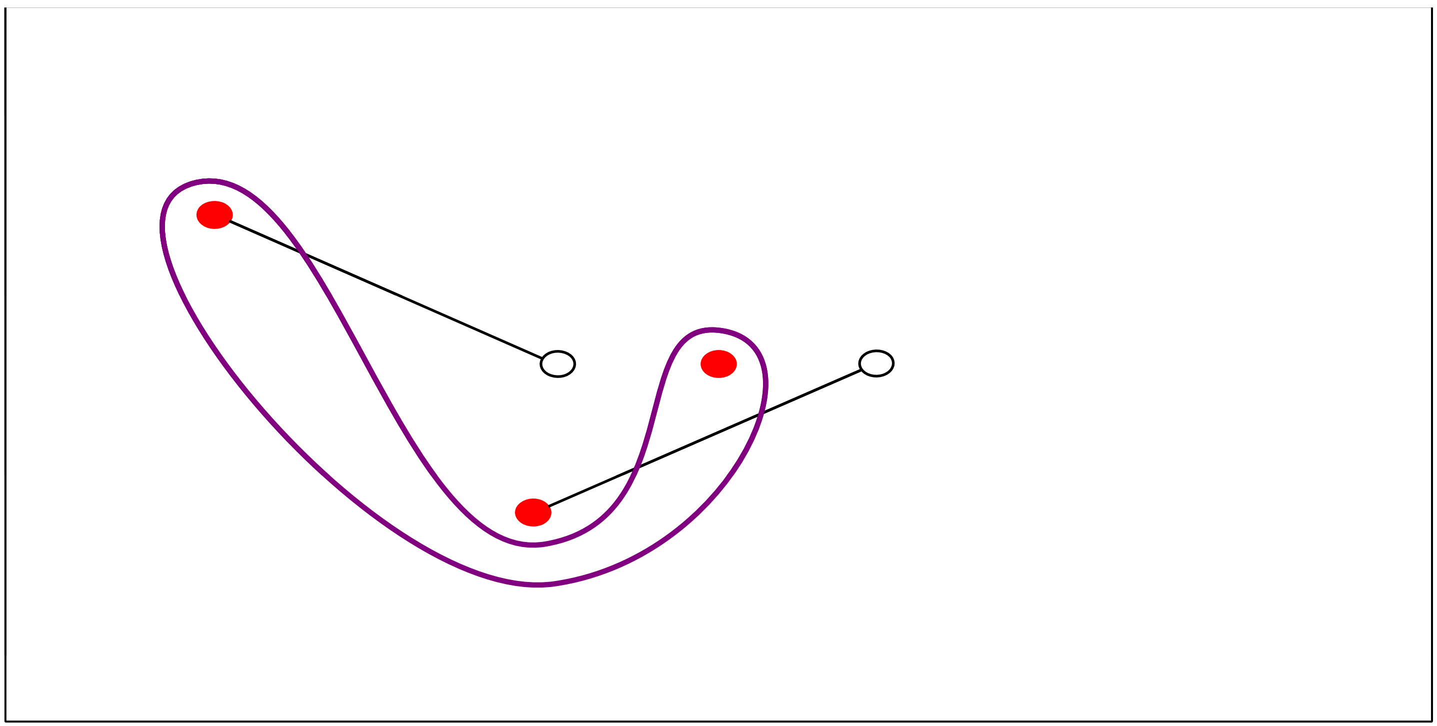
Dynamics of Surface Homeomorphisms

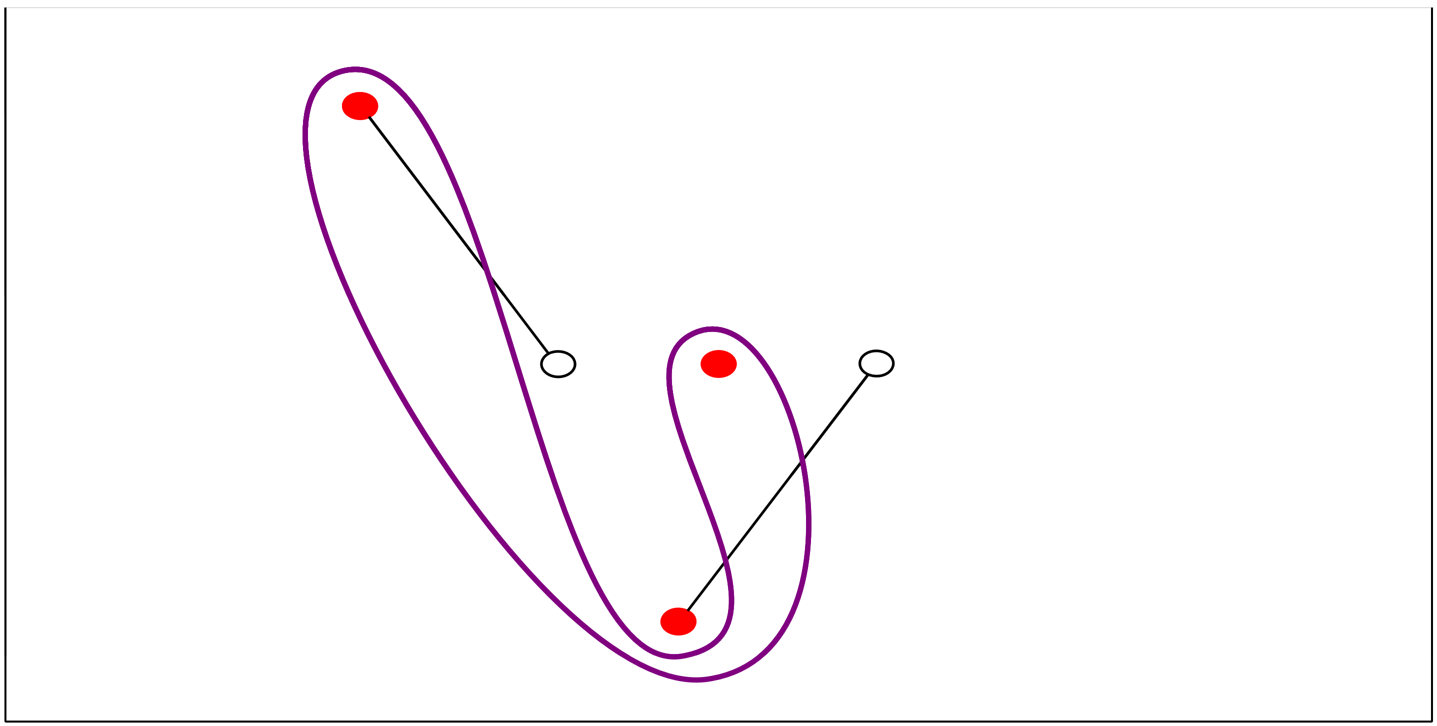
Yandi Wu
UC Berkeley
YMC 2017

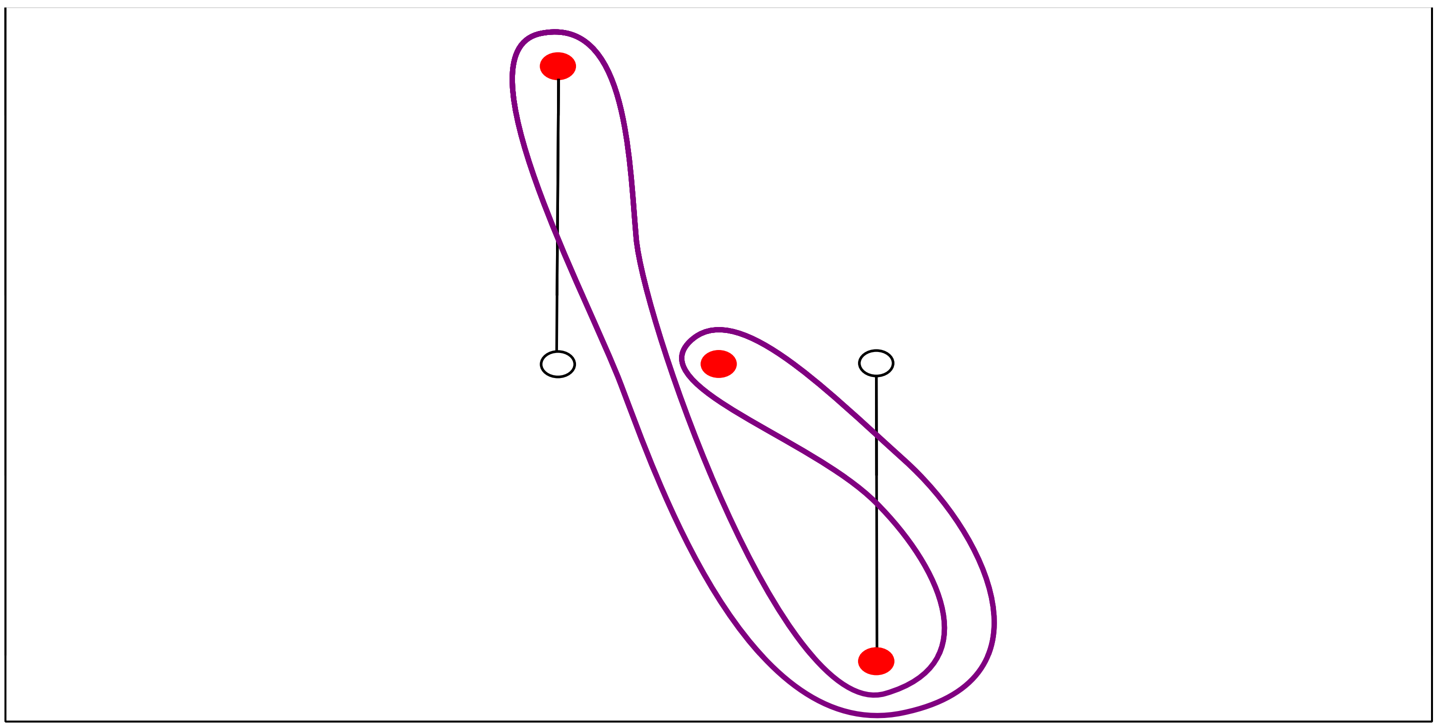
