Griffin Scafidi-McGuire

Exploring the Lorenz 96 Model

Introduction:

Edward Lorenz was a prominent figure within the mathematical and meteorological communities [1], and is remembered today as the founder of modern chaos theory: the study of dynamical systems which are heavily influenced by their initial conditions [2], and many know of chaos theory through the famous “butterfly effect”, the idea that a very small change in the initial conditions of a sensitive system can have lasting effects on the outcome of that system, with the popular analogy using the flapping of a butterfly's wings as a possible catalyst for large weather events like a tornado, even weeks later [3], and while this analogy has become relatively well known in popular culture, much of what chaos theory seeks to study, and how this study is conducted is less understood, because chaos theory studies deterministic systems (systems in which no randomness is involved in the development of future states of that system [4]), which start predictably, and after some amount of time appear to act randomly [2], and one of the key questions that chaos theorists seek to answer is the amount of time that the states of a system can be effectively predicted, which is influenced by three factors: how much uncertainty can be tolerated in the forecast, how accurately its current state can be measured, and a timescale depending on the dynamics of the system, called the Lyapunov time, with Lyapunov time being the characteristic timescale on which the system is chaotic, and by convention it is defined as the time for the distance between nearby trajectories of the system to increase by a factor of e [5], and in weather systems the Lyapunov time is usually a few days [2], and because the Lyapunov time increases the distance between trajectories exponentially, a meaningful prediction cannot be made past two or three Lyapunov time units, and hence the system will appear random or chaotic [2].

Chaos theory has found extensive applications in modeling and understanding the world around us, because it deals with systems often with a large number of interdependent variables that often do not follow predictable behavior, and three of the most explored areas of application are biology, economics, and weather modeling, with chaos being observable in population dynamics, hydrology (the study of the movement of water on earth), macroeconomic behavior, and as stated above, weather forecasting [2].

The Lorenz-96 system was first introduced by Edward Lorenz at the European Centre for Medium-Range Weather Forecasts in 1995, then later published in his 1996 paper titled, "Predictability–A Problem Partly Solved" [6]. This model was introduced as a toy model to study predictability, especially in atmospheric sciences [6], and it has also been found to be very useful within data assimilation models and ensemble forecasting techniques [7]. The model consists of N variables arranged in a cyclic structure, with partially coupled equations 'wrapping around from the 0th to the Nth, and a constant external forcing, and despite its simplicity, Lorenz-96 can display fixed points, periodic orbits, and chaotic behavior depending on the forcing parameter F and system size N.

With this overview of the Lorenz-96 system, I plan to explain in depth the structure of the system and what its variables represent, explore the stability of equilibrium, use numerical simulation to study behavior along different initial conditions and forcing, introduce my own program for studying this system, and finally explore its applications in data assimilation and give an overview of what data assimilation is.

Equation Overview:

Below is the equation for the Lorenz 96 System which we will abbreviate as L96:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2}) \bullet x_{i-1} - x_i + F$$

With X_i being the state variable representing the atmospheric quantity at site i, F being the constant forcing term, and N being the size of the system. Because of the cyclic coupling mentioned above, it is assumed that

$$x_{-1} = x_{N-1} \qquad x_0 = x_N \qquad x_{N+1} = x_1$$

By coupling the variables not surrounded on both sides by other variables to the variables on the opposite end of the system, we essentially wrap the coupling relationship from the start to end of the system, creating a system in which all terms are coupled to surrounding neighbors in a cyclical overall shape. This relates directly to the original model system: a latitudinal slice of climate blocks along earth's outside, in which a single atmospheric variable can be seen evolving through time [8].

Another important detail about L96, is that it has a minimum of four variables. This is due to the fact that the indexing spans 4 variables (x_{i-2}, x_{i-1}, x_i, x_{i+1}), and having fewer than four variables would change the structure of the equations entirely.

This structure reflects several physical analogies. When introducing this model, Edward Lorenz wrote, “The physics of the atmosphere is present only to the extent that there are external forcing and internal dissipation, simulated by the constant and linear terms, while the quadratic terms, simulating advection, together conserve the total energy” [9]. This means that the $-x_i$ term represents internal dissipation, which acts like friction or damping on the system. This makes sense as it is negative, so any positive value within the ith cell will negatively affect its own rate of change. In an atmospheric context this could be thought of as friction, heat radiating out of the system, or any process which dissipates the variable being modeled. Advection is defined as “the transport of a substance or quantity by bulk motion of a fluid. The properties of that substance are carried with it” [10]. Which in this context is the effect of neighboring variables influencing the quantity within the *i*th cell.

In my homework, I had not yet made the python programs I have now to easily experiment with a full scale lorenz system, so for a few examples below I worked with a reduced, three variable system of equations similar to the lorenz system because it includes wrapped coupling, but structurally different because it does not meet the four equation minimum. From now on I will call this system L96*. Below is the system I used.

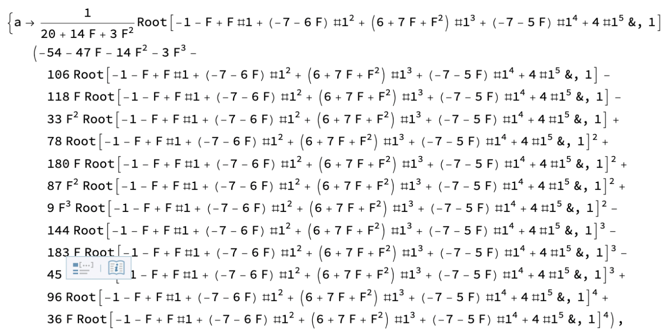
$$\dot{x}_1 = (x_2 - x_3) \bullet x_3 - x_1 + F$$

$$\dot{x}_2 = (x_3 - x_1) \bullet x_1 - x_2 + F$$

$$\dot{x}_3 = (x_1 - x_2) \bullet x_2 - x_3 + F$$

Equilibria:

At a glance, we guess when equilibrium will occur. Because F is a positive constant, our easiest solutions would be when F = x_i and the $(x_{i+1} - x_{i-2}) \bullet x_{i-1}$ term evaluates to 0. This will happen when $x_{i+1} = x_{i-2}$. One obvious situation where both of the conditions will be true, is when all x variables are equal to F. When I put this system of equations into the Mathematica solver you recommended, it does not yield any other useful equilibria. Below is an example of a non (F,F,F,F) solution that the solver outputs. I only show the value of x_1 to save space.



