AN OVERVIEW OF A DERIVATION OF THE EXPONENT OF CONVERGENCE OF A POINCARÉ SERIES

WILLIAM

Step 1.

Consider the following example: Let $\Gamma < \mathrm{PSU}(1,1)$ be a well-distributed Schottky group with m generators and its orbit $\Gamma(0)$ can be reconstructed by only using two operators T and R.

To construct the operator T, we start from the upper half-plane model: Define $T' \in \mathrm{PSL}(2,\mathbb{R})$ to be a hyperbolic isometry

$$T' := \begin{pmatrix} \Lambda & 0 \\ 0 & \Lambda^{-1} \end{pmatrix}$$

and T' fixes $\{0,\infty\}$, and $\Lambda < 1$.

Recall the Cayley transformation

$$\Psi(z) := \frac{iz+1}{z+i}, \Psi : \mathbb{H}^2 \to \mathbb{B}^2.$$

With $\Psi(z)$ in matrix representation:

$$\Psi = \begin{pmatrix} \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix}$$

and its inverse

$$\Psi^{-1} = -\begin{pmatrix} \frac{i}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix}.$$

Thus, the operator $T := \Psi T' \Psi^{-1} \in \mathrm{PSU}(1,1)$ can be derived as follows

$$T := \Psi T' \Psi^{-1} = \frac{-1}{2}\begin{pmatrix} -\Lambda - \Lambda^{-1} & -i\Lambda + i\Lambda^{-1} \\ i\Lambda - i\Lambda^{-1} & -\Lambda - \Lambda^{-1} \end{pmatrix} \in \mathrm{PSU}(1,1),$$

and it is easy to check that T fixes $\pm i$. Let $T(-i) = -i := T^+$ be denoted as the attractive fixed point, and $T(i) = i := T^- i$ be denoted as the repelling fixed point of T.

To define the rotation operator R, we would like it to be the function $R(z) = e^{ik\theta}z$ which is a counter-clockwise rotation operation, and $\theta := \frac{\pi}{m}$.

To achieve this goal we directly start from the Poincaré disk model

$$R = \begin{pmatrix} e^{\frac{i\theta}{2}} & 0 \\ 0 & e^{\frac{-i\theta}{2}} \end{pmatrix} \in \mathrm{PSU}(1,1)$$

which fixes $\{0,\infty\}$.

Step 2. Next, since in Poincaré disk model the hyperbolic length of a radial geodesic $[0,z]$ is

$$H(|z|) := \rho_{\mathbb{B}^2}(0,z) = \ln\left(\frac{1+|z|}{1-|z|}\right),$$

we know that the function $H(|z|)$ is a strictly monotone increasing real function.

Let $x = |z|$, then we can plot the graph of $H(|z|) = \rho_{\mathbb{B}^2}(0,z)$:

Date: February 14, 2022.

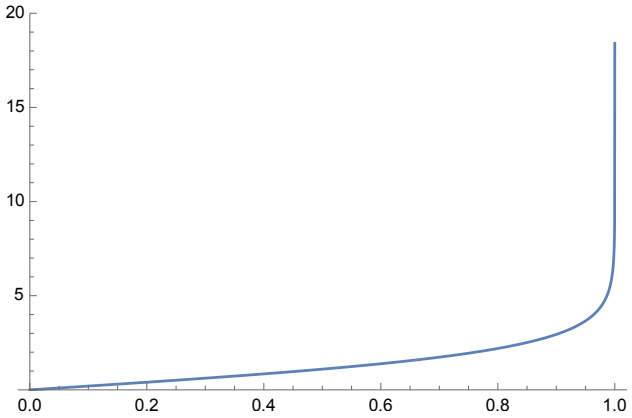

FIGURE 1. Plotting the function of the hyperbolic length function $H(x) = \ln\left(\frac{1+x}{1-x}\right), 0 \leq x < 1$.

In other words, instead of focusing on hyperbolic length of the radial geodesic $[0, z]$, to compare the hyperbolic length of two radial geodesic, $\rho_{\mathbb{B}^2}(0, z_1)$ and $\rho_{\mathbb{B}^2}(0, z_2)$, we merely need to compare the Euclidean distance to the origin, i.e. to compare $|z_1|$ and $|z_2|$.

Let

$$z_0 := re^{i\phi} \in D_0(T) \subset \mathbb{B}^2 \cap \Gamma(0),$$

where $D_0(T)$ is called the half-space of T with respect to 0. The half-space $D_0(T)$ is the interior point of the region bounded by $\partial\mathbb{B}^2$ and the perpendicular bisector of $T0$ and 0 denoted by $M_0(T)$, i.e. $M_0(T) := \{z : \rho_{\mathbb{B}^2}(z, 0) = \rho_{\mathbb{B}^2}(z, Tz)\}$.

Since $z_0 \in \Gamma(0)$, there exists $T_I \in \mathcal{W}_N$ such that $z_0 = T_I 0$.

Define symbols for the exponents of R and T in T_I as follows:

$$T_I := T^{\alpha_1} R^{\beta_1} T^{\alpha_2} R^{\beta_2} ... T^{\alpha_N} R^{\beta_N},$$

where $\alpha_1 = 1$, if $i > 1$, $\alpha_i \in \{0, 1\}, \sum_{i=1}^{N} \alpha_i = N$, and $0 \leq \beta_i \leq m - 1$.

Define the number of operators of T and R for a given reduced word T_I as follows

$$\#R(T_I) := \sum_{i=1}^{N} \beta_i,$$

and

$$\#T(T_I) := \sum_{i=1}^{N} \alpha_i.$$

Lemma 1. *Let $z_0 = re^{i\phi}$. Then, we have*

$$\rho_{\mathbb{B}^2}(0, Tz_0) > \rho_{\mathbb{B}^2}(0, TRz_0).$$

Proof. Based on the previous step, it is equivalent to prove:

$$|Tz_0| > |TRz_0|.$$

For simplicity, since $\Lambda < 1$ is fixed, let

$$A := -\Lambda - \Lambda^{-1},$$

and

$$B := i\Lambda - i\Lambda^{-1}.$$

Then

$$|Tz_0| = \left|\frac{Are^{i\phi} - iB}{iBre^{i\phi} - A}\right| = \left|\frac{(Ar\cos\phi) + i(Ar\sin\phi - B)}{(-A - Br\sin\phi) + i(Br\cos\phi)}\right|$$

$$= \sqrt{\frac{(Ar\cos\phi)^2 + (Ar\sin\phi - B)^2}{(Br\cos\phi)^2 + (-A - Br\sin\phi)^2}}$$
$$= \sqrt{\frac{(Ar)^2 - 2ABr\sin\phi}{(Br)^2 + 2ABr\sin\phi}}.$$

On the other hand,

$$|TRz_0| = \left|\frac{Are^{i(\phi+\theta)} - iB}{iBre^{i(\phi+\theta)} - A}\right|$$
$$= \sqrt{\frac{(Ar)^2 - 2ABr\sin(\phi+\theta)}{(Br)^2 + 2ABr\sin(\phi+\theta)}}.$$

Recall $\theta = \frac{\pi}{m}$, $m \in \mathbb{Z}^+$, $\frac{3\pi}{2} < \phi < \frac{7\pi}{4}$. Hence,

$$\frac{3\pi}{2} < \phi + \theta < \frac{\pi}{2} + \frac{7\pi}{4} = \frac{9\pi}{4}.$$

Since $\sin x$ is strictly monotone increasing in the range $\left(\frac{3\pi}{2}, \frac{9\pi}{4}\right)$ (which is illustrated in the graph below), we have

$$\sqrt{\frac{(Ar)^2 - 2ABr\sin(\phi+\theta)}{(Br)^2 + 2ABr\sin(\phi+\theta)}} < \sqrt{\frac{(Ar)^2 - 2ABr\sin(\phi)}{(Br)^2 + 2ABr\sin(\phi)}}.$$

The above inequality still holds when $\theta = \pi$. It follows that

$$|TRz_0| < |Tz_0|.$$

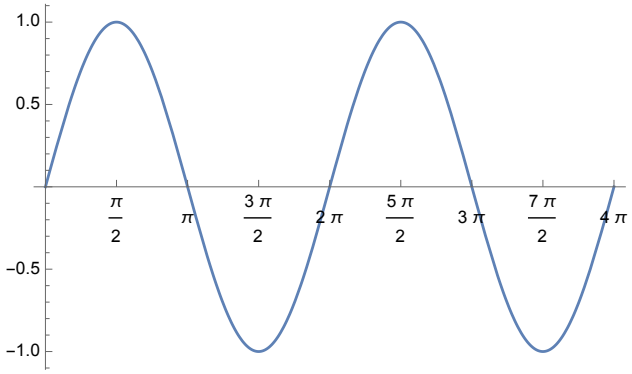

FIGURE 2. A graph of the sine function $G(x) = \sin(x), 0 \le x \le 4\pi$.