Joint with Ian Katz & Yihan Zhou
Advisors: Balázs Strenner, Dan Margalit
Georgia Institute of Technology

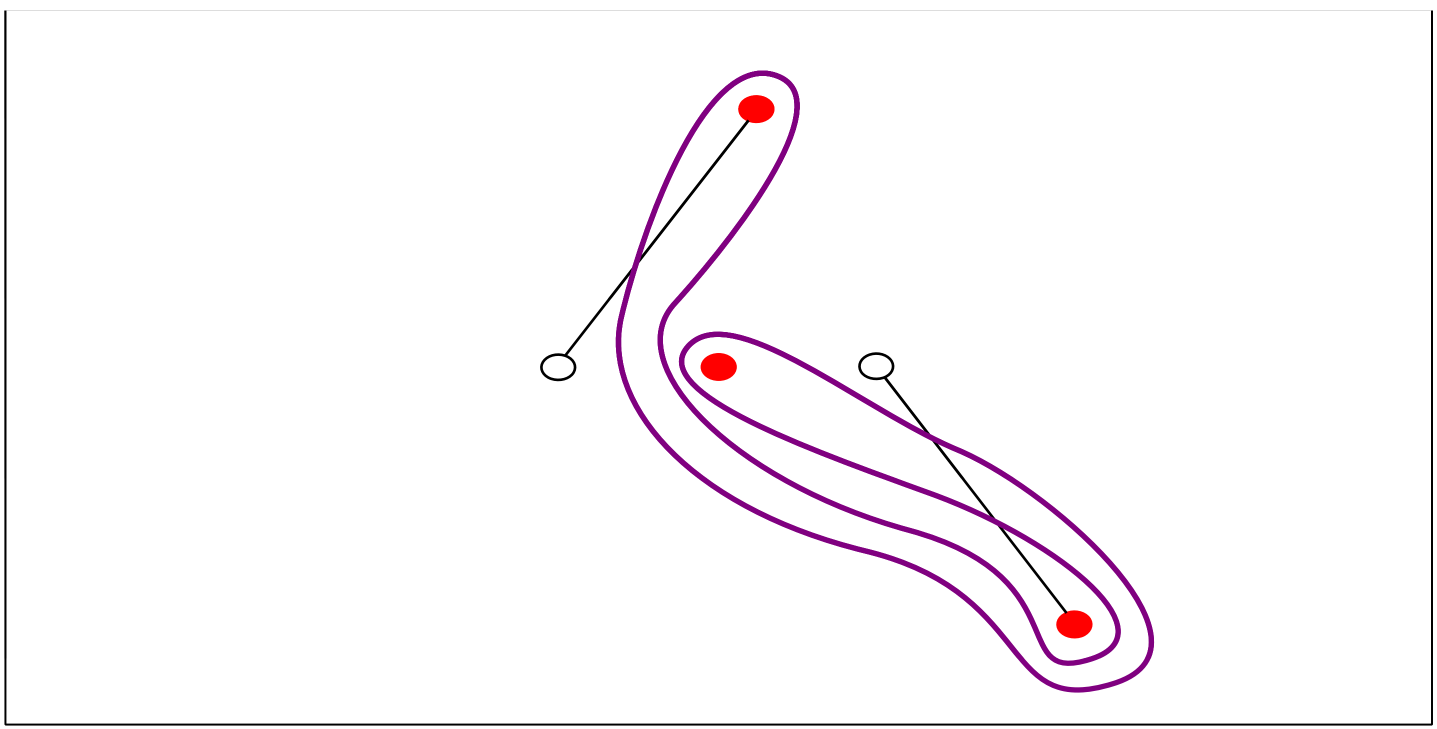
Q: How efficiently does this taffy puller stretch taffy?

