Chaos and Computation of Lyapunov Spectra

Bryan Garcia
AMS 114
December 9, 2018

1 Introduction

In the study of dynamical systems, understanding behavior known as chaos is of great importance due to the plethora of situations in which said behavior arises. The seemingly random nature of chaos can be better understood by obtaining the Lyapunov spectra. Named after Russian mathematician Aleksandr Lyapunov, the Lyapunov spectra gives us a way to quantify the sensitivity associated with chaotic behavior in a dynamical system\(^7\). With motivations established, we will now explore the derivation of Lyapunov formulae, demonstrate methods for computation of Lyapunov spectra in systems which governing equations are known, and finally, discuss an algorithm for calculating the maximal Lyapunov exponent from time series data.

2 The Lyapunov Spectra

2.1 Defining Lyapunov Exponents in 1-D

Chaos in a dynamical system can be characterized by aperiodic behavior and extreme sensitivity to changes in initial conditions\(^7\). The Lyapunov exponent aims to quantify this sensitivity of a dynamical system by measuring the "propagated error" at the infinitesimal level\(^2\).

Consider some initial error \(\delta_0 \in \mathbb{R}\) and say

\[
x'_0 = x_0 + \delta_0
\]

where \(x_0\) is some initial orbit, and \(x'_0\) is said orbit with our error included. Let us monitor the change in error at each iteration. That is, let us observe:

\[
\delta_0 = x'_0 - x_0
\]

\[
\delta_1 = x'_1 - x_1
\]

\[
\delta_2 = x'_2 - x_1
\]

\[
\delta_3 = x'_3 - x_3
\]
This change in error can be measured at each iteration simply by computing
\[
\frac{\delta_n}{\delta_{n-1}}
\]
Now, after some amount of steps \(i\), we can say our initial error \(\delta_0\) has been amplified to:
\[
\frac{\delta_i}{\delta_0} = \frac{\delta_i}{\delta_{i-1}} \cdot \frac{\delta_{i-1}}{\delta_{i-2}} \cdots \frac{\delta_2}{\delta_1} \frac{\delta_1}{\delta_0}
\]
Let us apply this reasoning to the linear function \(f(x) = cx\) with \(c > 1\). In general, we see the behavior of \(f(x)\) with error as:
\[
\begin{align*}
    f(x + \delta) &= c(x + \delta) \\
    &= cx + c\delta \\
    &= f(x) + c\delta \\
    f(f(x) + c\delta) &= c(f(x) + c\delta) \\
    &= f(f(x)) + c^2\delta
\end{align*}
\]
It is clear that after some amount of iterations \(i\), we will have had the error \(\delta\) in our system increase by a factor of \(c^i\). Equivalently, we can say:
\[
c^n = \left| \frac{\delta_n}{\delta_0} \right|
\]
\[
\ln(c) = \frac{1}{n} \ln \left| \frac{\delta_n}{\delta_0} \right|
\]
\[
\ln(c) = \frac{1}{n} \sum_{k=1}^{n} \ln \left| \frac{\delta_k}{\delta_{k-1}} \right|
\]
This behavior is at the heart of the Lyapunov exponent. In general, we can take any function \(f(x)\), apply the same treatment, and should values converge as we let \(n\) grow larger and larger, we may say our error grows similarly to \(c^n\) on average.
To officially define our Lyapunov exponent \(\lambda\), we observe the behavior as \(n \to \infty\) for some \(f(x)\), and obtain from [2]:
\[
\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \ln \left| \frac{\delta_k}{\delta_{k-1}} \right|
\]
It should be noted that:
\[
\frac{\delta_n}{\delta_{n-1}} = \frac{f(x_{n-1} + \delta_{n-1}) - f(x_{n-1})}{\delta_{n-1}}
\]
If we assume \( f \) is a smooth function, then by letting \( \delta_{n-1} \to 0 \), we see from [2]:

\[
\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \ln |f'(x_{k-1})|
\]

**Remark:** It should be understood that these equations outlined are only exactly applicable to one-dimensional dynamical systems. This behavior of "error measurement" we outlined in the derivation of Lyapunov exponents in one-dimension, however, will prove to be the basis for our methods of calculating Lyapunov exponents in n-dimensions. Before we discuss the N-D case, let us see what we have developed applied to a 1-D dynamical system.

