DEPENDENCE OF THE ROTATION NUMBER OF THE RESTRICTION OF BILLIARD MAPS TO A COMMON CAUSTICS

Victoria Wilczynski

2022

intro

First, we extract information from a .json file. This file provides us with the sequences a_n and b_n, and store them in global variables for use in our functions. We also make a function so that if the length of the two sequences are not equal, we append 0's onto the one of smaller length until a_n and b_n are equal in length.

check_interval

This function first created p' where $p' = pmod2\pi$, and q' where $q' = q + (p' - p)$. Then, if $q' < p'$, we continue to add 2π to q' until $q' > p'$. The function returns the updated p' and q'.

mod_2pi

This function returns a given angle mod 2π ($x \mapsto x \bmod 2\pi$).

check_equality

This function checks for "equality" between two floats. We check if the absolute difference between two floats is less than the tolerance (set at the beginning of our code). If it is true, the two floats are considered equal, otherwise, they are not.

check_eq_mod_mod_pi

This function is similar to `check_equality`. It checks if the absolute difference between two floats mod π is less than the tolerance (the same one set at the beginning of our code). If true, the two floats are equal, otherwise, they are not.

distance

This function is defined by the Euclidean distance, $d = \sqrt{(x_2 - x_1)^2 + (y_2 + y_1)^2}$. The function takes a=$(x_1, y_1)$ and b=(x_2, y_2) as inputs, and returns an integer.

trig_poly

This function computes $p(\theta)$ given the input θ. We first extract a_0 from the global variable a_n. We then run a loop that computes the summation $\sum_{k=2}^{N} a_k \cos(k\theta) + b_k \sin(k\theta)$. We return the final sum of $p(\theta) = a_0 + \sum_{k=2}^{N} a_k \cos(k\theta) + b_k \sin(k\theta)$.

convex_check

This function checks for a strongly convex domain. To check for a strongly convex domain, we want to find $\epsilon > 0$ such that if $\rho(n\epsilon) > M\epsilon \; \forall n \in 0, 1, ..., \frac{2\pi}{\epsilon}$, where we find M by setting $|\rho'| < M$. Then, $|\rho'| = |\sum_{k=1}^{N}(-ka_k sin(k\theta) + kb_k cos(k\theta)| \leq \sum_{k=1}^{N} |ka_k||sin(k\theta)| + |kb_k||cos(k\theta)| \leq \sum_{k=1}^{N} k(|a_k| + |b_k|) = M$. Thus, our function sets $\epsilon = \frac{1}{2}$, and finds M from $a_n$and b_n. If $\rho(n\epsilon) > M\epsilon, \forall n \in 0, 1, ..., \frac{2\pi}{\epsilon}$, then we return True, as it is strongly convex. Otherwise, we take half of epsilon, and check the inequality again. We keep looping and taking a smaller epsilon each time. If we reach our tolerance and $\rho(n\epsilon) < M\epsilon$, then we return False, as this is not a strongly convex domain.

x_theta

For this function, we need to compute the integral $\int_0^\theta p(\theta')\cos(\theta')\,d\theta'$. Let $\theta' = t$ in our computation for clarity. Note the two trig identities: $\cos(x)\sin(y) = \frac{\sin(y+x)+\sin(y-x)}{2}$ and $\cos(x)\cos(y) = \frac{\cos(x+y)+\cos(x-y)}{2}$.

$$
\begin{aligned}
x(\theta) &= \int_0^\theta p(t)\cos(t)dt \\
&= \int_0^\theta [a_0 + \sum_{k=2}^{N} a_k\cos(kt) + b_k\sin(kt)]\cos(t)dt \qquad \text{substituting in } p(t) \\
&= \int_0^\theta a_0\cos(t)dt + \sum_{k=2}^{N}\int_0^\theta a_k\cos(kt)\cos(t)dt + \int_0^\theta b_k\sin(kt)\cos(t)dt \qquad \text{distributing cos(t), splitting integral by addition} \\
&= a_0\int_0^\theta \cos(t)dt + \sum_{k=2}^{N} a_k\int_0^\theta \frac{\cos(kt+t)+\cos(kt-t)}{2}dt + b_k\int_0^\theta \frac{\sin(kt+t)+\sin(kt-t)}{2}dt \qquad \text{using identities above} \\
&= a_0(\sin(t))\Big|_0^\theta + \frac{1}{2}\sum_{k=2}^{N} a_k(\frac{1}{k+1}\sin(kt+t) + \frac{1}{1-k}\sin(t-kt))\Big|_0^\theta + b_k(\frac{-1}{k+1}\cos(kt+t) - \frac{1}{1-k}\cos(kt-t))\Big|_0^\theta \\
&= a_0(\sin(\theta)) + \frac{1}{2}\sum_{k=2}^{N} a_k(\frac{1}{k+1}\sin(k\theta+\theta) + \frac{1}{1-k}\sin(\theta-k\theta)) + b_k[(\frac{-1}{k+1}\cos(k\theta+\theta) - \frac{1}{1-k}\cos(k\theta-\theta)) - (\frac{-1}{k+1} - \frac{1}{1-k})]
\end{aligned}
$$

