TIME SERIES ANALYSIS OF CHAOTIC DYNAMICAL SYSTEMS WITH RECURRENT NEURAL NETWORKS

Lucas Wilson
Undergraduate: Mathematics, Computer Science
Colorado State University
Fort Collins, CO 80523
lkwilson96@gmail.com

May 1, 2019

ABSTRACT

The predictive power of the recurrent neural network known as the Echo State Network is not very strong for very chaotic problems. Measured by Lyapunov Characteristic Exponents (LCE), a time series forecasting analysis was applied to highly chaotic systems: Lorenz System, Double Pendulum, and Poincaré's solution to the restricted circular three body problem. The Lorenz system had an LCE of 0.935 with a predictability length of 1 Lyapunov time. The double pendulum had an LCE of 0.601, and a predictability length of less than 1 Lyapunov time. The restricted circular three body problem solution had an LCE of 0.176, much less chaotic, and a predictability power as strong as 10 Lyapunov times. Other methods may be applied to improve the results, but in general it seems the predictability is very low for more chaotic systems.

1 Introduction

Chaotic dynamical systems have been an important area of study especially in fields of atmospheric sciences. However, mathematical models are not always available, and if they are, very small errors in measurements can amplify very quickly making the prediction useless.

Deep learning techniques are known to be able to approximate any function given enough data and resources [1]. In light of this, the hope is that a deep learning based model, the Echo State Network, will be able to capture the complexity of chaotic systems and forecast accurately.

Using a tool created by me to perform the analysis, `echonn`, I will explore this problem and apply the Echo State Network to three classic chaotic problems. The python module can numerically solve these dynamical systems, provides an echo state network implemented by me, and has classes to help with experimentation and visualization.

One of the first chaotic systems is Henri Poincaré's solution to the restricted circular three body problem, a very useful simplification of the popular three body problem [2]. The second system will be the Lorenz equation, researched by the father of chaos, Edward Lorenz [3]. The system is commonly known as the Lorenz Butterfly for its shape, and it's the origin of the butterfly effect commonly used in pop culture. The last dynamical system is the double pendulum. This system is very commonly cited for its very random looking behavior despite being a completely deterministic system.

Chaos of the systems will be measured using the systems' maximum Lyapunov Characteristic Exponents, and data for the systems is collected from numerical solutions of the dynamical system's governing ODE equations and randomly generated initial conditions.

2 Chaos

In 1963, Edward N. Lorenz published an article researching a simplified system of ordinary differential equations modeling a convective system [3]. His research popularized the possibility of a system being highly sensitive to initial

conditions, i.e., chaos theory; while he wasn't the first to discover this phenomenon, he is considered to be the "official discoverer of chaos theory" [4].

When forecasting, the goal is to have accurate predictions. However, a chaotic system's sensitivity to initial conditions implies that a small perturbation as a result of error will produce incorrect solutions. Further, given a periodic solution, the hope is that a small perturbation, from error in measurement or from floating-point errors in numerical calculations, doesn't affect the solution or at least produces a quasi-periodic solution close to the periodic one. (A solution or trajectory F is quasi-periodic if and only if for all t and for some τ, $F(t+\tau)$ is arbitrarily close to $F(t)$ [3].)

An example of this is the trajectories of the sun and planets in our solar system. Newton's equations can be used to create a system of equations to model the bodies of mass (known as the n-body problem). However, with the existence of other bodies of mass from other solar systems within the universe, small perturbations are introduced into the system. If the system is chaotic, then over time, the actual trajectories will diverge from our model's [4]. One of the defining characteristics of chaos is that the error from perturbations grows exponentially [4]. At the time of Lorenz's paper, unstable error was not well understood, and this led to Lorenz's surprising discovery [4]. Lorenz reported that a single iteration in calculating the solution to the dynamical system took approximately one second [3]. In his paper, he shows one of his calculations having about 3000 iterations, which must have taken around 50 minutes. Given the amount of time it takes to perform the calculation, it's tempting to continue the calculation from the output of the previous calculation. However, the story of the discover is that while the computer used 6 digit accuracy, it only output 3 digits, so when the simulation was continued with the less accurate measurements, Lorenz found the results to be very different [4].

Chaos can appear in many different situations. Lorenz's paper demonstrates chaos in a dynamical system useful for forecasting convection in the atmosphere or liquids [3]. "Chaos theory has a few applications for modeling endogenous biological rhythms such as heart rate, brain functioning, and biological docks" [4]. It can also appear in solutions to Hamiltonian problems such as the 3 body problem (as mentioned before) and the double pendulum problem (as I will show later). Understanding the predictability of chaos is very useful to many different fields.

2.1 Lyapunov Constant

It's useful to know how chaotic a dynamical system is. Chaos is usually measured by the exponential rate at which nearby trajectories diverge. Constants describing the divergence are known as the Lyapunov Characteristic Exponents (LCEs) [5]. There is an LCE for each dimension, and since they all represent exponential growth, the error growing at the rate of the largest LCE will dominate over the others. Given a small perturbation ϵ to a dynamical system in the direction of the largest LCE, we can approximate the growth of the perturbation by the equation $\epsilon(\Delta t) = \epsilon_0 e^{\lambda \Delta t}$ where λ is the largest LCE [6]. For an LCE of zero, then the error doesn't grow exponentially, and thus, there is no chaos. Error created from smaller LCEs is negligible. The approximate time scale where perturbations become large is going to be proportional to $1/\lambda$. This is known as the Lyapunov time [6]. For example, the Lyapunov time of the orbits of our solar system is 10,000,000 years [4].

Researchers have used it to evaluate the effectiveness of forecasting models on chaotic problems in the past. Predicting the solution of a chaotic model over an interval of time which is multiple times the size of the Lyapunov time shows the predictability of a model. One group of researchers studied the Kuramoto-Sivashinsky chaotic model near an attractor and found that with a max LCE of 0.1, they were able to predict 8 Lyapunov times before their forecast diverged [7].

2.2 Calculating the Largest Lyapunov Characteristic Exponent

In order to find the Lyapunov time, we only need to solve for the largest LCE. To find this, we will use the following method [8].

The dynamical system will be defined by $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x})$ where $\mathbf{f}, \mathbf{x} \in \mathbb{R}^d$. Given an initial condition $\mathbf{x}_0$, the largest LCE, denoted λ_1, is calculated as follows:

$$\lambda_1 = \lim_{t \to \infty} \frac{1}{t} \log \sigma_1 \left(\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0} \right) \tag{1}$$

where σ_1 is the first singular value of the matrix $\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0}$ and $\log$ is the natural logarithm. The notation of $\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0}$, to me, is unfamiliar. I believe $\mathbf{x}_0$ is a vector representing the initial conditions, but not a specific point in $\mathbb{R}^n$, since that would leave the partial derivative meaningless. Regardless, it will be defined as the matrix obtained from the calculation outlined below. Since we can only numerically approximate the limit for λ_1, we will need to use a sufficiently large t until the limit converges to the accuracy we need.

In order to calculate $\frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0}$, we need to integrate the matrix differential equation:

$$\frac{\mathrm{d}\mathbf{M}(t)}{\mathrm{d}t} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}(t))\mathbf{M}(t) \tag{2}$$

where $\mathbf{M}(t) \in \mathbb{R}^{d\times d}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is the $d \times d$ Jacobian matrix of $\mathbf{f}(\mathbf{x})$ at $\mathbf{x}(t)$ and with $\mathbf{M}(0)$ equal to the identity matrix.

Since values for $\mathbf{x}(t)$ are needed as well, the systems will be integrated together. If the dynamical system has dimensions d, then the new system of differential equations will have dimension $d + d \times d$. For larger dynamical systems, this calculation becomes costly. The dynamical systems analyzed are small enough (having dimensions ranging from 3-4) that the calculation can be performed timely and accurately.

There are several issues with calculating λ_1 with this algorithm. Already mentioned, we need to choose a sufficiently large t for the limit to converge. Unfortunately, for chaotic systems, $\mathbf{M}$ will increase rapidly with time. The large floating point values will eventually overflow and λ_1 will not be accurate. I have found empirically that if values of $\mathbf{M}$ don't exceed 10^{100}, then λ_1 will be accurate.

Implemented in `echonn`, the algorithm was tested on the Lorenz system for various parameters and initial conditions. The LCEs match the expected values very accurately (see the Lorenz System section: 3.2). The 3 dimensional 3-body problem is an 18 dimensional dynamical system, so while the dynamical system is implemented in `echonn`, its max LCE can't be calculated accurately since the resulting system has 342 dimensions. The algorithm was also tested on the differential equation which produces a circle:

$$\begin{aligned} x' &= y \\ y' &= -x \end{aligned}$$

with the solution

$$\begin{aligned} x(t) &= r\sin(t) \\ y(t) &= r\cos(t) \end{aligned}$$

for any $r \in \mathbb{R}$.

Given an initial condition (x_0, y_0), it produces the trajectory of a circle with radius $r = \sqrt{x_0^2 + y_0^2}$. If perturbed, the radius will change, $r_\epsilon = \sqrt{(x_0 + \epsilon_x)^2 + (y_0 + \epsilon_y)^2}$, so the new trajectory is

$$\begin{aligned} x_\epsilon(t) &= r_\epsilon \sin(t) \\ y_\epsilon(t) &= r_\epsilon \cos(t). \end{aligned}$$

Now, the ℓ^2 norm of error is

$$\begin{aligned} \sqrt{(x_\epsilon - x)^2 + (y_\epsilon - y)^2} &= \sqrt{(r_\epsilon - r)^2\sin^2(t) + (r_\epsilon - r)^2\cos^2(t)} \\ &= \sqrt{(r_\epsilon - r)^2(\sin^2(t) + \cos^2(t))^2} \\ &= \sqrt{(r_\epsilon - r)^2} \\ &= |r_\epsilon - r|. \end{aligned}$$

The error doesn't grow at all over time, so the LCE should be less than or equal to zero as the system is not chaotic. The algorithm reports an LCE of -1.15954×10^{-06} as expected.

Empirically shown, `echonn` can be used to identify the max LCE of smaller dynamical systems, and special precaution has to be taken when choosing a time interval. Using a larger time will yield a more accurate λ_1, but it may also cause an overflow if the system is chaotic. Thus, maximum time values need to be chosen such that $||\mathbf{M}||_\infty$ doesn't exceed 10^{100} (where $||\cdot||_\infty$ is the ℓ^∞ norm).

3 Dynamical Systems

Three dynamical systems will be used to demonstrate the predictability of chaotic systems: the Lorenz system, Poincaré's 3-body problem solution, and the double pendulum system. The Poincaré system is the one of the first, and the Lorenz system is a classic example. The double pendulum is commonly referred to as chaotic, and I will show this later. Both Poincaré's restricted 3 body problem solution and the double pendulum are Hamiltonian systems representing physical systems within our universe.

3.1 Numerical Integration Methods

In order to generate the data to be analyzed, the solutions to these dynamical systems need to be computed. Each of the systems are defined by a system of differential equations which can be numerically solved. However, due to the chaotic nature, error introduced by the numerical solution will grow exponentially, so precaution needs to be taken to avoid having this sort of error introduced into the collected data. When numerically solving chaotic problems, it's possible for the chaotic behavior to be suppressed, or where there shouldn't be chaotic behavior, it could be created [9].

However, it's been shown that the Fourth Order Runge-Kutta Method (RK4) outperforms other numerical methods such as Euler's Method and the Midpoint Method for chaotic systems [10]. Not only does RK4 perform better than other methods, it serves as a good method for producing accurate numerical solutions to choatic systems with small enough time steps: the Lü chaotic system [11] and the Zhou system [12], for example. One article points out that a larger concern regarding these solutions is the accuracy of the initial conditions measured from the physical problem which the mathematical model is based, but otherwise, they echo the strength of correctly implemented numerical solutions [9]. Further, my initial conditions are randomly generated, so this is not a concern.