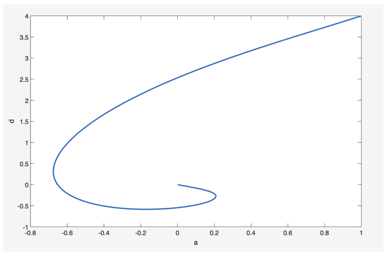
```
{a → 1/(20 + 14 F + 3 F^2) Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]
   (-54 - 47 F - 14 F^2 - 3 F^3 -
     106 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1] -
     118 F Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1] -
     33 F^2 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1] +
     78 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^2 +
     180 F Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^2 +
     87 F^2 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^2 +
     9 F^3 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^2 -
     144 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^3 -
     183 F Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^3 -
     45 [illegible]1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^3 +
     96 Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^4 +
     36 F Root[-1 - F + F ♯1 + (-7 - 6 F) ♯1^2 + (6 + 7 F + F^2) ♯1^3 + (-7 - 5 F) ♯1^4 + 4 ♯1^5 &, 1]^4),
```

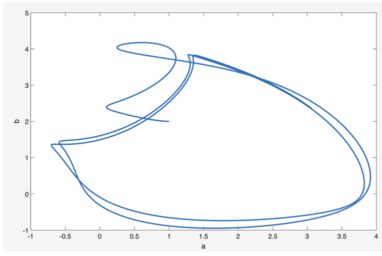
Examining this solution does not seem fruitful, but I will assume that these solutions are unstable.

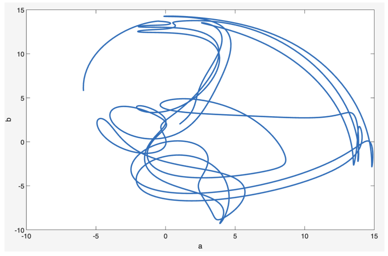
Stability:

One of the first ways I looked into the stability of L96, was by plotting two variables in relationship to each other. In the below example, I did not set the other variables to constants. Instead I let the solver run in the background, but only plotted two of the variables. I then observed as trajectories changed when varying forcing. Below are plots of x_1 in relation to x_2 in an N=4 system, with different forcing constants.

F = 0 F=3 F=15







Looking at the plots above can observe that the solutions begin to 'look' more chaotic, but let's investigate deeper. By setting $G_L(x) = (x_{i+1} - x_{i-2}) \bullet x_{i-1}$, and $x = (x_0, x_1, ..., x_{N-1}, x_N)$ Kerin and Engler are able to rewrite the equation for the whole of L96 as: $\dot{x} = G_L(x) - x + Fe$ with $e = (1, 1, ..., 1, 1)$. This is just a vectorized version of the original lorenz system I introduced above. They write, "It is easy to see that constant functions $x(t) = Fe$ always solve $G_L(x)$, and it is known that these solutions are stable for $-\frac{1}{2} < F < \frac{8}{9}$. As F increases, solutions with spatial and temporal periodicity appear" [10]. Below I have attached their modeling of this behavior.

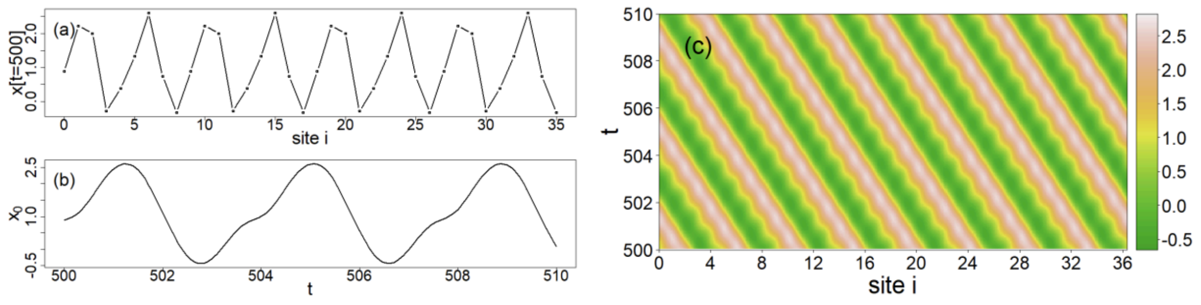


Figure 1: A solution of the L96 system with $F = 2$ starting from random initial data. (a) All 36 sites at $t = 500$. (b) First site for $500 \leq t \leq 510$. (c) Hovmoeller plot for $500 \leq t \leq 510$.

As F is increased, this periodic behavior disappears. Looking at any point (i site) along the horizontal access and following its value as t increases does not show any recognizable pattern like before.