### 2.2 The Logistic Map

We will apply our work so far on a classic one-dimensional dynamical system: the logistic map. This one-dimensional, discrete dynamical system is defined as:

\[
x_{n+1} = rx_n(1-x_n)
\]

with \( r \in \mathbb{R} \). This system is of particular interest due to chaotic behavior it exhibits. Let us observe two examples:

#### 2.2.1 Sensitivity: Visualized

![Logistic Map Plot: \( \Delta x_0 = 0.0001 \)](image)

\begin{align*}
\text{Figure 1: Blue: } x_0 &= 0.9; \ Red: x'_0 &= 0.9001; \ r &= 3.6 \\
\end{align*}

We see initially the two initial conditions of \( x_0 \) and \( x'_0 \) appear to be in sync. By iteration \( n = 20 \), this extremely small difference in initial conditions clear has amplified to a noticeable
difference. We see even more divergence around the $n = 30$ mark, and then a short window of synchronicity followed by more diverging values.

![Logistic Map Plot: $\Delta x_0 = 0.0001$](image)

Figure 2: Blue: $x_0 = 0.9$; Red: $x'_0 = 0.9001$; $r = 3.9$

Again, we see the "in-phase" nature at early iterations, but (even more so in this simulation) we can also see loss of synchronicity around the $n = 20$ mark. This difference in this system is only in parameter value, but it definitely exhibits much more sensitivity to initial conditions than the previous example.

### 2.2.2 Sensitivity: Quantified

Lyapunov exponents are a quantity to characterize this sensitivity to change and/or orbital divergence at the infinitesimal level. Let us employ the work we did earlier to compute the Lyapunov exponents of the figures above.

Applying the Lyapunov exponent formula for 1-D systems, we get:

$$
\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \ln \left| f'(x_{k-1}) \right|
$$

$$
\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \ln \left| r - 2rx_k \right|
$$

For $r = 3.6$ and $r = 3.9$, respectively, we obtain:

$$
\lambda \approx 0.1782
$$
\[
\lambda \approx 0.4723
\]

Notice the larger Lyapunov exponent value that corresponds with the more erratic plot in figure 2. Referring to the derivation of our Lyapunov exponent formula, it makes sense that a larger Lyapunov exponent will yield a much more sensitive system. In general, we can say that positive Lyapunov exponents are a strong indicator of chaos in a system. Going one step further, and using the same formula from above, we can obtain the following plot of Lyapunov exponent values for varying values of \( r \).

![Figure 3: \( r \) values from 3 to 4 with a 0.0001 step size.](image)

This plot of Lyapunov exponent values for the logistic map, coupled with the idea that positive Lyapunov exponents correspond to chaotic behavior, demonstrates why the \( r \) values above were chosen for our example. One should be able to deduce that negative and zero Lyapunov exponent values will yield much more stable systems that are not sensitive to the perturbations we saw in fig. 1 and 2. This qualitative interpretation of Lyapunov exponents is a tool for us to employ in analysis of dynamical systems; an idea we will revisit later when we discuss n-dimensional systems.
Here is a plot with the $r$ value corresponding to the first sharp dip in the Lyapunov exponent plot. There are still slight differences in our plots, but clearly chaotic behavior is not exhibited.

### 3 Computation of Lyapunov Spectra in N-Dimensional Systems

The previous section demonstrated a technique for finding Lyapunov exponent in one-dimensional systems. In general, however, there are $n$ Lyapunov exponents in $n$-dimensional systems; what we will refer to as the Lyapunov Spectra. Obtaining the entire spectrum is of use to us because they give us greater insight into the dynamical system we are analyzing. For example, consider a three-dimensional system. According to [1], we can find that:

- Strange attractors correspond to a spectra with values of $(\lambda_1 > 0, \lambda_2 = 0, \lambda_3 < 0)$
- Limit cycles correspond to $(\lambda_1 = 0, \lambda_2 < 0, \lambda_3 < 0)$
- Fixed points correspond to $(\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 0)$.