□

Lemma 2. *Let $z = r_0 e^{i\phi_0}$. Then, we have*

$$\rho_{\mathbb{B}^2}(0, z_0) < \rho_{\mathbb{B}^2}(0, Tz_0).$$

Proof. For simplicity, let us prove this in the upper half-plane model $\mathbb{H}^2$.

Recall that in the upper half-plane model, the hyperbolic length of the geodesic $[i, z]$ is

$$\rho_{\mathbb{H}^2}(i, z) = \cosh^{-1}\left(1 + \frac{|i - z|^2}{2\Im(z)}\right).$$

Also recall that $\operatorname{arccosh}(x) = \ln\left(x + \sqrt{x^2 - 1}\right)$. Since

$$1 + \frac{|i - z|^2}{2\Im(z)} \ge 1,$$

we have $\rho_{\mathbb{H}^2}(i,z) \geq 0$. Furthermore, since $\operatorname{arccosh}(x) = \ln\left(x+\sqrt{x^2-1}\right)$, we know that $\rho_{\mathbb{H}^2}(i,z)$ is again a strictly monotone increasing function.

Thus, it is equivalent to show

$$\frac{|i-\Lambda^2 r_0 e^{i\phi_0}|^2}{\Lambda^2} > |i-r_0 e^{i\phi_0}|^2.$$

By calculating $\frac{|i-\Lambda^2 r_0 e^{i\phi_0}|^2}{\Lambda^2}$, we can derive

$$\frac{|i-\Lambda^2 r_0 e^{i\phi_0}|^2}{\Lambda^2} = (\Lambda r_0)^2 - 2r_0\sin\phi_0 + \frac{1}{\Lambda^2}.$$

On the other hand,

$$|i-r_0 e^{i\phi_0}|^2 = r_0^2 - 2r_0\sin\phi_0 + 1.$$

We can find a trivial bound for r_0 by the unit, since $T^+ = 0$. Further, by directly using Euclidean metric, we can have another trivial but tighter bound for r_0:

$$r_0 < |Ti| + \frac{1-|Ti|}{2} = \frac{\Lambda^2+1}{2} < 1.$$

Since both $(\Lambda r_0)^2 - 2r_0\sin\phi_0 + \frac{1}{\Lambda^2}$ and $r_0^2 - 2r_0\sin\phi_0 + 1$ have $-2r_0\sin\phi_0$, we only have to compare the remaining terms.

Then we have

$$\begin{aligned}&\frac{1}{\Lambda^2} + \Lambda^2 r_0^2 - 1 - r_0^2\\ &= \left(1-\Lambda^2\right)\left(\frac{1}{\Lambda^2} - r_0^2\right)\end{aligned}$$

(recall $r_0 < \frac{\Lambda^2+1}{2}$)

$$> \left(1-\Lambda^2\right)\left(\frac{1}{\Lambda^2} - \frac{(1+\Lambda^2)^2}{4}\right) = \frac{(1-\Lambda^2)(4-\Lambda^2(1+\Lambda^2)^2)}{4\Lambda^2} > 0.$$

Because of $\Lambda < 1$, we can derive the last inequality. □

The third case is to compare $\rho_{\mathbb{B}^2}(0, TTRz_0)$ and $\rho_{\mathbb{B}^2}(0, TRTz_0)$, where $z_0 \in D_0(T)$. Since $TTRz_0 \in D_0(T^2) \setminus D_0(T^3)$, $TRTz_0 \in D_0(T) \setminus D_0(T^2)$, the two points $T(RTz_0)$ and $TT(Rz_0)$ are having the same starting point $z_0 \in D_0(T)$, hence $\rho_{\mathbb{B}^2}(0, TTRz_0) > \rho_{\mathbb{B}^2}(0, TRTz_0)$. In other words, when the number of T and R are the same, then by comparing the first few operators from the left before $z_0 = T_I 0$. That is, to compare $\rho_{\mathbb{B}^2}(0, T^n Rz_0)$ and $\rho_{\mathbb{B}^2}(0, T^m RT^l z_0)$, where $m+l = n, m \neq 0, l \neq 0$, we have

$$\rho_{\mathbb{B}^2}(0, T^n Rz_0) > \rho_{\mathbb{B}^2}(0, T^m RT^l z_0).$$

Finally, the fourth case is to compare with nodes with $R^{-k}, 1 \leq k \leq (2m-1)$. This case can be solved by applying the above results, since they are symmetric to the above cases. Thus, we have $\rho_{\mathbb{B}^2}(0, TR^{-1}z_0) = \rho_{\mathbb{B}^2}(0, TRz_0)$, and $\rho_{\mathbb{B}^2}(0, TRTR^{-1}z_0) > \rho_{\mathbb{B}^2}(0, TRTRz_0)$.

Lemma 3. *Let $T_{I^-} = T^{N-1}$, then the farthest node of $T_{I^-}T0 = T^N 0$ that shares the same parent node can be encoded as $T_{I^-}R^{m-1}T0$.*

Proof. Proof by induction. Consider the base case:

- If $T_{I^-} = T0$, then the farthest node of $T_{I^-}T$ is $TR^{m-1}T0$.
- Assume when $N = k$, $T_{I^-} = T^k 0$, the farthest node of $T_{I^-}T0 = T^k T0$ is $T^k R^{m-1}T0$.
- When $N = k+1$, $T_{I^-} = T^{k+1}0$, the farthest node of $T_{I^-}T0 = T^{k+1}T0$ is derived by mapping $R^{m-1}T0$ from $\mathcal{R}(\Gamma, R^{m-1}T)$ to $\mathcal{R}(\Gamma, T)$ by using the map T^{k+1}, i.e. such that $R^{m-1}T0$ is mapped into the isometric circle of T^{k+1}.

Hence, for the second level, $N = 2$, the farthest node of $T^2 0$ is always coded as

$$TR^{m-1}T0.$$

□

The next step is to find the general expression of the node that is the farthest node of the relative radial children node of $TR^{m-1}T0$.

The radial children node of $TR^{m-1}T0$ is simply an image from the radial path which is

$$TR^{m-1}T^20.$$

Based on this, we can derive the following lemma:

Lemma 4. *Let $T_{I^-} = T(R^{m-1}T)^{N-1}0$ be the parent node. Then, its relative radial children node is $T(R^{m-1}T)^{N-1}T0$, and the farthest children node of $T(R^{m-1}T)^{N-1}T0$ is $T(R^{m-1}T)^N0$.*

Proof. Proof by induction. Consider the base case:

- For $N = 2$, we have $T_{I^-} = T0$, then the farthest node of T^20 is $TR^{m-1}T0$.
- Assume when $N = k$ the farthest node of $T_{I^-} = T(R^{m-1}T)^{k-1}T0$ is $T(R^{m-1}T)^k0$.
- When $N = k + 1$, the farthest node of $T_{I^-} = T(R^{m-1}T)^kT0$ is derived by mapping $R^{m-1}T0$ from $\mathcal{R}(\Gamma, R^{m-1}T)$ to $\mathcal{R}(\Gamma, T)$ by using the map $T(R^{m-1}T)^k$, i.e. such that $R^{m-1}T0$ is mapped into the isometric circle of $T(R^{m-1}T)^k$. Hence, we have the claimed result.

□

By combining the above two claims, we have for the parent node T_I0, it has

$$T_IT0$$

to be its relative radial children node, and

$$T_I(R^{m-1}T)0$$

to be the farthest node of T_IT0.

Therefore, we have theorem 2.33. Examples of this theorem are figure and .

Theorem 1 (2.33)**.** *Let $\Gamma = \langle T_1, ..., T_m\rangle$ be the well-distributed Schottky group and be the special example as mentioned above, so we let $T_1 = T$. Let $\mathcal{R}(T)$ be the right critical region bounded by the radial path on the imaginary axis, boundary path in the fourth quadrant, and the unit disk.*

Let $S_I, S_J, S_K \in \mathcal{W}_N$ such that $S_I0 \in \mathcal{R}(T)$, $S_J0 \in \mathcal{R}(T)$, $S_K0 \in \mathcal{R}(T)$, and S_I0 is on the radial path, S_K0 is on any path between the radial and boundary path, and S_J0 is on the boundary path.