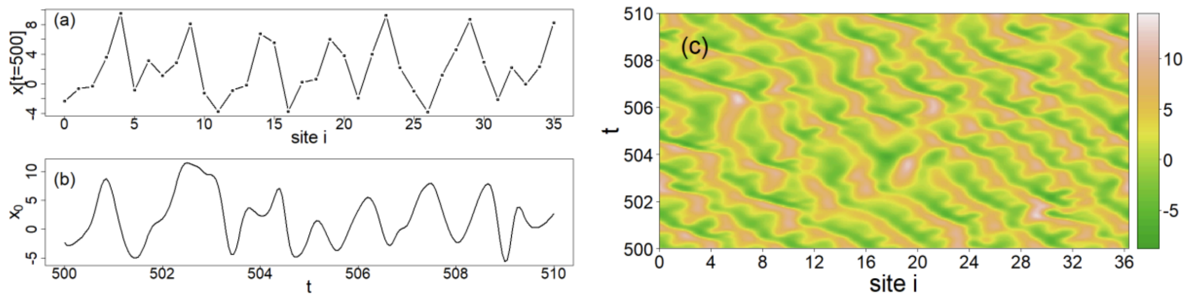


Figure 2: A solution of the L96 system with $F = 8$ starting from random initial data. (a) All 36 sites at $t = 500$. (b) First site for $500 \leq t \leq 510$. (c) Hovmoeller plot for $500 \leq t \leq 510$.

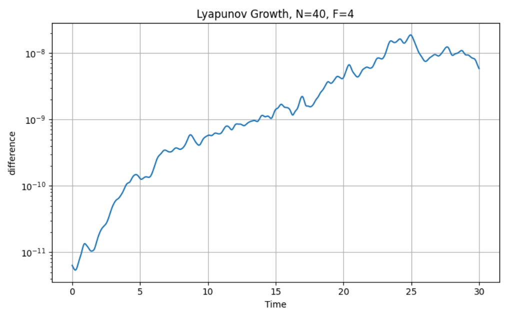
Later on, I will linearize around this Fe equilibrium, and show a hopf bifurcation.

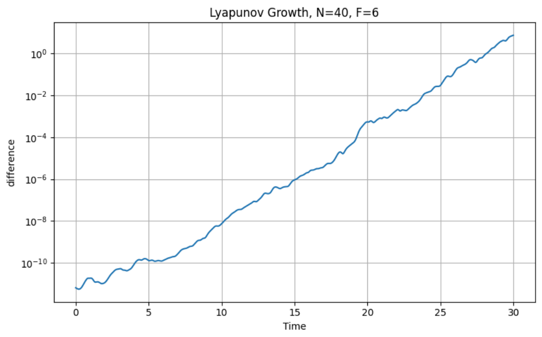
Effects of Perturbation:

As established before, chaotic dynamical systems are unique in that very small differences within the initial conditions of a system can lead to large differences within that same system in the long term. The way I approached studying this is influenced by the techniques established by Wolf et al in their 1985 paper on Lyapannov times and time series [11], but was first implemented by Michael T. Rosenstein and colleagues. In their paper on calculating lyapunov exponents from time series, they write, “Lyapunov exponents quantify the exponential divergence of initially close state-space trajectories and estimate the amount of chaos in a system.” [?]. Rosenstein notes that at the time they wrote this (1992), the most popular algorithm for quantifying chaos was The Grassberger-Procaccia algorithm or ‘GPA’. GPA is simple and easy to use because intermediate calculations are used to estimate both dimension and entropy, but GPA is also sensitive to parameters, especially the dimension of a system [?]. This poses a big problem when working with L96, because it has no fixed size, and varying the size of the system is commonplace when studying it. Rosenstein and colleagues detail a method that depends on techniques too complicated to explore for this project including delay reconstruction and dimension embedding. Instead, below I find the average slope of the lyapunov over time to estimate the lyapunov exponent.

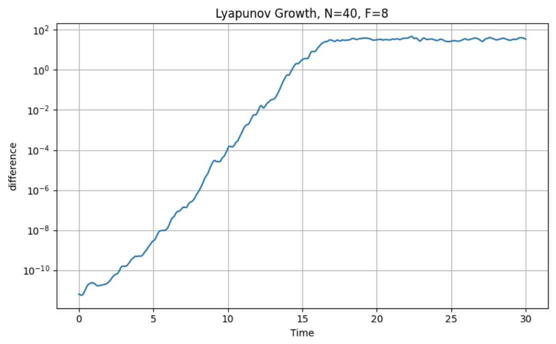
We create a new system with N variables and initial conditions randomly selected within some range A to B. Solve this system (using Euler's method). Then create a new system, with identical initial conditions, but then perturb all of these initial conditions by a small value. Now, after recording the value of each variable on each time t for both systems, compute the norm of each system compared to the other system at each time t. The norm functions as a generalization of ‘distance’ between the solutions, and therefore gives us an idea of how different the two solutions become with only a small perturbation of initial conditions. Norm is defined as the square root of the sum of squares of differences between each corresponding value in both solutions. $||x|| = \sqrt{(a_0 - b_0)^2 + (a_1 - b_1)^2 + ... + (a_N - b_N)^2}$ with a_i being the value in system 1 at i, and b_i being the same in system 2.

Below are the results of plotting the perturbation for several forcing values. In all examples, the initial conditions range between 0 and 5, the perturbation has size 1×10^-6 and Euler's solver runs to time 50 with a stepsize of 0.01.





By computing the aver



nov exponents for each forcing. For F = 4, lyap . For F = 4, lyapunov =1.416. For these specific initia n very close, showing only minimal divergence. In tories to begin diverging more noticeably, with the "distance" between them fluctuating, but staying under 10. The same pattern occurs for F=8, except it ends up hovering right below 100. One other thing to note is that because the graph is on a log scale, the growth that appears to be linear is exponential.

This behavior illustrates how increasing the forcing in the Lorenz-96 system induces chaotic behavior: below a certain threshold, solutions starting from nearby initial conditions remain similar, while above this threshold, small differences grow rapidly. As forcing increases further, chaos emerges earlier in time. In the Lorenz-96 system of size 40, F=8 is commonly observed to induce chaos, though the precise onset depends on the initial conditions.

