Numerical determination of Lyapunov exponents of discrete and continuous dynamical systems

Daniel SMOLDERS

University of Cambridge, Department of Physics

Abstract

We examine classical methods of determining Lyapunov exponents of both discrete and continuous-time dynamical systems, and present a Python script that calculates the full spectrum of Lyapunov exponents for arbitrary systems given an equation of state. QR-decomposition was used to maintain orthogonality of the eigenvectors throughout the calculations. The exponents were calculated for four discrete and four continuous systems. Ten simulations starting at different initial values were done for each dynamical system, allowing for a rough estimate of the error in the exponents, which averaged 1%. Most results showed agreement within one standard deviation of the literary values, and all within two. Finally, graphs of the trajectories in phase space, as well as the evolution of the exponents over time were plotted to allow visual inspection of the behaviour of each system, which also matched the literature. (138 words)

1 Introduction

Dynamical systems play a key role in the modelling of countless phenomena in an ever-increasing number of fields. There exists numerous quantities used to characterise such systems, examples being the Kolmogorov entropy and the Lyapunov energy function, which determine the stability and complexity of solutions. A popular tool is the spectrum of Lyapunov Characteristic Exponents (LCEs), which quantify the evolution of perturbations in solutions of a system. These describe the average exponential separation of two nearby orbits in the system's phase space. The existence of these numbers is proved by Oseledec's theorem [4], and numerous algorithms have been developed to find these exponents numerically, especially as more computational power becomes more accessible.

2 Theoretical Background

A discrete-time dynamical system is defined by an n-dimensional map f that acts on state vectors $x_t \in \mathbb{R}^n$, which follow the state equation

$$x_{t+1} = f(x_t), t \in \mathbb{N}. \tag{1}$$

Given an initial state x_0, the system evolves by repeatedly applying f, thus generating a discrete set of states $x_t = f^t(x_0)$, namely an orbit.

Consider a perturbation u_0 between the points x_0 and $x_0 + u_0$. After t time steps, the perturbation will have evolved to $u_t = f^t(x_0 + u_0) - f^t(x_0) = D_{x_0} f^t(x_0) \cdot u_0$, from linearising f^t [7].

The LCE λ quantifies the evolution of this perturbation as

$$\|u_t\| \approx e^{\lambda t} \|u_0\| \tag{2}$$

For an n-dimensional system, there exist a total of n LCEs called the spectrum of Lyapunov exponents. These quantify the evolution of perturbations in different directions along an orbit, and are usually ordered from largest to smallest, i.e. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

The same idea applies for continuous-time systems, with the difference being the state equation consisting of differential equations in the form

$$\frac{\mathrm{d}x}{\mathrm{d}t} = F(x) \tag{3}$$

where $x(t) \in \mathbb{R}^n$ and F a function describing the state equation.

An important notion in dynamical systems is attracting limit-sets, as explained in [7]. These fall into four categories: fixed points, periodic motions, quasiperiodic motions, and chaotic motions. LCEs are a powerful tool in characterising which category a limit-set falls into. In fact, the number of LCEs and their signs determine the dynamics of an attractor, and are summarised in Table 1. There, the Hausdoff dimension corresponds

to the fractal dimension of the attractor, which is often very tedious to find. The Kaplan-Yorke dimension D_{KY}, however, can easily be calculated with the spectrum of LCEs and gives an approximation of the Hausdoff dimension [3]. D_{KY} is defined as follows

$$D_{KY} = k + \sum_{i=1}^{k} \frac{\lambda_i}{|\lambda_{k+1}|} \tag{4}$$

where k is the largest integer such that $\sum_{i=1}^{k} \lambda_i \geq 0$.