So, we can make our function compute $x(\theta)$ with the above equation.

y_theta

For this function, we need to compute the integral $\int_0^\theta p(\theta')\sin(\theta')\,d\theta'$. Let $\theta' = t$ for clarity. It will be a similar computation to x_theta. Note the following trig identities: $\sin(x)\sin(y) = \frac{-\cos(x+y)+\cos(x-y)}{2}$ and $\cos(x)\sin(y) = \frac{\sin(x+y)+\sin(y-x)}{2}$.

$$
\begin{aligned}
y(\theta) &= \int_0^\theta p(t)\sin(t)dt \\
&= \int_0^\theta [a_0 + \sum_{k=2}^{N} a_k\cos(kt) + b_k\sin(kt)]\sin(t)dt \qquad \text{substituting in } p(t) \\
&= \int_0^\theta a_0\sin(t)dt + \sum_{k=2}^{N}\int_0^\theta a_k\cos(kt)\sin(t)dt + \int_0^\theta b_k\sin(kt)\sin(t)dt \qquad \text{distributing sin(t), splitting integral by addition} \\
&= a_0\int_0^\theta \sin(t)dt + \sum_{k=2}^{N} a_k\int_0^\theta \frac{\sin(kt+t)+\sin(t-kt)}{2}dt + b_k\int_0^\theta \frac{-\cos(kt+t)+\cos(kt-t)}{2}dt \qquad \text{using identities above} \\
&= a_0(-\cos(t))\Big|_0^\theta + \frac{1}{2}\sum_{k=2}^{N} a_k(\frac{-1}{k+1}\cos(kt+t) - \frac{1}{1-k}\cos(t-kt))\Big|_0^\theta + b_k(\frac{-1}{k+1}\sin(kt+t) + \frac{1}{1-k}\sin(t-kt))\Big|_0^\theta \\
&= a_0(-\cos(\theta)) + \frac{1}{2}\sum_{k=2}^{N} a_k[(\frac{-1}{k+1}\cos(k\theta+\theta) - \frac{1}{1-k}\cos(\theta-k\theta)) - (\frac{-1}{k+1} - \frac{1}{k-1})] + b_k(\frac{-1}{k+1}\sin(k\theta+\theta) + \frac{1}{1-k}\sin(\theta-k\theta))
\end{aligned}
$$

So we can calculate $y(\theta)$ using this equation above.

length

For this function, we need to first compute the integral $\int_\theta^{\theta'} p(t)dt$.

$$\begin{aligned}
Length(\theta, \theta') &= \int_\theta^{\theta'} p(t)dt \\
&= \int_\theta^{\theta'} [a_0 + \sum_{k=2}^{N} a_k \cos(kt) + b_k \sin(kt)]dt \qquad \text{substituting in p(t)} \\
&= \int_\theta^{\theta'} a_0 dt + \sum_{k=2}^{N} \int_\theta^{\theta'} a_k \cos(kt)dt + \int_\theta^{\theta'} b_k \sin(kt)dt \qquad \text{splitting integral} \\
&= a_0(t)\Big|_\theta^{\theta'} + \frac{1}{k}\sum_{k=2}^{N} a_k(\sin(kt))\Big|_\theta^{\theta'} + b_k(-\cos(kt))\Big|_\theta^{\theta'} \qquad \text{evaluating anti-derivatives} \\
&= a_0(\theta' - \theta) + \sum_{k=2}^{N} \frac{1}{k}[a_k(\sin(k\theta') - \sin(k\theta)) + b_k(-\cos(k\theta') + \cos(k\theta))]
\end{aligned}$$

So we can use this equation for our function. Let p=θ and q=θ' in our function. We also need to check that $q > p$ and that $q, p \in [0, 2\pi)$ before we compute this function by calling check_interval.