Then, we have

$$\rho_{\mathbb{B}^2}(0, S_I0) > \rho_{\mathbb{B}^2}(0, S_K0) > \rho_{\mathbb{B}^2}(0, S_J0).$$

Proof. Denote

$$\ell_I = \rho_{\mathbb{B}^2}(0, S_I0),$$
$$\ell_J = \rho_{\mathbb{B}^2}(0, S_J0),$$

and

$$\ell_K = \rho_{\mathbb{B}^2}(0, S_K0).$$

Recall for the group generator T_1 in Γ, we have the following radial path:

$$[0, T_10, T_1^20, ..., T_1^N0, ...].$$

For simplicity, let $T = T_1$ for the the following proof.

Next, we apply method 2 to reconstruct $\Gamma(0)$ by only using R and T.

Then, by applying lemma 3 and lemma 4, we can decode the expressions of all nodes on radial path and boundary path for any given level N of Γ, i.e. on radial path we have T^N0, and on the boundary path we have $T(R^{m-1}T)^{N-1}0$.

Then,

$$S_I0 = T^N0,$$
$$S_K0 = T(R^{m-1}T)^{N-1}0,$$

and

$$S_J0 = TR^{i_1}T...TR^{i_N}T0.$$

Furthermore, since we focused on the half-plane $D_0(T)$, we have N many T's, and $-(m-1) \le i_a \le (m-1)$.

The next step is by using lemma 1 to each pair of two nodes that we would like to compare between

$$\{S_I 0, S_J 0, S_K 0\}.$$

Then, by substituting z_0 to each pair of two nodes that we would like to compare repeatedly, we firstly have

$$\ell_I > \ell_K.$$

Next, since for S_J there exists at least one nonzero exponent i_a of R, lemma 1 implies

$$\ell_I > \ell_J.$$

Furthermore, by the proof of lemma 4 (by looking at other children nodes for each level N mapped from $\mathcal{R}(\Gamma, R^{m-1}T)$ to $\mathcal{R}(\Gamma, T)$ with the same parent node), we can see that as long as a node is not on the boundary path it will have the total number of exponent of R to be less than

$$(m-1)\cdot(N-1).$$

Therefore, by using lemma 1 again we have

$$\ell_J > \ell_K.$$

This completes the proof.

□

Definition 2. *Let N be fixed.*

- *Denote the node $S_{J_1}0$ where $S_{J_1} \in \mathcal{W}_N$ and $S_{J_1} \neq S_K$ and $S_{J_1}0$ is on the right of the relative radial path with respect to its parent node, then in the azimuth direction, i.e. fix the polar coordinate r within a range, and increase the angle θ from the point where the node $S_{J_1}0$ is located to $S_{J_2}0$, where $S_{J_2} \in \mathcal{W}_N$ and S_{J_2} is on the right of the relative radial path with respect to its parent node.*
- *Define the notation $S_{J_4}0$ for denoting the node where $S_{J_4} \in \mathcal{W}_N$ and $S_{J_4} \neq S_I$ and $S_{J_4}0$ is on the left of the relative radial path with respect to its parent node, then in the azimuth direction, i.e. fix the polar coordinate r within a range, and increase the angle θ from the point where the node $S_{J_4}0$ is located to $S_{J_3}0$, where $S_{J_3} \in \mathcal{W}_N$ and S_{J_3} is on the left of the relative radial path with respect to its parent node.*

Lemma 5. *Let $S_{J_1}, S_{J_2}, S_{J_3}$, and S_{J_4} be defined as above, then*

- $\rho_{\mathbb{B}^2}(0, S_{J_1}0) > \rho_{\mathbb{B}^2}(0, S_{J_2}0)$, *and*
- $\rho_{\mathbb{B}^2}(0, S_{J_4}0) > \rho_{\mathbb{B}^2}(0, S_{J_3}0)$.

Proof. This follows from the above lemmas and theorem. □

Let $\Gamma(0)$ be an orbit of a well-distributed Schottky group, then we can define the following important definitions:

Definition 3. *For each level N and for each node $T_I 0 \in \Gamma(0)$, where $T_I \in \mathcal{W}_N$, and $T_I 0 \in \mathcal{R}(T) \subset D_0(T)$, by connecting 0 to $T_I 0$ with a geodesic $[0, T_I 0]$, we can have:*

- *a hyperbolic length of the geodesic $\rho_{\mathbb{B}^2}(0, T_I 0)$, and*
- *an average of all $e^{-t\rho_{\mathbb{B}^2}(0,T_I 0)}$ terms in the Poincaré series*

$$E_N(t) = \frac{\sum_{T_I \in \mathcal{W}_N} e^{-t\rho_{\mathbb{B}^2}(0,T_I 0)}}{\#(\mathcal{W}_N)}$$

where

$$\#(\mathcal{W}_N) = (2m-1)^N.$$

First of all, with $E_N(t)$ to be defined we can have the original Poincaré series to be rewritten as

$$\mathcal{P}(\Gamma, t) = (2m)\sum_{N=1}^{\infty}(2m-1)^N\left(E_N(t)\right).$$

Secondly, we can partition the Poincaré series into two partial series by using $E_N(t)$. Consider the example when $\Lambda = 0.3, m = 2$, and $0 \leq N \leq 15$.

To make sure this example is a well-distributed Schottky group, it is only necessary to check whether $D_0(T)$ is bounded within the region $\frac{-3\pi}{2} - \frac{\pi}{2m} \leq \phi \leq \frac{-3\pi}{2} + \frac{\pi}{2m}$. To ensure this can be the case:

Definition 4. *Let* $B = \frac{2\Lambda^2}{\Lambda^4+1} + i\frac{\Lambda^4-1}{\Lambda^4+1}$ *be called the classification point of* Γ.

The formula of the classification point is derived by using the idea that the perpendicular bisector $M_0(T)$ is a semicircle center at real line in the upper half-plane model. Hence, by mapping everything back to $\mathbb{H}^2$ and then mapping everything back to $\mathbb{B}^2$, the Cartesian coordinates of B can solved which is $\left(\frac{2\Lambda^2}{\Lambda^4+1}, \frac{\Lambda^4-1}{\Lambda^4+1}\right)$.

Furthermore, to have $D_0(T)$ to be bounded within the region $\frac{-3\pi}{2} - \frac{\pi}{2m} \leq \phi \leq \frac{-3\pi}{2} + \frac{\pi}{2m}$, let $t = \frac{-\pi}{2} - \frac{\pi}{2m}$, a point on the unit circle is denoted by $K = e^{it}$.

A criterion to justify whether Γ is a well-distributed Schottky group, it is only necessary to check whether the Euclidean distance between $T0$ to K is greater than or equal to the Euclidean distance from $T0$ to B.

This method is implemented as the following code:

```
def classification_point(Lambda):
    return [float((2*Lambda**2)/(Lambda**4+1)),float((Lambda**4-1)/(Lambda**4+1))]

def check_T_generate_a_Schottky(Lambda,m):
    t=float(-math.pi/2)+float(math.pi/(2*m))
    K=Polar_complex_complex_to_Cartesian([1,t])
    B=classification_point(Lambda)
    T=operator_T(Lambda)
    T0=Cartesian_complex_divide(T[0][1],T[1][1])
    discriminant=float(Cartesian_complex_modulus(Cartesian_complex_add(K,
    Cartesian_complex_scalar_mul(-1,T0)))**2)-
    float(Cartesian_complex_modulus(Cartesian_complex_add(B,
    Cartesian_complex_scalar_mul(-1,T0)))**2)
    print(discriminant)
    if discriminant>0:
        return True
    else:
        return False
```

By plotting out $\Gamma(0) \cap D_0(T)$ for $0 \leq N \leq 15$, we can obtaing the following:

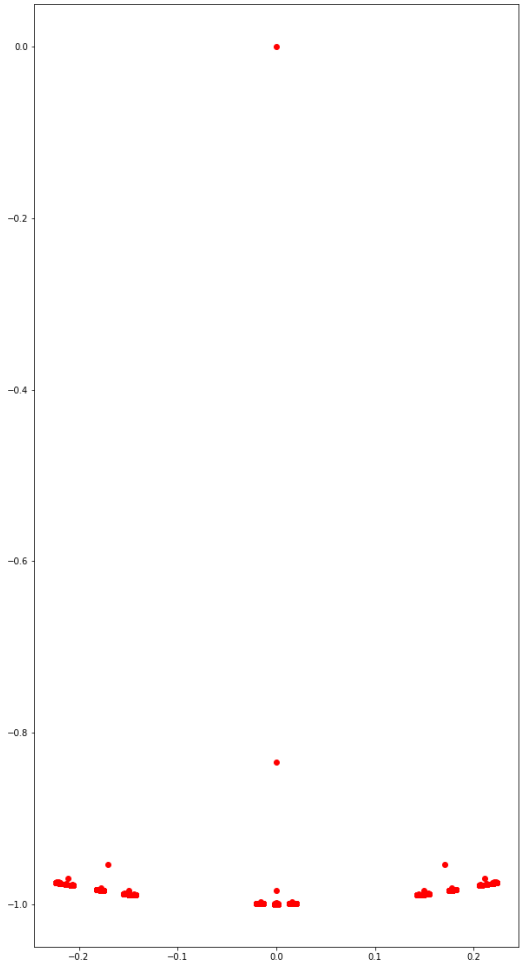

FIGURE 3. A figure for the example with $\Lambda = 0.3, m = 2, 0 \leq N \leq 15$.