Linearization:

Now we will linearize around the solution $x_0 = x_1 = ... x_N = F$ solution we found earlier. The jacobian is a matrix consisting of the partial derivative of each differential equation with relation to each variable.

$$\mathbf{J} = \frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} \cdots \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_u} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_u} \\ \vdots & & \vdots \\ \frac{\partial f_v(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_v(\mathbf{x})}{\partial x_u} \end{bmatrix}$$

In the homework, I looked at a linearization of L96*. Here I got

$$J = \begin{pmatrix} -1 & x_3 & x_2 - 2x_3 \\ x_3 - 2x_1 & -1 & x_1 \\ x_2 & x_1 - 2x_2 & -1 \end{pmatrix}$$

At the equilibrium (F,F,F), this becomes.

$$J_{\mathrm{eq}} = \begin{pmatrix} -1 & F & -F \\ -F & -1 & F \\ F & -F & -1 \end{pmatrix}$$

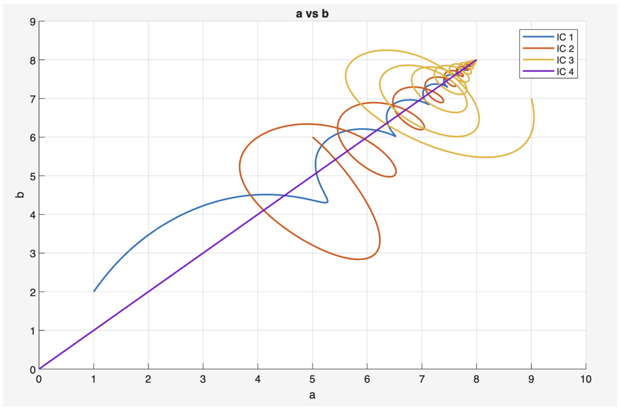
Then, using mathematica I was able to find the eigenvalues of

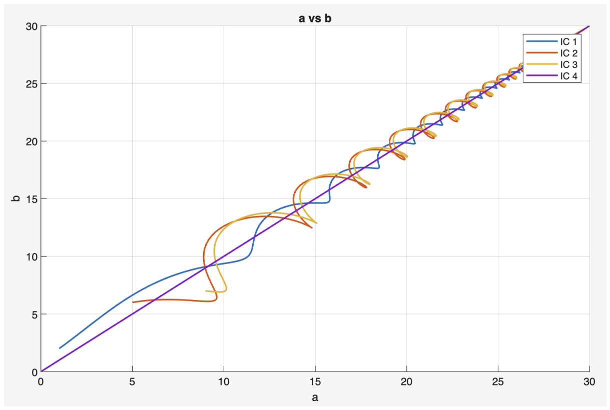
$\lambda_1 = F - 1, \quad \lambda_2 = \frac{1}{2}(- i\sqrt{7}F - F - 2), \quad \lambda_3 = \frac{1}{2}(i\sqrt{7}F - F - 2)$

As long as $F \neq -4, 1$ then there exists real eigenvalues not equal to 0. That means that the linearization is qualitatively similar at all equilibrium where $(x_1, x_2, x_3) = (F, F, F) \neq -2, 1$.

In the case that F = 1, we would have two complex eigenvalues of that linearization with negative real part, so the linearization and original L96* system would have rotating decaying solutions. In the case that F = -4, we would have one real negative eigenvalue, and two complex eigenvalues with real part equal to zero. This solution would also correspond to a rotating decaying trajectory for all solutions towards the equilibrium. We do not know if these behaviors are qualitatively similar to the regular system, because they have eigenvalues with real parts equal to zero.

To visualize these trajectories I used a similar method to my stability portraits, where I plotted a projection of two of the variables of the system. Below are the graphs for $x_1$and x_2 with a forcing of 8-right and 30-left.





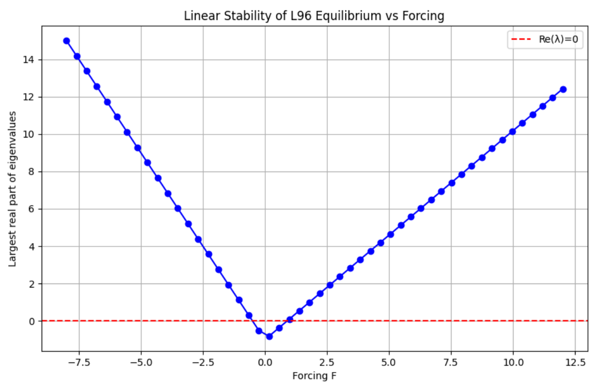
While these are both only plotting A and B, plots of B and C, and A and C yield incredibly similar patterns so it did not make sense to include them. In these examples, solutions spiral and converge at the equilibrium point $(x_1, x_2, x_3) \ = \ (F, F, F) \ \neq - \ 2, 1$. Because of the change of variables by linearization, F becomes the origin, and eigenvalues have negative real parts and non zero imaginary parts, leading to solutions spiraling inwards. This process is also analogous to the dissipation embedded within these equations, where they ‘pass’ energy off to their neighbors, which in this case are only 2 other variables. This leads to the total ‘energy’ of the system oscillating between variables, which is another reason we see the similar spiral shape between all 3 sets of variables.

Now, we will move on to the actual L96 system. Let us briefly observe that each equation is only dependent on $x_{i-2}, x_{i-1}, x_i, and\ x_{i+1}$ This means that all other partial derivatives will be equal to 0. This means the structure of the jacobian is a sparse matrix (majority zero entries) of the form below.

$$J_{ij} = \begin{cases} x_{i+1} - x_{i-2}, & j = i-1, \\ x_{i-1}, & j = i+1, \\ -x_{i-1}, & j = i-2, \\ -1, & j = i, \\ 0, & \text{otherwise.} \end{cases}$$

To contextualize this, we will look at the jacobian analytically across different values of F. Below is a plot of the largest real parts of the eigenvalues of the jacobian for each forcing. We look at real parts of the eigenvalues of each matrix to understand its stability, because real parts of the eigenvalues determine if solutions are converging or diverging. We only look at the largest real part because this convergence or divergence is exponential, and so the largest term will dominate and predict the overall behavior of solutions not directly on other eigenvectors.