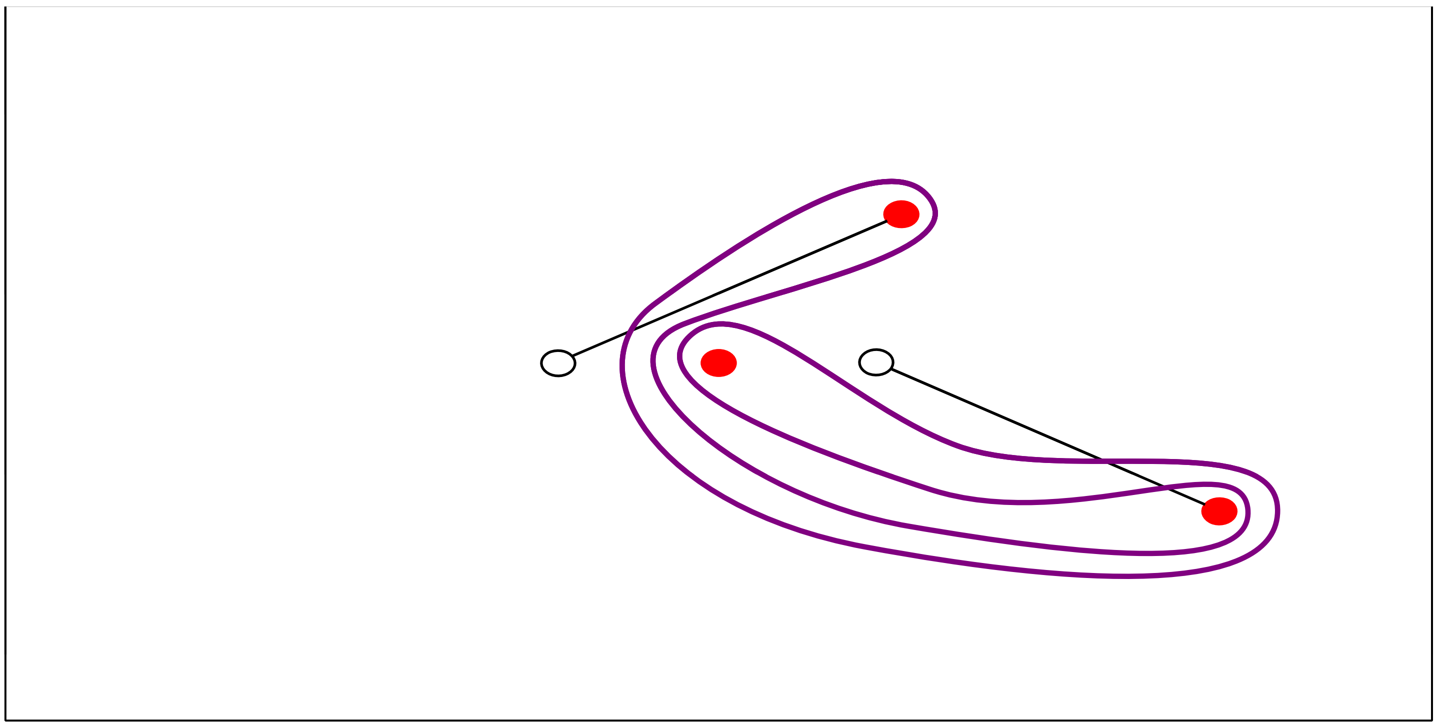


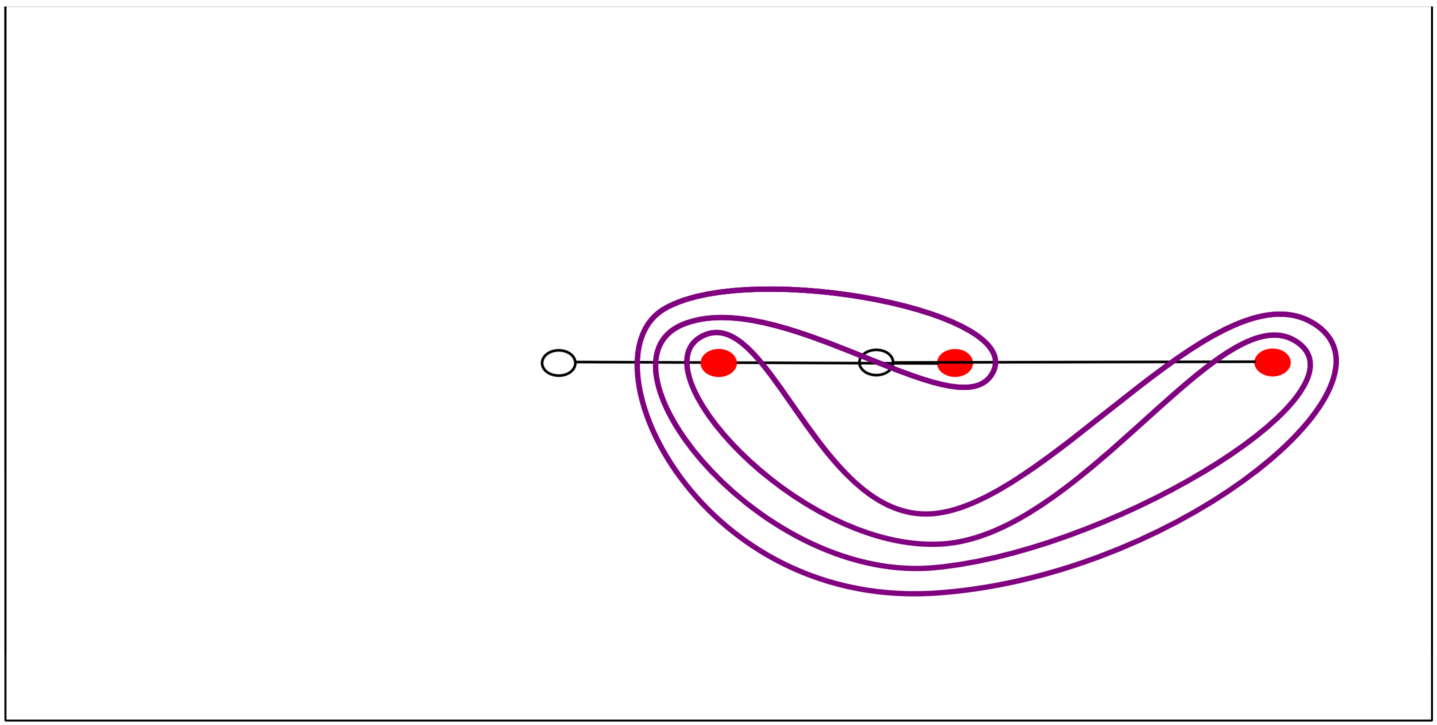


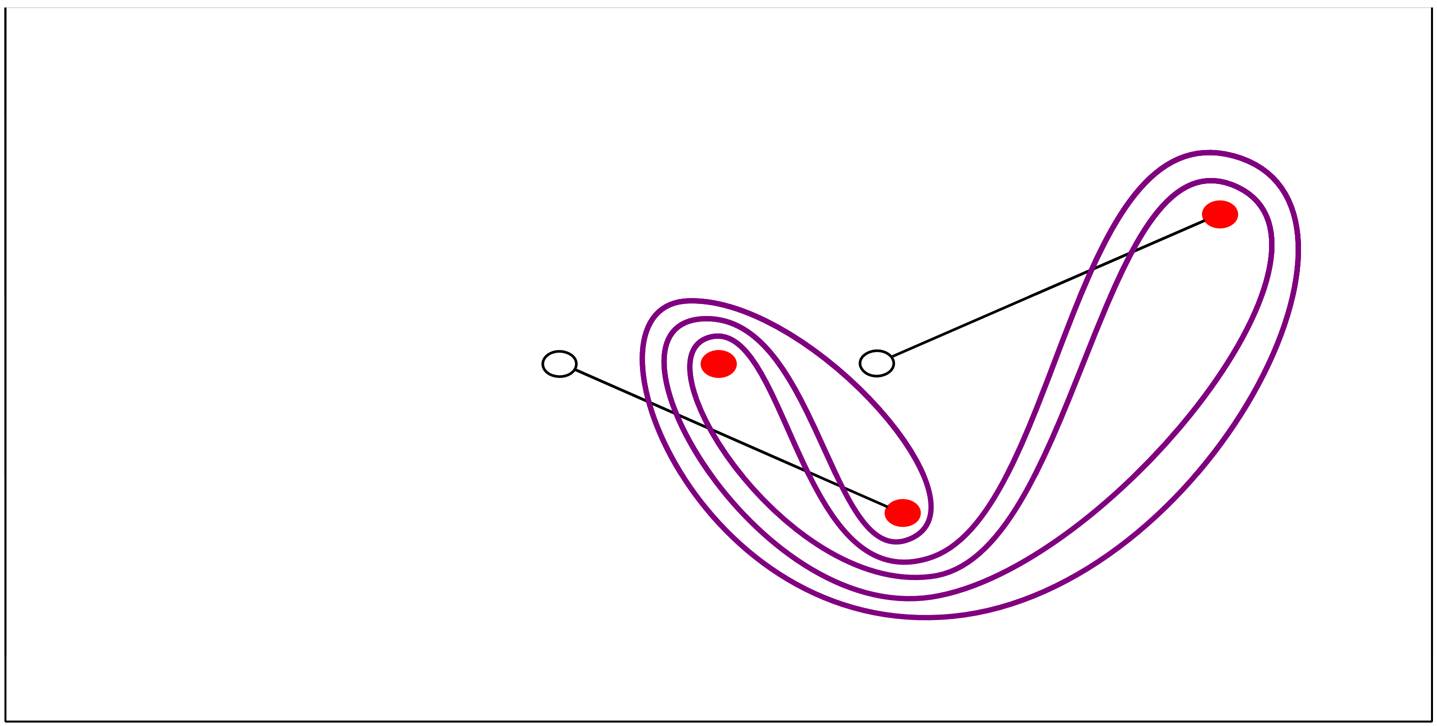


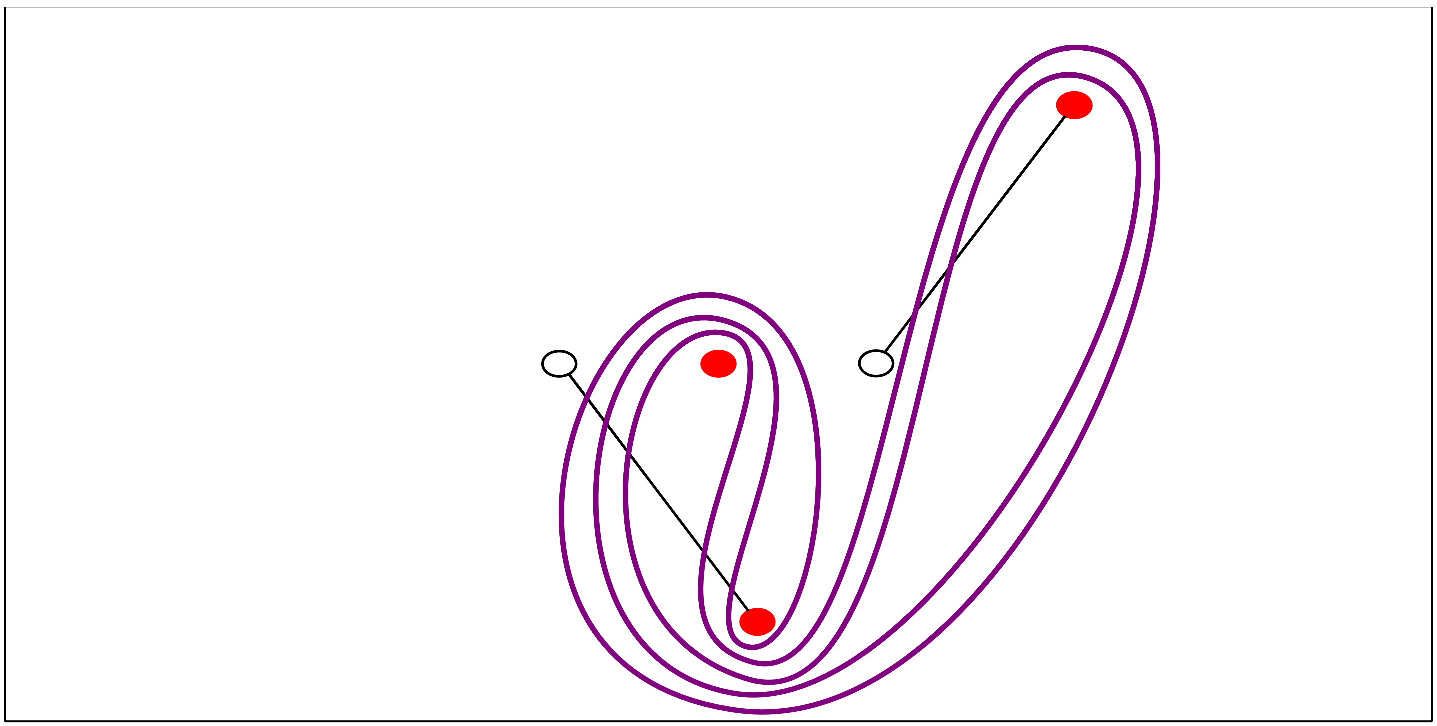