This is just one example of usage of the Lyapunov spectra. Let us now discuss a method for computing the spectrum.
3.1 Lyapunov Spectra (re)Defined

For the sake of the method we will employ, we must slightly adjust our idea of what the Lyapunov exponent says. The measurement of "error" will still be at heart, so not all is lost in n-dimensions.

In this new setting, consider an n-dimensional continuous dynamical system. To compute the Lyapunov spectra will be monitoring the long-term behavior of an n-sphere of initial conditions. What happens is our equations of motion will deform this initial n-sphere into an n-ellipsoid as we move forward in time. We then say the i-th one-dimensional Lyapunov exponent is defined in terms of the length of the ellipsoidal principal axis \( p_i(t) \) [1]. This is our "error" from earlier sections, and thus, we may say that:

\[
\lambda_i = \lim_{t \to \infty} \frac{1}{t} \log_2 \left| \frac{p_i(t)}{p_i(0)} \right|
\]

where \( \lambda_i \) are ordered largest to smallest. [1]

Our logarithm base of 2 is simply to keep notation consistent with that of author Alan Wolf.\(^1\), but the same qualitative analysis can be made.

3.2 Our Method for Computation

For our method, we will employ a "phase-space plus tangent-space" approach. That is, we will monitor a fiducial trajectory in the phase-space, and an initially orthonormal vector frame anchored to the fiducial trajectory as they propagate with time. [1]

Said fiducial trajectory is propagated by the equations of motion for our dynamical system. In the case of a continuous system, one should employ a numerical integration method in which control over time-step is not a problem. Our motivation is to discretize the continuous system (for the sake of our algorithm).

The orthonormal vector frame is propagated by the Jacobian matrix of the linearized equations of motion acting as a linear transformation. In the two-dimensional case, we would have something in the form of:

\[
\begin{pmatrix}
  dX_{n+1} \\
  dY_{n+1}
\end{pmatrix}
= J_{(X_n,Y_n)}
\begin{pmatrix}
  dX_n \\
  dY_n
\end{pmatrix}
\]

After each propagation step, we will apply the following formula so that we may measure the orbital divergence of each principal axis vector:

\[
L_i = \log_2 \left| \frac{\|p_i(t_n)\|}{\|p_i(0)\|} \right|
\]

Each \( L_i \) value will correspond to a Lyapunov exponent that we then obtain by dividing \( L_i \) by our entire integration time (or number of iterations) \( \tau \):

\[
\lambda_i = \frac{L_i}{\tau}
\]

\(^1\)In fact, this logarithm base is employed to keep everything in information theoretic terms
3.3 Some Issues and their Remedy

Computation of Lyapunov spectra with this outlined method is not perfect. Specifically, there are two issues that we must remedy.

3.3.1 Magnitude Divergence of Principal Vectors

The first issue we encounter is divergence in magnitude of our initially orthonormal vector frame. As we propagate our vector frame in the tangent space, we find that successive applications of the Jacobian matrix will cause the vectors to become inoperable on due to computer limitations with regards to storing values.

3.3.2 Loss of Orientation

Our second issue arises from the propagation of our vector frame as well. In this case, however, it is the loss of orthogonality of the vector frame. As we noted, the initially orthonormal vector frame will diverge in magnitude, but it will additionally "collapse" in the direction of largest growth. This collapse causes the angles between vectors to become indistinguishable, again being an issue for our computers. We need to be able to distinguish the orientation of each vector [1].
3.3.3 Our Remedy: Gram-Schmidt Reorthogonalization

Both of these issues are resolved by applying the Gram-Schmidt procedure. This procedure will allow us to obtain an orthonormal vector frame at each iteration, thus allowing our machines to work their magic. It should be noted GSR does not corrupt our algorithm. Even with repeated vector replacement, the dynamics of the system still cause each individual vector to seek out the direction in which it is growing most significantly.

Figure 6: Propagation of initially orthonormal vector frame. Divergence of principle axes and loss of orthogonality occur rapidly even when working in the tangent space.