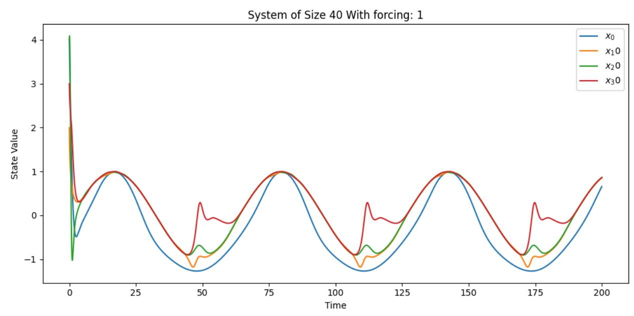
We look at linearizations around equilibriums with forcing between -12 and 12, as this is a similar range where we expect behavior to qualitatively change.

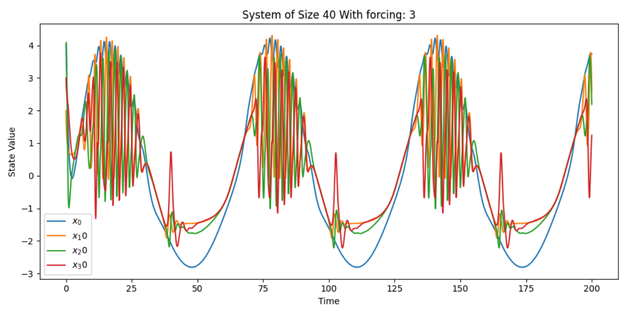


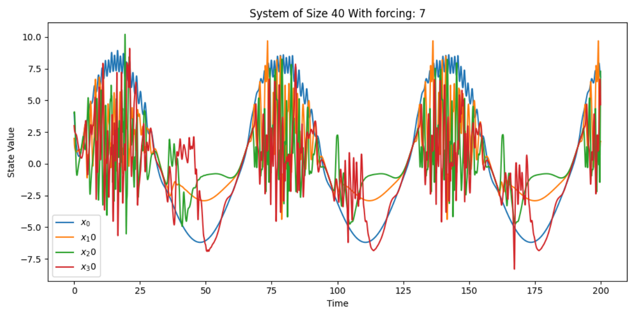
The eigenvalues with the largest real parts, still have real parts equal to and then below zero when the blue line crosses the red line. This means that all solutions must be decaying to zero in the range. Once we exit this range, all solutions become repelling again. Therefore there exists a bifurcation within the stability of solutions of L96 when varying forcing.

Different Forcing:

Above we have looked at the effects of increasing constant forcing values on a system. But now we will explore sinusoidal and heavyside forcing on the Lorenz system. We will do this through a forcing function F(t) = $a \bullet sin(bt)$ with constants a and b, evaluated at each time t and substituted into each differential equation as the forcing at that time. On the homework I visualized this using an animated ring of circles that change color as their value changes, where each circle represents one i site. This sort of dynamic visualization will not work in this paper, so I have instead done timeplots of i sites 0, 10, 20, and 30 for multiple values of A. In other words, I have varied the amplitude of the sinusoidal firing. In these examples, I have set B = 0.1 so that the period of forcing is large, and we can easily observe it going up and down within the system.

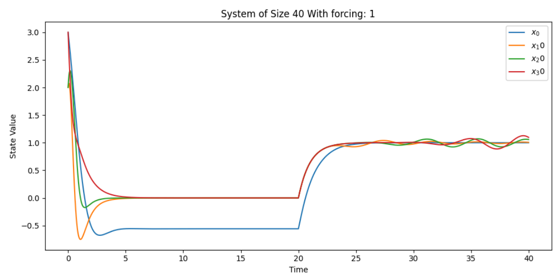


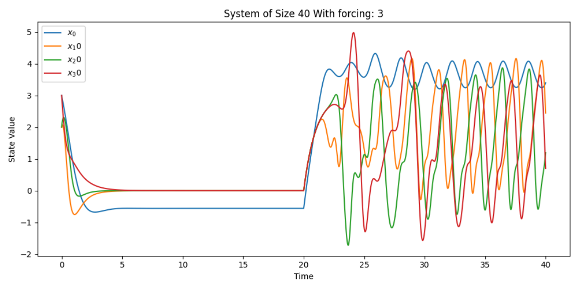


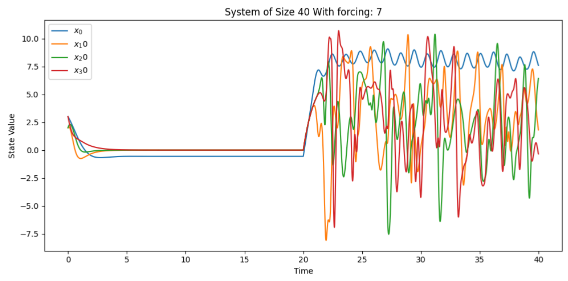


at around t≈23. This follows a similar pattern as constant forcing, where increasing the forcing leads to chaotic behavior which eventually cannot be predicted. Although the solutions for a =1,3, and 7 initially appear to follow a repeating pattern, closer inspection reveals that they do not exhibit true periodicity. In particular, the case a = 7 shows an interesting feature: the system's behavior between successive peaks of the forcing is inconsistent from cycle to cycle. I interpret this as a signature of chaos. When the forcing becomes strongly negative, the system's intrinsic sensitivity to initial conditions dominates, causing small differences in state to diverge into noticeably different trajectories. When the forcing becomes positive again, portions of the state space become more energetically driven, and the system appears to momentarily re-align—not because it becomes predictable, but because the external sinusoid partially entrains the dynamics. This combination of divergence during low forcing and partial synchronization during high forcing is characteristic of chaotic systems subjected to periodic external driving.