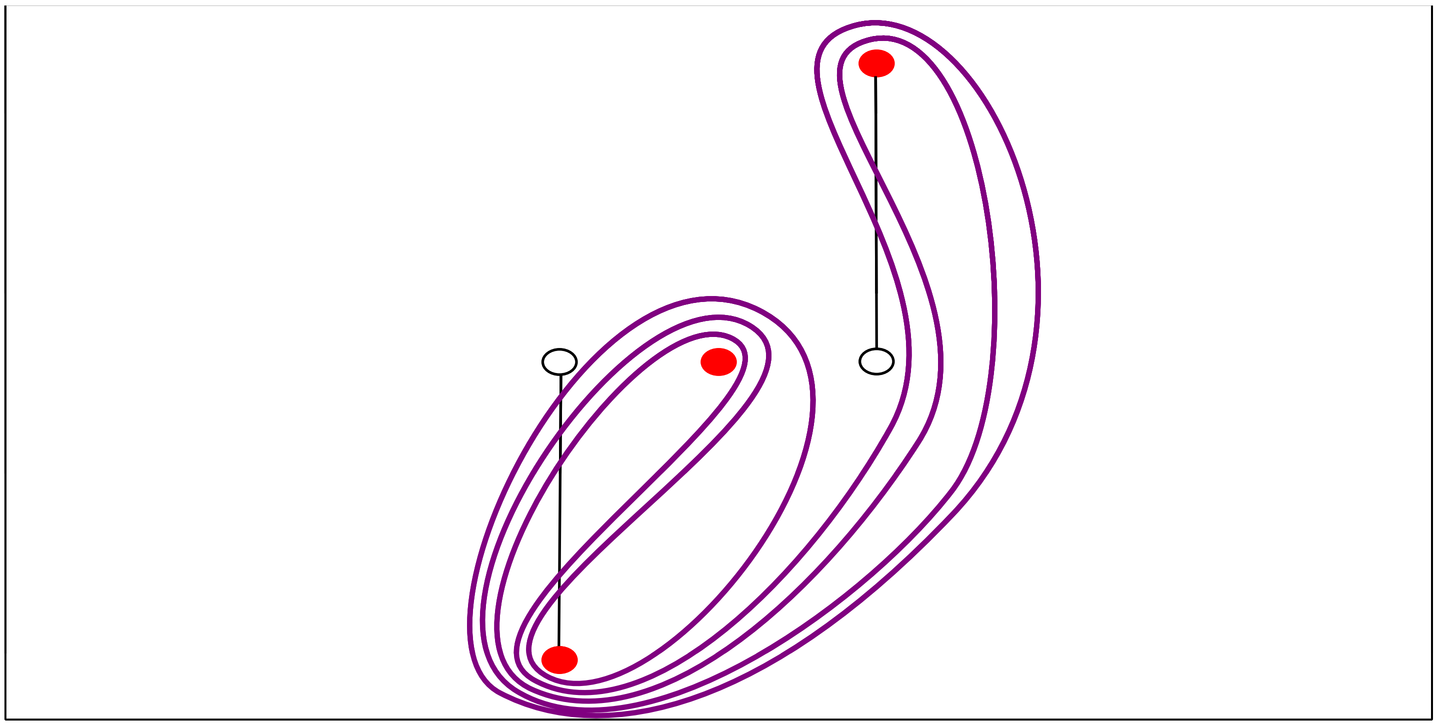


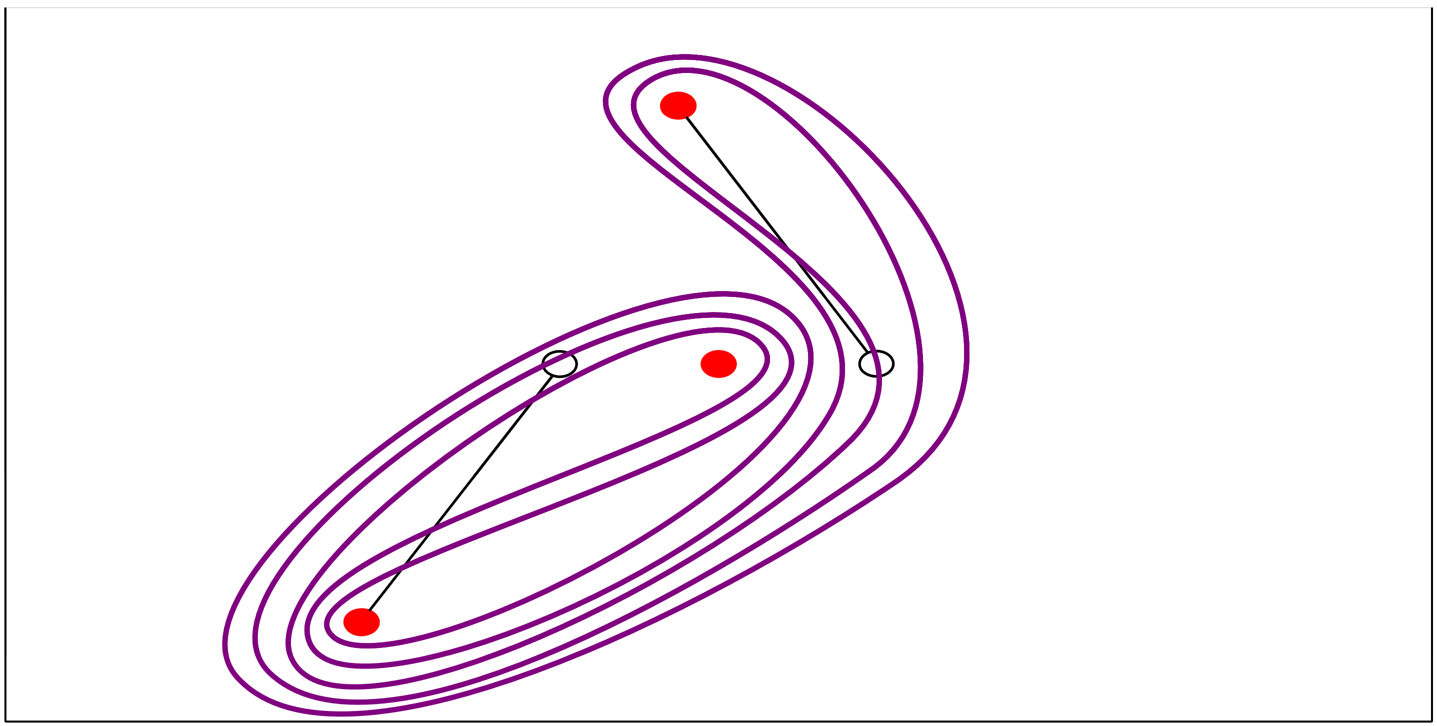


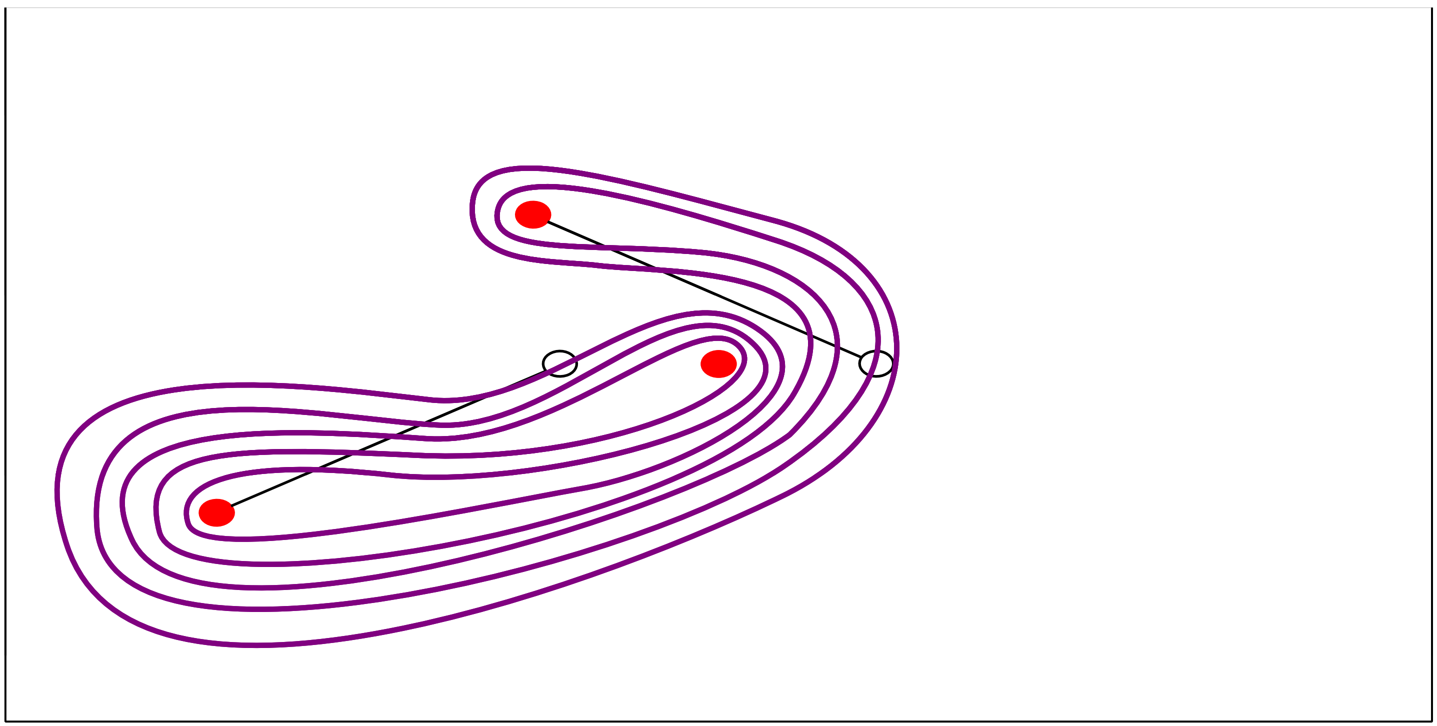