I will be using an improved version of RK4 which incorporates the fifth order Runge-Kutta: RK4(5) [13]. This is the method implemented in the Python function `scipy.integrate.solve_ivp`, and my numerical solver wraps this function to integrate the chaotic systems. Using small time steps will yield accurate samples from the chaotic systems.

3.2 Lorenz System

The Lorenz system was originally based on a system of equations modeling convection created by Saltzman [3] [14]. Simplified by Lorenz, it is defined as follows [3]:

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma(y-x),\\ \frac{\mathrm{d}y}{\mathrm{d}t} &= x(\rho-z)-y,\\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy-\beta z.\end{aligned} \tag{3}$$

Not all parameters for the Lorenz equation will produce chaotic behavior. In Lorenz's article, he uses the parameter values $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ [3], and it has a maximum LCE of 0.90566 [8]. There are other parameters which produce chaotic results outlined in Table 1. Also shown in Table 1 are the values calculated by my Python module `echonn` are listed. These values demonstrate the accuracy of `echonn`.

σ	ρ	β	actual λ_1	calculated λ_1	relative error of calculated λ_1
4	45.92	16	1.50255	1.49869	-2.5694 $\times 10^{-3}$
4	40	16	1.37446	1.37297	-1.0876 $\times 10^{-3}$
8/3	28	10	0.90566	0.91188	6.8694 $\times 10^{-3}$

Table 1: Lorenz Parameters and Largest Lyapunov Exponent [8]

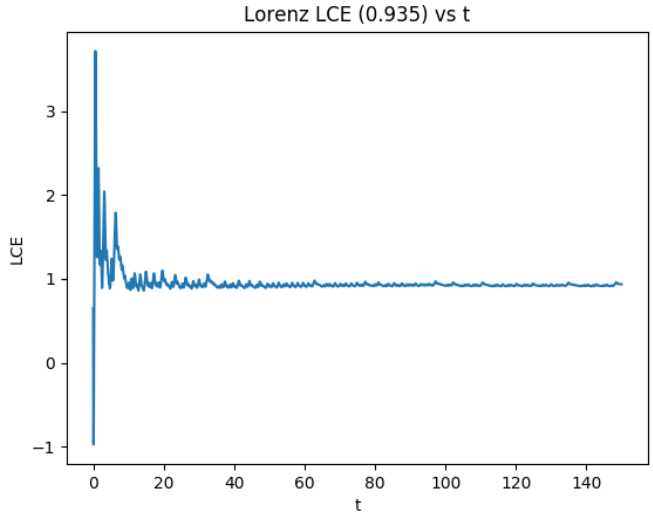
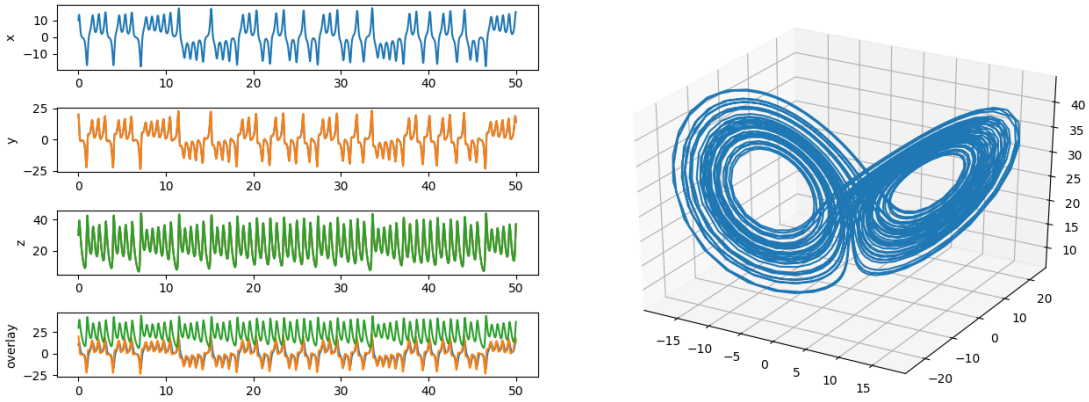


Figure 1: Convergence of LCE with large t

The Lorenz system is named after its creator, but it is also known as the Lorenz butterfly due to the shape of the system. The Lorenz Butterfly can be seen in Figure 2.



(a) Plot of each dimension of Lorenz System

(b) The Lorenz Butterfly

Figure 2: The Lorenz System Visuals

3.3 3-Body Problem

The 3-body problem, or the generalized n-body problem, is a famous problem which many have analyzed; most notably, Herni Poincaré wrote many papers developing the area significantly [2]. One of Poincaré's solutions, known as the circular restricted 3-body problem, demonstrates chaotic properties [4]. Poincaré performs several reductions of the original 3-body problem in order to come to a much more useful form of the problem. The mass of one of the objects is assumed to be negligible. This is the case for Sun-Earth-Moon orbits: the moon has little effect on the orbit of the sun. Another aspect is the rotating reference frame. Since the mass of one of the bodies is negligible, the problem simplifies to first solving a two body problem. While much easier to solver, Poincaré went further to stop the motion of the two planets by using a rotating reference frame. Rotating with the planets, the problem can be modeled seemingly by two static planets and a third mass-less planet orbiting around those in a two dimensional plane.

The N body problem is defined as follows [2]:

Let $\mathbf{r}_i$ be the position of the i^{th} body in the N body problem. Then, the force of gravity is the sum of gravity from every other body in the system. That is,

$$F_g^{(i)} = m_i \frac{\mathrm{d}^2 \mathbf{r}_i}{\mathrm{d}t^2} = \sum_{j \neq i} G \frac{m_i m_j \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^3} \tag{4}$$

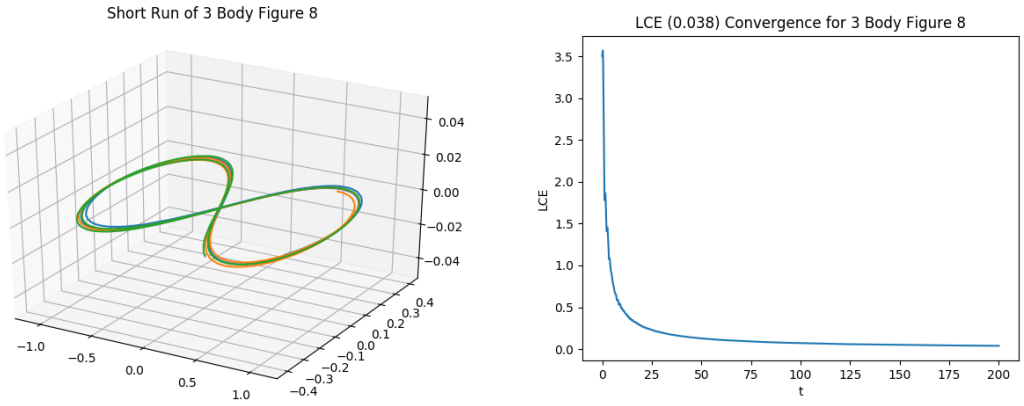
where $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, m_i is the mass of body i, and G is the gravitational constant.

An interesting solution (originally calculated by Carles Simó) can be found where the trajectories of the bodies are periodic and form a figure 8 [15].

The gravitational constant, G was redefined to equal 1, the masses of the objects were equal to 1, and the initial conditions are as follows:

$$\begin{aligned}
\mathbf{r}_1 &= (0.97000436, -0.24308753) \\
\mathbf{r}_2 &= -\mathbf{r}_1 \\
\mathbf{r}_3 &= (0, 0) \\
\mathbf{r}_1' &= -\mathbf{r}_3/2 \\
\mathbf{r}_2' &= -\mathbf{r}_3/2 \\
\mathbf{r}_3' &= (-0.93240737, -0.86473146)
\end{aligned}$$

However, the resulting solution doesn't demonstrate chaotic properties. The LCE convergence chart (Figure 3b) shows the LCE converging to zero.



(a) Figure 8 solution to 3 body problem

(b) Nonchaotic LCE convergence

Figure 3: Nonchaotic Figure 8 Solution to 3 Body

For our purposes, the reduced system is more useful since it is much easier to calculate than numerically solving the original 3-body problem. With three bodies in three dimension, their motion is described by 9 second order differential equations. Therefore, reducing the system to first order differential equations yields 18 first order equations. The reduced problem only requires 4 first order differential equations since the position is defined by two second order differential equations.

Derived in two independent papers [16] and [17], the system of equations defined with dimensionless rotating (synodic) coordinates:

$$
\begin{aligned}
x'' - 2y' &= x - \frac{\alpha}{r_1^3}(x-\mu) - \frac{\mu}{r_2^3}(x+\alpha) \\
y'' + 2x' &= \left(1 - \frac{\alpha}{r_1^3} - \frac{\mu}{r_2^3}\right) y
\end{aligned}
\tag{5}
$$

where

$$
\begin{aligned}
\mu &= \frac{m_1}{m_1 + m_2} \\
\alpha &= 1 - \mu \\
r_1(t) &= \left[(x(t)-\mu)^2 + y(t)^2\right]^{\frac{1}{2}} \\
r_2(t) &= \left[(x(t)+\alpha)^2 + y(t)^2\right]^{\frac{1}{2}}
\end{aligned}
\tag{6}
$$

and where mass body one and two are located at $-\alpha$ and μ.

Although the notation in both papers was different, the equations were the same. It's also worth mentioning that Eberle's paper was for his Masters Thesis, and Frnka's paper has no details associated with it, but I was able to find a mathematician with the exact name who achieved his doctorate. With that said, I believe the math is reliable, and the simulations seem to work and at least demonstrate chaotic properties (the only requirement for this project). With my lacking physics knowledge, I was unable to derive these equations myself from the assumptions made by Henri Poincaré outlined in [2].

If we define the following system:

$$
\begin{aligned}
x_1 &= x \\
x_2 &= x' \\
y_1 &= y \\
y_2 &= y',
\end{aligned}
\tag{7}
$$

then, we can take the derivative of each:

$$
\begin{aligned}
x_1' &= x' \\
&= x_2,
\end{aligned}
$$

and

$$
\begin{aligned}
x_2' &= x'' \\
&= 2y' + x - \frac{\alpha}{r_1^3}(x-\mu) - \frac{\mu}{r_2^3}(x+\alpha) \\
&= 2y_2 + x_1 - \frac{\alpha}{r_1^3}(x_1-\mu) - \frac{\mu}{r_2^3}(x_1+\alpha),
\end{aligned}
$$

and

$$
\begin{aligned}
y_1' &= y' \\
&= y_2,
\end{aligned}
$$

and

$$
\begin{aligned}
y_2' &= y'' \\
&= -2x' + \left(1 - \frac{\alpha}{r_1^3} - \frac{\mu}{r_2^3}\right) y \\
&= -2x_2 + \left(1 - \frac{\alpha}{r_1^3} - \frac{\mu}{r_2^3}\right) y_1.
\end{aligned}
$$

Then, we can represent the second order system of motion as four first order equations:

$$
\begin{aligned}
x_1' &= x_2 \\
x_2' &= 2y_2 + x_1 - \frac{\alpha}{r_1^3}(x_1 - \mu) - \frac{\mu}{r_2^3}(x_1 + \alpha) \\
y_1' &= y_2 \\
y_2' &= -2x_2 + \left(1 - \frac{\alpha}{r_1^3} - \frac{\mu}{r_2^3}\right) y_1, \qquad (8) \\
& \qquad (9)
\end{aligned}
$$

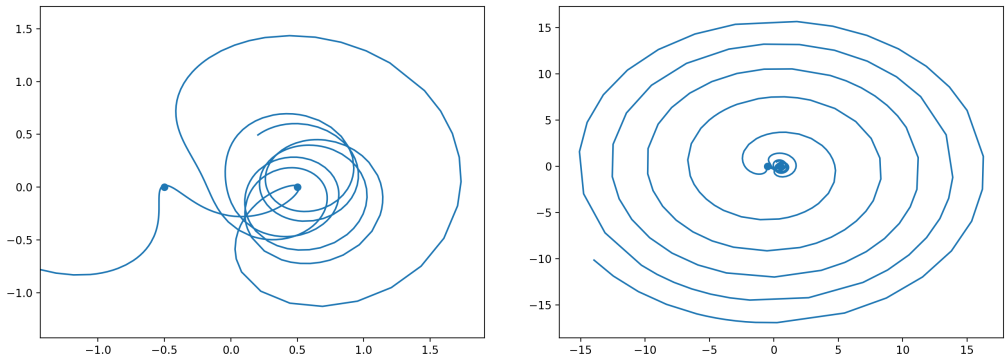
where

$$
\begin{aligned}
r_1 &= \left[(x_1 - \mu)^2 + y_1^2\right]^{\frac{1}{2}} \\
r_2 &= \left[(x_1 + \alpha)^2 + y_1^2\right]^{\frac{1}{2}}. \qquad (10)
\end{aligned}
$$

Here, x_1 and y_1 represent the position of the third body relative to the other two, but not in the standard Cartesian coordinates of 3D space. Outlined in the both papers, [16] and [17], the transformation back to Cartesian coordinates is given, but we don't need the exact position of the bodies since we are only concerned with the chaotic motion, so that isn't shown here.

