Abstract

This report corresponds to the explanation of the code and evaluation of the results obtained in the fourth project of Simulation Methods. In order to solve the proposed questions, I have prepared a script in C, called *dissipativeHenon.c*, which is also attached in the delivery. I shall divide the report in two parts. The first one, will be destined to the explanation of the theoretical exercise (Exercise 1) and, in the second one, I will discuss the code and the remaining exercises.

1 Theoretical exercise

Our focus in this project will be on the dissipative Hénon map. This particular map takes on the following form:

$$\begin{cases} \overline{x} = 1 + y - ax^2 \\ \overline{y} = bx \end{cases}$$

where a and b are given parameters. In the exercise, we choose the classical values $a = 1.4$ and $b = 0.3$. In what follows, we will denote this map by H. The theoretical exercise that we need to solve is the following:

Exercise 1.1. *Show that H is an invertible map. Given any set $S \subset \mathbb{R}^2$ with positive measure, can you compute the limit of the measure of $H^{(n)}(S)$ when $n \to \infty$?*

Proof. Let us prove that H is an invertible map. In fact, we will prove that H a is diffeomorphism, which will be useful for the second part of the exercise.

Definition 1.2. *Given two manifolds M and N, a differential map $f\colon M \to N$ is called a diffeomorphism if it is a bijection and its inverse $f^{-1}\colon N \to M$ is differentiable as well.*

It is clear that our function H is a differential map. The only thing left to prove is that H is a bijection and its inverse is differentiable. To prove this, we will use the Inverse Function Theorem, IFT. To apply the theorem, we need the determinant of the Jacobian matrix to be different from 0 at some point, let us check it:

$$\det(J_H(x,y)) = \det \begin{bmatrix} -2ax & 1 \\ b & 0 \end{bmatrix} = -b \neq 0.$$

Hence, by the IFT, the map H has a local continuously differentiable inverse at each point of $\mathbb{R}^2$. Notice that, if we prove that H is an injective map, we will have proved that H is a diffeomorphism, since the injectivity of H will imply the existence of a global continuously differentiable inverse. To prove that H is injective, assume that $H(x,y) = H(x',y')$ and observe that $(x,y) = (x',y')$[1]. Hence, H is injective and therefore, H is a diffeomorphism.

Let us continue with the second part of the exercise. Given any set $S \subset \mathbb{R}^2$ with positive measure, notice that

$$\lim_{n\to\infty} \mu(H^n(S)) = \lim_{n\to\infty} \int_{H^n(S)} 1 d\mu = \lim_{n\to\infty} |\det(J_{H^{(n)}})| \int_S 1 d\mu$$

[1] This is an easy computation which follows directly from the definition of H.

where the last equality is due to the change of variable theorem for diffeomorphisms. Notice that

$$|\det(J_{H^{(n)}})| = \left|\det\left(\prod_{k=1}^{n-1} J_H(H^{n-k})J_H\right)\right| = b^n.$$

Hence, since $b = 0.3$,

$$\lim_{n\to\infty} \mu(H^n(S)) = \lim_{n\to\infty} b^n \int_S 1 d\mu = \begin{cases} 0 & \text{if } S \text{ is a set of finite measure,} \\ \infty & \text{otherwise.} \end{cases}$$