Now, instead of accumulating nodes in each level, let us only focus on one specific level, say $N = 15$, then we have the following figure:

FIGURE 4. A figure for the example with $\Lambda = 0.3, m = 2, N = 15$.

The above figure can be transformed into the following figure by collecting all of the geodesic $[0, T_I 0]$, then plotted them to the $\rho - x$ diagram, i.e. the y axis is the hyperbolic length, and each of $[0, T_I 0]$ is assigned an $x-$coordinated on a positive integer in the order from left to right in the above figure (since y-axis is a symmetric axis of the figure, from right to left will give an identical result): **Example of the theorem 1.**

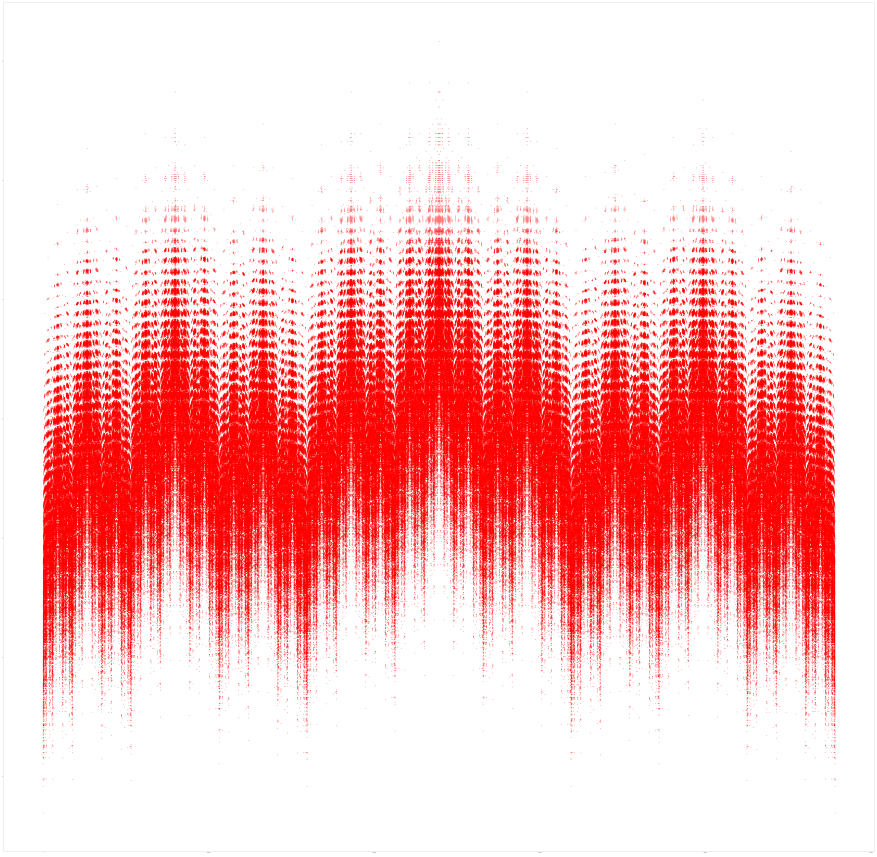

FIGURE 5. A figure for the example with $\Lambda = 0.3, m = 2, N = 6$.

Please mind that the difference of $x-$coordinate between any tow dots in the above figure is at least 1. The following example can illustrate this by plotting the $N = 6$ level for the same orbit:

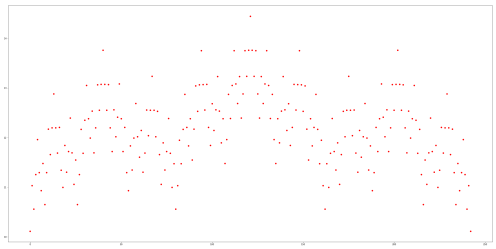

FIGURE 6. A figure for the example with $\Lambda = 0.3, m = 2, N = 6$.

Next, taking negative exponential on all of these hyperbolic length of geodesics as $e^{-t\rho_{\mathbb{B}^2}(0,T_I0)}$, and then computing the average of all of them, we have

$$E'_N(t).$$

Using $E'_N(t)$ as a criterion, we can collect all $e^{-t\rho_{\mathbb{B}^2}(0,T_I0)}$ that is greater than $E'_N(t)$. Then, we can define the following dictionary:

$$\mathcal{D}_N := \left\{ T_I \in \mathcal{W}_N : e^{-t\rho_{\mathbb{B}^2}(0,T_I0)} > E'_N(t) \right\}.$$

Next, let us still focus on the example with $t = 0.25, \Lambda = 0.3, m = 2$, and $0 \leq N \leq 15$ being specified. We know that the function $^{-tx}, x \in (0,1)$ is concave downward and strictly monotone decreasing when $t \in \left(, \frac{1}{2}\right)$, so as long as the value of t is within this range, we can derive the same conclusion in the following.

Firstly, by using the function (defined in the appendix)

```
examples_of_10000_with_t(initial=0.3, increment=0.00001,t0=0.25)
```

we can generate more than ten thousands examples in about one day to back theorem 1, and the results follows the theorem. For instance,

denote $E_N(t) := e^{-t\rho'_N}$ to be the average of all $e^{-t\rho_{\mathbb{B}^2}(0,T_I0)}$ of this level, and denote $e^{-t\rho_N}$ to be the average of all $e^{-t\rho_{\mathbb{B}^2}(0,T_I0)}$ of this level that is greater than the value $E_N(t)$, then starting from $N = 2$, we can obtain:

N	occurrence	$e^{-t\rho_N}$	$e^{-t\rho'_N}$	$\frac{e^{-t\rho_N}}{e^{-t\rho'_N}}$
2	1	0.355348548685	0.336899032457	1.05476274626
3	5	0.232076246076	0.336899032457	1.11535722686
4	15	0.145451898381	0.208073467841	1.13151123142
5	45	0.0917719929372	0.128546579426	1.15557243238
6	129	0.0571998712962	0.0794169109314	1.16581334503
7	393	0.0359208806166	0.0490643476848	1.18502597371
8	1201	0.0225608089023	0.0303123150155	1.20470990885
9	3573	0.0140744987991	0.0187271713602	1.21648756768
10	10651	0.00877894665843	0.0115697843308	1.22818578622
11	32567	0.00550382089226	0.00714789794584	1.24632947024
12	98757	0.00344314800678	0.00441602403191	1.26203425237
13	298293	0.00215103171706	0.00272825242287	1.27617105167
14	894987	0.00134081593204	0.00168553558259	1.28758525877
...	...	...	...	...

It is obvious that the number of occurrences of each level is growing with a rate at least equals $3 = (2m-1)$ (that is, the number of occurrences= a constant $\cdot\, 3^N$), and the ratio of $\frac{e^{-t\rho_N}}{e^{-t\rho'_N}}$ is also growing exponentially.

By using the Extremal Principal, one can check that it is impossible to balance the partition using the above process without the number of occurrences growing exponentially.

Results from the properties of $e^{-t\rho_N}$, $e^{-t\rho'_N}$, $\frac{e^{-t\rho_N}}{e^{-t\rho'_N}}$, theorem 1, and the set-up of the well-distributed Schottky group, the ratio of $\frac{e^{-t\rho_N}}{e^{-t\rho'_N}}$ is going to infinity, and the number of occurrences of each level is always at least a constant $\cdot\, 3^N =$ a constant $\cdot\, (2m-1)^N$.