There are however two issues when integrating this problem. As objects approach each other, the distance between them approaches zero. If collision occurs, then the diverging trajectory approaches infinity. There is no collision detection in this model, and each body has a radius of zero, so this case is not handled. Instead, a divide by zero error occurs, and the model cannot be calculated further.

Further, near collisions are also an issue. With the distance between bodies approaching zero, their potential energy becomes significantly smaller than other values, such as those describing kinetic energy [18]. This produces a stiff system of equations, and introduces error. Since energy is conserved in this Hamiltonian system, the error introduced represents energy being created. This can cause the singularity property seen in Figure 4. This problem can be solved by using a symplectic integrator [18].



(a) Restricted 3 Body Problem with collision (b) The resultant divergence of the body

Figure 4: The Restricted 3 Body Problem Error from Collision

While it is possible to use a symplectic integrator, it is not necessary. Under the right initial conditions, the third body can diverge regardless (e.g., fast initial velocity away from the bodies). The simple divergent spiral isn't chaotic, so instead orbital and collision less initial conditions will be used. Without divergence and without collisions, the symplectic integrator is unnecessary.

3.4 Double Pendulum

The double pendulum is commonly cited to be chaotic [19] [20]. Using the system of equations derived by Stachowiak [19], the system is defined as follows:

$$\begin{aligned}
\frac{\mathrm{d}\theta_1}{\mathrm{d}t} &= \omega_1, \\
\frac{\mathrm{d}\theta_2}{\mathrm{d}t} &= \omega_2, \\
\frac{\mathrm{d}\omega_1}{\mathrm{d}t} &= \frac{\sin(\theta_1-\theta_2)[l_1\cos(\theta_1-\theta_2)\omega_1^2+\omega_2^2]}{2l_1[1+m_1-\cos^2(\theta_1-\theta_2)]} - \frac{(1+2m_1)\sin\theta_1+\sin(\theta_1-2\theta_2)}{l_1[1+m_1-\cos^2(\theta_1-\theta_2)]}, \\
\frac{\mathrm{d}\omega_2}{\mathrm{d}t} &= \sin(\theta_1-\theta_2)\frac{(1+m_1)(\cos\theta_1+l_1\omega_1^2)+\cos(\theta_1-\theta_2)\omega_2^2}{1+m_1-\cos^2(\theta_1-\theta_2)}.
\end{aligned} \tag{11}$$

θ_1 and ω_1 represent the angle and angular velocity of the inner pendulum where $\theta_1 = 0$ implies the pendulum is straight down, and $\theta_1 = \epsilon$ implies the pendulum is rotated ϵ radians counter-clockwise. The same is true for the outer pendulum relative to its pivot point, the end of the first pendulum. l_1 and l_2 are the lengths of the inner and outer pendulums, respectively. m_1 and m_2 are the masses of the balls on the end points of the inner and outer pendulums, respectively. Then, the positions of the endpoints (x_1, y_1) and (x_2, y_2) of the pendulum, with the center pivot at the origin, $(0, 0)$, can be calculated as follows:

$$\begin{aligned}
x_1 &= l_1\sin(\theta_1), \\
y_1 &= -l_1\cos(\theta_1), \\
x_2 &= x_1 + l_2\sin(\theta_2), \\
y_2 &= y_1 - l_2\cos(\theta_2),
\end{aligned} \tag{12}$$

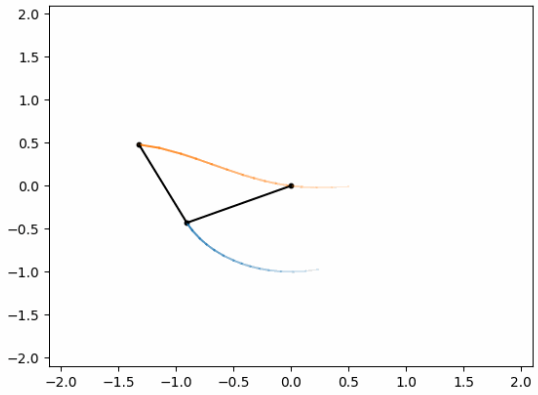


Figure 5: The Double Pendulum

While the system is a Hamiltonian system, Stachowiak decided to define the first order derivative components as angular velocity as opposed to momentum. This won't affect anything regarding the purpose of this paper.

Interestingly, the degree of chaos is dependent on the initial conditions [20]. In the experiment, the initial conditions had angular velocities of zero, the outer pendulum was straight down, and the inner pendulum varied from 0 degrees to 180 degrees [20]; the results showed that the LCE increased with theta.

Repeating the same experiment, but with different parameters ($l_1 = l_2 = 1$, and $m_1 = m_2 = 1$), the results are similar (see Figure 6). As inner theta increases, the largest LCE of the system seems to approach 1.4.

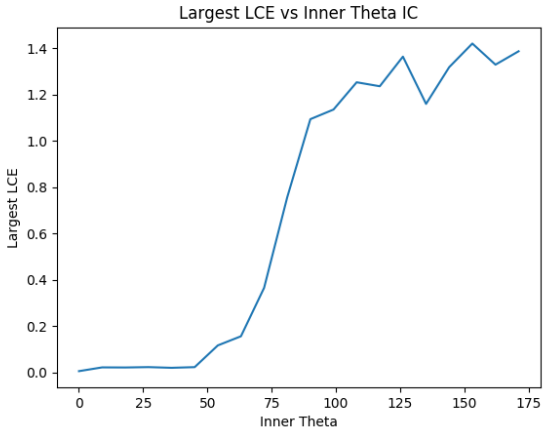


Figure 6: Largest Lyapunov Characteristic Exponent vs Inner Theta Initial Condition

4 The Echo State Network

The Echo State Network (ESN) is a type of Recurrent Neural Network (RNN) used for time series analysis, but it is trained differently than most RNNs and is characterized by having the echo state property. I implement the design detailed by Jaeger in is 2002 book [21] and further detailed in his 2007 article [22].

A time series data set will contain samples of data which change over time. Part of the data at each time point is measured or known before, called input data and denoted by $\mathbf{u}(t)$ with K dimensions. The rest of the data will be the goal for prediction, called output data and denoted $\mathbf{d}(t)$ with L dimensions.

For T time samples, $t_1, t_2, \ldots, t_T$, we have a $T \times K$ matrix, $\mathbf{U}$, with sample input vectors as rows. We also have a $T \times L$ matrix, $\mathbf{D}$, with sample output vectors as rows. A model will predict the values of $\mathbf{d}$, and this prediction will be denoted $\mathbf{y}$.

4.1 Recurrent Neural Networks and Traditional Training Method

RNNs are usually designed by having K input units, L output units, and some internal units organized by some network structure. These units have evolving values for every time point. The structure of networks describes how units are connected. The input units at time t_i are set to $\mathbf{u}(t_i)$. However, output units and internal units are calculated from the values of other units. These other units are said to be connected to the resultant unit. Output units are calculated from current input units, current internal units, and from previous output units. Internal units are calculated from the values of the current input units, previous internal units, and previous output units.

Although, it's rare for the ESN that the output units are calculated directly from previous output units [21]. Sometimes for training, the output units are set to the value of $\mathbf{d}$, called teacher forcing, so that the model trains on actual values as opposed to its accumulated errors [21].

To calculate the value of a unit, an activation function is applied to a linear combination of the units connected to it [23]. The constants for the linear combination are called weights, and they are parameters of the model. Which activation function is used is also a parameter of the function. Typically, some non-linear function is used, such as $\tanh$, sigmoid, or rectified linear unit.

During training, we search for weights which maximize the model's performance on data which the model hasn't seen. A common training method for RNNs is to use a form of backpropagation [21]. Backpropagation for RNNs is slightly different from traditional Neural Networks, but the idea is the same.

For example, given some neural network model $\mathbf{y}(\mathbf{x}; \mathbf{w})$, where $\mathbf{x}$ is known input data, $\mathbf{w}$ is the model's parameters, and $\mathbf{y}$ approximates some values, $\mathbf{d}$, then we define an objective function as the sum of squared error:

$$\mathbf{E}(\mathbf{x}; \mathbf{w}) = ||\mathbf{y}(\mathbf{x}; \mathbf{w}) - \mathbf{d}||_2^2. \tag{13}$$

Using steepest descent to solve this minimization problem (as shown in [23]), we can reduce $\mathbf{E}$ in (13) by updating the weights as follows:

$$\mathbf{w} = \mathbf{w} - \lambda \nabla_{\mathbf{w}} \mathbf{E} \tag{14}$$

for some learning rate λ.

4.2 Echo State Network Implementation

The implementation design used for this paper is detailed by Herbert Jaeger in [21] and [22]. I explain it here to clarify my exact process (and with a slight change of notation to be consistent with the rest of this paper).

Like RNNs, ESNs have an input layer with K units and with values $\mathbf{u}(t_1), \mathbf{u}(t_2), \ldots, \mathbf{u}(t_T)$ where $\mathbf{u}(t) \in \mathbb{R}^K$. K is zero if there are no input units. There is an internal layer with N units, denoted by $\mathbf{x}(t_i) \in \mathbb{R}^N$ for $i = 1, 2, \ldots, T$, and there is an output layer with L units, denoted by $\mathbf{y}(t_i) \in \mathbb{R}^L$ for all $i = 1, 2, \ldots, T$.

How the units are connected can vary, but the suggested way [21] is to connect the current input layer, the previous internal layer, and the previous output layer to the internal layer. These connections will have weights described in 4.1. Respectively, these weights will be denoted by the matrices: $\mathbf{W}_{in}$, $\mathbf{W}$, and $\mathbf{W}_{back}$. The activation function for the internal layer is $\tanh$.

Connected to the output layer is the current input layer and current internal layer. Additionally, a bias input is connected to this layer. The weight for these connections will be denoted $\mathbf{W}_{out}$ (we only have one matrix since the connected layers are concatenated into a single vector, per Jaeger's notation [21]). The activation function for these connections is linear. Although, the recommended choice is $\tanh$, the output layer doesn't necessarily lie in the range of $\tanh$, so it can't be used without some form of normalization.

