Exploring the Relationship Between Coupled Chaotic Systems and Their Higher Order Lyapunov Exponents: Final Report Summer 2008

Dustin Keys and Alex Blount
Advisor: Dr. Robert Indik

August 7, 2008

1 Introduction

The study of nonlinear dynamical systems has uncovered rich structures and provided us with insight into events for which are beyond grasp of common intuition. One such fascinating result is in the coupling of chaotic systems. When multiple chaotic systems are coupled in such a way that their future state depends not only on the individual systems current state, but also on the state of the other systems, they can actually synchronize. The multiple systems, each chaotic in their own respect, will actually have the behavior of just one chaotic system, but chaotic nonetheless. For this to occur the amount by which the individual systems share in each other's dynamics, the coupling strength, need not be huge, and depends on how chaotic the individual systems are. The conditions necessary for chaotic systems to synchronize have been studied and are supplemented by rigorous proofs [?][?][?]. The goal of this project is to study how the dynamics of coupled systems depend on other properties of the individual systems, and the nature of the coupling.

2 Theory

The question of how to quantify chaos is a tricky one. Since an important property of a chaotic system is its sensitivity to initial conditions, a widely agreed upon measure of chaos is the *Lyapunov Exponent*, LE for short. For a discrete scalar dynamical system, $x_{n+1} = f(x_n)$ this number quantifies the amount by which small perturbations from a trajectory of a scalar dynamical system are amplified, and is defined as

$$\lambda = \lim_{N\to\infty} \frac{1}{N} \ln \left| \prod_{k=0}^{N-1} f'(x_k) \right| \tag{1}$$

In the scalar case, the derivation is relatively straightforward. The second iteration of the system above is $x_2 = f(x_1) = f(f(x_0))$ and the derivative is $f'(x_1) = f'(f(x_0))f'(x_0)$ which is just the chain rule. Then a linearization of the system will give an estimation of the growth at each step, which may widely fluctuate, and an average as $n \to \infty$ will give the average rate of growth. Taking the derivative for the linearization at the *nth* step, (remember the chain rule):

$$f'(x_n) = f'(x_{n-1})f'(x_{n-2})\dots f'(x_1) = \prod_{k=0}^{n-1} f'(x_k)$$

and the geometric mean gives Equation 1. Then e^{λ} measures the average separation per step after N iterations of the initially perturbed trajectories, infinitesimally close at the start.

When we consider the linearization of a multi-dimensional map, the derivative at the *nth* iteration is given in the form of the Jacobian Matrix, F'_n. The chain rule applies as it did in the scalar case, and to get the average rate of growth when these are multiplied at each step we have to instead consider a sort of geometric mean in the operator sense and take the limit as $n \to \infty$:

$$\lim_{n\to\infty}\left(\sqrt{(F'^*_n)F'_n}\right)^{1/n}$$

Where F'^*_n is the adjoint of F'_n. Then when taking the eigenvalues of the matrix that results gives the *singular values* corresponding to the map, which measure the amount of stretching (or contracting) of a linear operator in each dimension it operates on, and the logs of the eigenvalues give the LEs. For example if you apply a 2×2 matrix to the unit circle turning it into some arbitrary ellipse, the first and largest singular value gives the length of the semi-major axis and the second singular value gives the length of the semi-minor axis. In order to find these values for a matrix, a process called *Singular Value Decomposition*, or SVD, must be used on the matrix, see Section 5. This stretching gives a direct parallel with LEs, so it is not hard to see, conceptually at least, why the two are related. Positive LEs means chaos, and they tell us that the system is stretching and shifting in some way along at least one direction. These correspond to singular values which are greater than one, where the shift is described by the singular vectors. Negative LEs means stability, which means that the system is heading asymptotically toward some fixed state. These correspond to singular values between one and zero which compress the system along some direction, given by the singular vectors.

There are some subtleties in chaos that can help create an analogy with which to understand it. Consider a map which simply stretches its domain. The behavior of this map alone does not exhibit chaotic qualities since a simple stretch, no matter the magnitude, is very predictable. However, add another transformation, a folding, such that the domain for the map is very dynamic, and chaos emerges. Now initially close points are growing apart and being reordered, much like how a baker would mix dough. This notion of stretching and folding turns out to be a very useful analogy for understanding chaos. It illustrates the difference between randomness and chaos: chaos is deterministic, that is the state of a chaotic system is predetermined based on its initial conditions, even if similar conditions can have very different trajectories.

To define synchronization we require that states of the systems converge to each other in the limit as $n \to \infty$. In the case of two maps whose values after each iteration are given by x_n and y_n, this can be written as $\lim_{n\to\infty}(x_n - y_n) = 0$. The example that follows is a symmetric method of coupling in which the arguments for two copies of a map are replaced with a weighted mean of the state of both of the systems. Another method is to leave one copy alone and to replace the other copy's arguments with the weighted mean of the two system's values, so there is a sort of master/slave relationship. It is relatively straight forward to show that the point at which synchronization occurs is directly related the LE for a one dimensional system, and also to the method of coupling.

