PEC 4

Pablo Riutort Grande

June 11, 2023

M0.534 - Chaotic Dynamical Systems

Master's Degree in Computational and Mathematical Engineering

1

Prove that $e^{2\pi i F(x)} = f(e^{2\pi i x})$, for all $x \in R$.

F is a lift function of f if

$$\pi \circ F = f \circ \pi \tag{1}$$

and

$$\pi(x) = exp(2\pi i x) \tag{2}$$

In our case:

$$F(x) = x + \omega + \frac{\epsilon}{2\pi} sin(2\pi x) \tag{3}$$

$$f_{\omega,\epsilon}(\theta) = \theta + 2\pi\omega + \epsilon sin(\theta) \tag{4}$$

$$e^{2\pi i F(x)} = e^{2\pi i(x+\omega+\frac{\epsilon}{2\pi} sin(2\pi x))} \tag{5}$$

$$f(e^{2\pi i x}) = e^{2\pi i x} + 2\pi\omega + \epsilon sin(e^{2\pi i x}) \tag{6}$$

to prove the lift we could solve analytically:

$$\begin{aligned} e^{2\pi i(x+\omega+\frac{\epsilon}{2\pi} sin(2\pi x))} &= e^{2\pi i x} + 2\pi\omega + \epsilon sin(e^{2\pi i x}) \\ e^{2\pi i x} e^{2\pi i \omega} e^{i\epsilon sin(2\pi x)} &= e^{2\pi i x} + 2\pi\omega + \epsilon sin(e^{2\pi i x}) \\ &\dots \end{aligned} \tag{7}$$

2

Prove that $F(x+1) = F(x) + 1$, for all $x \in R$.

$$\begin{aligned} F(x+1) = (x+1) + \omega + \frac{\epsilon}{2\pi} sin(2\pi(x+1)) \\ x + 1 + \omega + \frac{\epsilon}{2\pi} sin(2\pi x + 2\pi) * \\ x + 1 + \omega + \frac{\epsilon}{2\pi} sin(2\pi x) \end{aligned} \tag{8}$$

$$
\begin{aligned}
*sin(2\pi x + 2\pi) &= sin(2\pi x)cos(2\pi) + sin(2\pi)cos(2\pi x) \\
sin(2\pi x + 2\pi) &= sin(2\pi x) * 1 + 0 * cos(2\pi x) \\
sin(2\pi x + 2\pi) &= sin(2\pi x)
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
F(x) + 1 &= x + \omega + \frac{\epsilon}{2\pi} sin(2\pi x) + 1 \\
&x + 1 + \omega + \frac{\epsilon}{2\pi} sin(2\pi x)
\end{aligned}
\tag{10}
$$

As we can see, Equation 8 is equal to Equation 10, hence proving that $F(x+1) = F(x) + 1$.

3

Choose (your favourite) number $0 < \epsilon < 1$, we denote this number by ϵ_0, and compute numerically an approximation of the rotation number $\rho(f_{\omega,\epsilon_0})$

$$
\rho(f_{\omega,\epsilon_0}) = \lim_{n\to\infty} \frac{F^n_{\omega,\epsilon_0}(x)}{n}
$$

To do this numerical computation, you can follow the following steps

- Fix N $>$ 1 and make a discretization of the parameter ω, so consider the discrete values

$$
\omega_k = \frac{k}{N} \quad k = 0, \ldots, N \tag{11}
$$

- For each ω_k and your favorite ϵ_0, compute

$$
\rho(f_{\omega_k,\epsilon_0}) = \lim_{n\to\infty} \frac{F^n_{\omega_k,\epsilon_0}(x)}{n}
$$

The way to represent this function is in the x-axis you put the ω_k values and in the y-axis you plot the corresponding rotation number $\rho(f_{\omega_k}, \epsilon_0)$.