The dimensions of the weight matrices are as follows: $\mathbf{W}_{in} \in \mathbb{R}^{N \times K}$, $\mathbf{W} \in \mathbb{R}^{N \times N}$, $\mathbf{W}_{back} \in \mathbb{R}^{N \times L}$, and $\mathbf{W}_{out} \in \mathbb{R}^{N \times (K+1+N)}$,

In mathematical notation, the model is defined as follows:

$$\mathbf{x}(t_i) = \tanh\left(\mathbf{W}_{in}\mathbf{u}(t_i) + \mathbf{W}\mathbf{x}(t_{i-1}) + \mathbf{W}_{back}\mathbf{y}(t_{i-1})\right) \tag{15}$$

$$\mathbf{y}(t_i) = \mathbf{W}_{out}(\mathbf{u}(t_i), 1, \mathbf{x}(t_i))_{\text{concat}}. \tag{16}$$

"Feeding data" into the model means iterating using values for $\mathbf{u}$. If the feeding is teacher-forced, then the output layer's values are set to $\mathbf{d}$. Otherwise, the calculated output values are used.

$\mathbf{x}$ and $\mathbf{y}$ are calculated from previous unit values but for the first time step, there are none, so we need to initialize the units to zeros [21]. This is done by defining $\mathbf{x}(t_0)$ and $\mathbf{y}(t_0)$ to zeros.

The weights are initialized such that the model has the defining characteristic of echo state networks: the echo state property (Section 4.3). During the training phase (Section 4.4), $\mathbf{W}_{out}$ is fit to the training data.

Once the network has been initialized and trained, it can be used to perform time series analysis.

4.3 The Echo State Property

The echo state property is defined by the following [21]:

> Assume an untrained network with weights $\mathbf{W}_{in}$, $\mathbf{W}$, and $\mathbf{W}_{back}$ is driven by teacher input $\mathbf{u}(n)$ and teacher-forced by teacher output $\mathbf{d}(n)$ from compact intervals $\mathbf{U}$ and $\mathbf{D}$. The network ($\mathbf{W}_{in}$, $\mathbf{W}$, and $\mathbf{W}_{back}$) has echo states w.r.t. $\mathbf{U}$ and $\mathbf{D}$, if for every left-infinite input/output sequence $(\mathbf{u}(n), \mathbf{d}(n-1))$, where $n = \ldots, -2, -1, 0$, and for all state sequences $\mathbf{x}(n)$, $\mathbf{x}'(n)$ compatible with the teacher sequence, i.e. with
>
> $$\mathbf{x}(n) = \tanh\left(\mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}(n-1) + \mathbf{W}_{back}\mathbf{y}(n-1)\right) \tag{17}$$
>
> $$\mathbf{x}'(n) = \tanh\left(\mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}'(n-1) + \mathbf{W}_{back}\mathbf{y}(n-1)\right) \tag{18}$$
>
> it holds that $\mathbf{x}(n) = \mathbf{x}'(n)$ for all $n \leq 0$.

The idea is that if the network is run long enough, it will, in a sense, forget long past input and focus only on the most recent input. If two networks with differing internal states are run with the same input and teacher-forced output, they will eventually converge to each other. It's name suggests data fed to the model will echo around but eventually fade as echos do.

Jaeger outlines a process to create weights which will likely have echo state properties. The process is posed as conjecture and doesn't guarantee the echo state property. However, the echo state is very likely and can be assumed confidently [21] [22].

To demonstrate the echo state property, initialize the network with random values. Set user input (if any) and teacher-forced output units to zero, After enough iterations, the internal state should converge to zero.

The process to initialize the echo state weights, $\mathbf{W}$, is as follows [21]:

1. Initialize an $N \times N$ matrix, $\mathbf{W}_0$, with values from a uniform distribution in $[-1, 1]$.
2. Calculate the spectral radius, $|\lambda_{\text{max}}|$, of $\mathbf{W}_0$, i.e., the maximum absolute value of the eigenvalues of $\mathbf{W}_0$.
3. Normalize $\mathbf{W}_0$ to $\mathbf{W}_1$ so that it has unit spectral radius: $\mathbf{W}_1 = \frac{1}{|\lambda_{\text{max}}|}\mathbf{W}_0$
4. Choose parameter $\alpha < 1$. Typically, α lies in the range $(0.7, 0.98)$ [21]. This number will represent the new spectral radius of $\mathbf{W}$.
5. Scale $\mathbf{W}_1$ to $\mathbf{W}_2$ to have a spectral radius of α: $\mathbf{W}_2 = \alpha\mathbf{W}_1$. Now, let $\mathbf{W} = \mathbf{W}_2$, and the model will have the echo state property [21].

If the spectral radius of $\mathbf{W}$ is found to be greater than 1, then it can be shown that the model doesn't demonstrate the echo state property; Jaeger's process uses this fact by ensuring that this isn't the case [21]. While it can't guarantee an echo state property, the resultant matrix does have the property frequently enough that it's a reliable process. In his words, the model "has always been found to be an echo state network" [21].

Applying linear algebra, this makes a lot of sense. Let A be an $N \times N$ matrix with spectral radius $r = \rho(A) > 0$ and denote its eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ for $i = 1, 2, \ldots, N$. Then, define $B = \frac{\alpha}{r}A$.

By definition, $\mathbf{v}_i A = \lambda_i \mathbf{v}_i$, so

$$\begin{aligned}\mathbf{v}_i B &= \frac{\alpha}{r}\mathbf{v}_i A \\ &= \frac{\alpha}{r}\lambda_i \mathbf{v}_i\end{aligned}$$

Therefore, B has eigenvalues $\frac{\alpha}{r}\lambda_i$ for all i. Further, its spectral radius is then

$$\begin{aligned}||\frac{\alpha}{r}\lambda||_\infty &= \frac{\alpha}{r}||\lambda||_\infty \\ &= \frac{\alpha}{r}r \\ &= \alpha\end{aligned}$$

where λ is a vector of eigvenvalues and $||\cdot||_{\text{max}}$ is the ℓ^∞ norm.

As the model iterates, the matrix transformation of B is repeatedly applied. With $\mathbf{u}$ and teacher-forced $\mathbf{d}$ as zeros, the internal state has the value $\mathbf{x}(t_{k+1}) = \tanh(B\mathbf{x}_k)$ for iteration k.

Since the spectral radius of B is less than 1,

$$\begin{aligned}\lim_{k\to\infty} B^k\mathbf{v}_i &= \lim_{k\to\infty}\lambda_i^k\mathbf{v}_i && \text{(where } \lambda_i \text{ and } \mathbf{v}_i \text{ is an eigvenvalue-eigenvector pair of } B\text{)} \\ &= 0\mathbf{v}_i && \text{(since } |\lambda_i| < \alpha < 1 \text{ and } \lim_{k\to\infty}\alpha^k = 0\text{)} \\ &= 0\end{aligned}$$

This suggests that as the model iterates, the internal state converges to zero. Also, $\tanh$ is contractive further supporting this idea.

The rest of weights can be initialized however desired, so we will use a common technique.

1. Initialize an $N \times M$ matrix, $\mathbf{W}_0$, with values from a uniform distribution in $[-1, 1]$. M and N here are not the same values from before for the sake for generality.
2. Scale the matrix by $\frac{1}{\sqrt{M}}$ to calculate the weight matrix.

Then, $\mathbf{W}_{in}$ and $\mathbf{W}_{back}$ can be calculated as follows:

$$\mathbf{W}_{in} = \frac{\mathbf{W}_0}{\sqrt{K}}$$
$$\mathbf{W}_{back} = \frac{\mathbf{W}_0}{\sqrt{L}}$$

Different from most training techniques, $\mathbf{W}_{in}$, $\mathbf{W}$, and $\mathbf{W}_{back}$ are never changed after initialization.

4.4 Echo State Network Training

In order to train the echo state network, we need to find good weight values for $\mathbf{W}_{out}$. Training the echo state network is different from the backpropagation approach in that we can solve for and calculate directly the weights matrix; more modern methods use an iterative approach which converges to good weight values. Part of the reason is due to the calculation cost of the iterative methods. Today, that it is no longer an issue, but historically, it was a strong quality of the echo state network [22].

First, the data (input, $\mathbf{u}$, and output, $\mathbf{d}$, values) is partitioned into three groups:

1. Initialization values at time points $t_0, t_1, \ldots, t_{T_0-1}$.
2. Training values from $t_{T_0}, \ldots, t_T$
3. Testing values from $t_{T+1}, \ldots, t_{T_f}$

Second, the initialization data is fed into the model with teacher-forced output. This initializes the inner state of the model, and the initial state (zeros) is lost due to the echo state property. T_0 should be chosen such that this is truly the case. Like α, T_0 is a parameter of the echo state network and differs depending on the problem. Good values for T_0 are 10 to 500 depending on the problem complexity.

Third, the training values are fed into the model to sample the network with teacher-forced output. We define a matrix $M \in \mathbb{R}^{(K+1+N)\times(T-T_0+1)}$ to store the input unit values, bias, and internal state value pairs, $(\mathbf{u}(t_{T_0}), 1, \mathbf{x}(t_{T_0})), \ldots, (\mathbf{u}(t_T), 1, \mathbf{x}(t_T))$, in its columns. We define a matrix $T \in \mathbb{R}^{L\times(T-T_0+1)}$ to store the output unit values: $\mathbf{d}(t_{T_0}), \ldots, \mathbf{d}(t_T)$.

To train the model, we want $g(\mathbf{W}_{out}M)$ to approximate accurately T, where g is the activation function for the output layer. Since we use an idempotent output layer activation function, $g^{-1}(X) = g(X) = X$, so

$$\begin{aligned} g(\mathbf{W}_{out}M) &= T \\ \mathbf{W}_{out}M &= g^{-1}(T) \\ \mathbf{W}_{out}M &= T \end{aligned}$$

Therefore, $\mathbf{W}_{out}$ should be found such that $\mathbf{W}_{out}M = T$ [21]. Solving for $\mathbf{W}_{out}$,

$$\begin{aligned} \mathbf{W}_{out}M &= T \\ (\mathbf{W}_{out}M)^T &= T^T && \text{(where } A^T \text{ is the transpose of } A\text{)} \\ M^T\mathbf{W}_{out}^T &= T^T \\ (M^T)^{-1}M^T\mathbf{W}_{out}^T &= (M^T)^{-1}T^T && \text{(where } A^{-1} \text{ is the Moore-Penrose pseudoinverse [24] of } A\text{)} \\ \mathbf{W}_{out}^T &= (M^T)^{-1}T^T \\ \mathbf{W}_{out} &= \left((M^T)^{-1}T^T\right)^T \\ \mathbf{W}_{out} &= T\left((M^T)^{-1}\right)^T \end{aligned}$$

Using packages like `numpy.linalg.eig` and `numpy.linalg.pinv`, solving for eigenvalues and pseudoinverses for $\mathbf{W}_{out}$ is very easy. After solving for $\mathbf{W}_{out}$, the echo state network is trained.

To evaluate the network, the testing values are fed into the model. However, this time, the output layer is not teacher-forced since in real application, the output values are unknown. To measure how well the echo state network approximates the desired output we define the error to be the root mean squared error (RMSE) of the actual output values $\mathbf{d}$ and predicted values $\mathbf{y}$: $\sqrt{\sum_{i=T+1}^{n}(\mathbf{d}(t_i) - \mathbf{y}(t_i))^2}$ where $n \in [T+1 \dots T_f]$. The error represents the predictive power of the model for n time steps in the future. If RMSE is low for larger values of n, the model has a strong long term predictive power.

5 Analysis

5.1 Method

First, data from the models is collected. Using the process described above (Section 2.2), the Largest Lyapunov Exponent is calculated, λ_1. Then, the Lyapunov Time is calculated by $\frac{1}{\lambda_1}$. We will then integrate the model with the algorithm RK4(5) over the interval $[0, \frac{50}{\lambda_1}]$ with time steps of $\frac{1}{100\lambda_1}$. This will result in 5,000 data points.