Notice that $e^{-t\rho_N} > e^{-t\rho'_N}$ for all $N \in \mathbb{Z}^+$. Recall by the definition of the number of occurrences of each level, i.e. it is always less than or equals to $(2m-1)^N$, although it grows at least as a constant $\cdot$ $3^N =$ a constant $\cdot\, (2m-1)^N$, hence the denominator of $e^{-t\rho_N}$ is smaller than the $e^{-t\rho'_N}$. Secondly, for the numerator, since the occurrences are counted as a $e^{-t\rho_{\mathbb{B}^2}(0,T_I 0)} > E_N(t)$ happens, so this two reasons results in $e^{-t\rho_N} > e^{-t\rho'_N}$ for all $N \in \mathbb{Z}^+$.

For the following development, denote the number of occurrences by

$$\#\left(\mathcal{D}_N\right)$$

where $\mathcal{D}_N$ is the set collects all reduced words $T_I \in \mathcal{W}_N$ such that $e^{-t\rho_{\mathbb{B}^2}(0,T_I 0)} > E_N(t) = e^{-t\rho'_N}$.

We also have

$$\#\left(\mathcal{D}_N\right) \geq \text{a constant} \cdot (2m-1)^N$$

Recall the definition of the exponent of convergent δ:

$$\delta = \inf\left\{t > 0 : \mathcal{P}(\Gamma, t) < \infty\right\}.$$

By generating a contradiction that is due to a false assumption, we can prove the following lemma.

Lemma 6. *If $\delta(\Gamma) < \delta_{max}$, then $\mathcal{P}(\Gamma, t)|_{t>\delta} \to \infty$.*

Proof. Assume to the contrary, we have $\delta < \delta_{\max}$.

Step 1. Since with $t \to \delta$, by the definition of δ, we have the Poincaré series $\mathcal{P}(\Gamma, t) < \infty$. This implies a partial series of $\mathcal{P}$ is

$$P(\Gamma, t) = \sum_{N=1}^{\infty} (2m-1)^N e^{-t\rho'_N} < \infty,$$

where

$$\mathcal{P}(\Gamma, t) = 2mP(\Gamma, t).$$

Step 2. Using words in $\mathcal{D}_N$, we can have one more infinite series which is a partial series of $\mathcal{P}$:

$$P'(\Gamma, t) = \sum_N \mathcal{D}_N e^{-t\rho_N}.$$

Since we have $\#(\mathcal{D}_N) \geq$ a constant $\cdot\, (2m-1)^N$, and we only focus on the convergent behavior of $P'(t)$ and the Poincaré series $\mathcal{P}$, so a partial series of P' after relabeling with N is

$$P''(\Gamma, t) = \sum_N (2m-1)^N e^{-t\rho_N}.$$

Step 3. Please mind that there is a difference in the form of this type of functions between $P(\Gamma, t)$ and $P''(\Gamma, t)$ which is

$$e^{-t\rho_N} > e^{-t\rho'_N}, \forall \mathbb{Z}^+.$$

By the definition of $\delta(\Gamma)$, we have both $P(\Gamma, t) < \infty$ and $P''(\Gamma, t) < \infty$ as $t \to \delta(\Gamma)$, meaning that $\delta(\Gamma)$ is the infimum such that $P(\Gamma, t) < \infty$ and $P''(\Gamma, t) < \infty$.

However, looking at both of the functions $P''(\Gamma, t)$ and $P(\Gamma, t)$, we know that although they are the same type of the functions with the form

$$f(x) = \sum_N (2m-1)^N e^{-tx}, x \in \mathbb{R}_{\geq 0},$$

there is an intrinsic distinction between them based on the previous step. Hence, based on this intrinsic distinction

$$e^{-t\rho_N} > e^{-t\rho'_N}, \forall \mathbb{Z}^+,$$

meaning that

$$t\rho_N < t\rho'_N.$$

Recall that $P(\Gamma, t)$ and $P''(\Gamma, t)$ are the same type of functions, since both of them have the form such as

$$f(x) = \sum_N (2m-1)^N e^{-tx}, x \in \mathbb{R}_{\geq 0},$$

if $\delta(\Gamma)$ is the exponent of convergence of $P(\Gamma, t)$, then it can also be the the exponent of convergence of $P(\Gamma, t)$ if $P(\Gamma, t) = P''(\Gamma, t)$, i.e. if

$$t\rho_N = t\rho'_N$$

which contradicts to the fact that

$$\rho_N < \rho'_N, \forall \mathbb{Z}^+.$$

□

Remark: This contradiction can be solved when we increase $\delta(\Gamma)$ until it becomes $\delta(\Gamma)'' = \delta(\Gamma)_{\text{max.}}$, since at that value for all $t > \delta(\Gamma)_{\text{max.}}$. If $\delta(\Gamma)$ is the exponent of convergence of $P(\Gamma, t)$, then we ought to have a larger exponent of convergence for $P''(\Gamma, t)$, say $\delta(\Gamma)''$ such that

$$\delta(\Gamma)''\rho_N = \delta(\Gamma)\rho'_N, \forall \mathbb{Z}^+,$$

and

$$\delta(\Gamma)'' > \delta(\Gamma),$$

thus

$$\delta(\Gamma)'' \neq \delta(\Gamma).$$

Any value smaller than that can lead to the same argument showed above. Though we still have $\delta(\Gamma)_{\text{max.}}\rho_N < \delta(\Gamma)_{\text{max.}}\rho'_N$, with $\delta = \delta(\Gamma)_{\text{max.}}$ we will have not only $P(\Gamma, t) < \infty$, but also $P''(\Gamma, t) < \infty$.

It follows the lemma that we have

$$\delta(\Gamma) = \frac{\ln(2m-1)}{\ln\left(\lim_{N\to\infty}(\rho_{\mathbb{B}^2}(0, T(R^{m-1}T)^N 0) - \rho_{\mathbb{B}^2}(0, T(R^{m-1}T)^{N-1}0))\right)}.$$

Furthermore, the value $(R^{m-1}T)^{N-1}0$ and $(R^{m-1}T)^N 0$ can be easily found by using the following lemma:

Lemma 7. *Let* $T \in PSU(1,1), T := \begin{pmatrix} \alpha & \overline{\gamma} \\ \gamma & \overline{\alpha} \end{pmatrix}$, *and let* $A := \sqrt{\alpha^2 - 2|\alpha|^2 + \overline{\alpha}^2 + 4|\gamma|^2}$. *Then,*

$$T^N = \frac{1}{2^{N+1}A}\begin{pmatrix} \left((\alpha+\overline{\alpha}-A)^N(-\alpha+\overline{\alpha}+A) + (\alpha+\overline{\alpha}+A)^N(A+\alpha-\overline{\alpha})\right) & 2\overline{\gamma}\left((\alpha+\overline{\alpha}+A)^N - (\alpha+\overline{\alpha}-A)^N\right) \\ 2\gamma\left((\alpha+\overline{\alpha}+A)^N - (\alpha+\overline{\alpha}-A)^N\right) & \left((\alpha+\overline{\alpha}-A)^N(\alpha-\overline{\alpha}+A) + (\alpha+\overline{\alpha}+A)^N(A-\alpha+\overline{\alpha})\right) \end{pmatrix},$$

for all $N \in \mathbb{Z}^+$.

Proof. Proof by induction.