Topological dimension	Dynamics of the attractor	LCE spectrum	Hausdroff dimension
1	Fixed point	–	0
2	Periodic motion	0 –	1
3	Torus $\mathbf{T}^2$	0 0 –	2
	Chaos $\mathbf{C}^1$	+ 0 –	$2 < D < 3$
4	Hypertorus $\mathbf{T}^3$	0 0 0 –	3
	Chaos on $\mathbf{T}^3$	+ 0 0 –	$3 < D < 4$
	Hyperchaos $\mathbf{C}^2$	+ + 0 –	$3 < D < 4$

Table 1: LCE spectrum of continuous-time attractors, from [7].

Calculation of the largest LCE is quite straightforward in dynamical systems. In fact, a perturbation with a non-zero component in the direction of largest λ will tend to fully align with that direction, as shown in Figure 1. Thus measuring the norm of arbitrary tangent vectors evolving through time will give the largest LCE. We will therefore focus on finding the entire spectrum of LCEs.

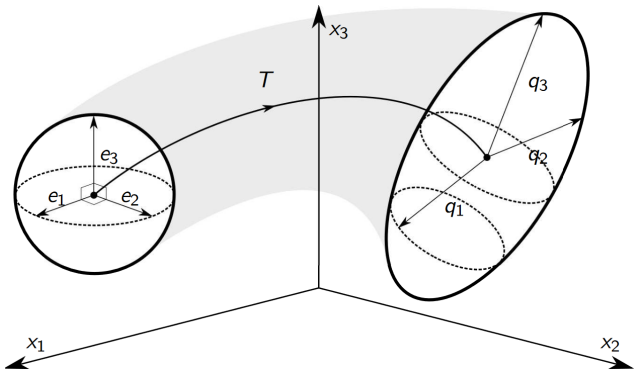

Figure 1: Perturbations all tending to align in the direction of largest (positive) LCE. The same idea applies if the largest LCE is negative, as it will be the direction of slowest convergence. Sketch from [1].

3 Method

3.1 Discrete Maps

Following the chain rule, as shown in [7], we can get

$$D_x f^t(x_0) = J(f^{t-1}(x_0)) \cdots J(f(x_0)) \cdot J(x_0) \tag{5}$$

where $J(x_0) = D_x f(x)|_{x=x_0}$ is the Jacobian of f evaluated at a point. The algorithm we will use is based on [2]. We first use QR-decomposition to write $J(x_0) = Q_1 R_1$ where Q_1 is orthogonal and R_1 is upper triangular. Then with the definition $J_k^* = J(f^{k-1}(x))Q_{k-1}$ for $k \geq 2$, we apply QR-decomposition on each $J_k^* = Q_k R_k$. This recursively leads to

$$D_x f^t(x) = Q_t R_t R_{t-1} \cdots R_1. \tag{6}$$

Now it can be shown that the LCEs are given by the diagonal elements of $\mathcal{R}^{(t)} = R_t \cdots R_1$ as follows [7]:

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} \ln \mathcal{R}_{ii}^{(t)}. \tag{7}$$

Therefore λ_i is simply an average of the logs of the diagonal elements of R_t.

3.2 Continuous Flows

Lyapunov exponents for continuous dynamical systems can be calculated very similarly to that of discrete cases, the difference being that between each QR-decomposition, we must integrate the entire system.

In addition to the equation of state, the tangent vector evolves according to the variational equation [5]:

$$\dot{\Phi}_t(x_0) = D_x F(f^t(x_0)) \cdot \Phi_t(x_0), \quad \Phi_0(x_0) = I \tag{8}$$

where $\Phi_t(x_0) = D_{x_0} f^t(x_0)$.

We therefore need to integrate the entire system together:

$$\begin{cases} \dot{x} = F(x) \\ \dot{\Phi} = D_x F(x) \cdot \Phi \end{cases}, \qquad \begin{cases} x(t=0) = x_0 \\ \Phi(t=0) = I \end{cases} \tag{9}$$

Fourth-order Runge-Kutta is used to numerically integrate the system, however, other simpler methods can also be used to save time over precision. Euler's method and other Runge-Kutta methods are implemented as part of the main function for calculating LCEs for continuous systems, and can be chosen with the `order` argument.