Then, the data is partitioned into a training set, a validation set, and a testing set. I use a 90% split, so testing is the last 10% of the data. The training and validation set are the first 90%. Of the 90%, training is the first 90%, and validation is the final 10%. In total, it's a 81%-9%-10% split.

The training set is used to train the model. The validation set is used to tune the parameters of the model (α, N, T_0). The testing set is used to evaluate the performance.

The first data point of the training set is $t = 0$. The parameter T_0 marks where the model begins training on the data (see Section 4.4). Using slightly different notation to account for the extra validation set, we will denote the first validation time point by $t = T$. We will denote the first testing time point by $t = T_f$

Performance of the model is calculated by RMSE for a given time interval $[t_0, t_f]$:

$$\text{RMSE} = \sqrt{\sum_{t=t_0}^{t_f}(\mathbf{d}(t) - \mathbf{y}(t))^2} \tag{19}$$

where $(\cdot)^2$ is the dot product of the error with itself, $\mathbf{d}(t)$ is the true value at time t, and $\mathbf{y}(t)$ is the model's approximation at time t.

First, we generate a list of parameter combinations to try. Usually, this is the Cartesian product of typical parameter values for the model. For each combination, we train several models on the training data. Then, we predict the validation data to measure the general performance of the model under those parameters.

$\text{RMSE}_{\text{train}}$ is calculated on the interval $[T_0, T)$, and the prediction uses teacher-forced values on $[0, T_0)$. $\text{RMSE}_{\text{validation}}$ is calculated on the interval $[T, T_f)$, and the prediction uses teacher-forced values on $[0, T)$.

Then, the models which performed best on the validation set should also perform well on the testing set. The models are retrained on the testing and validation set, and the error is calculated: $\text{RMSE}_{\text{full train}}$ is calculated on the interval $[T, T_f)$, and the prediction uses teacher-forced values on $[0, T)$.

Finally, the error for the testing set is calculated. The performance on the testing set is an unbias representation of how well the ESN can fit the model.

However, the $\text{RMSE}_{\text{test}}$ is calculated slightly different since we want to know how far the model can predict before deviating too far. The $\text{RMSE}_{\text{test}}(t)$ will be calculated on the interval $[T_f, t)$ and the prediction uses teacher-forced values on $[0, T_f)$. Lastly, $\text{RMSE}_{\text{test}}$ as a value instead of a function will notate the RMSE for the entire test set.

We will look at how far past T_f the ESN model's $\text{RMSE}_{\text{test}}$ grows too large. The best model will be the model with the lowest validation RMSE as well as a low testing RMSE.

5.2 Results

Overall, the systems proved too difficult to forecast; however, this could be due to lacking data. While more data can generated, numerical errors might grow too large. Since we have about 5,000 data points, we can't have an internal state dimension of more than 500 or we risk over-fitting [21].

The Lorenz model showed strong forecasting results, but due to the shape of the attractor, it was easy for the small perturbations to cause quick divergence as the trajectories followed the incorrect "wing" of the butterfly.

The double pendulum model showed very poor results. Occasionally, the pendulum would move much quicker causing spikes in the data. The ESN tried to predict when these spikes occurred, but was unable. Even the forecasted shape of the rapid movement was inaccurate. This is likely due to the lack of data.

The restricted circular three body problem performed much better than the other two, but this is due to the existence of a simple attractor: a single circular orbit as opposed to two rings in the Lorenz system. Also, the max LCE was much smaller for this system (0.1759 vs 0.9 and 0.6) suggesting that it was less chaotic. While it's possible that the chaotic behavior was created by the numerical solution, it suggests the strong predictability of lesser chaotic systems.

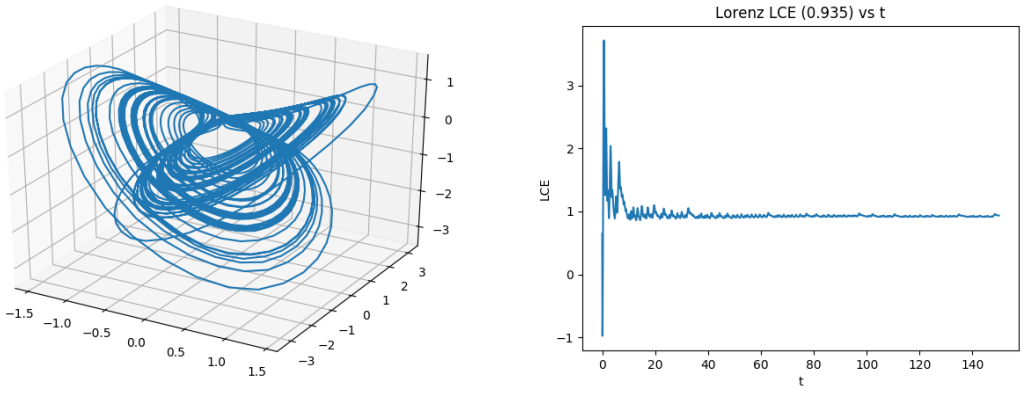
In the case of researchers performing a similar analysis mentioned before [7], they too had strong performance for a less chaotic problem. I wasn't able to predict 8 Lyapunov times, but there might be reasons why my results weren't as strong other than the fact that the ESN might need more data that I used. The lorenz attractor is more chaotic than the attractor they used. The double pendulum lacks an attractor and shows a higher level of chaos. The 3 body problem has a lower LCE and a simple attractor, and showed strong results as well.

5.2.1 Lorenz

To integrate the model, I use the initial conditions: $(x, y, z)_{t=0} = (-13.78990503, -11.6069343, 36.17332518)$. The calculated Lyapunov value was $\lambda_1 = 0.9348566$.

One of the issues run into while forecasting with the Lorenz is that the continuous nature of the equation led to the ESN just outputing the previous value. For initial values far from the center of the Lorenz Butterfly, it would tend towards the center and stop when it does arrive there. Most models trained poorly and gave $\text{RMSE}_{\text{test}}$ values of 16 or well above 10,000, implying the model diverged.

To fix this issue, we instead have the ESN train on differences between time points. This reduces our data set to 4,999 values, and it creates a new shape of the data. The Lorenz Butterfly had a similar shape, two rings, but in this case, the rings intersect (See figure 7a).



(a) A plot of the differences between time steps

(b) LCE converges to 0.935 for the Lorenz system

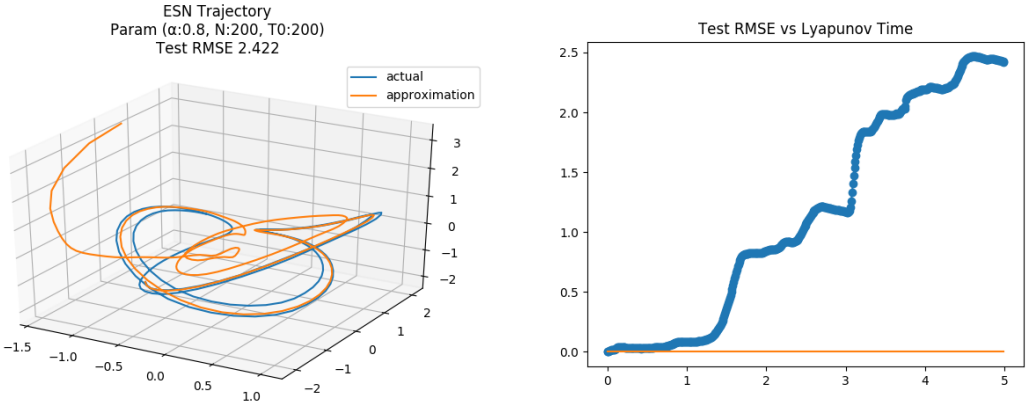
Figure 7: The difference butterfly and LCE convergence

After this transformation of the data, the model performed significantly better. $\text{RMSE}_{\text{test}}$ dropped to ranges of 1-2, the model exploded (diverged to infinity) less frequently, and it was no longer outputting previous values.

Most well performing models could predict around one Lyapunov time constant. From the graphs, we can see that most models deviated once reaching the intersection of the butterfly wings.

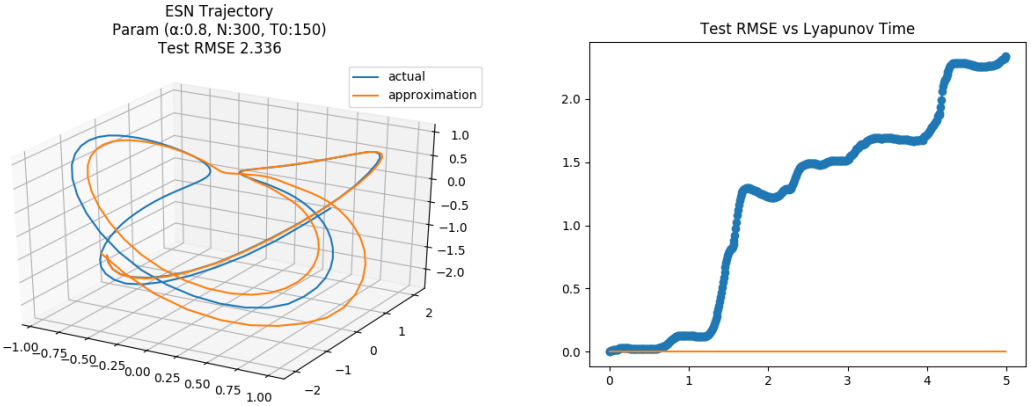
Once reaching the intersection, several things can happen. First, it can deviate completely and stop following either wings. Second, it can follow the wrong wing. Which wing the trajectory follows seems almost random, so it makes sense if it chooses the wrong one. Third, it will continue to follow the trajectory, but slightly perturbed. With every passing of the intersection, any of these can happen. Eventually, even the perturbations causes problems. The model was trained on data that existed in a specific range of space. When the perturbations lead the model into space not explored, it performs even worse, and this can cause very sporastic behavior.

The chaotic behavior of the Lorenz model dictates which wing the trajectory will follow. This is difficult for the ESN to be able to model. However, it is possible that I am not using enough data to train the ESN, and due to resource limitations, I may have not been able to find the correct parameters to achieve a better performance.



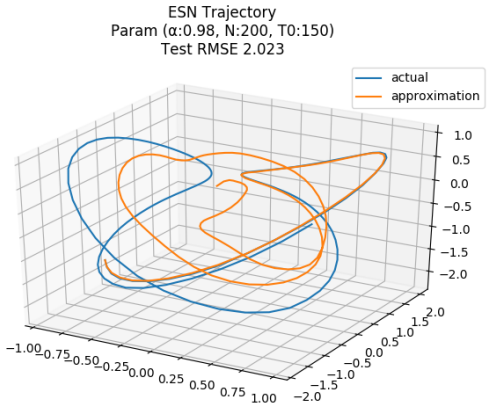
(a) The True Figure vs ESN Prediction (b) RMSE Over Time

Figure 8: Diverges when entering space outside training data.

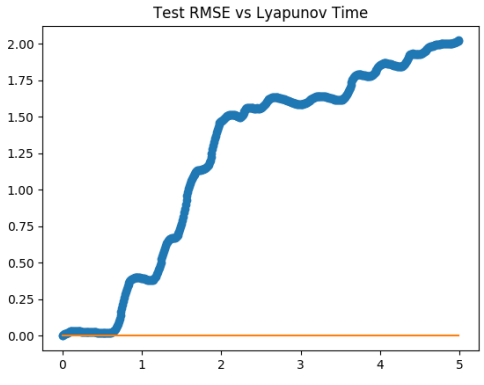


(a) The True Figure vs ESN Prediction (b) RMSE Over Time

Figure 9: Continues in wrong direction at intersection.