We now look at the same forcing values, but constant and heavyside, where they only begin to act at t=20.







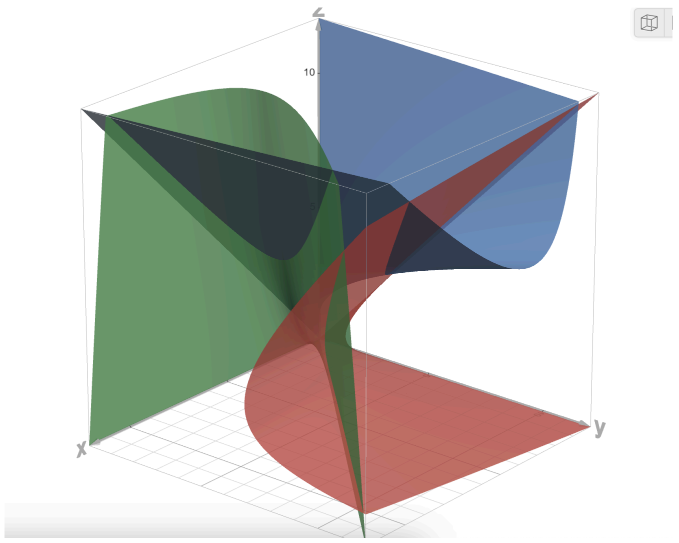
Here we observe similar behavior, of chaotic behavior increasing and eventually blowing up with high enough forcing.

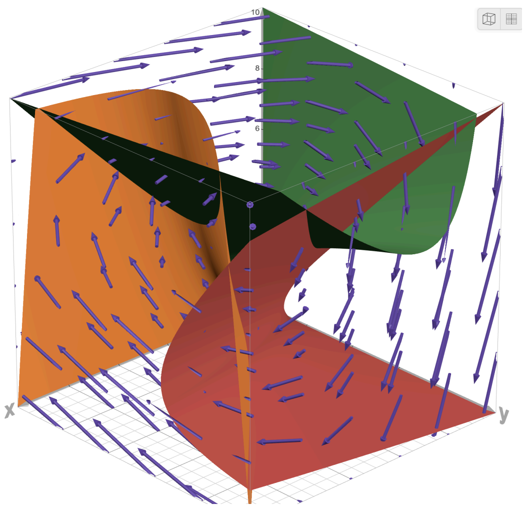
One thing to note is that in these examples we are using classic Euler's methods for the numerical simulation. Euler's method had an error proportional to its step size squared, and so simulating for large amounts of time may lead to compounding error. To avoid any additional risk I am using the numpy library in python which is known to not cause a lot of floating point errors.

Nullclines of L96*:

We will quickly take a look at the nullclines of the reduced three dimensional Lorenz 96 - inspired system to provide a visual representation of nullclines in the context of a system like L96. Below I have attached 3d plots of these nullclines along with a vector field.

Each surface represents the set of points in which $\dot{x}_i$ is equal to 0. In this example all solutions converge into the origin.





The reason we include these visuals is because its self similarity is similar to that of L96. Although L96* converges into the equilibrium and L96 does not, both have equations that are all structurally similar. This self similarity leads to symmetric patterns and the cyclical behavior we see above. But instead of converge towards the original, L96 grows outwards in cyclical pattern analogous to L96* shown above, but in much higher dimensions.

Data Assimilation:

Data assimilation is a fundamental technique used in numerical weather prediction, oceanography, and other geophysical modeling to improve estimates of the current state of a system by combining observational data with forecasts generated by a mathematical model [12]. In practice, both the model and the observations are imperfect: models may contain simplifications of reality or numerical error, and observations may be sparse, noisy, or incomplete. Data assimilation aims to optimally reconcile these two sources of information, producing a more accurate and dynamically consistent estimate of the system's state than either the model or the observations could provide alone [12].

A commonly used method is the Ensemble Kalman Filter, which represents the system state as an ensemble of model simulations, each slightly perturbed to represent uncertainty in the initial conditions and the model itself. When new observational data becomes available, the ensemble is adjusted so that the forecasted state of the system is nudged closer to the observations, weighted by the relative uncertainties of the model and the data. This process is repeated at each observation time, producing a dynamically evolving estimate of the system that incorporates both the deterministic evolution of the model and the information from observations [13]. One other approach is variational data assimilation, often written as 3D-Var or 4D-Var depending on whether it incorporates observations at a single time or over a time window. In these methods, the goal is to minimize a cost function representing the difference between the model and the observations, while also penalizing deviations from the background state or prior knowledge. The result is an optimal estimate of the initial conditions or the trajectory of the system that is consistent with both the observations and the model dynamics.

The Lorenz-96 system provides an ideal testbed for exploring data assimilation because it is a controlled, low-dimensional chaotic system that nevertheless exhibits the hallmark features of real-world atmospheric dynamics, including sensitivity to initial conditions and nonlinear coupling between variables [7]. By generating synthetic observations from a known "true" trajectory of the system, possibly with added noise to simulate observational error, one can systematically evaluate the performance of different data assimilation schemes. For example, one can test how the number and frequency of observations, the magnitude of observational noise, or the size of the ensemble in Ensemble kallman filter methods affect the accuracy of the reconstructed system state. These experiments illustrate that in chaotic systems, errors grow exponentially over time, so even small inaccuracies in the initial conditions or observational data can rapidly lead to divergence from the true trajectory if the assimilation scheme does not account for uncertainty properly [13].