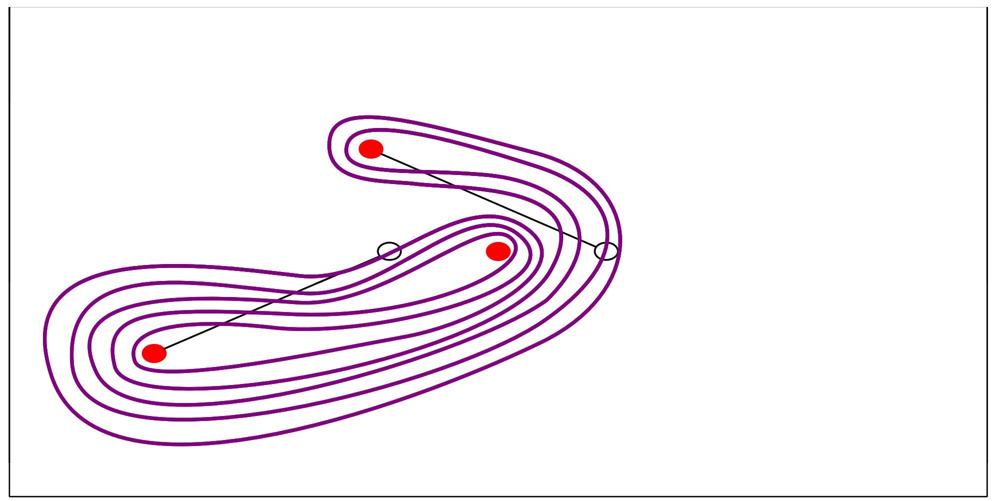








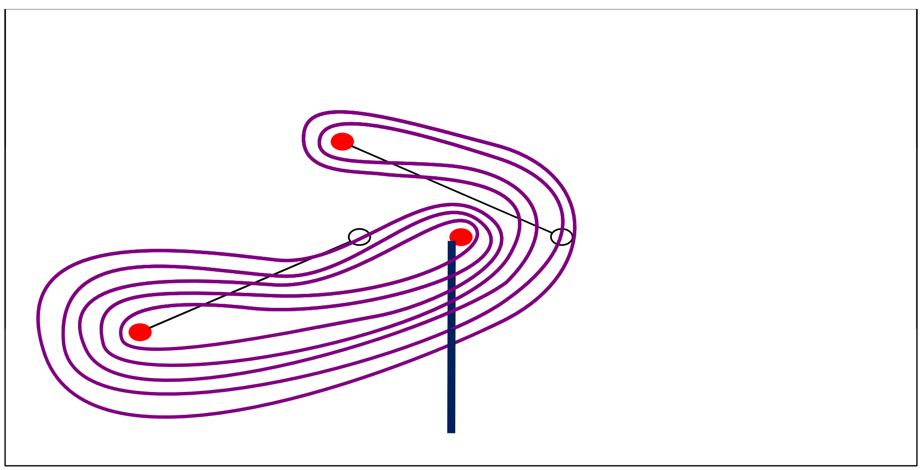
Q: How efficiently does this taffy puller stretch taffy?