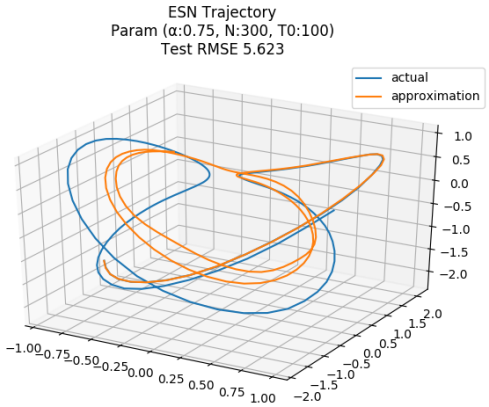


(a) The True Figure vs ESN Prediction

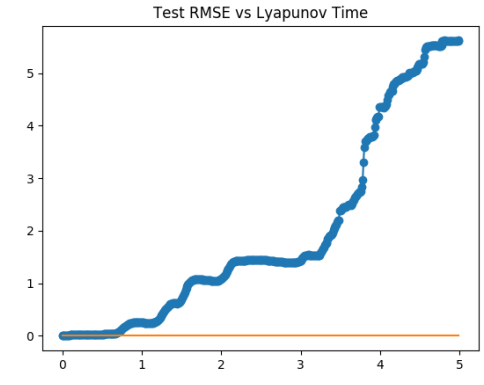


(b) RMSE Over Time

Figure 10: Deviates after an intersection.

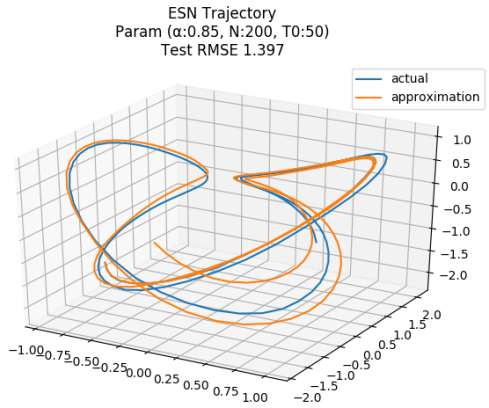


(a) The True Figure vs ESN Prediction

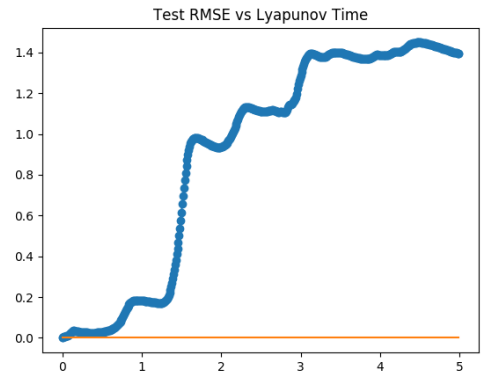


(b) RMSE Over Time

Figure 11: Fits model well but follows wrong wings.

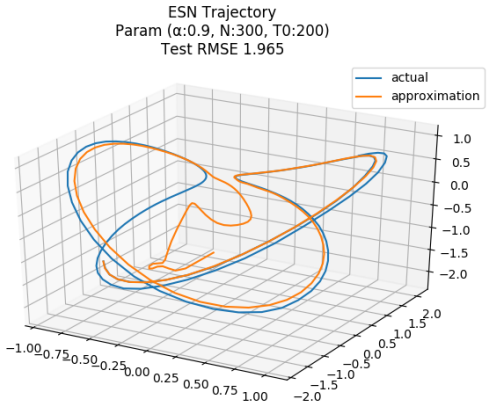


(a) The True Figure vs ESN Prediction

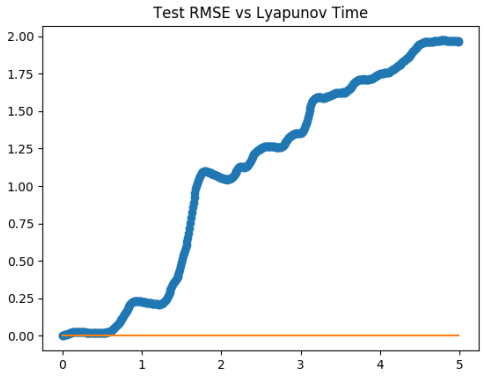


(b) RMSE Over Time

Figure 12: Fits model well but trajectory is ahead in time.

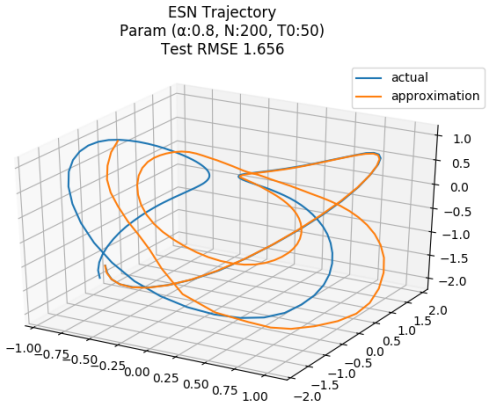


(a) The True Figure vs ESN Prediction

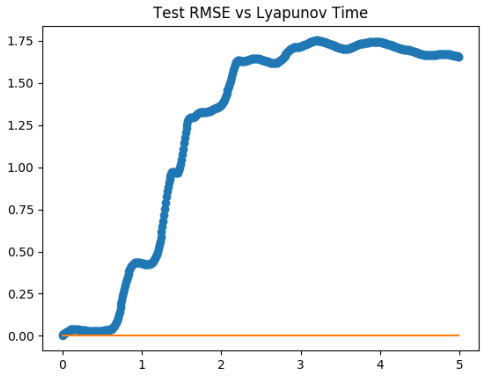


(b) RMSE Over Time

Figure 13: Diverges after reaching unknown space.

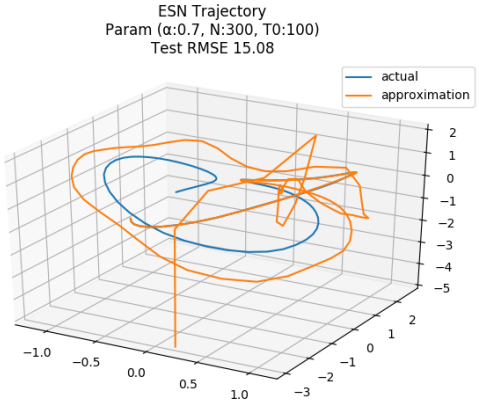


(a) The True Figure vs ESN Prediction

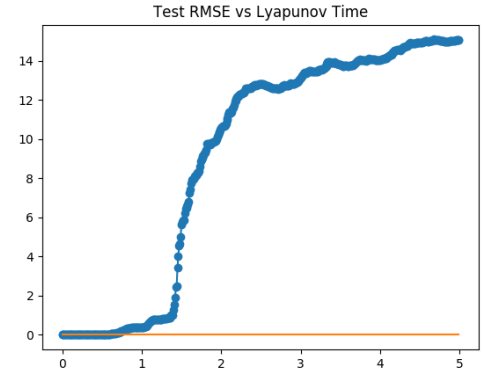


(b) RMSE Over Time

Figure 14: Diverges after intersection.



(a) The True Figure vs ESN Prediction



(b) RMSE Over Time

Figure 15: Sporadic behavior shown after divergence.

5.2.2 Double Pendulum

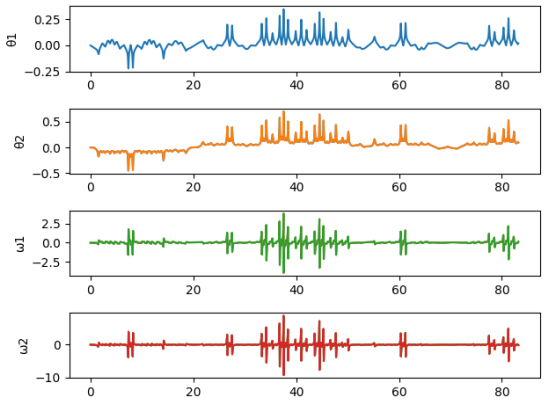
In this case, we need θ_1 large enough that the system is chaotic, but the rest of the parameters were chosen randomly (with rotational velocities close to zero). The double pendulum's initial condition was

$$(\theta_1, \theta_2, \omega_1, \omega_2)_{t=0} = (4.66648088, 4.29820501, 8.01585663 \times 10^{-4}, 3.71175422 \times 10^{-3}), \tag{20}$$

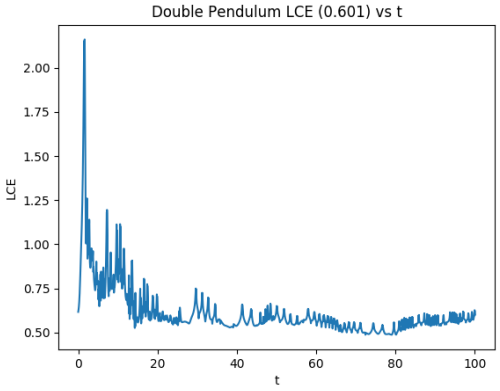
and the resulting Lyapunov Exponent is $\lambda_1 = 0.60074$. Like the Lorenz, the differences between time points were used to train the ESN since the ESN was showing poor results otherwise. As the pendulum kept looping the center, θ_1 and θ_2 kept growing; this shouldn't be an issue, but using the differences keeps the system variables in a closer range.

Figure 16a shows a plot of each of the dataset's dimensions. It seems like the dataset's activity is fairly sparse. There is only gradual but seemingly random variations until a few spots around $t = 10, 30, 60,$ and 75. The results for the ESN weren't very strong, and none could predict more than one Lyapunov Times before diverging.

Because there are portions of the dataset which are calm, the ESN then has to pickup on signals which lead to the spikes in movement. We see in the graphs that while it predicts the testing set well, it eventually tries to predict spikes but does so at the wrong time, and the spike pattern never ends, has too large of magnitude, and is very rapid. Overall the model performs poorly, and it's reflected in the RMSE which quickly diverges well above 3. The likely reason for this is that the ESN can't learn the complex patterns from such a small and sparse dataset. With more data or more complex internals, maybe the ESN could learn to forecast this dataset.