Theorem 1. *Consider the following method of coupling:*

$$\begin{aligned}x_{n+1} &= f((1-\epsilon)x_n + \epsilon y_n)\\ y_{n+1} &= f((1-\epsilon)y_n + \epsilon x_n)\end{aligned}$$

Synchronization occurs when $\epsilon < \frac{1-e^{-\lambda}}{2}$

Proof. Let $\bar{x}_0 = \frac{x_0+y_0}{2}$ and $\delta_n = x_n - y_n$. Then $\bar{x}_n + \frac{1}{2}\delta_n = x_n$,
$\bar{x}_n - \frac{1}{2}\delta_n = y_n$ and

$$\begin{aligned}x_{n+1} &= f((1-\epsilon)(\bar{x}_n + \delta_n/2) + \epsilon(\bar{x}_n - \delta_n/2))\\ &= f(\bar{x}_n + \tfrac{1-2\epsilon}{2}\delta_n)\\ y_{n+1} &= f((1-\epsilon)(\bar{x}_n - \delta_n/2) + \epsilon(\bar{x}_n + \delta_n/2))\\ &= f(\bar{x}_n - \tfrac{1-2\epsilon}{2}\delta_n)\end{aligned}$$

We linearize near synchronization, i.e. around $\delta_n = 0$

$$\begin{aligned}\delta_{n+1} &= \left(f(\bar{x}_n) + \tfrac{1-2\epsilon}{2} f'(\bar{x}_n)\delta_n + ...\right) \\ &\quad - \left(f(\bar{x}_n) - \tfrac{1-2\epsilon}{2} f'(\bar{x}_n)\delta_n + ...\right) \\ &= (1-2\epsilon) f'(\bar{x}_n)\delta_n\end{aligned}$$

Let $T^n(x_n) = f'(x_0)f'(x_1)...f'(x_n)$

We note that for synchronization to occur, $\lim_{n\to\infty} |(1-2\epsilon)^n f'(\bar{x}_n)| < 1$ since δ_n must go to zero. Also since we are linearizing around $\delta_n = 0$, $f(\bar{x}) = f(x_n)$. We insist that the geometric mean will be less than one if the limit is and we find the following relationship:

$$\begin{aligned}\lim_{n\to\infty} |(1-2\epsilon)^k \textstyle\prod_{k=0}^n f'(x_k)|^{1/k} &< 1 \\ \lim_{n\to\infty} \ln |(1-2\epsilon)^k \textstyle\prod_{k=0}^n f'(x_k)|^{1/k} &< 0 \\ \lim_{n\to\infty} \ln |(1-2\epsilon)| + \tfrac{1}{k} \ln |\textstyle\prod_{k=0}^n f'(x_k)| &< 0 \\ \ln |(1-2\epsilon)| + \lambda &< 0 \\ 1-2\epsilon &< e^{-\lambda} \\ \epsilon &< \frac{1-e^{-\lambda}}{2}\end{aligned}$$

$\square$

Thus synchronization for copied systems is fundamentally dependent on the LEs of the original system. For multidimensional systems the derivative becomes a Jacobian Matrix, and it becomes much harder to show that even in the case where there are multiple LEs total synchronization is still decided by the 1st LE for coupling with the weighted mean. The general proof for higher dimensions can be found in an earlier paper by Dutson [?]. The basis of the relationship between synchronization and the LEs is the linearization, which means that the dynamics of a synchronized system are determined by the 1st LE only *near synchronization*, according to the proofs. The working goal of this project is to find any relationship between synchronization and the higher order LEs that may be lurking. To understand the reasoning behind this motive, we must introduce the concept of dimension.

2.1 Dimension

Dimension comes in many different mathematical flavors, each important in their own respect. The basic concept behind defining dimension is how structure changes over scale. For example, if you take a line and scale it down by a half, then it will take two copies to reconstruct the original line. A square scaled by one half will require four copies to reconstruct, and a cube will require eight. This relationship is described by a power law, that is the number of copies required to reconstruct the original structure, N, depends on the scale, $1/s$, by:

$$N(s) = \left(\frac{1}{s}\right)^{d_c} \quad \text{(Covering Dimension)}$$

where d_c is the covering dimension of the structure. To put this on a firm basis, the covering dimension describes how the number of open sets required to cover a space changes with the size of the covering set. Now consider a set of points with three coordinates, if these points densely fill up a plane, though they may be embedded in a 3-dimensional space the covering dimension is still 2 because the points scale 2-dimensionally. The structure that develops from chaotic systems often has a non-integer, or fractional, dimension. We call these sets *fractals* and they exhibit a self-similarity different from that of the space in which they reside (the *embedding dimension*). For example the Koch Curve, a well known fractal, when scaled by a third takes four copies to recreate, resulting in a fractal dimension of $\frac{\log(4)}{\log(3)} = 1.2619$. The terms 'covering dimension' and 'fractal dimension' can be used interchangeably though fractal dimension usually specifies a covering dimension which is non-integer. Conceptually, the reason chaotic systems exhibit fractal structure can be understood with the stretching and folding analogy. They can operate on a domain with an integer dimension, and through stretching and folding can transform the domain into a non-trivially self-similar structure. Other types of dimension include the Hausdorff Dimension which is usually the same as the covering dimension though its definition is slightly more technical, and the Information Dimension which quantifies the amount of information gained when decreasing the size of covering sets, and is less than or equal to the Hausdorff Dimension.[1] This can be thought of as the amount of information gained when analyzing a set with increasing resolution. Another dimension we'll be dealing with is the Lyapunov Dimension.