Figure 7: Divergence of orthonormal vector frame in the Hénon Map after just three iterations. GSR resolves this issue. [1]
3.4 Algorithm summary for spectra computation in N-D

We will now summarize our procedure for the computation of Lyapunov spectra.

1. Define initial conditions. We choose an initial condition for our equations of motion, and an orthonormal vector frame that contains said initial condition in its n-sphere.

2. Integrate non-linear equations forward some time-step from some $t_i$ to $t_{i+1}$ so that we obtain a portion of the trajectory.

3. Obtain the Jacobian matrix at $t_i$ from the linearized equations of motion so that we may propagate our orthonormal vector frame forward to $t_{i+1}$.

4. Update orbital divergence values $L_i$ from our new set of propagated (non-orthonormal) vector frame.

5. Apply GSR to non-orthonormal vector frame to prepare for the next propagation step.

6. After integration time or number of iterations are completed, compute $\lambda_i = \frac{L_i}{\tau}$ to obtain the Lyapunov spectrum.
4 Results of Algorithm

Here are some results from employing the algorithm outlined above. All computations were performed in MATLAB. For the continuous systems, RK4 integration was employed with step-size $h = 0.001$ and $h = 0.01$ for the Lorenz and Rössler systems respectively. All results obtained were compared against the primary paper by Alan Wolf detailing the algorithm.

4.1 Hénon Map

**Governing Equations**

\[
X_{n+1} = 1 - aX_n^2 + Y_n \\
Y_{n+1} = bX_n
\]

**Linearized Equations**

\[
J(X_n, Y_n) = \begin{pmatrix} -2aX_n & 1 \\ b & 0 \end{pmatrix}
\]

![Hénon Map](image)

Figure 8: Hénon Map with parameter values: $a = 1.4, b = 0.3$. Number of iterations $n = 1000$.

**Published Paper Results for Lyapunov spectra:**

- $\lambda_1 = 0.603$
- $\lambda_2 = -2.34$

**Implementation Results for Lyapunov spectra:**

- $\lambda_1 = 0.6015$
- $\lambda_2 = -2.3385$
4.2 Lorenz System

Governing Equations
\[
\begin{aligned}
\dot{X} &= \sigma (Y - X) \\
\dot{Y} &= X(\rho - Z) - Y \\
\dot{Z} &= XY - \beta Z
\end{aligned}
\]

Linearized Equations
\[
J_{(X,Y,Z)} =
\begin{pmatrix}
-\sigma & \sigma & 0 \\
\rho & -1 & -X \\
Y & X & -\beta
\end{pmatrix}
\]

Figure 9: Lorenz Equations with parameter values: \(\sigma = 16, \rho = 45.92, \beta = 0.4\).
Initial Lorenz Conditions: \(\bar{x} = (0.2, 0.3, 0.5)\)
RK4 details: \(h = 0.001, t = [0, 100]\).

Published Paper Results for Lyapunov spectra:
- \(\lambda_1 = 2.16\)
- \(\lambda_2 = 0\)
- \(\lambda_3 = -32.4\)

Implementation Results for Lyapunov spectra:
- \(\lambda_1 = 2.0686\)
- \(\lambda_2 = -0.0115\)
- \(\lambda_3 = -32.3537\)
4.3 Rössler System

**Governing Equations**
$$\begin{align*}
\dot{X} &= -(Y + Z) \\
\dot{Y} &= X + aY \\
\dot{Z} &= b + Z(X - c)
\end{align*}$$

**Linearized Equations**
$$J_{(X,Y,Z)} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & a & 0 \\ Z & 0 & X - c \end{pmatrix}$$

Figure 10: Rössler Equations with parameter values: $a = 0.15, b = 0.20, c = 10$.
Initial Rössler Conditions: $\bar{x} = (0.2, 0.3, 0.5)$
RK4 details: $h = 0.01, t = [0, 1000]$.