(a) A plot of the differences between time steps



(b) LCE converges to 0.601 for the Double Pendulum system

Figure 16: The difference graph and LCE convergence

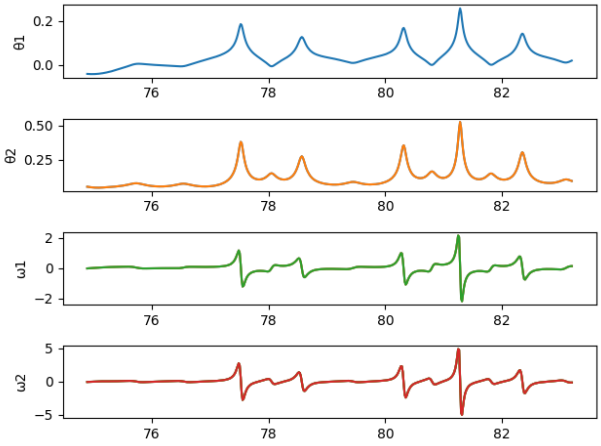
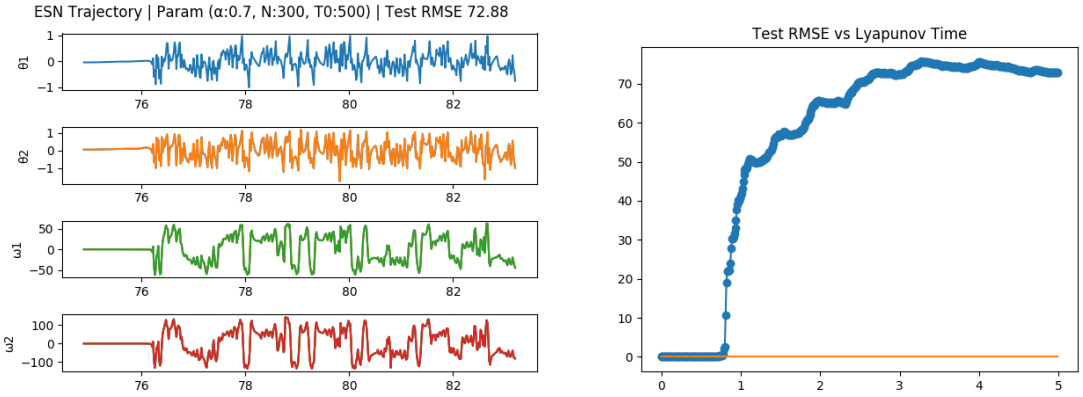
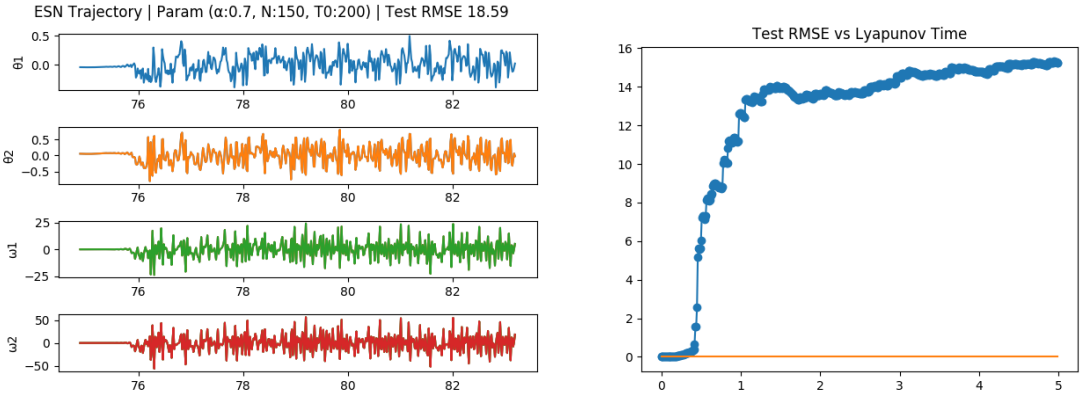


Figure 17: The testing dataset partition



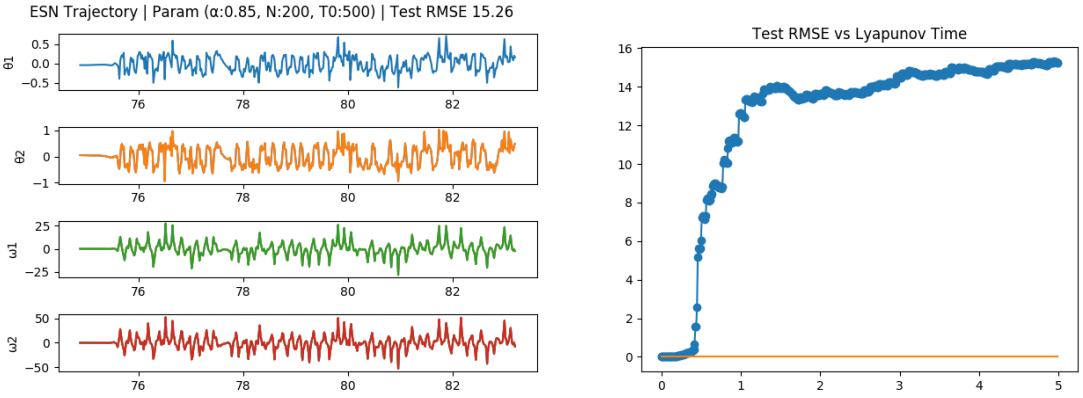
(a) Delayed spike prediction (b) RMSE Over Time

Figure 18: Poor forecast of double pendulum



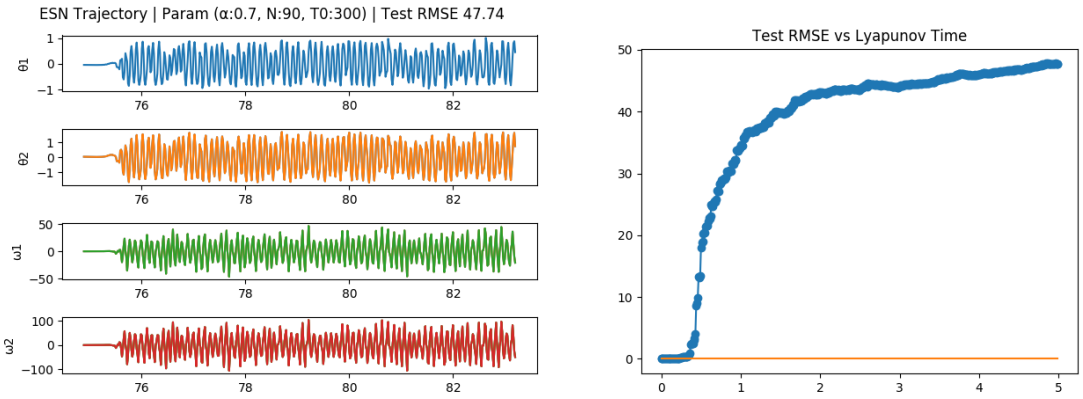
(a) Delayed spike prediction (b) RMSE Over Time

Figure 19: Poor forecast of double pendulum



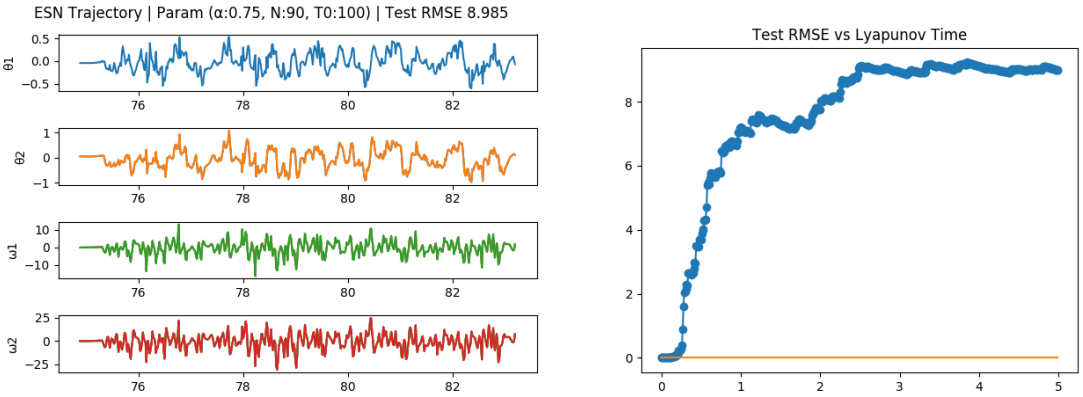
(a) Delayed spike prediction (b) RMSE Over Time

Figure 20: Poor forecast of double pendulum



(a) Delayed spike prediction (b) RMSE Over Time

Figure 21: Poor forecast of double pendulum



(a) Delayed spike prediction (b) RMSE Over Time

Figure 22: Poor forecast of double pendulum

5.2.3 Restricted Circular 3 Body

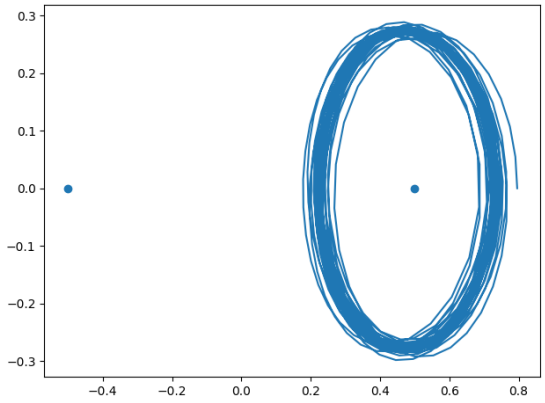
Generating data for the 3 body problem is more difficult since we have to be careful that the trajectory doesn't diverge or collide with the other bodies.

When the bodies collide, then the problem becomes a stiff problem and the trajectory might be inaccurate, and when the bodies diverge, the trajectory becomes a simple spiral without chaotic properties. Initial conditions which don't run into these issues are usually orbital. Such solutions exist [17], and `echonn` finds them by randomly generating initial conditions, and testing for the conditions in the numerical solution. Orbital patterns happen frequently enough that they can be found easily with this method.

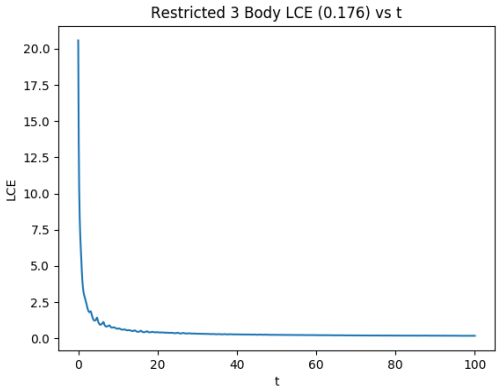
The case I analyze, the initial conditions were found to be $(x, x', y, y')_{t=0} = (0.79563053, 0, 0, 0.96833226)$ with a largest Lyapunov Exponent of $\lambda_1 = 0.1759$. This case is much different than the other systems since it doesn't have strong chaotic properties. This can be seen in the orbital pattern which seems quasi-periodic (which it may be). In light of this, it's also very possible that the measured chaotic properties are solely due to error in the numerical calculation. However, the orbiting doesn't repeat exactly, so there are still small complexities for the ESN to pick up on. In the results, we see that the ESN performs very well, but it's possible that the ESN is just converging to the mean of the orbits. The steep increase in error and the subsequent slowing of the increase in error, seen in the graphs below, suggests that this might be the case. Since the orbits are fairly close, this results in a sufficiently accurate approximation.

Most RMSE graphs look very similar. There is a steep increase in RMSE at first but then it slows. Overall, the RMSE is really strong. Most don't rise above 0.2, and one shown below (Figure 25) shows an RMSE not rising above 0.1 despite

its not as elliptical looking shape. Maybe this model is truly capturing the complexity of the system. Further, this model is impressive because it only has 5 internal units. Most other top performers have over 100 internal units.