2.2 Lyapunov Dimension

The Lyapunov dimension is an abstract concept that deals with how the LEs of a system scale. Consider a general dynamical system with n LEs, ordered from largest to smallest. If we define a function

$$\gamma(k) = \sum_{i=1}^{k} \lambda_i \quad k = 1, ..., n \quad (\gamma(0) = 0)$$

and extend the function to include all real numbers in a piece-wise linear fashion, then the point (other than zero) where γ is zero is called the Lyapunov Dimension, d_L, that is $\gamma(d_L) = 0$. In other words

$$d_L = m + \frac{1}{|\lambda_{m+1}|} \sum_{k=1}^{m} \lambda_k \quad \text{(Lyapunov Dimension)} \tag{2}$$

where m is the maximum integer for which $\gamma(m) \geq 0$ The Kaplan-Yorke Conjecture is that in most cases this dimension is the same as the Information dimension, a lower bound for the Hausdorff Dimension. [?][?], which is nice because in most cases calculating the Hausdorff dimension is much more difficult.

The dimension of the attractor for a coupled system provides some insight into synchronization. The obvious implication is that when synchronization occurs the dimension will decrease to that of the attractor for the single system. Since Kaplan-Yorke suggests a deep relationship between this dimension and the LEs of the system, it is worth exploring *how* the dimension of the attractor changes as this is one way to see if the higher order LEs are affecting the synchronization.

3 Test Cases

In order to investigate these dynamics a few typical maps were chosen.

$$\textbf{General Quadratic Map} \quad x_{n+1} = a - x_b^2 \tag{3}$$

$$\textbf{Logistic Map} \quad x_{n+1} = ax_n(1 - x_n) \tag{4}$$

$$\textbf{Hénon Map} \quad \begin{aligned} x_{n+1} &= y_n + 1 - ax_n^2 \\ y_{n+1} &= -bx_n \end{aligned} \tag{5}$$

and a generalized map in which the LEs are easily tuned:

$$\textbf{Baker's Map} \quad \begin{aligned} x_{n+1} &= \left\{ \begin{array}{cc} \mu_a x_n & \text{if } y_n > \alpha \\ \frac{1}{2} + \mu_b x_n & \text{if } y_n < \alpha \end{array} \right\} \\ y_{n+1} &= \left\{ \begin{array}{cc} \frac{1}{\alpha} y_n & \text{if } y_n > \alpha \\ \frac{1}{1-\alpha}(y_n - \alpha) & \text{if } y_n < \alpha \end{array} \right\} \end{aligned} \tag{6}$$

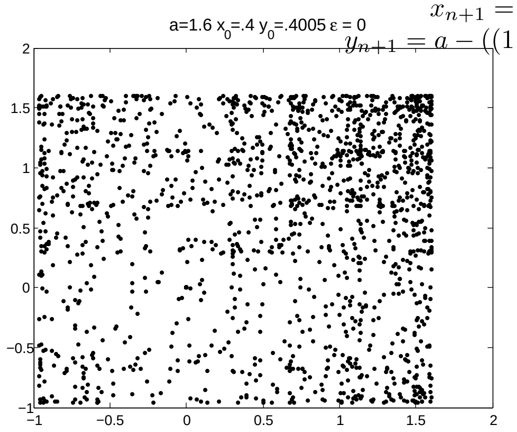
[1]These dimensions are part of a collection of dimensions known as Rényi Dimensions, which uses an expanded notion of information to define a whole family of dimensions

3.1 The Coupled Quadratic Map

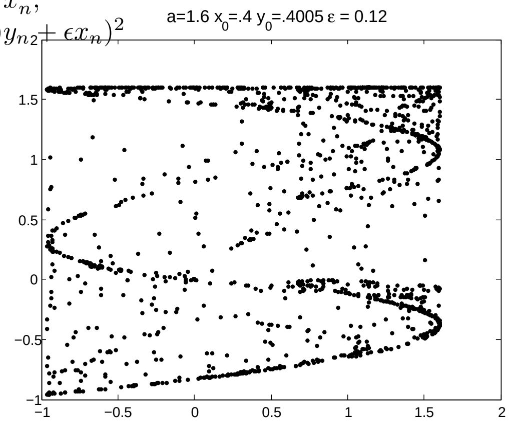
The behavior of quadratic maps is a fascinating subject in and of itself. The first goal of the project was to understand this behavior and the previous work that has been done on it, but the subject does not merit revisiting in this paper. What is important about these maps for our purposes is how they behave when synchronized. To visualize this we can consider two copies of the system and plot their respective values against each other. If for example two copies of a map were synchronized then plotting the respective values of the system at each iteration against the other system would produce a diagonal line. After every iteration the next point would also be on the line but its behavior on that line is chaotic.