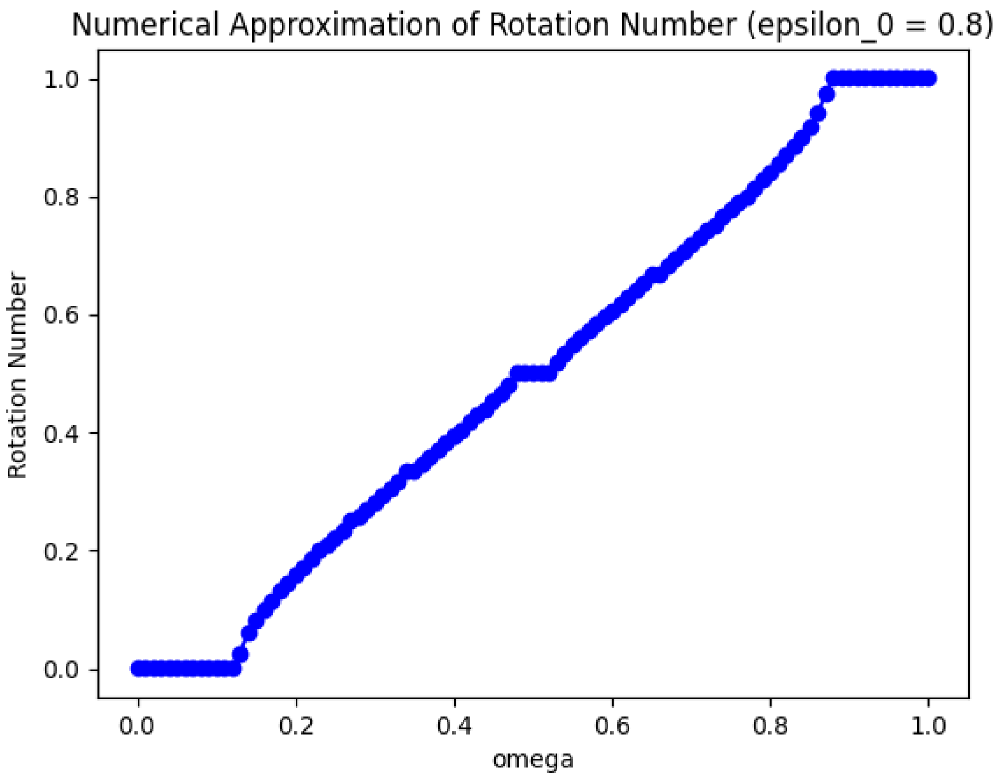

Figure 1: Approximation of the rotation number.

We define the lift function $F(x)$ for a given x, omega, and epsilon and we compute with compute_rotation_number function the iteration to compute the rotation number for those given parameters. Finally, plot_rotation_numbers function discretizes the parameter omega, computes the rotation numbers for each omega, and plots them (Listing 1).

```
import numpy as np
import matplotlib.pyplot as plt

def lift(x, omega, epsilon):
    return x + omega + epsilon/(2*np.pi) * np.sin(2*np.pi*x)

def compute_rotation_number(omega, epsilon, x0=0.5, max_iterations=1000):
    x = x0
    for _ in range(max_iterations):
        x = lift(x, omega, epsilon)
    return x / max_iterations

def plot_rotation_numbers(N, epsilon_0):
    omegas = np.linspace(0, 1, N+1)
    rotation_numbers = [
        compute_rotation_number(omega, epsilon_0)
        for omega in omegas
    ]

    plt.plot(omegas, rotation_numbers, 'bo-')
    plt.xlabel('omega')
```

```
    plt.ylabel('Rotation Number')
    plt.title(f'Numerical Approximation of Rotation Number (epsilon_0 = {epsilon_0})')
    plt.savefig('cantor.png')
    plt.close()

N = 100
epsilon_0 = 0.8

plot_rotation_numbers(N, epsilon_0)
```

Listing 1: Approximation of the rotation number.