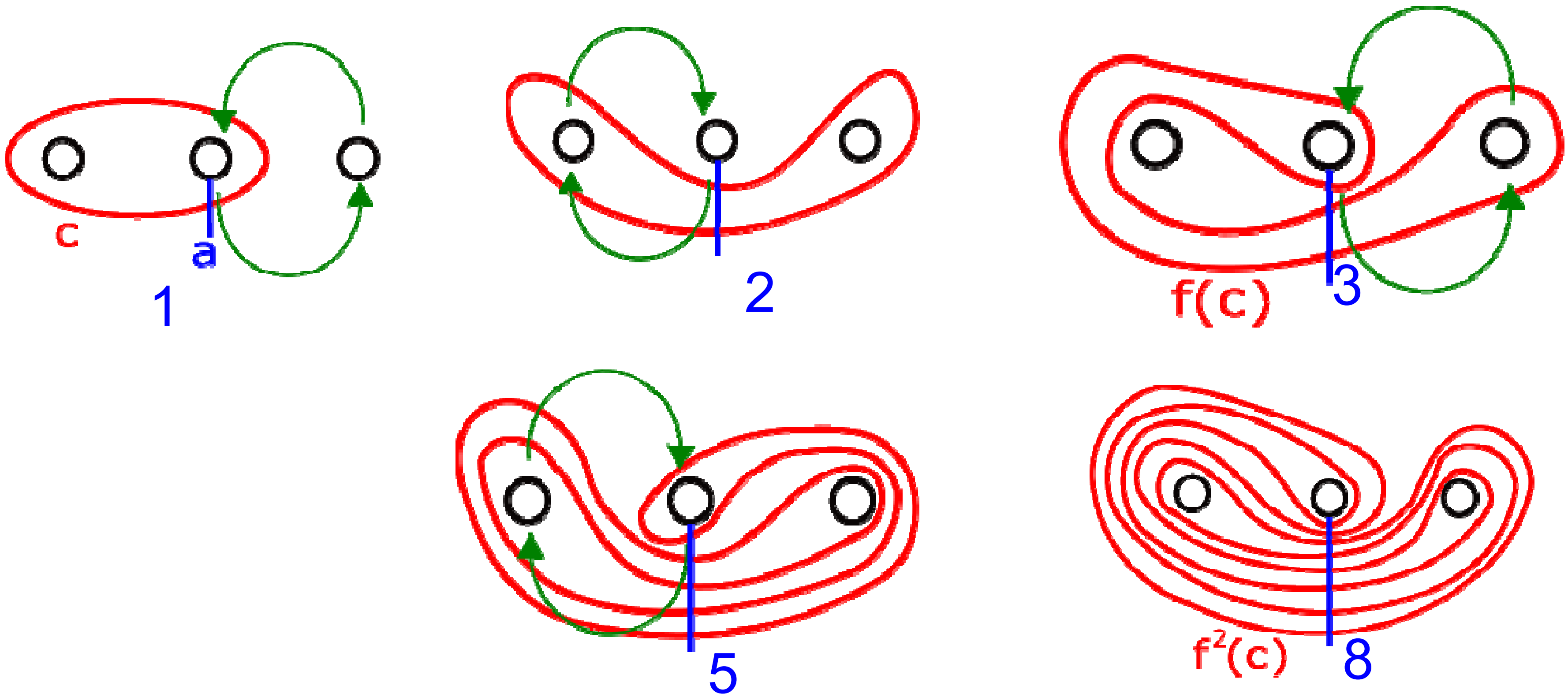
Q: How efficiently does this taffy puller stretch taffy?

Let b_k = # intersections with vertical arc

Q: What is the growth rate of b_k?



$$\lim_{k\to\infty} \frac{b_{k+1}}{b_k}$$

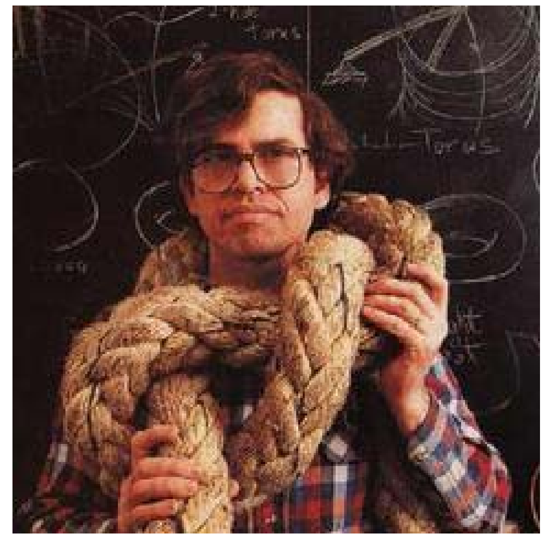


Stretch factor: $\boldsymbol{\varphi}^2 \approx 2.618$

Nielsen-Thurston Theory

The Nielsen-Thurston Classification

Every homeomorphism on a surface is one of the following types:



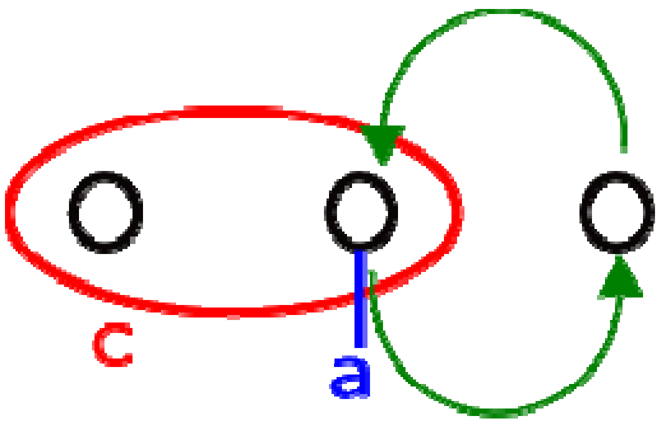
1) Finite Order (ie rotation)

2) Reducible: fixes disjoint union of curves

3) **pseudo-Anosov**: stretches every curve exponentially (ie taffy puller)

Measuring the Stretch Factor

The taffy puller is an example of a homeomorphism on $\mathbb{R}^2 - \{3 \text{ points}\}$



c = simple closed curve

a = reference arc

f = homeomorphism

Stretch factor = $\lim_{N \to \infty} \frac{i(a, f^{N+1}(c))}{i(a, f^N(c))}$

The Nielsen-Thurston Classification Problem

Fix s_i , generators for Homeo(S)/~.

<u>INPUT</u>

$f = s_1 s_2 \ldots s_n$ → Algorithm →

<u>OUTPUT</u>

Finite order

Reducible

pseudo-Anosov

& stretch factor

Theorem

Margalit-Strenner-Yurttas: There exists a quadratic-time algorithm for the Nielsen-Thurston Classification Problem.

The Algorithm

Setup: Compute piecewise linear action of each s_i generator on "space of curves."