Figure 1: The Coupled Master/Slave Quadratic Map:

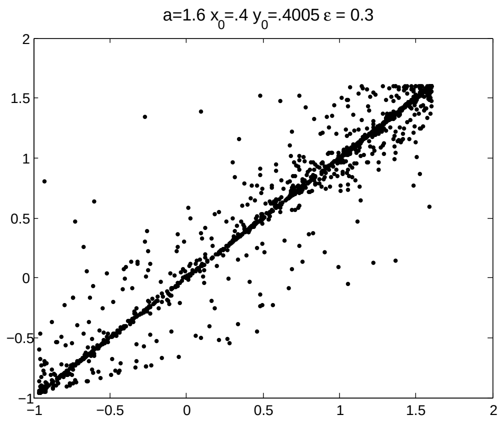
$$x_{n+1} = a - x_n^2,$$
$$y_{n+1} = a - ((1-\epsilon)y_{n2} + \epsilon x_n)^2$$



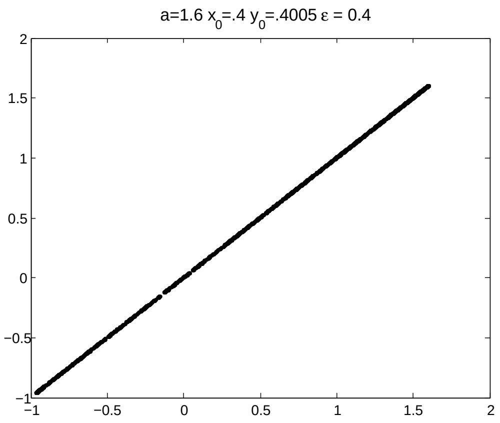
(a) Unsynchronized



(b) Far from synchronization



(c) Near synchronization



(d) Fully synchronized

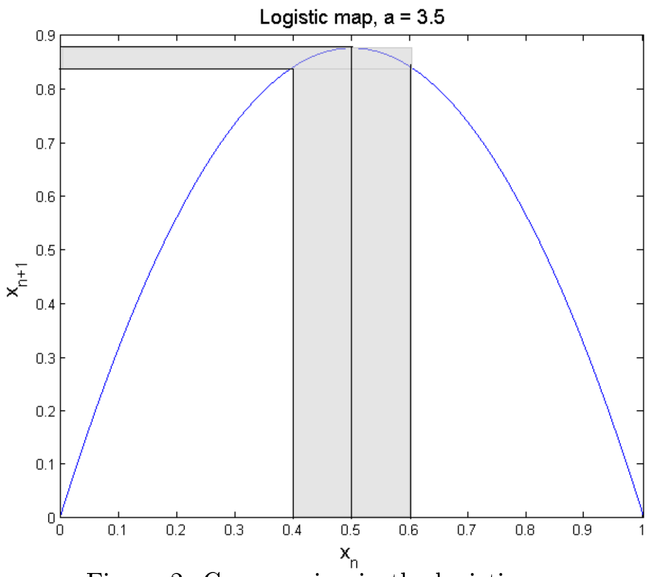


Figure 2: Compression in the logistic map

Figure 1.a shows a fully unsynchronized quadratic map. The x-axis is the master system and the y-axis is the slave. Since there is relatively little correlation between the two systems, the graph just shows a box in which the system's values are located. Figure 2.b is slightly closer to synchronization but still far from it. The interesting thing about this frame is that there is definitely structure present. There is a certain condensation of the chaotic regions on ambiguous curves which move through the frame. This new structure is one which is very interesting, but clearly isn't related to higher order LEs as the system only has one. More likely, the condensation comes from the geometry of the semi-coupled dynamic system. Condensation in the quadratic map is due to the fact that the next value is decided by a parabolic curve, which has an extremum. This extremum condenses a section of the domain into a smaller section in the range, not affecting the chaotic nature of the transformation but merely weighting the outcome, as Figure 2 shows. Now in the coupled map the next y-value is decided by both the previous y-value and the previous x-value, so the new underlying geometry is a surface, not a curve, providing more possibility for condensation. This is what appears to be happening to this coupled system, though the phenomenon has not been fully explored. In Figure 1.c, the system is nearing synchronization and is restricting itself as most of the points are on the diagonal. Figure 1.d shows the completely synchronized map, so the graph shows a diagonal line indicating that the two systems values are equal at every step. The fact that there is so much structure already present merely from the nature of the system makes it very difficult to explore any that may be due to higher order LEs. The best approach for us is to calculate the fractal dimension of the entire system and look for some significant change in dimension which can be linked to the higher order LEs. Even with powerful computers, this is not an easy task.