When $N = 1$, we have

$$\begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix},$$

where

$$T_{11} = \frac{(\alpha+\overline{\alpha}-A)(-\alpha+\overline{\alpha}+A) + (\alpha+\overline{\alpha}+A)(\alpha-\overline{\alpha}+A)}{4A} = \alpha,$$

$$T_{21} = \frac{\gamma((\alpha+\overline{\alpha}+A) - (\alpha+\overline{\alpha}-A))}{2A} = \gamma,$$

$$T_{12} = \frac{\overline{\gamma}((\alpha+\overline{\alpha}+A) - (\alpha+\overline{\alpha}-A))}{2A} = \overline{\gamma},$$

and

$$T_{22} = \frac{(\alpha+\overline{\alpha}-A)(\alpha-\overline{\alpha}+A) + (\alpha+\overline{\alpha}+A)(-\alpha+\overline{\alpha}+A)}{4A} = \alpha.$$

Therefore, we have

$$\begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} = T.$$

Secondly, assume when $N = n$ we have the claimed result. Then when $N = n + 1$ we have

$$T^{n+1} = T^n \circ T = \frac{1}{2^{n+1}A}\begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$$

where

$$t_{11} = \alpha\left((\alpha+\overline{\alpha}-A)^n(-\alpha+\overline{\alpha}+A)+(\alpha+\overline{\alpha}+A)^n(\alpha-\overline{\alpha}+A)\right)+2|\gamma|^2((\alpha+\overline{\alpha}+A)^n-(\alpha+\overline{\alpha}-A)^n),$$
$$t_{21} = 2\gamma\alpha((\alpha+\overline{\alpha}+A)^n-(\alpha+\overline{\alpha}-A)^n)+\gamma((\alpha+\overline{\alpha}-A)^n(\alpha-\overline{\alpha}+A)+(\alpha+\overline{\alpha}+A)^n(-\alpha+\overline{\alpha}+A)),$$
$$t_{12} = \overline{\gamma}\left((\alpha+\overline{\alpha}-A)^n(-\alpha+\overline{\alpha}+A)+(\alpha+\overline{\alpha}+A)^n(\alpha-\overline{\alpha}+A)\right)+2\overline{\gamma\alpha}((\alpha+\overline{\alpha}+A)^n-(\alpha+\overline{\alpha}-A)^n),$$

and

$$t_{22} = 2|\gamma|^2((\alpha+\overline{\alpha}+A)^n-(\alpha+\overline{\alpha}-A)^n)+\overline{\alpha}((\alpha+\overline{\alpha}-A)^n(\alpha-\overline{\alpha}+A)+(\alpha+\overline{\alpha}+A)^n(-\alpha+\overline{\alpha}+A)).$$

Finally, after a few cancellations, the above results lead to

$$\frac{1}{2^{n+1}A}\begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$$

$$= \frac{1}{2^{n+2}A}\begin{pmatrix} \left((\alpha+\overline{\alpha}-A)^{n+1}(-\alpha+\overline{\alpha}+A)+(\alpha+\overline{\alpha}+A)^{n+1}(A+\alpha-\overline{\alpha})\right) & 2\overline{\gamma}\left((\alpha+\overline{\alpha}+A)^{n+1}-(\alpha+\overline{\alpha}-A)^{n+1}\right) \\ 2\gamma\left((\alpha+\overline{\alpha}+A)^{n+1}-(\alpha+\overline{\alpha}-A)^{n+1}\right) & \left((\alpha+\overline{\alpha}-A)^{n+1}(\alpha-\overline{\alpha}+A)+(\alpha+\overline{\alpha}+A)^{N}(A-\alpha+\overline{\alpha})\right) \end{pmatrix}$$

□

Based on the above lemma, $|T_I^N 0|, T_I \in \mathcal{W}_N$ can always be easily computed for all $N \in \mathbb{Z}^+$.