□

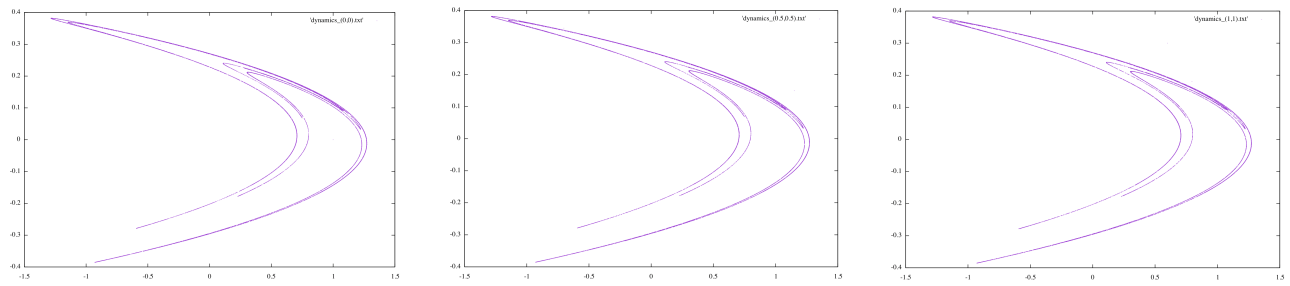
2 Script

For the remaining exercises, I have prepared a script in C called *dissipativeHenon.c*, in which I solve the required numerical exercises. I am going to explain the script as I explain the solution of the exercises.

Exercise 2.1. *Make a plot of the dynamics of H. Note that there is an attractor set for many initial conditions. Make a plot of this attracting set. Feel free to zoom into the structure of this attracting set.*

To tackle this exercise, I developed a function named *dynamics* specifically designed to plot the dynamics of H. This function generates a text file that contains a series of points resulting from iterations of H, based on a given initial condition. To achieve this, I created a separate function called H which calculates the output of the dissipative Henon map for a given point. Consequently, the *dynamics* function is capable of performing any desired number of iterations for H.

To illustrate the functionality of the code, I defined three initial conditions: $(0, 0)$, $(0.5, 0.5)$, and $(1, 1)$. By executing the *dynamics* function, three distinct files are generated, each labeled as *dynamics_(x,y).txt* corresponding to the aforementioned points. Utilizing Gnuplot[2], I visualized the contents of these files. Figures 1 and 2 showcase the emergence of the attractor set in all three cases.



(a) dynamics_(0,0).txt (b) dynamics_(0.5,0.5).txt (c) dynamics_(1,1).txt

Figure 1: Attractor set of the dissipative Henon map for different initial conditions.

[2]Gnuplot is a portable command-line driven graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms.