4 Calculating Fractal Dimension

Recall that the basic concept behind calculating the fractal dimension is to find how the number of covering sets required to contain the data points scales with the radius of the covering sets. So the algorithm amounts to specifying the size of a neighborhood, picking a random point in the set and eliminating its neighborhood from the data set, pick another point and repeat. Eventually, all the data points will be removed and however many times a neighborhood was removed is how many covering sets was required. For a two, and even three, dimensional coupled system this is made easy by the use of a histogram. Simply specify the resolution of the histogram, s, and count the number of non-empty boxes, N. From the definition of covering dimension we can derive the equation for the dimension:

$$N(s) = \left(\frac{1}{s}\right)^{d_f}$$

Take the log of both sides and bring the power down to the front.

$$\log(N(s)) = d_f \log\left(\frac{1}{s}\right)$$

So making a log-log plot of N vs. $1/s$ will give you a line whose slope is the fractal dimension. Note that s for a histogram is the number of boxes per side of the frame, the radius of the covering set is then $1/s$ in terms of the length of that side. For higher dimensional attractors analysis a more direct approach is needed.

The principal is the same as the histogram, or box-counting, approach except that we must define neighborhoods by their euclidean distances and that means calculating the distance to *every* point in the data set, otherwise there is no way to tell which points are in the neighborhood. The actual algorithm amounts to:

Step 1: Pick a random point in the set, and specify a radius for the covering set

Step 2: Calculate the distance to every other point in the set

Step 3: Remove all points which are within the neighborhood of specified radius from the data set

Step 4: Pick another point and do the same until all points have been removed, the number of neighborhoods removed is the number of covering sets for that radius

Step 5: Vary the radius and generate a log-log plot whose slope is the dimension

4.1 Issues

There are a few critical issues with this approach from both a numerical and a theoretical stand point. Numerically, this is a *very* expensive calculation, even though the number of points in the data set is decreasing after each neighborhood is removed, there are still thousands of distances to calculate. Also, once the size of the neighborhood approaches the smallest (or largest) distances between the data points, the number of covering sets will max out (or reach 1, the obvious minimum if the size is too large) resulting in a skewing of the line fit and hence a lower dimension. To deal with this problem, only radii in a certain middle range should be considered. So a step can be added to find the mean distance between the points when the number of covering sets is exactly half of the number of points in the data set. This makes it possible to use less radii for the line fit, since only the linear part of the graph is even being considered. From a theoretical standpoint, there is a problem with using random points in the set as one can never be sure that the output is a good representation of scaling in the set, really the only way to overcome this is to average the calculations over many trials, which once again increases the amount of calculations which need to be executed. There are also theoretical problems with using a finite number of points on the attractor to begin with, as one is never sure these points are an accurate representation of the attractor. If the system is *ergodic*, that is a trajectories are evenly distributed throughout the attractor, then simply increasing the number of points used will ensure that a good representation of the attractor is being examined. These issues are very important in getting a good accurate number for the fractal dimension. It helps to check the algorithm against a map which has a known or solvable fractal dimension. The Baker's Map is one such map.

4.2 Baker's Map

The nice thing about this map is that it is designed to give manageable analytical solutions for certain properties of the map like fractal dimension and LEs. The map requires three parameters μ_a, μ_b, and α. Also let $\beta = 1 - \alpha$ The map was designed to have a diagonal Jacobian matrix.

This means if we take the product of the Jacobian matrices at each step then the geometric mean of each entry along the diagonal give the LEs over this product:

$$\lambda_1 = \alpha \log \frac{1}{\alpha} + \beta \log \frac{1}{\beta} \tag{7}$$

$$\lambda_2 = \alpha \log \mu_a + \beta \log \mu_b \tag{8}$$

It is also possible to obtain an equation for the Lyapunov dimension, using the definition from Equation 2:

$$D_L = 1 + \frac{\lambda_1}{\alpha \log \frac{1}{\mu_a} + \beta \log (1 + \mu_b)} \tag{9}$$

and even an equation for the capacity dimension can be obtained in the form of a transcendental equation:

$$1 = \mu_a^{(d_c - 1)} + \mu_b^{(d_c - 1)} \tag{10}$$

The derivations of these equations are found in a paper by J. Farmer, E. Ott, and J. Yorke [?].

4.3 Calculations

Using the parameters $\mu_a = 0.8$, $\mu_b = 0.4$, and $\alpha = 0.7$, Equation 10 gives a number for the capacity dimension of about $d_C = 1.4215$. This gives a way to check for errors in the fractal dimension program. The following table shows the fractal dimension attractor, embedded in 4 dimensions, as it varies over the coupling strength. The first column is the coupling parameter, the second is the mean fractal dimension over 10 trials, and the third is the standard deviation over those 10 trials.

As the coupling strength is increased in the table, the dimension of the attractor is decreasing because the amount of freedom the system has is decreasing. The most important thing to realize about this table is that the dimension given by Equation 10 was 1.4215, whereas the calculated dimension after synchronization here is about 1.51. This is evidence that there is a small bias in the program which is *consistently* pushing up the dimension, since the standard deviations are relativly small. This bias is sensitive to the number of points used [2] and that hints that maybe the Information Dimension might be a better measure of dimension for our purposes. The effect is currently being investigated.