**Published Paper Results for Lyapunov spectra:**
- $\lambda_1 = 0.13$
- $\lambda_2 = 0$
- $\lambda_3 = -14.1$

**Implementation Results for Lyapunov spectra:**
- $\lambda_1 = 0.1258$
- $\lambda_2 = 0.0047$
- $\lambda_3 = -14.1455$
5 Computation of Maximal Lyapunov Exponent from Time-Series Data

In the previous sections, Lyapunov spectra was computed from systems in which we explicitly knew the governing equations. In many real-world scenarios, however, we do not have access to this information and often times only have data in the form of a time-series. Because we lack knowledge of the governing equations for some suspect chaotic system, all techniques we have discussed so far go out the window. With that said, all hope is not lost. In this section, we will discuss a method for computing the maximal Lyapunov exponent from time-series data.

5.1 Phase-Space Reconstruction: An Quick Chat

In our new setting, we only have access to one "dimension" of the system we want to determine the maximal Lyapunov exponent of. This is problematic. Consider the task of knowing what the Lorenz system looks like by only being given the $\dot{X}$ component. This issue, however, can be resolved with Takens' Embedding Theorem [4] [6]. Admittedly, Takens' Theorem was exploited in this work to obtain the later results without a strong intuition on how the theorem truly functions. On the other hand, it is understood that we are able to reconstruct the phase-space in which an attractor resides, and, through this, reconstruction from a given time-series data will allow us to obtain the Lyapunov exponents of the original system [4] [6]. This is a truly incredible result. Many other interesting applications follow from Takens' Theorem, but we will only be reconstructing attractors here.

5.2 Phase-Space Reconstruction: Time-Delay Embedding

Given a time-series of data, it was claimed we can reconstruct the dynamics of the phase-space. This will be done through time-delay embedding. With our time series $x(t)$, we can define an $(n+1)$ dimensional embedding as

$$[x(t), x(t + \tau), \ldots, x(t + n\tau)]$$

where $\tau$ determines the amount of "lag" and essentially the "accuracy" of our reconstructed system[6]. So through time-delay embedding, we can obtain a pseudo $\dot{Y}$ and $\dot{Z}$. Here are some plots of the Lorenz System with varying $\tau$ values. Our "time-series" is simple the $\dot{X}$ of our original equations and nothing else.

The sweet-spot $\tau$ value appears to be at $\tau = 30$, and attempts to compute the maximal Lyapunov exponent from the reconstructed attractor were done on said data set and reconstructed attractor.

Remark: This is a not a trivial task. Working with time-series data has felt very much like an art due to the extreme sensitivity to changes in parameters that were encountered. In fact, some crucial parameters for implementing this method have no "best" method of having their value determined. That is not to say methods do not exist, but much trial and error was conducted in order to get the results that will be presented. Essentially, the enthusiastic beginner should proceed with caution.
Figure 11: $\tau = 1$

Figure 12: $\tau = 30$

Figure 13: $\tau = 90$
5.3 Our method for computing the Maximal Lyapunov Exponent

Our goal is to mimic the previous algorithm for computing the Lyapunov spectra of n-dimensional systems as much as possible. The ideas cannot be directly implemented because of the reasons discussed, but we will attempt to recreate our "phase-space plus tangent-space" in previous setting.

5.3.1 Outlining the Algorithm

Since our time series recreates an attractor from a single trajectory, we must look at neighboring points in the data file and treat them as neighboring trajectories. That is, we will monitor the long-term evolution of a nearby pair of points on the reconstructed attractor [1]. With this idea, we will say two points whose distance apart is sufficiently small will represent the initial state of our principal axis we are trying to determine the Lyapunov exponent of. We will take this distance to be called $L(t_0)$.

In order to mimic the propagation of vector our vector frame, we will simply traverse through our data file a fixed time step and monitor the new distance $L'(t_1)$ that occurs between data points.

To measure the orbital divergence $D$ in a similar manner to previous algorithm, we simply say

$$D = \frac{L'(t_1)}{L(t_0)}$$

This effectively measures the divergence in our pseudo trajectories.