Input: Homeo *f*: S → S

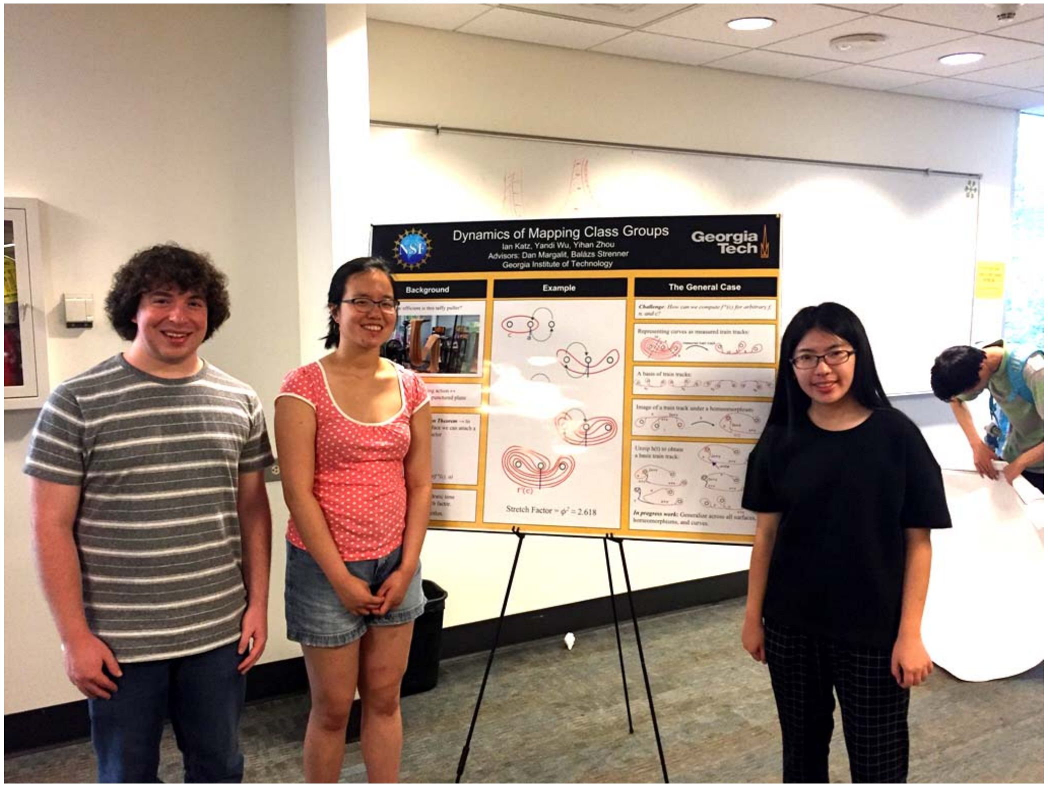
1) Compute $f^q(c)$

Margalit-Strenner-Yurtass: q exists

2) Find matrix M for *f* acting on $f^q(c)$

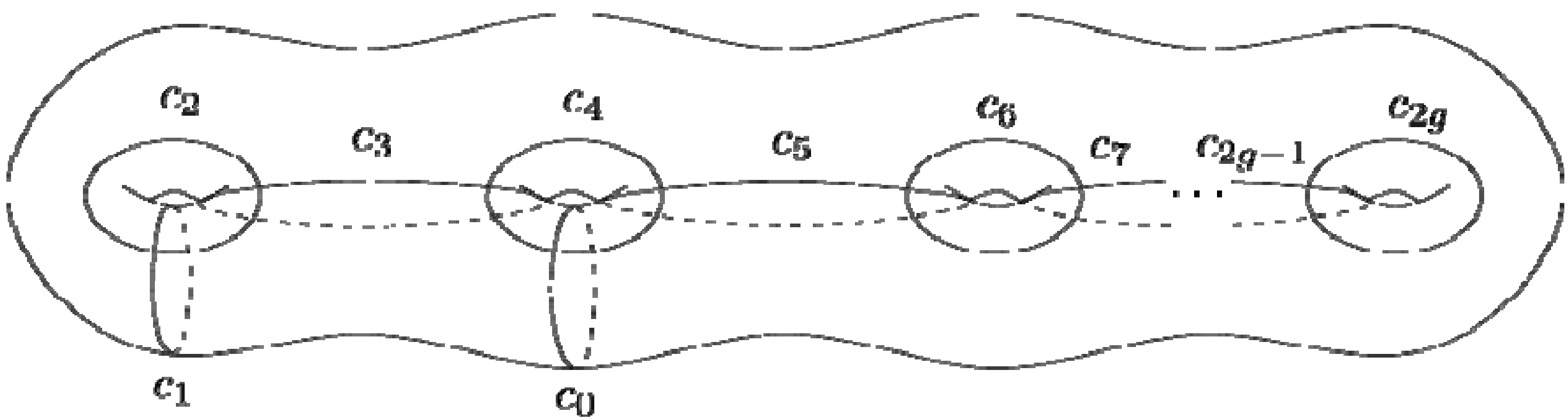
3) Compute largest eigenvalue of M (stretch factor)