Ignoring the bias for the time being, there are certain interesting things happening in the data. For instance it is very interesting that before total synchronization, there are a few places in which the fractal dimension drops significantly. Though this is probably not related to our hopes of seeing a drop in fractal dimension which can be tied to the higher order LEs, it is an interesting question in and of itself. The reason for ruling out LEs as the source for these drops is that the dimension goes back up rather quickly. We would expect the dimension to stay down if the system was partially synchronized by the higher order LEs. The reason for the drop also merits further investigation. Eventually, we will need to deal with systems which have many positive higher order LEs. This means more dimensions, more calculations and more numerical issues.

If dimension calculations are to be used as a measure of synchronization is would be wise to understand the reason behind the changes in dimension so that it is easy to tell what can be attributed to syncrhonization.

5 Calculating the Lyapunov Exponents

One of the motivations for this research is that perhaps synchronization will provide a simpler method to calculate a systems Lyapunov exponents. If we knew the relationship between synchronization and the LEs then we could get those LEs simply by finding when a system synchronizes, which is much easier then direct methods. For one dimensional maps the Lyapunov exponent is

[2] 5000 points were used in generating this data

ϵ	d_f	std
0.000	2.7693	0.0548
0.100	2.5450	0.0291
0.200	2.2988	0.0377
0.210	2.1719	0.0499
0.220	2.1355	0.0465
0.221	2.0215	0.0819
0.222	1.9565	0.0465
0.223	1.9806	0.0706
0.224	1.9777	0.0687
0.225	2.0792	0.0648
0.226	1.9542	0.0638
0.227	2.0705	0.0400
0.228	1.9697	0.0672
0.229	1.5174	0.0376
0.230	2.1305	0.0560
0.221	2.0040	0.0811
0.222	2.0062	0.0495
0.223	2.0103	0.0877
0.224	1.9598	0.0750
0.225	2.0955	0.0409
0.226	1.9944	0.0594
0.227	2.0640	0.0258
0.228	2.0319	0.0736
0.229	1.4989	0.0486
0.230	2.1649	0.0358
0.231	1.5336	0.0223
0.232	1.5074	0.0340
0.233	1.5197	0.0529
0.234	1.5148	0.0377
0.235	1.5240	0.0536
	avg	0.0531

Table 1: Fractal Dimension Vs. Coupling Parameter

easy to calculate. A large number of iterations is used and each is plugged into the derivative. The primary, and only, LE is then either the log of their products, or the sum of their logs. The latter was used in order to keep the value of the LE under control at every step.

As we begin to look at higher dimensional systems, or coupled lower dimensional systems, things not only become more complicated theoretically, but numerically. The introduction of matrices creates a new set of problems. In order to easily handle matrices and their operations in C, the GNU Scientific Library (GSL) [?] was used. Many of its matrix and linear algebra routines were based on BLAS and LAPACK.

The Lyapunov exponent of the Hénon map posed some problems. Now, we are dealing with the product of matrices as opposed to scalars. And instead of taking the log and geometric mean, we find the singular values. The Jacobian of map is relatively simple for the Hènon map.

$$F' = \begin{pmatrix} -2ax_n & 1 \\ b & 0 \end{pmatrix}$$

Unfortunately, when we begin to multiply these matrices together, they become ill-conditioned extremely quickly, which means the upper two entires have become extremely large, while the bottom two have become extremely small and this brings up numerical issues. As a start, only the primary exponent was calculated. To accomplish this, two systems were iterated, with one slightly perturbed from the other and their separation was measured after each step. In order to prevent the two trajectories from becoming parallel, they were re-orthoganalized after every step. This method was outlined by Sprott [?]. The C function calculated the first exponent to be .419277. Then the second exponent was then calculated with a trick. We know the total stretching or compressing done by a matrix is equal to the absolute value of its determinant. And in the case of the Hénon map the determinant of the Jacobian is:

$$\begin{vmatrix} -2ax_n & 1 \\ b & 0 \end{vmatrix} = -b$$

So the total stretching or compression is given by $|b|$. We also know that the total stretching or compressing done by the map is given by $e^{\sigma_1+\sigma_2}$, with σ_n as the n-th LE. This leads to the following relationship.

$$\begin{aligned} |b| &= e^{\sigma_1+\sigma_2} \\ \ln|b| &= \sigma_1+\sigma_2 \\ \ln|b|-\sigma_1 &= \sigma_2 \end{aligned}$$

And in most cases we use the values $a = 1.4$ and $b = .3$.

$$\sigma_2 = \ln(.3) - .419277$$

$$\sigma_2 = -1.62325$$

As well as this method works in this case, it scales up very poorly. It has too many unknowns in higher dimensions.

$$\sigma_n = \ln|\det(F')| - (\sigma_1+\sigma_2+\cdots+\sigma_{n-1})$$

In addition the determinant may not be as simple as $\ln(.3)$. It could contain x's y's (or any other variables for that matter.