Furthermore, data assimilation emphasizes the interplay between predictability and chaos. In systems like Lorenz-96, the Lyapunov time sets a limit on how far into the future meaningful forecasts can extend. Assimilation methods are therefore designed to continually update the system state using observations, effectively "resetting" the forecast and extending the window over which predictions remain accurate [12]. Ensemble methods, in particular, allow one to quantify the forecast uncertainty and propagate it through time, offering both a best estimate of the state and an indication of the confidence in that estimate. This is directly analogous to real-world weather forecasting, where forecasters combine imperfect models and sparse measurements to provide probabilistic predictions [12].

Data assimilation provides a bridge between theory and application. It demonstrates how even in deterministic but chaotic systems, information from observations can be systematically incorporated to improve forecasts, and it highlights the limits imposed by chaos on predictability. Within the Lorenz-96 framework, data assimilation can be used to test these methods in a controlled setting, providing insight into the strengths and limitations of ensemble and variational approaches, and ultimately help with understanding the challenges of forecasting in chaotic, nonlinear systems [12, 13]. Below I will give an example of implementing the URDA data assimilation method.

The general implementation of the ensemble kalman filter that I did, goes as follows. First, we create f models of our system, representing denoting this 'ensemble' as X^f.

$$\mathbf{X}^f := \left[\mathbf{x}^{f(1)}, \ldots, \mathbf{x}^{f(m)}\right],$$

In our case, this model contains f copies of L96, each with initial conditions scattered in a small area. In my simulation, for each model I created an N length array of random values between -1 and 1 (these are the values that worked best for the model within my undocumented experimentation), then I added this array to the initial values within each model, essentially creating a spread of initial conditions so we have some variation within our forecasts. Below is the vector of perturbations expressed within Kawasaki et al [14].

$$\delta \mathbf{X}^f := \left[\mathbf{x}^{f(1)} - \overline{\mathbf{x}}^f, \ldots, \mathbf{x}^{f(m)} - \overline{\mathbf{x}}^f\right].$$

We then run all of these models forward one time step, in our case using the Runge Kutta RK4 solver. In some of the previous answers, I had used Eulers because it was easy for me to code in python. But I was getting pretty bad results on my Ensemble Kalman Filter, so I switched to RK4 because it is more accurate. This solver works by taking the slope at the start of the time step (k1) $k_1 = f(t_n, y_n),$, predicting the slope at the half way point (k2) using k1 $k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$, then using this new predicted slope to repeat the process and predict a new slope also at the halfway point , but calculated using k2, $k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$. Finally, we predict k4, the slope at the end of this time step using k3, $k_4 = f(t_n + h, y_n + hk_3)$. All of these slope predictions are then weighted and used to predict the next value of our system [15].

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ t_{n+1} &= t_n + h \end{aligned}$$

After using RK4 to predict our forecast forward one time step for all models, we take the mean of these models, or the “middle most” model. $\overline{\mathbf{x}}^f := \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}^{f(i)},$

Now we have all model forecasts for the next time

Step, as well as a mean model. At this point we either receive an observation of the true system, or we do not. If we do not receive an observation, we begin these predictions to the next step, if we do, we must update our models in context with our observations to make our next predictions more reliable. We do this by calculating the error co-variance (expressed below).

This co-variance is a measure o $\mathbf{P}^f := \frac{1}{m-1}\delta\mathbf{X}^f\left(\delta\mathbf{X}^f\right)^\top$ capturing both the uncertainty in the forecast and the way errors in one state variable influence errors in another. We then produce this variance for the analysis as well, using these two to get the kalman gain. I had trouble finding understandable descriptions of how to calculate this, but ChatGPT told me that it could be obtained through matrix multiplication of each covariance with their mean.

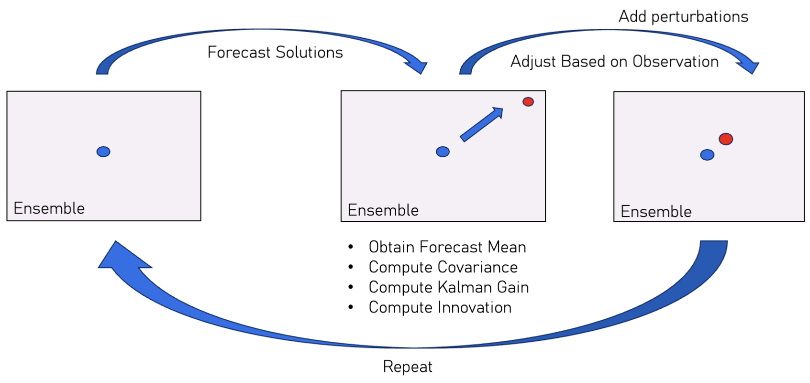
Doing this we obtain an N by N matrix K which will be kalman the kalman gain. We then use this kalman to obtain the innovation. This innovation is defined as d^{o-f} which uses the H operator to reduce our mean vector forecast to the size of our observation (which may have missing values due to faulty instrumentation). $\mathbf{d}^{o-f} := \mathbf{y}^o - H\left(\overline{\mathbf{x}}^f\right) \in \mathbb{R}^p$. This innovation tells you how wrong your forecast was in relation to the observation. Finally, we update our analysis using all of this information to obtain a corrected estimate of the system state.

$$\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^f + \mathbf{K}\left(\mathbf{y}^o - H\left(\overline{\mathbf{x}}^f\right)\right) = \overline{\mathbf{x}}^f + \mathbf{K}\mathbf{d}^{o-f},$$

This estimate blends what the model believed (the forecast) with what the data actually reported (the observation). Mathematically, this update nudges the forecast back toward reality, with the size of this "nudge" controlled by the Kalman gain. If the ensemble forecast shows a large spread in some direction (high uncertainty), the Kalman gain becomes large, meaning we trust the observation more strongly. If the ensemble is tightly clustered (low uncertainty), the Kalman gain becomes small, meaning we trust the model more than the noisy data.