3.3 Error calculation

To get an estimate of the error, we run multiple simulations starting at different initial values. Since all the maps are chaotic in this case, choosing a slightly different initial value will suffice as the trajectories will have diverged after the transient iterations. In fact, assuming the largest LCE is small, i.e. of the order $\lambda = 0.01$, and that the initial perturbation is of the order $\delta x = 0.01$, then the time t after two orbits separated by δx will diverge satisfies

$$e^{\lambda t} \approx \frac{1}{\delta x} \quad \iff \quad t \approx 500. \tag{10}$$

This means 10^4 transient iterations are enough for the systems considered here.

4 Results

LCEs for eight systems from the appendix of [8] were calculated. 10 trials of 10^4 transient iterations and 10^5 normal iterations were used. The step size used for numerical integration of continuous systems was 0.005, and the initial random perturbations were of the order of 0.01.

All the numerical results, as well as literary values taken from [8] are in Table 2. Graphs of trajectories and evolution of LCEs are in Figures 2 and 3 in the appendix.

Running the program took 20 minutes on Google's Python 3 Compute Engine (Google Colab).

5 Discussion

Most of the errors on the calculated LCEs are less than 1% and agree within one standard deviation to the literary values, indicating that the program works as expected.

All trajectories in Figures 2 and 3 visually match very well with the expected shapes of attractors also found in [8]. In fact, for continuous systems, the time step of 0.005 for numerical integration was chosen so that each point in the trajectories were neither too close nor too sparse, as that would lead to too little or too erratic movement in the phase space respectively.

A few values such as for the damped driven pendulum and the hyperchaotic Rossler attractor deviate a bit more than the other values. This is likely due to too few iterations as looking at Figure 3 shows that the LCEs of each trial have not fully converged by 10^5 iterations.

5.1 Errors

Errors in the LCEs were calculated based on 10 trials of the same system starting from slightly different intial positions. For systems that converge quickly and are stable, such as the Henon map, the precision of the results could likely be increased by increasing the number of trials while decreasing the number of iterations. However, the same cannot be said about all systems, as the hyperchaotic Rossler attractor has one very negative exponent, which therefore has a much slower convergence rates.

Discrete Maps	LCEs	D_{KY}	Literary LCEs	Literary D_{KY}
Henon Map	0.4194 ± 0.0002 -1.6234 ± 0.0002	1.25837	0.41922 -1.62319	1.25827
Tinkerbell Map	0.1899 ± 0.0002 -0.5214 ± 0.0007	1.36417	0.18997 -0.52091	1.36468
Predator-Prey Map	0.1968 ± 0.0002 0.0328 ± 0.0006	2	0.19664 0.03276	2
Lorenz 3D Map	0.078 ± 0.003 $(3 \pm 4) \cdot 10^{-6}$ -0.078 ± 0.003	3	0.07456 0 -0.07456	2
Continuous flows				
Lorenz Attractor	0.905 ± 0.003 -0.0001 ± 0.0005 -14.572 ± 0.003	2.06210	0.9056 0 -14.5723	2.06215
Damped Driven Pendulum	0.151 ± 0.005 0.0001 ± 0.0002 -0.201 ± 0.005	2.75136	0.1414 0 -0.1914	2.7387
Rossler Attractor	0.071 ± 0.002 0.0006 ± 0.0002 -5.407 ± 0.001	2.01330	0.0714 0 -5.3943	2.0132
Hyperchaotic Rossler Attractor	0.115 ± 0.003 0.020 ± 0.002 0.0005 ± 0.0009 -24.0 ± 0.6	3.00563	0.112 0.019 0 -25.188	3.00052

Table 2: Average LCEs calculated for 10 trials of 10^5 iterations with 10^4 transient iterations.