In order to calculate both LE's using the singular value decomposition, the matrix product must be kept from becoming ill conditioned. As all the matrices are multiplied together, the primary LE dominates and the product begins to reflect *only* the primary LE as the the trajectories are stretched the most in that direction, and they slowly become parallel. The first attempt to control this was with QR factorization using householder reflections. Let A be the product of matrices.

$$A = A_nA_{n-1}\dots A_2A_1$$

After each step, the product up to the first k matrices, A_k, would be factored into a Q_k and R_k. Then before A_{k+1} is multiplied, it is factored into Q' and R'.

$$\begin{aligned} A_k &= Q_kR_k \\ A_{k+1} &= Q'R'Q_kR_k \end{aligned}$$

Then R' and Q_k are multiplied together and refactored.

$$A_{k+1} = Q'Q''R''R_k$$

And finally, the Q's and R's are multiplied together giving Q_{k+1} and R_{k+1}. This method helped to slow the bleeding somewhat, but it didn't go far enough.

The first method of simply multiplying all the Jacobian matrices together gives a value of .409876 at 1000 iterations. At first this looked promising, and it was hoped that it would converge, however by 1500 iterations it reached .421202, and before to long, it gave inf. At no point did this method give a worthwhile value for the second exponent. It went quickly to $-inf$.

Using the QR refactorization, we saw similar behavior. The first exponent converged to approximately the same value as before, however it did so after fewer iterations. As for the second exponent, while giving a no more accurate answer, it held up for more iterations before "hitting" $-inf$.

In order to look at things from the ground up, a new matrix pseudo-class was written in C along with linear algebra functions to go with it. The QR decomposition function was done using *householder transformations.* Given a unit vector $\mathbf{u}$, the householder matrix of $\mathbf{u}$ is defined as follows.

$$\mathcal{H}(\mathbf{u}) = \mathbf{I} - 2\mathbf{u}\mathbf{u}^{\mathrm{T}}$$

If a matrix is multiplied by a householder matrix on its right side, the columns of the matrix will be reflected around $\mathbf{u}$. If multiplied on the left, the rows will be reflected. In the case of the QR decomposition, we are not given what vector to reflect around, but rather what we want to reflect into. But given this, the householder vector can be calculated easily. If we wish to reflect $\mathbf{a}$ into $\mathbf{b}$, we can just take their difference and normalize.

$$\frac{\mathbf{a} - \mathbf{b}}{||\mathbf{a} - \mathbf{b}||} = \mathbf{u}$$

5.1 The QR Factorization

We begin with A, an $n \times n$ matrix. The first step is to transform the first column of the matrix, call it $\mathbf{v}_1$ into $||\mathbf{v}_1||\mathbf{e}_1$. This column could also be seen as the first column of the $(0, 0)$ minor of A $(A_{0,0})$. To construct the first householder matrix we first find the householder vector.

$$\frac{\mathbf{v_1} - ||\mathbf{v}_1||\mathbf{e}_1}{||\mathbf{v_1} - ||\mathbf{v}_1||\,\mathbf{e}_1||} = \mathbf{u}_1$$

Here, $\mathbf{u}_1$ is an n-dimensional unit vector. Then $\mathcal{H}(\mathbf{u}_1)$ is constructed, this matrix is the first step to finding Q, and $A\mathcal{H}(\mathbf{u}_1)$ is the first step to R.

$$\begin{aligned} Q_1 &= \mathcal{H}(\mathbf{u}_1) \\ R_1 &= AQ_1 \end{aligned}$$

Now we consider $A_{1,1}$, which is an $(n-1) \times (n-1)$ matrix. Just like before, the first column, $\mathbf{v}_2$ must be transformed into $||\mathbf{v}_2||\mathbf{e}_1$. After the householder matrix, $\mathcal{H}(\mathbf{u}_2)$, is calculated, we place into the $n \times n$ identity matrix as the $(1, 1)$ submatrix.

$$\mathcal{H}'(\mathbf{u}_2) = \begin{pmatrix} I & 0 \\ 0 & \mathcal{H}(\mathbf{u}_2) \end{pmatrix}$$

Q and R are then updated.

$$\begin{aligned} Q_2 &= \mathcal{H}(\mathbf{u}_1)\mathcal{H}'(\mathbf{u}_2) \\ R_2 &= AQ_2 \end{aligned}$$

This process is continued to the $(n-1, n-1)$ submatrix, at which point Q will be the product of unitary matrices, and R will be upper triangular.

5.2 Bidiagonalization

While the QR factorization only operates on the columns, it's possible to extend the idea in order to simplify both the rows and column. . This process is outlined by Golub and Kahan in "Calculating the singular values and pseudo-inverse of a matrix"[**?**]. It is then extended to operate on a product of matrices in a paper by Golub, Sølna and Van Dooren with the specific purpose of computing the SVD[**?**]. After the product has been bidiagonalized, the singular values of the matrix can then be extracted through bulge chasing [**?**].

6 Future Plans

Once the kinks are worked out of the fractal dimension program, all the necessary tools will be available to really tackle the project goal. The problem, as discussed earlier, is that the direction of the largest singular value undergoes such radical growth that it is drowning out any other directions in which the system might have some structure. It is something like measuring the expansion of a bullet due to heat as it is speeding away. A possible way of revealing structure from higher order LEs is to consider more complicated methods of synchronization. For example, it would be ideal if there was a way to suppress the largest singular value, leaving only the other LEs to factor into the systems behavior, a way to catch up to the bullet so to speak and enter its reference frame. This would mean multidimensional coupling as the direction of maximum stretch, given by the singular vector of the largest singular value, would have to be coupled much more strongly than the other directions. But this vector changes as the trajectory visits different parts of the attractor. Executing such a procedure would amount to:

Step 1: Calculate the Jacobian for some N iterations

Step 2: Find the singular values and their corresponding vectors using QR decomposition

Step 3: Reconfigure the coupling to strongly couple the system in the direction of the largest singular value's vectors for ever step.