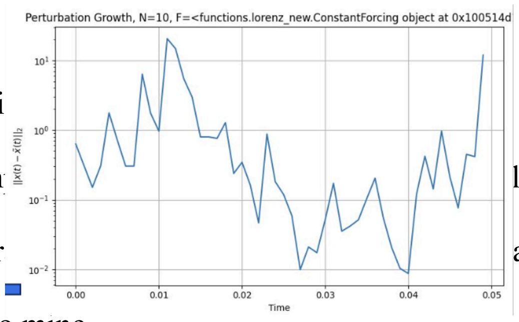
The update is applied to the entire ensemble, not just the mean, which ensures that the spread of the ensemble also changes—reflecting the reduced uncertainty after incorporating real-world information. The corrected ensemble (which we call the analysis ensemble) becomes the starting point for the next forecast cycle. In this way, the ensemble Kalman filter alternates between the forecast: which pushes each model forward according to the system dynamics, and the analysis: which pulls the ensemble back closer to the observation once it arrives.

Here is a chart I made to describe my implementation. Here the blue dots represent the mean forecast, and the red dots represent the observation. The blue arrow represents us updating our analysis to be closer to the observation.



- Obtain Forecast Mean
- Compute Covariance
- Compute Kalman Gain
- Compute Innovation

Here I plotted the distance between my Kalman Forecast prediction means of L96 over time and a stochastic L96 with the same starting condition that I assumed to be a 'real system' to observe



I am not sure why it di did not include noise within my process. Within the im lter I looked at, there was added noise with specific par and updates, but I found this a little daunting to add to mine.

Finishing off my exploration of this system there are a few things I am taking away. Firstly, dynamical systems are unique - at least compared to the math I've learned up until this point - in that they feel a little alive. To study this system, I had to approach it from a few different perspectives, and take more of an experimental approach rather than a direct path, like I am used to for math related projects. The rolling behavior of the system and overall structure of the atmospheric dynamics it represents could be seen through multiple lenses. The second thing I got out of this project is exposure to data driven and numerical methods with my exploration of the ensemble kalman filter and my own study of L96 through python. Earlier in the semester, I was having trouble studying this system, and as a personal little project I made a program that initialized a L96 system object with individual equations I could create, access, and manipulate. Later, I added the Euler's solver and RK4, alongside visualizations to understand my simulations. I had a ton of fun, and got a lot of exposure to vectorized operations in python, alongside a good excuse to read more about computational mathematics, which I really enjoyed. Overall, this project was a blast, and it makes me excited for future studies.

Bibliography

1. Wikipedia contributors. (2025, September 29). Edward Norton Lorenz. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:20, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Edward_Norton_Lorenz&oldid=1314102950

2. Wikipedia contributors. (2025, December 1). Chaos theory. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:07, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Chaos_theory&oldid=1325154313

3. Wikipedia contributors. (2025, November 7). Butterfly effect. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:15, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Butterfly_effect&oldid=1320973755

4. Wikipedia contributors. (2025, February 19). Deterministic system. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:16, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Deterministic_system&oldid=1276521658

5. Wikipedia contributors. (2025, October 19). Lyapunov exponent. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:03, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Lyapunov_exponent&oldid=1317663123

6. Kerin, J., & Engler, H. (2020). *On the Lorenz '96 model and some generalizations*. arXiv. https://doi.org/10.48550/arXiv.2005.07767

7. Lorenz, E. N., and K. A. Emanuel, 1998: Optimal Sites for Supplementary Weather Observations: Simulation with a Small Model. J. Atmos. Sci., 55, 399–414, https://doi.org/10.1175/1520-0469(1998)055<0399:OSFSWO>2.0.CO;2.

8. Wikipedia contributors. (2025, November 15). Lorenz 96 model. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:07, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Lorenz_96_model&oldid=1322235079

9. *Lorenz, E. N. (1995). Predictability: A problem partly solved (Proceedings of the Seminar on Predictability, 4–8 September 1995). ECMWF.*

10. Wikipedia contributors. (2025, September 14). Advection. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:16, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Advection&oldid=1311248193

11. Wolf, A., Swift, J. B., Swinney, H. L., & Vastano, J. A. (1985). *Determining Lyapunov exponents from a time series*. *Physica D: Nonlinear Phenomena, 16*(3), 285–317. https://doi.org/10.1016/0167-2789(85)90011-9

12. Wikipedia contributors. (2025, October 25). Data assimilation. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:05, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Data_assimilation&oldid=1318674427

13. Wikipedia contributors. (2025, November 26). Ensemble Kalman filter. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:20, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Ensemble_Kalman_filter&oldid=1324274963

14. Kawasaki, F., Okazaki, A., Kurosawa, K., & Kotsuki, S. (2025). *Exploring ultra rapid data assimilation based on ensemble transform Kalman filter with the Lorenz 96 model*. arXiv. https://doi.org/10.48550/arXiv.2511.12620

15. Wikipedia contributors. (2025, October 17). Runge–Kutta methods. In *Wikipedia, The Free Encyclopedia*. Retrieved 21:06, December 12, 2025, from https://en.wikipedia.org/w/index.php?title=Runge%E2%80%93Kutta_methods&oldid=1317325865

Inspiration for Ensemble Kalman Filter Design

- Slud, E. (2017). *A tutorial on the ensemble Kalman filter* (RITF17). University of Maryland. https://www.math.umd.edu/~slud/RITF17/enkf-tutorial.pdf

- Labbe, R. (n.d.). *Kalman and Bayesian Filters in Python* (GitHub repository). GitHub. https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python