1. Appendix

```
__author__="William Huanshan Chuang"
__version__="1.0.1"
__email__="wchuang2@mail.sfsu.edu"

#define elementary functions
import math
import statistics
def Cartesian_complex_mul(numb1,numb2):
    #numb1 a+bi, [a,b], numb2 c+di, [c,d]
    x=numb1[0]*numb2[0]-numb1[1]*numb2[1]
    y=numb1[0]*numb2[1]+numb1[1]*numb2[0]
    return [x,y]

def Cartesian_complex_scalar_mul(alpha,numb1):
    #numb1 a+bi, [a,b], numb2 c+di, [c,d]
    x=numb1[0]*alpha
    y=numb1[1]*alpha
    return [x,y]

def Cartesian_complex_add(numb1,numb2):
    #numb1 a+bi, [a,b], numb2 c+di, [c,d]
    x=numb1[0]+numb2[0]
    y=numb1[1]+numb2[1]
    return [x,y]

def Cartesian_complex_divide(numb1,numb2):
    #numb1 u+vi, [u,v], numb2 x+yi, [x,y]
    d=numb2[0]*numb2[0]+numb2[1]*numb2[1]
    nx=numb1[0]*numb2[0]+numb1[1]*numb2[1]
    ny=numb1[1]*numb2[0]-numb1[0]*numb2[1]
    X=float(nx/d)
    Y=float(ny/d)
    return[X,Y]

def Cartesian_complex_modulus(numb):
    return math.sqrt(numb[0]*numb[0]+numb[1]*numb[1])

def Cartesian_complex_complex_conjugate(numb):
    return [numb[0],-numb[1]]

def Cartesian_complex_complex_to_polar(numb):
    r=Cartesian_complex_modulus(numb)
```

```
    if numb[0]>0:
        t=math.atan(float(numb[1]/numb[0]))
    elif numb[0]<0:
        t=math.atan(float(numb[1]/numb[0]))+math.pi
    else:
        if numb[1]>0:
            t=float(math.pi/2)
        elif numb[1]<0:
            t=float(-math.pi/2)
        else:
            t="null"
    return [r,t]

def Polar_complex_complex_to_Cartesian(numb):
    return [numb[0]*math.cos(numb[1]),numb[0]*math.sin(numb[1])]

def Polar_complex_conjugate(numb):
    return [numb[0],-numb[1]]

def Polar_complex_mul(numb1,numb2):
    #numb1 r_1e^(it_1), [r_1,t_1], numb2 r_2e^(it_2), [r_2,t_2]
    r=numb1[0]*numb2[0]
    t=numb1[1]+numb2[1]
    return [r,t]

def Polar_complex_divide(numb1,numb2):
    #numb1 r_1e^(it_1), [r_1,t_1], numb2 r_2e^(it_2), [r_2,t_2]
    r=float(numb1[0]/numb2[0])
    t=numb1[1]-numb2[1]
    return [r,t]

def Polar_complex_add(numb1,numb2):
    N1=Polar_complex_complex_to_Cartesian(numb1)
    N2=Polar_complex_complex_to_Cartesian(numb2)
    tot=Cartesian_complex_add(N1,N2)
    return Cartesian_complex_complex_to_polar(tot)

def real_matrix_addition(m1,m2):
    #m1=[[M11,M12],[M21,M22]]
    a=m1[0][0]+m2[0][0]
    b=m1[0][1]+m2[0][1]
    c=m1[1][0]+m2[1][0]
    d=m1[1][1]+m2[1][1]
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l

def Cartesian_complex_matrix_addition(m1,m2):
    #m1=[[M11,M12],[M21,M22]]
    #M11=[a,b]
    a1=m1[0][0][0]+m2[0][0][0]
    a2=m1[0][0][1]+m2[0][0][1]
    b1=m1[0][1][0]+m2[0][1][0]
    b2=m1[0][1][1]+m2[0][1][1]
    c1=m1[1][0][0]+m2[1][0][0]
    c2=m1[1][0][1]+m2[1][0][1]
    d1=m1[1][1][0]+m2[1][1][0]
    d2=m1[1][1][1]+m2[1][1][1]
    a=[a1,a2]
    b=[b1,b2]
    c=[c1,c2]
    d=[d1,d2]
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l
```

```
def real_matrix_multiplication(m1,m2):
    #m1=[[M11,M12],[M21,M22]]
    a=m1[0][0]*m2[0][0]+m1[0][1]*m2[1][0]
    b=m1[0][0]*m2[0][1]+m1[0][1]*m2[1][1]
    c=m1[1][0]*m2[0][0]+m1[1][1]*m2[1][0]
    d=m1[1][0]*m2[0][1]+m1[1][1]*m2[1][1]
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l

def Cartesian_complex_matrix_multiplication(m1,m2):
    #m1=[[M11,M12],[M21,M22]]
    #M11=[a,b]
    a=Cartesian_complex_add(Cartesian_complex_mul(m1[0][0],m2[0][0]),
    Cartesian_complex_mul(m1[0][1],m2[1][0]))
    b=Cartesian_complex_add(Cartesian_complex_mul(m1[0][0],m2[0][1]),
    Cartesian_complex_mul(m1[0][1],m2[1][1]))
    c=Cartesian_complex_add(Cartesian_complex_mul(m1[1][0],m2[0][0]),
    Cartesian_complex_mul(m1[1][1],m2[1][0]))
    d=Cartesian_complex_add(Cartesian_complex_mul(m1[1][0],m2[0][1]),
    Cartesian_complex_mul(m1[1][1],m2[1][1]))
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l

def real_matrix_inverse(m1):
    #m1=[[M11,M12],[M21,M22]]
    det=m1[0][0]*m1[1][1]-m1[0][1]*m1[1][0]
    a=float(m1[1][1]/det)
    b=float(-m1[0][1]/det)
    c=float(-m1[1][0]/det)
    d=float(m1[0][0]/det)
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l

def Cartesian_complex_matrix_inverse(m1):
    #m1=[[M11,M12],[M21,M22]]
    #M11=[a,b]
    det=Cartesian_complex_add(Cartesian_complex_mul(m1[0][0],m1[1][1]),
    Cartesian_complex_scalar_mul(-1,Cartesian_complex_mul(m1[0][1],m1[1][0])))
    inverse_det=Cartesian_complex_divide([1,0],det)
    a=Cartesian_complex_mul(m1[1][1],inverse_det)
    b=Cartesian_complex_mul(Cartesian_complex_scalar_mul(-1,m1[0][1]),inverse_det)
    c=Cartesian_complex_mul(Cartesian_complex_scalar_mul(-1,m1[1][0]),inverse_det)
    d=Cartesian_complex_mul(m1[0][0],inverse_det)
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l


def Cartesian_radial_hyperbolic_distance(z):
    r=float(Cartesian_complex_modulus(z))
    return math.log(float((1+r)/(1-r)))

def operator_T(Lambda):
    D=2
    a1=(-Lambda-float(1/Lambda))*float(-0.5)
    a2=0
    b1=0
```

```
    b2=(-Lambda+float(1/Lambda))*float(-0.5)
    c1=0
    c2=(Lambda-float(1/Lambda))*float(-0.5)
    d1=(-Lambda-float(1/Lambda))*float(-0.5)
    d2=0
    l1=[[a1,a2],[b1,b2]]
    l2=[[c1,c2],[d1,d2]]
    l=[]
    l.append(l1)
    l.append(l2)
    return l

def operator_R(theta):
    a=Polar_complex_complex_to_Cartesian([1,float(0.5*theta)])
    b=[0,0]
    c=[0,0]
    d=Polar_complex_complex_to_Cartesian([1,float(-0.5*theta)])
    l1=[a,b]
    l2=[c,d]
    l=[]
    l.append(l1)
    l.append(l2)
    return l

def classification_point(Lambda):
    return [float((2*Lambda**2)/(Lambda**4+1)),float((Lambda**4-1)/(Lambda**4+1))]

def check_T_generate_a_Schottky(Lambda,m):
    t=float(-math.pi/2)+float(math.pi/(2*m))
    K=Polar_complex_complex_to_Cartesian([1,t])
    B=classification_point(Lambda)
    T=operator_T(Lambda)
    T0=Cartesian_complex_divide(T[0][1],T[1][1])
    discriminant=float(Cartesian_complex_modulus(Cartesian_complex_add(K
    ,
    Cartesian_complex_scalar_mul(-1,T0)))**2)
    -float(Cartesian_complex_modulus(Cartesian_complex_add(B,
    Cartesian_complex_scalar_mul(-1,T0)))**2)
    print(discriminant)
    if discriminant>0:
        return True
    else:
        return False

def Tz(T,z):
    a=T[0][0][0]
    b=T[0][0][1]
    c=T[1][0][0]
    d=T[1][0][1]
    x=z[0]
    y=z[1]
    A=float(a*x+c-b*y)
    B=float(-d+b*x+a*y)
    C=float(c*x+a-d*y)
    D=float(-b+d*x+c*y)
    return [float((C*A+B*D)/(C**2+D**2)),float((B*C-D*A)/(C**2+D**2))]


#Generate the orbit Gamma(0)
def Gamma0(Lambda,m,N):
    T=operator_T(Lambda)
    theta=float(math.pi/m)
    R=operator_R(theta)
    L=[Tz(T,[0,0])]
    tmp1=[]
    tmp2=[]
    nodes_in_DT=[[0,0]]
    j=1
    while j<=N:
        #N=1
```

```
        if j==1:
            z=L[0]
            tmp1=[]
            tmp2=[]
            for i in range(2*m-1):
                z=Tz(R,z)
                tmp1.append(z)
                if i!=m-1:
                    tmp2.append(z)
                else:
                    tmp2.append(L[0])
            L=[]
            for k in tmp2:
                L.append(k)
            nodes_in_DT=[Tz(T,[0,0])]

        #N>1
        else:
            nodes_in_DT=[]
            tmp1=[]
            tmp2=[]
            tmp3=[]
            for k in L:
                z=Tz(T,k)
                nodes_in_DT.append(z)
                tmp1.append(z)
                tmp2.append(z)
            L=[]
            for i in range(2*m-1):
                if i!=m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                        L.append(k)
                    tmp3=[]
                elif i==m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                    tmp3=[]
                    for k in tmp2:
                        L.append(k)

        j+=1
    return nodes_in_DT
# measuring hyperbolic distance
def Hyperbolic_Distance_Gamma0(Lambda,m,N):
    T=operator_T(Lambda)
    theta=float(math.pi/m)
    R=operator_R(theta)
    L=[Tz(T,[0,0])]
    tmp1=[]
    tmp2=[]
    nodes_in_DT=[[0,0]]
    j=1
    while j<=N:
        #N=1
        if j==1:
            z=L[0]
            tmp1=[]
            tmp2=[]
            for i in range(2*m-1):
                z=Tz(R,z)
                tmp1.append(z)
                if i!=m-1:
                    tmp2.append(z)
                else:
```

```
                tmp2.append(L[0])
            L=[]
            for k in tmp2:
                L.append(k)
            nodes_in_DT=[Tz(T,[0,0])]

        #N>1
        else:
            nodes_in_DT=[]
            tmp1=[]
            tmp2=[]
            tmp3=[]
            for k in L:
                z=Tz(T,k)
                nodes_in_DT.append(z)
                tmp1.append(z)
                tmp2.append(z)
            L=[]
            for i in range(2*m-1):
                if i!=m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                        L.append(k)
                    tmp3=[]
                elif i==m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                    tmp3=[]
                    for k in tmp2:
                        L.append(k)

        j+=1
    Hyperbolic_distance=[]
    for k in nodes_in_DT:
        Hyperbolic_distance.append(Cartesian_radial_hyperbolic_distance(k))
    return Hyperbolic_distance

# measuring Exp(-hyperbolic distance)
def Exp_negative_Hyperbolic_Distance_Gamma0(Lambda,m,N):
    T=operator_T(Lambda)
    theta=float(math.pi/m)
    R=operator_R(theta)
    L=[Tz(T,[0,0])]
    tmp1=[]
    tmp2=[]
    nodes_in_DT=[[0,0]]
    j=1
    while j<=N:
        #N=1
        if j==1:
            z=L[0]
            tmp1=[]
            tmp2=[]
            for i in range(2*m-1):
                z=Tz(R,z)
                tmp1.append(z)
                if i!=m-1:
                    tmp2.append(z)
                else:
                    tmp2.append(L[0])
            L=[]
            for k in tmp2:
                L.append(k)
            nodes_in_DT=[Tz(T,[0,0])]
```

```
            #N>1
            else:
                nodes_in_DT=[]
                tmp1=[]
                tmp2=[]
                tmp3=[]
                for k in L:
                    z=Tz(T,k)
                    nodes_in_DT.append(z)
                    tmp1.append(z)
                    tmp2.append(z)
                L=[]
                for i in range(2*m-1):
                    if i!=m-1:
                        for k in tmp1:
                            tmp3.append(Tz(R,k))
                        tmp1=[]