Errors within the calculation of the LCEs were minised firstly with the use of `np.linalg.qr`, which utilises Householder reflectors and is less prone to the accumulation of numerical errors compared to the Gram-Schmidt process [6]. Moreover, the fourth-order Runge-Kutta numerical integration was used for continuous systems. When tested against other integration methods such as Euler's method, second, and third-order Runge-Kutta methods, the fourth-order proves to be the most accurate as expected, although at the expense of lower speeds: almost twice as slow as Euler's method.

5.2 Improvements

Vectorising functions throughout the code to run simulations in parallel will likely improve the efficiency of the code. For example, if the jacobian function `J_func` were to be vectorised to accept multiple parameters at once, one could be able to see find how small changes in parameters of a system affect its behaviour. However, this seems to be difficult to achieve working with `sympy` symbols.

More powerful integration methods such as using `scipy.integrate.odeint` do lead to higher precision, but were much slower and more complicated to run due to the nature of the problem involving matrix multiplication. Using these library functions would likely be an improvement if there is a way to keep matrices intact and not need to convert them repeatedly into flattened arrays.

6 Conclusions

The numerical calculation of LCEs proves to be very feasible on a modern computer, and using `numpy`'s QR-decomposition methods alongside any simple numerical integration method from Euler's method to the fourth-order Runge-Kutta methods proves to be efficient and accurate at determining the full spectrum of LCEs to within 1%. In addition to producing beautiful plots of attractors, calculating LCEs is useful for gaining insight into aspects other than the divergence of perturbations in such systems, such as finding the Kaplan-Yorke dimension.

References

[1] A. O. Belyakov. On the numerical calculation of lyapunov exponents. In *11th Workshop on Optimal Control, Dynamic Games and Nonlinear Dynamic*, 2010.

[2] J. P. Eckmann. and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617–656, 1985.

[3] P. Frederickson et al. The liapunov dimension of strange attractors. *Journal of Differential Equations*, 49(2):185–207, 1983.

[4] V. I. Oseledets. A multiplicative ergodic theorem: Lyapunov characteristic numbers for dynamical systems. *Trans. Mosc. Math. Soc.*, 19:197–231, 1968.

[5] T. S. Parker and L. O. Chua. *Practical Numerical Algorithms for Chaotic Systems*. Springer New York, 1989.

[6] A. Pikovsky and A. Politi. *Lyapunov Exponents: A Tool to Explore Complex Dynamics*. Cambridge University Press, 2016.

[7] M. Sandri. Numerical calculation of lyapunov exponents. *Math. J.*, 6, 01 1996.

[8] J. C. Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, 2003.

Appendix

`Systems.py` includes the definition of numerous discrete and continuous-time state equations, with each being an instance of the `System` class. All expressions are defined using `sympy` symbols, as it allows for symbolic calculation of the Jacobian matrix. Should there be an error, one can also explicitly define the Jacobian (usually for expressions involving moduli).

`LCEs.py` has two functions to calculate LCEs, for discrete and continuous systems. It prints the final LCEs of the systems defined in `DiscreteSystems.py`, as well as plotting the evolution of the LCEs. Finally, the trajectory of the state in phase space is also plotted for 2D and 3D systems.

`plotter.py` plots all of the phase space trajectories and evolution of LCEs to the screen if wanted, and saves them if wanted (can be useful to create a folder). The script also writes all the results to a plain text file alongside printing them to the console.

`main.py` combines the previous files to easily calculate LCEs of any system but passing them all in the `Maps` list. Parameters such as number of iterations and size of time steps can also easily be changed.