Step 4: Vary the coupling strength in the other directions of stretch to see when the system synchronizes.

A coupled system might be organized as follows

$$\mathbf{y}_{n+1} = f\left(\left[\mathbf{I} - (1-\epsilon)\mathbf{C}\right]\mathbf{x}_n + (1-\epsilon)\mathbf{C}\mathbf{y}_{n+1}\right) \tag{11}$$

Where the direction of coupling is handled in a matrix C such that

$$C = \left[\mathbf{I} - \mathbf{v}_{1,n}\mathbf{v}_{1,n}^*\right] + \rho\left(\mathbf{v}_{1,n}\mathbf{v}_{1,n}^*\right) \tag{12}$$

$$\rho = \frac{1-\gamma}{1-\epsilon} \tag{13}$$

In the SVD of the Jacobian, $\mathbf{J} = \mathbf{U}\Sigma\mathbf{V}^*$ the first column of $\mathbf{V}$ corresponds to the direction we would like to suppress at the *nth* step, which is the vector $\mathbf{v}_{1,n}$ in the equation for $\mathbf{C}$. The transformation $[\mathbf{I} - \mathbf{v}\mathbf{v}^*]$ represents the projection onto the *orthogonal complement* of the singular vector. The strength of this directional coupling is given by γ, and ρ is designed so that $(1-\epsilon)\rho = 1-\gamma$.

This will not be an easy task as there is much more to do at every iteration of the map, but right now it is our best bet at isolating any hidden structure. There may also be numerical issues with this method as it involves finding the initial vectors corresponding to the directions of largest stretch. The problem is that the exact value of these vectors might be incredibly precise, and any errors could easily render the synchronization useless as it would fail to suppress the largest LE. However, after much discussion of this idea we are becoming increasingly convinced that the it should at least theoretically work, even if it is beyond the reach of our capabilities, and therefore irrelevant to our goal of finding a cheap way to get at higher order LEs. One possible way to sidestep this sensitivity is to consider maps which are similar and whose study can be illuminating to the dynamics of the original system, but tuned such that the original direction of maximum stretch is known. For simpler systems, it may also be possible to convert them into tent maps, via coordinate transformations such that their LEs are preserved and made drastically easier to find by the tent maps simplicity. The motivation behind this thinking is the universal behavior of quadratic maps. The idea is that chaotic quadratic maps all have fundamentally the same behavior, though the details may vary, based on the underlying transformation they perform: stretch the domain and fold it in half. If systems can be classified by their underlying transformations it may be possible to consider simpler tent maps which perform the same transformation and have the same LEs, but which are much easier to partially synchronize. Complications arise in that often the mixing is much more complicated for larger systems, hence the difficulty with our original attempts at partial synchronization, but the fact that these systems come in families is definitely something that needs to be explored for calculating LEs as it may be possible to choose which system is easiest to use and then simply compensate for differences in the systems. All these will be considered in the next stage of the project.

In addition to the new idea above, we would like to be able make a formal assessment of the methods previously used to calculate the Lyapunov exponents. The various methods will be implemented in Matlab and C. The bidiagonalization procedure for a single matrix [?] has already been coded. This method simply needs to be applied to the product, and then bulge chasing is used to find the SVD of the bidaigonalized matrix. Next is the using the QR decomposition to find the eigenvalues of the the product of the Jacobian transposes and the Jacobians [?]. Lastly is the method using the exterior square. When the exterior squares of the Jacobians are multiplied together, the result is a matrix whose singular values are the product of pairs of singular values of the plain product. For instance the first singular value of the exterior product product, will be $\sigma_1 * \sigma_2$. This allows for the same sort of trick used to calculated the second exponent of the Hénon map earlier, if we know one of them we can extract them all. However because we are dealing with pairs rather than the sum of all the elements, it works much better in higher dimensions.

6.1 Goals

1. Partial synchronization has real potential and if the numerical difficulties can be overcome it may end up being a viable way of calculating the higher order LEs.

2. More research needs to be done on the classification of dynamical systems as there must be literature on the subject. If there is a way to consider simplified versions of the system this also may be helpful in terms of partial synchronization or maybe even directly calculating the LEs by established methods.

3. There is also much to be understood about the geometry of dynamical systems, as the explorations of the coupled quadratic map has shown. The dimension calculations of the baker's map also show some anomalies which need to be further understood.

4. Compare methods for calculating Lyapunov exponents, looking for one that is accurate, scales smoothly with dimension, and is easy to apply to different maps (and perhaps flows).

5. Create easy to use libraries/software that allow users to calculate the Lyapunov exponents with these different methods.