At this point in the previous algorithm, we employed GSR. In order to do this with our reconstructed attractor, we must settle on finding the nearest neighbor to the point on our fiducial trajectory at time $t_1$. The nearest neighbor criteria is to have a sufficiently short distance, but also to have an orientation error within a defined threshold. When this replacement point $P$ is found, we say the vector between the current point on the fiducial trajectory and $P$ is the new principle axis vector that we repeat the propagation process on.
We continue this process of updating $D$ and finding replacement points $P$ until our fiducial trajectory has reached the end of the data file. To then obtain the maximal Lyapunov exponent, we simply compute

$$\lambda_{\text{max}} = \frac{D}{T_u}$$

where $T_u$ is the number of time steps in our time-series data file.

### 5.4 Summary of MLE Algorithm with Illustration

![Figure 15: [5]](image)

1. We begin at $t_0$ on our fiducial trajectory (the first point in our reconstructed attractor data file).
2. Upon finding the nearest neighbor, we step forward a fixed time-step value $N$ and measure $L'(t)$.
3. Our pseudo GSR is employed, finding a new point $P$ that gives us the distance $L(t_1)$ pictured.
4. We repeat until fiducial trajectory reaches the end of the data file at time $t_n$.
5. Compute $\lambda_{\text{max}} = \frac{D}{T}$.

**Remark:** This is the very baseline for implementing this method. Additionally, one should define some constants such as distMin and distMax in order to limit the scope of the nearest neighbor search. This effectively helps combat noisy data and extraneous cases where the "nearest-neighbor" happens to be on the other side of the attractor (thus giving inaccurate orbital divergence values). On top of this distance window we define, we must also define some angleMax to give us a threshold for orientation error. As stated at the beginning of this section, this method has proven to be extremely tricky to nail due to the sensitivity on parameter values such as our time-delay $\tau$, our fixed time-step value $N$, distMin, distMax, and angleMin. At the end of the day, it has become apparent working with time-series data is an entirely different beast (see [1] for an extensive analysis).
6 Results of MLE Algorithm

Here are the results upon attempting to implement this algorithm on the Lorenz system. The algorithm logic implemented in MATLAB is without a doubt sound, but the method for choice of parameter values is definitely in question. Here is a rundown of what was achieved:

6.1 Reconstructed Lorenz

Parameters of System

- $\sigma = 16, \rho = 45.92, \beta = 0.4$
- Initial Lorenz Conditions: $\bar{x} = (0.2, 0.3, 0.5)$
- RK4 Time-Step: $h = 0.01$
- RK4 Integration Time: $[0 : h : 100]$
- Time-Delay Parameter: $\tau = 30$
- Fixed-Time-Step: $N = 100$ (1 second).
- $\text{distMin} = 0.001$
- $\text{distMax} = 10$
- $\text{angleMax} = \frac{\pi}{6}$

Published Paper Results for MLE:

- $\lambda_{\text{max}} = 2.16$

Implementation Results for MLE

- $\lambda_{\text{max}} = 2.305$

Figure 16: Corresponding Reconstructed Attractor
7 Future Work

Fully understanding Takens’ Embedding Theorem and the process of choosing parameters for phase space reconstruction is on the top of my list. From here, the process of calculating Lyapunov exponents from general time-series chaotic data can be done. In addition to this, I plan on implementing the procedure for computing the positive Lyapunov spectra from time series data. In [1], the algorithm is outlined, but even author Alan Wolf states it is a challenge. Finally, I was asked the question of ”what if there is more than one attractor in the phase-space”, and admittedly, I had never thought of such a scenario (if it is possible).

8 Acknowledgements

I would like to thank my mentor Ken Caluya for guiding me through all the hurdles that came my way. I look forward to many more late nights debugging code and reading math papers next quarter. Additionally, I would like to thank Professor Venturi and Tenavi for their efforts in coordinating the dynamical systems course this quarter. Thank you both for being great educators; always taking the time to answer my questions or even strike up short conversation has made all the difference.

Thank you all for being a part of my first quarter here at UC Santa Cruz, and thank you for inspiring me to reach new heights.
References


https://plus.maths.org/content/extracting-beauty-chaos


http://cnx.org/contents/939eff33-c4f0-43e4-ae2b-ca6d5b58c267@2.


http://node99.org/tutorials/ar/