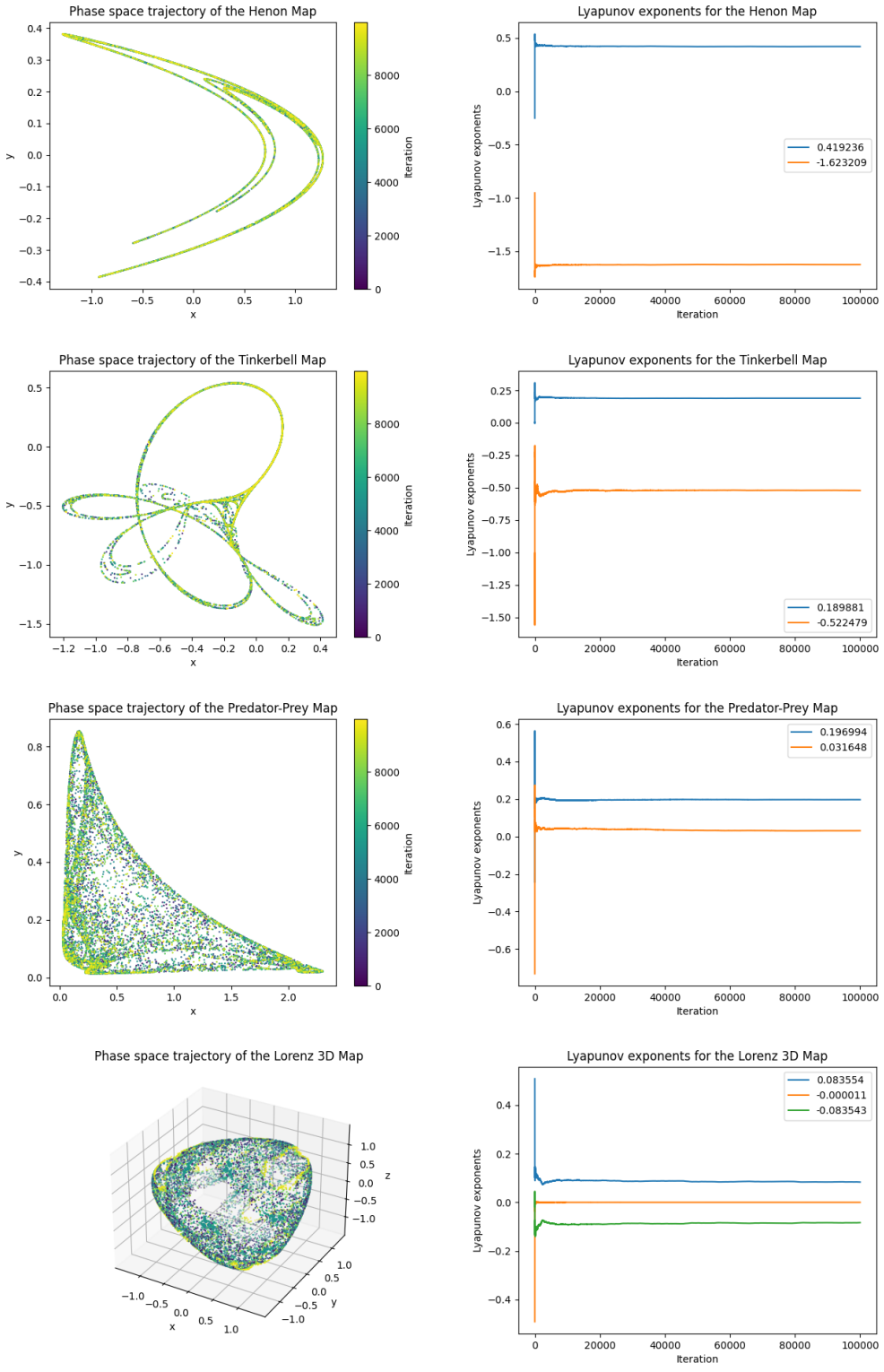

Figure 2: Phase space trajectories and evolution of LCEs for a single trial of each discrete dynamical system. The trajectories allows for visual inspection of whether the points lie on the attractor, and the evolution of LCEs give an indication of the convergence rate for each system.