                        for k in tmp3:
                            tmp1.append(k)
                            L.append(k)
                        tmp3=[]
                    elif i==m-1:
                        for k in tmp1:
                            tmp3.append(Tz(R,k))
                        tmp1=[]
                        for k in tmp3:
                            tmp1.append(k)
                        tmp3=[]
                        for k in tmp2:
                            L.append(k)

            j+=1
        Hyperbolic_distance=[]
        for k in nodes_in_DT:
            Hyperbolic_distance.append(math.exp(-Cartesian_radial_hyperbolic_distance(k)))
        return Hyperbolic_distance

# measuring Exp(-hyperbolic distance)
def Exp_negative_Hyperbolic_Distance_Gamma0_with_t(Lambda,m,N,t):
    T=operator_T(Lambda)
    theta=float(math.pi/m)
    R=operator_R(theta)
    L=[Tz(T,[0,0])]
    tmp1=[]
    tmp2=[]
    nodes_in_DT=[[0,0]]
    j=1
    while j<=N:
        #N=1
        if j==1:
            z=L[0]
            tmp1=[]
            tmp2=[]
            for i in range(2*m-1):
                z=Tz(R,z)
                tmp1.append(z)
                if i!=m-1:
                    tmp2.append(z)
                else:
                    tmp2.append(L[0])
            L=[]
            for k in tmp2:
                L.append(k)
            nodes_in_DT=[Tz(T,[0,0])]

        #N>1
        else:
            nodes_in_DT=[]
            tmp1=[]
            tmp2=[]
            tmp3=[]
```

```
            for k in L:
                z=Tz(T,k)
                nodes_in_DT.append(z)
                tmp1.append(z)
                tmp2.append(z)
            L=[]
            for i in range(2*m-1):
                if i!=m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                        L.append(k)
                    tmp3=[]
                elif i==m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                    tmp3=[]
                    for k in tmp2:
                        L.append(k)

        j+=1
    Hyperbolic_distance=[]
    for k in nodes_in_DT:
        Hyperbolic_distance.append(math.exp(-t*Cartesian_radial_hyperbolic_distance(k)))
    return Hyperbolic_distance

# measuring Exp(-t*(hyperbolic distance))
def Improved_Exp_negative_Hyperbolic_Distance_Gamma0_with_t(Lambda,m,N,L,t):

    T=operator_T(Lambda)
    theta=float(math.pi/m)
    R=operator_R(theta)
    if len(L)==0:
        L=[Tz(T,[0,0])]
        j=1
    else:
        j=N
    tmp1=[]
    tmp2=[]
    nodes_in_DT=[[0,0]]

    while j<=N:
        #N=1
        if j==1:
            z=L[0]
            tmp1=[]
            tmp2=[]
            for i in range(2*m-1):
                z=Tz(R,z)
                tmp1.append(z)
                if i!=m-1:
                    tmp2.append(z)
                else:
                    tmp2.append(L[0])
            L=[]
            for k in tmp2:
                L.append(k)
            nodes_in_DT=[Tz(T,[0,0])]

        #N>1
        else:
            nodes_in_DT=[]
            tmp1=[]
            tmp2=[]
            tmp3=[]
            for k in L:
```

```
                z=Tz(T,k)
                nodes_in_DT.append(z)
                tmp1.append(z)
                tmp2.append(z)
            L=[]
            for i in range(2*m-1):
                if i!=m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                        L.append(k)
                    tmp3=[]
                elif i==m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                    tmp3=[]
                    for k in tmp2:
                        L.append(k)

        j+=1
    Hyperbolic_distance=[]
    for k in nodes_in_DT:
        Hyperbolic_distance.append(math.exp(-t*Cartesian_radial_hyperbolic_distance(k)))
    return [Hyperbolic_distance,L]


# measuring Exp(-hyperbolic distance)
def Improved_Exp_negative_Hyperbolic_Distance_Gamma0(Lambda,m,N,L):

    T=operator_T(Lambda)
    theta=float(math.pi/m)
    R=operator_R(theta)
    if len(L)==0:
        L=[Tz(T,[0,0])]
        j=1
    else:
        j=N
    tmp1=[]
    tmp2=[]
    nodes_in_DT=[[0,0]]

    while j<=N:
        #N=1
        if j==1:
            z=L[0]
            tmp1=[]
            tmp2=[]
            for i in range(2*m-1):
                z=Tz(R,z)
                tmp1.append(z)
                if i!=m-1:
                    tmp2.append(z)
                else:
                    tmp2.append(L[0])
            L=[]
            for k in tmp2:
                L.append(k)
            nodes_in_DT=[Tz(T,[0,0])]

        #N>1
        else:
            nodes_in_DT=[]
            tmp1=[]
            tmp2=[]
            tmp3=[]
```

```
            for k in L:
                z=Tz(T,k)
                nodes_in_DT.append(z)
                tmp1.append(z)
                tmp2.append(z)
            L=[]
            for i in range(2*m-1):
                if i!=m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                        L.append(k)
                    tmp3=[]
                elif i==m-1:
                    for k in tmp1:
                        tmp3.append(Tz(R,k))
                    tmp1=[]
                    for k in tmp3:
                        tmp1.append(k)
                    tmp3=[]
                    for k in tmp2:
                        L.append(k)

        j+=1
    Hyperbolic_distance=[]
    for k in nodes_in_DT:
        Hyperbolic_distance.append(math.exp(-Cartesian_radial_hyperbolic_distance(k)))
    return [Hyperbolic_distance,L]


def examples_of_10000(initial, increment):
    counter=initial
    useful_example=0
    while counter < initial+10000*increment:
        print("************"+str(counter)+"************")
        out=[]
        if check_T_generate_a_Schottky(Lambda=counter,m=2):
            try:
                rho=[]
                for i in range(15):
                    sum1=0
                    sum2=0
                    test=Exp_negative_Hyperbolic_Distance_Gamma0(Lambda=counter,m=2,N=i)
                    ave_of_all_exp_of_negative_rho_of_this_level=statistics.
                    mean(test)
                    occurence=0
                    tmp=[]
                    for node in test:
                        if node < ave_of_all_exp_of_negative_rho_of_this_level:
                            occurence+=1
                        else:
                            tmp.append(node)
                    print("N="+str(i))
                    print("occurence="+str(occurence))
                    if len(tmp)!=0:
                        ave_of_all_short_exp_of_negative_rho_of_this_level=statistics.
                        mean(tmp)
                        rho.append(ave_of_all_large_exp_of_negative_rho_of_this_level)
                        print("ave_of_all_large_exp_of_negative_rho_of_this_level:"
                        +str(ave_of_all_large_exp_of_negative_rho_of_this_level))
                        print("ave_of_all_exp_of_negative_rho_of_this_level:"
                        +str(ave_of_all_exp_of_negative_rho_of_this_level))
                        if ave_of_all_large_exp_of_negative_rho_of_this_level!=0:
                            rho.append(ave_of_all_large_exp_of_negative_rho_of_this_level
                            /ave_of_all_exp_of_negative_rho_of_this_level)
                            print("ave_of_all_large_exp_of_negative_rho_of_this_level
                            /ave_of_all_exp_of_negative_rho_of_this_level:"
                            +str(ave_of_all_large_exp_of_negative_rho_of_this_level
```

```
                            /ave_of_all_exp_of_negative_rho_of_this_level))

            except:
                print("---")
        if len(rho)>2:
            if rho[-1]>1:
                useful_example+=1

        counter+=increment
        print("counter:"+str(counter))
        print("useful_example:"+str(useful_example))

def examples_of_10000_with_t(initial, increment,t0):
    counter=initial
    useful_example=0
    while counter < initial+10000*increment:
        print("************"+str(counter)+"************")
        out=[]
        if check_T_generate_a_Schottky(Lambda=counter,m=2):
            try:
                rho=[]
                for i in range(15):
                    sum1=0
                    sum2=0
                    test=Exp_negative_Hyperbolic_Distance_Gamma0_with_t(Lambda=counter,
                    m=2,N=i,t=t0)
                    #test=Exp_negative_Hyperbolic_Distance_Gamma0_with_t(Lambda=0.3,m=2,
                    N=i,t=t0)
                    ave_of_all_exp_of_negative_rho_of_this_level=statistics.
                    mean(test)
                    occurence=0
                    tmp=[]
                    for node in test:
                        if node < ave_of_all_exp_of_negative_rho_of_this_level:
                            occurence+=1
                        else:
                            tmp.append(node)
                    print("N="+str(i))
                    print("occurence="+str(occurence))
                    if len(tmp)!=0:
                        ave_of_all_large_exp_of_negative_rho_of_this_level=statistics.
                        mean(tmp)
                        rho.append(ave_of_all_large_exp_of_negative_rho_of_this_level)
                        print("ave_of_all_large_exp_of_negative_rho_of_this_level:"+
                        str(ave_of_all_large_exp_of_negative_rho_of_this_level))
                        print("ave_of_all_exp_of_negative_rho_of_this_level:"+
                        str(ave_of_all_exp_of_negative_rho_of_this_level))
                        if ave_of_all_large_exp_of_negative_rho_of_this_level!=0:
                            rho.append(ave_of_all_large_exp_of_negative_rho_of_this_level
                            /ave_of_all_exp_of_negative_rho_of_this_level)
                            print("ave_of_all_large_exp_of_negative_rho_of_this_level
                            /ave_of_all_exp_of_negative_rho_of_this_level:"+
                            str(ave_of_all_large_exp_of_negative_rho_of_this_level
                            /ave_of_all_exp_of_negative_rho_of_this_level))

            except:
                print("---")
        if len(rho)>2:
            if rho[-1]>1:
                useful_example+=1

        counter+=increment
        print("counter:"+str(counter))
        print("useful_example:"+str(useful_example))
```