Our Work: Implementing MaCAW

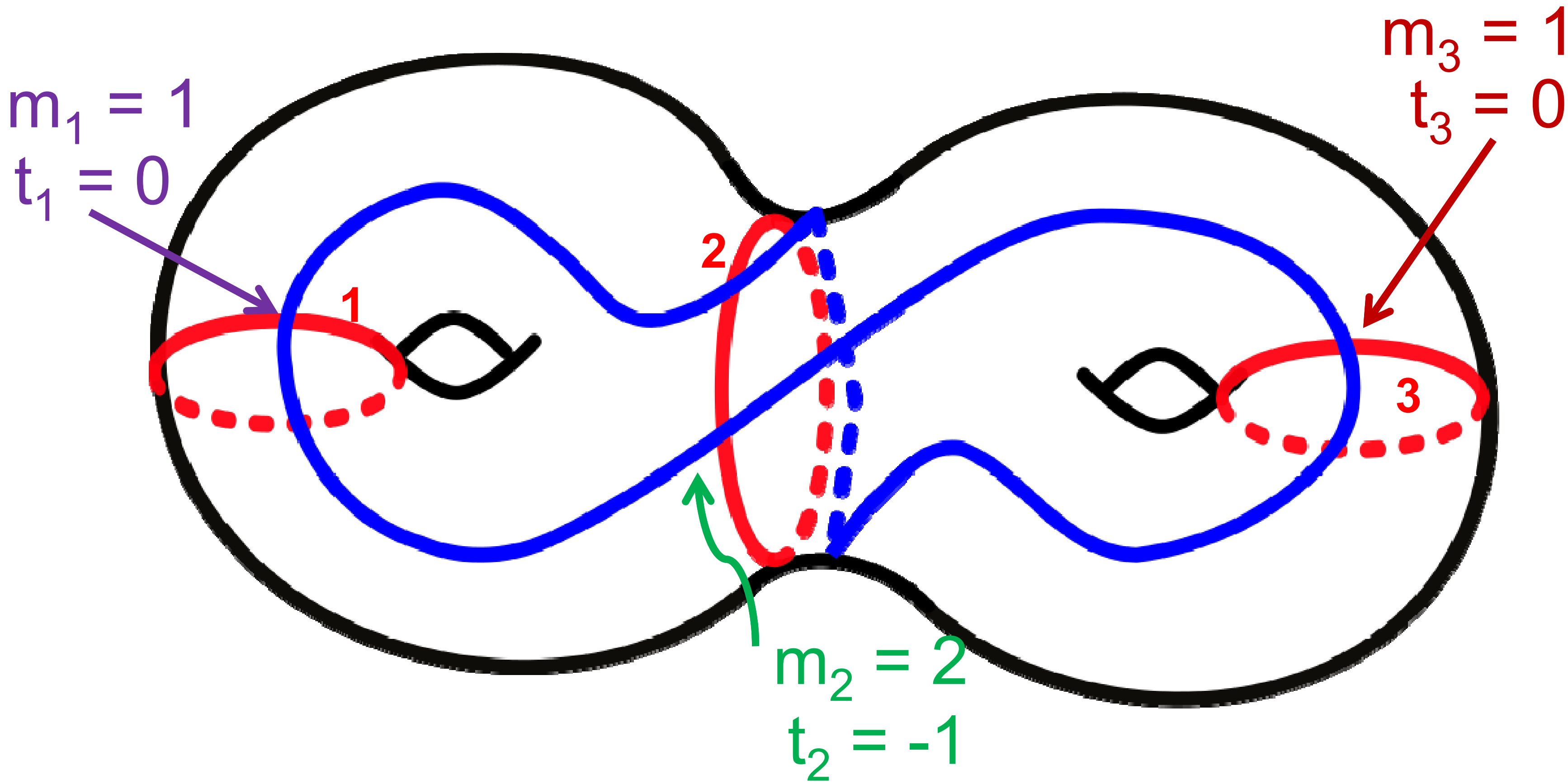


Challenge

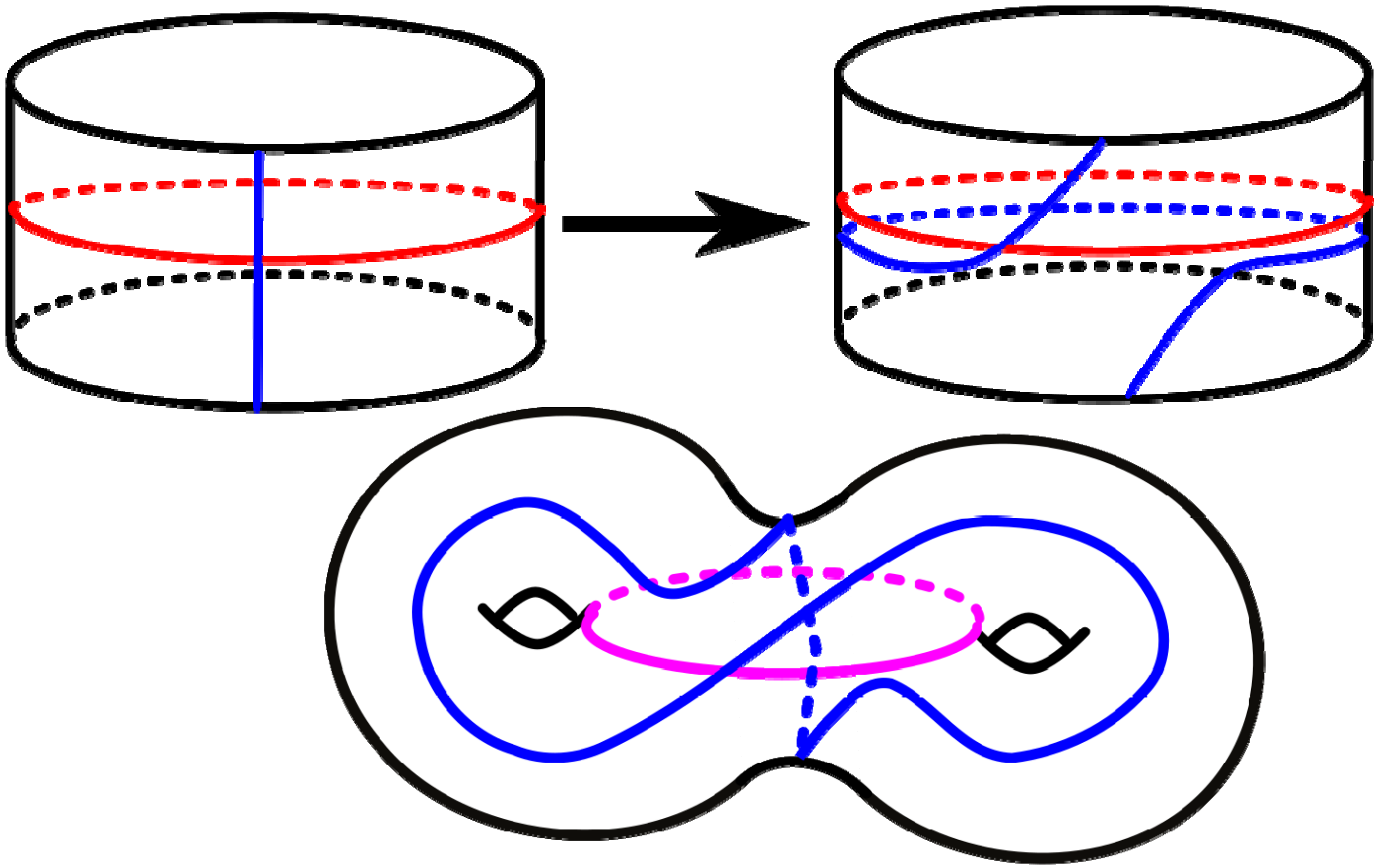
How to compute $f^q(c)$, or even $s_i(c)$?



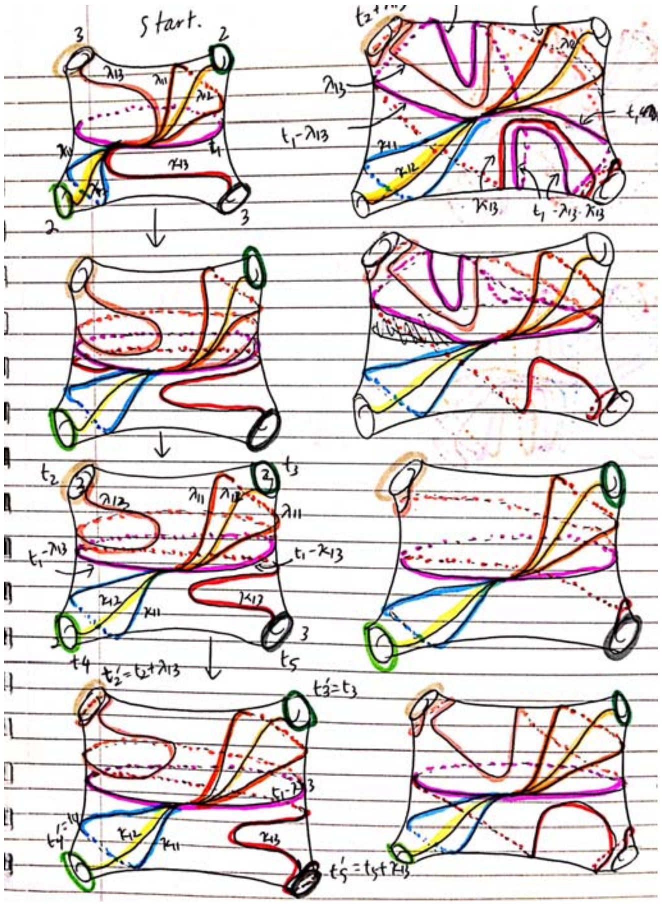
Dehn Thurston Coordinates

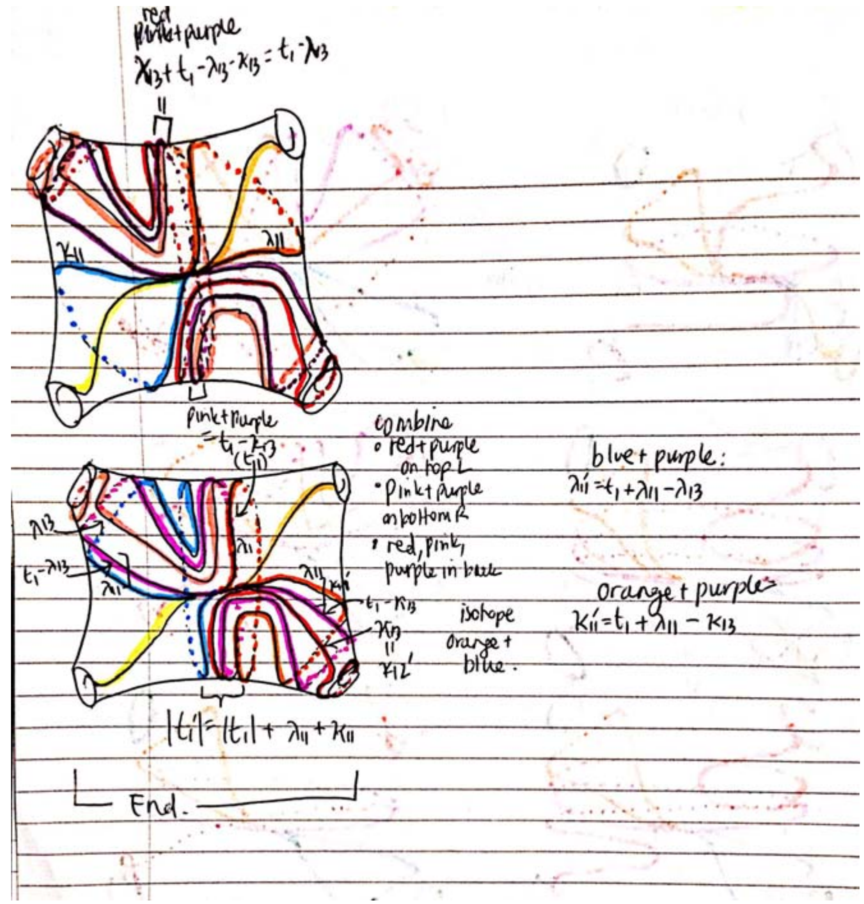


Generators: Dehn Twists



A small sample of our work





Train tracks represent curves

MaCAW

1. Solves word problem (word on generators = identity) on closed surfaces.

2. Approximates stretch factors on closed surfaces.

3. Determines if a homeomorphism is periodic and finds its order.

```
sage: %runfile pants_decomposition.py
sage: A, B, c = humphries_generators(6)
sage: A[0]*B[0] == B[0]*A[0]
False
sage: A[0]*B[0]*A[0] == B[0]*A[0]*B[0]
True
sage: A[3]*A[4] == A[4]*A[3]
True
sage: f = A[0]*B[0]^(-1)
sage: f.stretch_factor()
2.61803398874989
sage: n((3+sqrt(5))/2)
2.61803398874989
```