(a) A plot of the trajectory of third body



(b) The LCE is 0.176 for the Restricted 3 Body Problem

Figure 23: The orbital pattern of the restricted 3 body problem

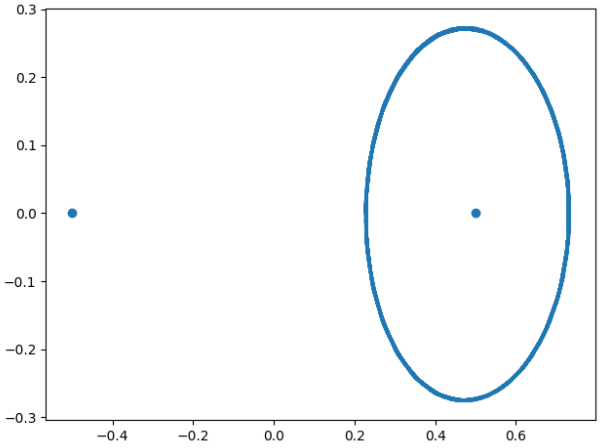
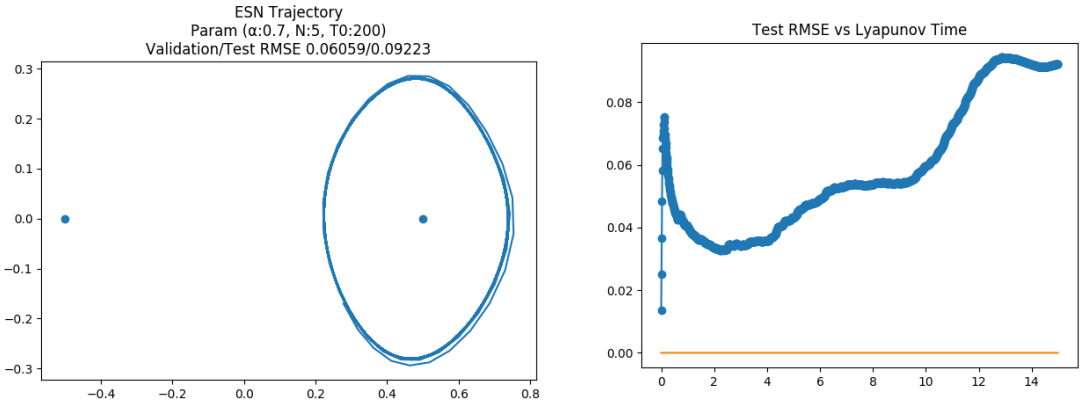
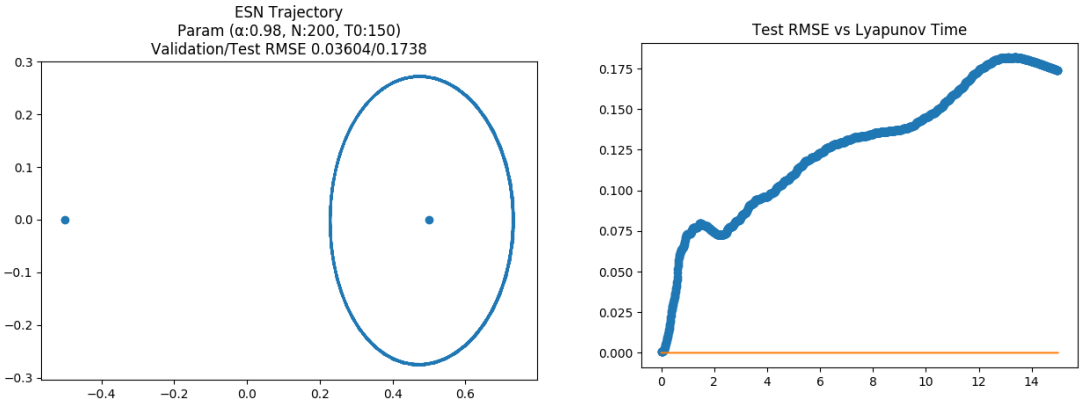


Figure 24: The testing dataset partition



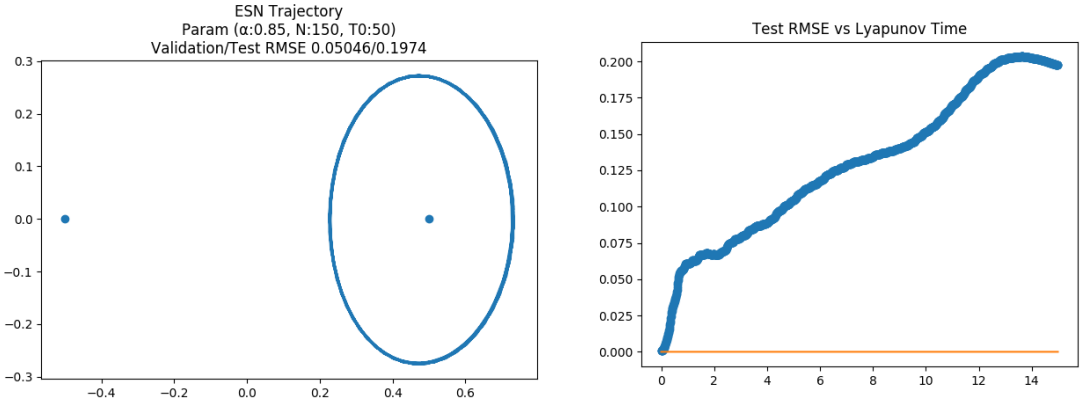
(a) Predicted Trajectory (b) RMSE Over Time

Figure 25: The best validation and testing performance and only 5 internal units



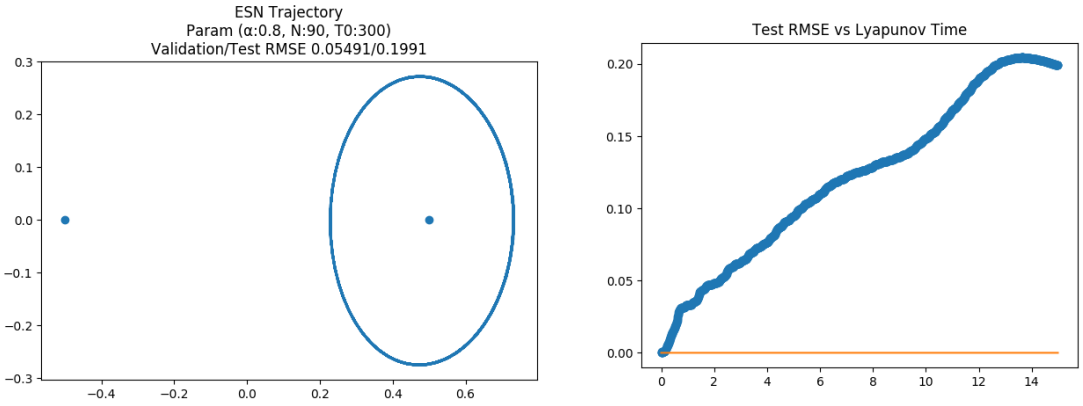
(a) Predicted Trajectory (b) RMSE Over Time

Figure 26: Best performer on validation set



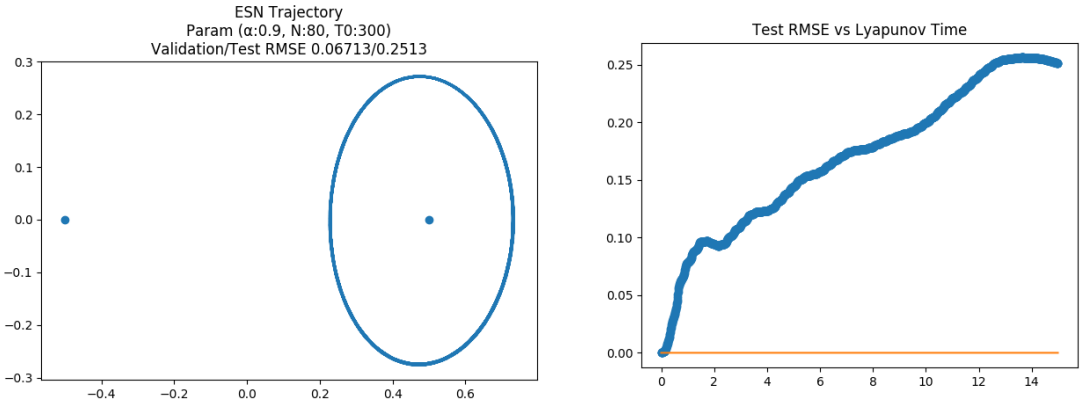
(a) Predicted Trajectory (b) RMSE Over Time

Figure 27: A top performer on validation set



(a) Predicted Trajectory (b) RMSE Over Time

Figure 28: A top performer on validation set



(a) Predicted Trajectory (b) RMSE Over Time

Figure 29: A top performer on validation set

6 Conclusion

The degree of chaotic behavior was measured by the largest Lyapunov Characteristic Exponent of the system, and three dynamical systems were used: Lorenz's convection equation, the Double Pendulum system, and Henri Poincaré's solution to the Restricted Circular Three Body Problem. The performance of the Echo State Network was determined from RMSE of the trajectory after some time. The number of the Lyapunov times that the model was able to predict accurately determined strength of the ESN.

The Lorenz equation's trajectory was reported to have an LCE of 0.935, and the predictability of the model was around 1 Lyapunov time for the best performers. In general, the ESN forecasted poorly on the Lorenz model due to the shape of the attractor. The ESN struggled to predict accurately which "wing" to follow and usually diverged at the intersection point.

The double pendulum performed the worst with an LCE of 0.601. Its best predictors were unable to predict at least one Lyapunov time. Much of the pendulums' motion was small with occasionally very large spikes of motion. There wasn't enough data for the ESN to capture the complexity of the signals which trigger the large spikes of motion. When the ESN begins to predict spikes (often incorrectly), the RMSE increases significantly. It did however produce the correlation found between dimensions, so some of the pendulum's patterns were captured successfully.

The restricted three body problem performed very well with a much smaller LCE value of 0.176. Similar to the Lorenz, there was an apparent pattern, but in this case, only a single point for the trajectory to orbit around. Due to this, the

performance of the ESN was much better. Unbias testing RMSE would remain below 0.1 for several Lyapunov times, similar to the results of another paper with a small maximum LCE value.

It's possible that the ESN model needs more data to train on, or other methods need to be applied to improve the model's performance, but it seems that highly chaotic systems can't be predicted very easily, but lesser chaotic systems with attractor patterns can be forecasted with more success.

References

[1] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[2] Alain Chenciner. Poincaré and the three-body problem. In *Henri Poincaré, 1912–2012*, pages 51–149. Springer, 2015.

[3] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[4] Christian Oestreicher. A history of chaos theory. *Dialogues in clinical neuroscience*, 9(3):279, 2007.

[5] Marco Sandri. Numerical calculation of lyapunov exponents. *Mathematica Journal*, 6(3):78–84, 1996.

[6] Boris P Bezruchko and Dmitry A Smirnov. *Extracting knowledge from time series: An introduction to nonlinear empirical modeling*. Springer Science & Business Media, 2010.

[7] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

[8] Divakar Viswanath. Lyapunov exponents from random fibonacci sequences to the lorenzequations. Technical report, Cornell University, 1998.

[9] Robert M Corless. What good are numerical simulations of chaotic dynamical systems? *Computers & Mathematics with Applications*, 28(10-12):107–121, 1994.

[10] M Affan Zidan, Ahmed Gomaa Radwan, and Khaled Nabil Salama. The effect of numerical techniques on differential equation based chaotic generators. In *ICM 2011 Proceeding*, pages 1–4. IEEE, 2011.

[11] Sadiq A Mehdi and Rabiha Saleem Kareem. Using fourth-order runge-kutta method to solve lü chaotic system. *American Journal of Engineering Research (AJER)*, 6(1):72–77, 2017.

[12] Ummu'Atiqah Mohd Roslan, Zabidin Salleh, and A Kılıçman. Solving zhou chaotic system using fourth-order runge-kutta method. *World Applied Sciences Journal*, 21(6):939–944, 2013.

[13] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[14] Barry Saltzman. Finite amplitude free convection as an initial value problem—i. *Journal of the Atmospheric Sciences*, 19(4):329–341, 1962.

[15] Alain Chenciner and Richard Montgomery. A remarkable periodic solution of the three-body problem in the case of equal masses. *Annals of Mathematics-Second Series*, 152(3):881–902, 2000.

[16] William Jason Eberle. Case studies of the circular restricted three body problem. 2007.

[17] Richard Frnka. The circular restricted three-body problem. 2010.

[18] John E Chambers. A hybrid symplectic integrator that permits close encounters between massive bodies. *Monthly Notices of the Royal Astronomical Society*, 304(4):793–799, 1999.

[19] Tomasz Stachowiak and Toshio Okada. A numerical analysis of chaos in the double pendulum. *Chaos, Solitons & Fractals*, 29(2):417–422, 2006.

[20] RB Levien and SM Tan. Double pendulum: An experiment in chaos. *American Journal of Physics*, 61(11):1038–1044, 1993.

[21] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, volume 5.

[22] Herbert Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007.

[23] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[24] Adi Ben-Israel and Thomas NE Greville. *Generalized inverses: theory and applications*, volume 15. Springer Science & Business Media, 2003.