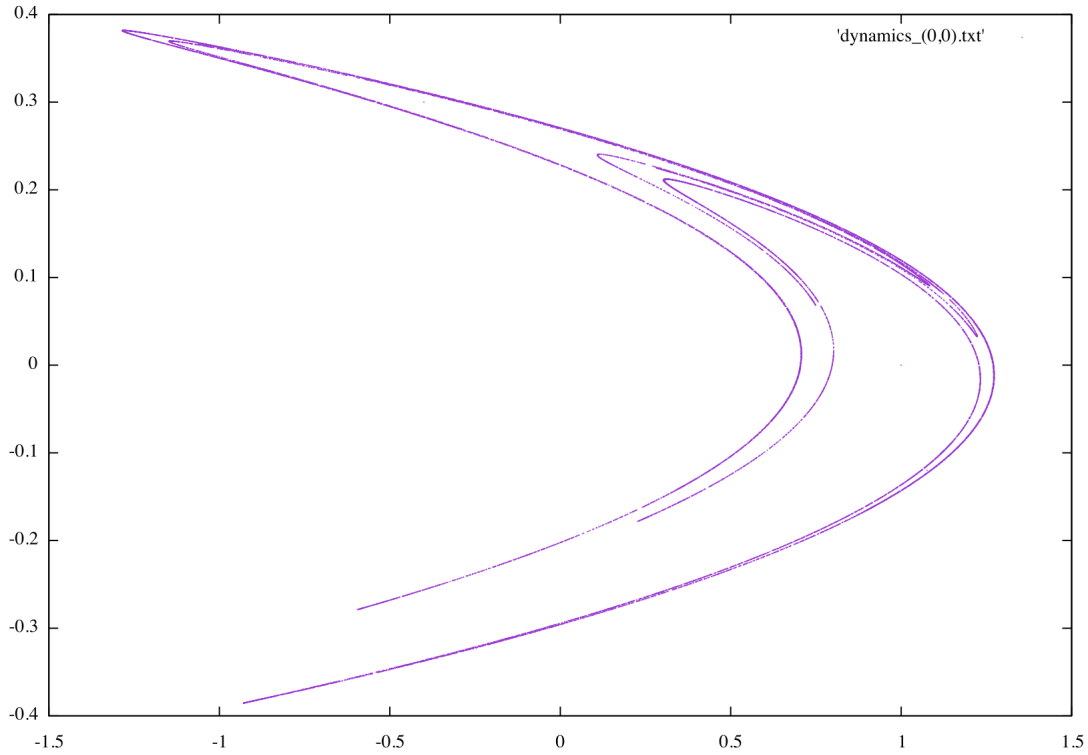


Figure 2: Better visualization of the attractor set for the initial condition $(0, 0)$.

After completing the visualization of the dynamics of H, we are tasked with locating a fixed point of H in close proximity to the coordinates $(0.63, 0.19)$. To be more precise,

Exercise 2.2. *This map has a fixed point p_0 near $(0.63, 0.19)$. Compute p_0 and its stability.*

To complete this exercise, I have prepared a function named *Newton* that utilizes Newton's method to find a zero of the function $G(x, y) := H(x, y) - (x, y)$. By employing this method, we can effectively determine a fixed point of H. Consequently, when providing the initial guess $(0.63, 0.19)$ as a parameter to the function, it executes the Newton's method algorithm until two consecutive iterates are considered "sufficiently close"[3]. Alternatively, if a maximum of 50 iterates is reached, the process terminates. Recall that, to perform the $k + 1$ Newton step, we need to solve the following system in $\mathbf{x}$:

$$0 = G(\mathbf{x}^k) + J_G(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k)$$

where $J_G(\mathbf{x}^k)$ is the Jacobian matrix of G evaluated in the previous step. By calling $\mathbf{v} = \mathbf{x} - \mathbf{x}^k$, one can easily solve the system and find the final result, which will be given by $\mathbf{v} + \mathbf{x}^k$. To solve the linear system, I have prepared a function called *gaussEliminationLS*, that solve the linear system using Gaussian elimination with partial pivoting.

I have obtained that, the fixed point near $(0.63, 0.19)$, computed using a threshold of $\epsilon = 10^{-15}$, is $p_0 = (0.6313, 0.1894)$. The method has needed 4 iterations to reach the result.

To conclude this section, we are also required to assess the stability of the fixed point denoted as p_0. To accomplish this, it is necessary to calculate the eigenvalues of the Jacobian matrix, evaluated at that specific point. For this purpose, I have prepared two functions: *solver_real* and *solver_complex*. These functions are designed to determine the eigenvalues of a 2×2 matrix,

[3] To determine the proximity, a threshold parameter is required within the function.

with one catering to the real case and the other handling the complex case. Additionally, there is a function named *find_eigenvalues* which determines whether the eigenvalues will be real or complex by examining the discriminant of the matrix. Based on this determination, the function utilizes one of the previously mentioned functions to compute the eigenvalues of the matrix. Using these functions, I obtain that the eigenvalues are both real, $\lambda_1 \approx 0.156$ and $\lambda_2 \approx -1.924$. Therefore, since both eigenvalues are real and, $|\lambda_1| < 1, |\lambda_2| > 1$, we get a saddle point, which is an unstable fixed point.