intersection

Given two lines L and K, where L and K are given as lists or tuples with $[x(\theta), y(\theta), \theta]$, where x and y are the coordinates of a point on the line and the angle θ is the angle the line makes with the horizontal. If the angle of both lines is the same, they are parallel, and thus there is no intersection point. If the angle of both lines are anti-parallel, they also never meet, and thus there is no intersection point. If the points on the two lines are the same, that is their intersection point, so we are done. Next, we can parameterize the lines as such: $p + tv$, where p is a point on the line, and $v = [cos(\theta), sin(\theta)]$, where θ is the angle the line makes with the horizontal. Then, once we have their parameterizations, we find that we get a system of equations as follows:

$$\begin{aligned}
a \cdot \cos(t) - b \cdot \cos(s) &= w - x \\
a \cdot \sin(t) - b \cdot \sin(s) &= z - y
\end{aligned}$$

We can then transform this system of equations into matrix form, where

$$\begin{aligned}
A = &\begin{bmatrix} a \cdot \cos(t) & -b \cdot \cos(s) \\ a \cdot \sin(t) & -b \cdot \sin(s) \end{bmatrix} \\
B = &\begin{bmatrix} w - z \\ z - y \end{bmatrix}
\end{aligned}$$

Then, we can use the numpy.linalg function `numpy.linalg.solve(A,B)`, and this gives us the solved values of a and b. We then can plug in a to our parameterization of line L, or b into our parameterization of line K. We choose the former, and thus we find the intersection point of line L and line K.

string_len

This function finds the length of the string, given the formula $\lambda(p, q) = \text{Length}(q, p) + |\overline{pr}| + |\overline{rq}|$. p and q are given as their respective tangent angles. We first use our x_theta and y_theta to find the points that correspond with these two angles. Then, we will find r using our intersection method. Then, we will find $|\overline{pr}|$ and $|\overline{rq}|$. To find $|\overline{pr}|$ and $|\overline{rq}|$, we can use our distance function using the points given and our found intersection point r. Then, to find Length(q,p), we can use the angles given (but adjusting p to be $p = p + 2\pi$ for the check_interval function to work properly) and our function length. Then, we add all these values together and we get our string length, ℓ.

domain_len

This function simply finds $|\Gamma|$ (the length of the curve). We add length$(\theta, \theta + \pi)$ and length$(\theta + \pi, \theta)$. This gives us $|\Gamma|$ as needed.

string_curve_ineq

This function checks to make sure that $\ell > |\Gamma|$. We find the length of the string using the function string_len. Then, we use domain_len to get the length of the curve. Then, we check if the length of the string is greater than the length of the curve. If true, returns True, otherwise it returns False.

find_q

This method inputs an angle θ tangent to the point p, and a string length, ℓ. We first get the tangent angle of the anti-podal point p', where $\theta' = \theta + \pi$. Then, we begin the bisection method. We first let $q = \frac{\theta+\theta'}{2}$. We then compute the string length from θ to q. If the string length is greater than ℓ, our desired q is in the first half of the arc, and we then set our interval to be from θ to q. If the string length is less than ℓ, our desired q is in the second half of the arc, and we then set our interval to be from q to θ'. Otherwise, we return q. Then, we continue looping until our interval is sufficiently small, and return the optimal q between p and its antipodal point.

rotations

This function finds the rotation number for a given domain. We first set M as the number of times we want to loop ($M >> 1$). Then, we set $\theta = 0$ as our starting point. We also set p=0 as our counter for how many times we loop around our domain. Then, we begin looping M times, and as we loop we set theta equal to find_q(θ, ℓ), to find the next point (we want to map $\theta \mapsto F_\ell(\theta)$). Then, if we find our new $\theta > 2\pi$, we have crossed our original spot (at 0), and we need to increment our rotation counter and fix theta to now start at $\theta \mod 2\pi$. We keep going until our loop is done, and then we return $\frac{\text{number of trips around domain}}{\text{number of loops in for loop}} = p/M$.

plot_rotation

This function plots rotation numbers over many values of ℓ. We choose an epsilon ($\epsilon = 10e-3$), and calculate the rotation number for values from $|\Gamma|$ to $1000 \cdot |\Gamma|$ by an increment of ϵ each loop. When we calculate the rotation number, we add it to a list. Once we are done our loop, we plot the rotation number against the value of ℓ used to get that rotation number.