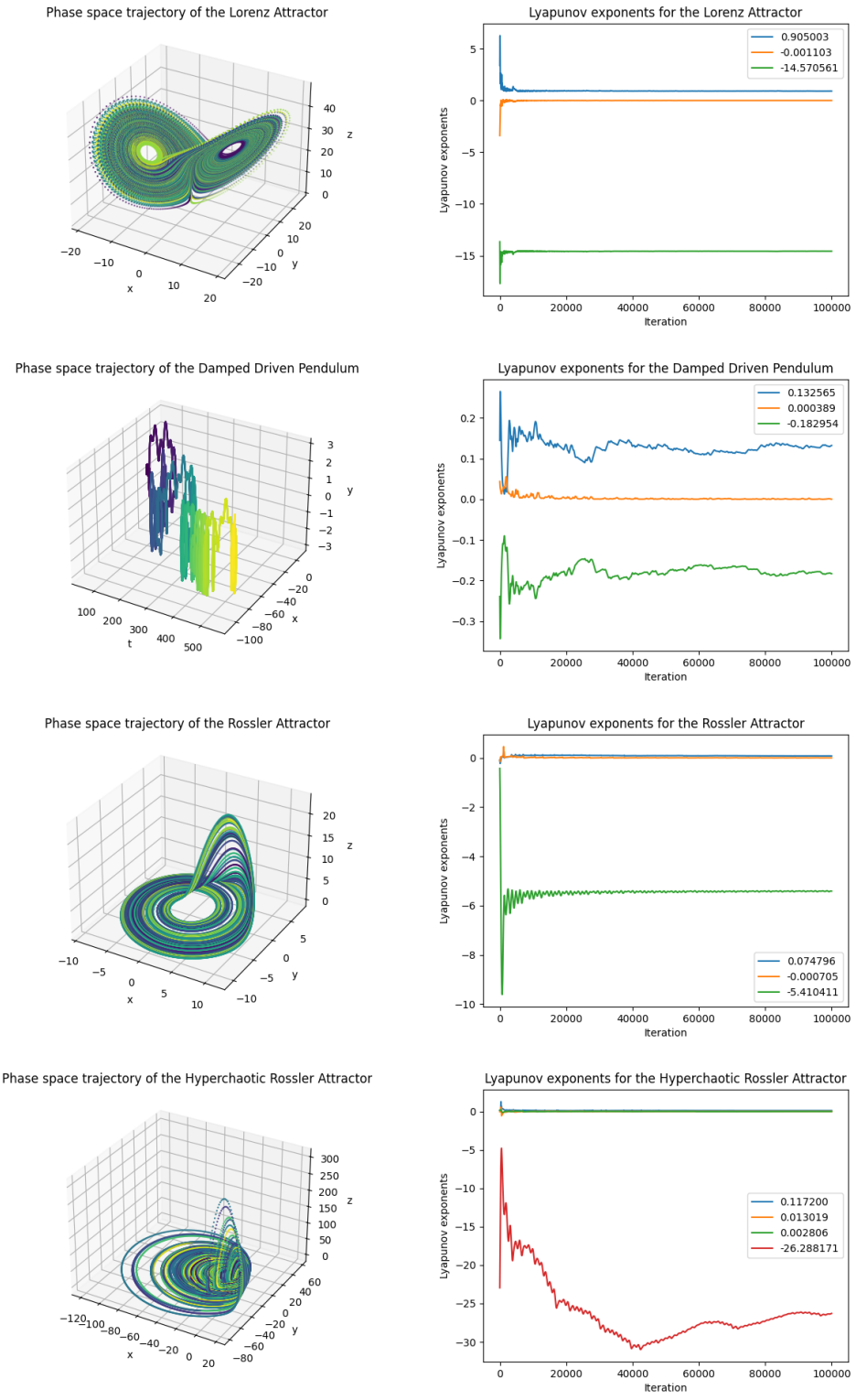

Figure 3: Continuation of Figure 2 with continuous systems.