To finish the project, we are asked to perform some calculation within the framework of the Lyapunov exponents. More precisely,

Exercise 2.3. *Compute an approximation to the Lyapunov exponent of the attractor for different (at least two) initial conditions. Explain the computational method and discuss the results.*

For this exercise, I have created a function called *Lyapunov* that implements the necessary algorithm to compute the maximal Lyapunov exponent. Let me explain it in detail. The Lyapunov function receives as a parameter a initial point (x_0, y_0) in the basin of attraction Λ, a positive integer s and a vector of norm one. For the given initial point, the function computes 10.000 iterates of H and assumes that $H^n(0,0) := (x_n, y_n) \in \Lambda$. Then, the function computes 10.000 iterates more to see if they fill the attractor. At that point, for the given s an v, the function computes $\lambda_k = \frac{1}{ks}\log(|Df^{ks}(x_{n+m}, y_{n+m})v|)$ until $|\lambda_k - \lambda_{k-1}|$ is less than some threshold, in our case, 10^{-15}. This is the most delicate step, let me explain how I have computed λ_k in detail.

We fix $s \in \mathbb{N}$ and a vector $v \in \mathbb{R}^2$, $||v|| = 1$. We define recursively:

$$\begin{aligned} w_0 &= v, \\ \alpha_k &= |Df^s(f^{(k-1)s}(x))w_{k-1}|, \\ w_k &= \frac{Df^s(f^{(k-1)s}(x))w_{k-1}}{\alpha_k}. \end{aligned}$$

Then,

$$|Df^{ks}(x)v| = |Df^s(f^{(k-1)s}(x))Df^s(f^{(k-2)s}(x)) \cdots Df^s(x)v| = \alpha_1 \cdots \alpha_k. \tag{2.1}$$

Therefore, the computation of $|Df^{ks}(x_{n+m}, y_{n+m})v|$, is reduced to the multiplication of $\alpha_1 \cdots \alpha_k$. Hence, we can easily compute λ_k by $\frac{1}{ks}\sum_i^k log(\alpha_i)$. In the script, to perform the previous recursion scheme, I have made a function called *DH_stimes* to compute the differential matrix Df^s evaluated at a point $\mathbf{x}$ and another one called *mat_times_vect* that computes the multiplication of a matrix times a vector. To calculate the matrix Df^s, we have to remember that, due to the chain rule, the computation is reduced to a matrix multiplication evaluated on different points, as is shown in (2.1). The computation made in the *DH_stimes* is the following:

$$Df^s(\mathbf{x}) = Df(f^{(s-1)}(\mathbf{x}))Df(f^{s-2}(\mathbf{x})) \cdots Df(\mathbf{x}).$$

Finally, the function *Lyapunov* returns the maximal Lyapunov exponent $\chi_1 = \lim_{k\to\infty} \frac{1}{ks}\sum_{i=1}^k \log\alpha_i$. We have computed the maximal Lyapunov exponent (MLE) for different initial conditions and different values of s and, as is shown in the theory, the Lyapunov exponent does not depend neither on the starting point, nor in s, nor in v. For detailed information, a comparison table is shown in Table 1 and 2.

s	MLE
1	0.4208
2	0.4206
3	0.4209
4	0.4214
5	0.4210

Table 1: MLE for the initial condition $(0, 0)$.

s	MLE
1	0.4169
2	0.4178
3	0.4192
4	0.4196
5	0.4195

Table 2: MLE for the initial condition $(1, 1)$.

With these values of MLE, we can conclude that this system is chaotic. Finally, the last question of the project is the following:

Exercise 2.4. *From the previous computation, it is possible to derive the full set of Lyapunov exponents of this attractor?*

For this question, we only have to observe that, if we assume that $\chi_1 + \chi_2 = \log|b|$, as stated in the theory, we can easily obtain the second Lyapunov exponent (SLE) and derive the full set of Lyapunov exponents for this attractor. For detailed information, a comparison table is shown in Table 3 and 4.

s	SLE
1	-1.6248
2	-1.6245
3	-1.6249
4	-1.6254
5	-1.6249

Table 3: SLE for the initial condition $(0, 0)$.

s	SLE
1	-1.6209
2	-1.6218
3	-1.6231
4	-1.6235
5	-1.6237

Table 4: SLE for the initial condition $(1, 1)$.