Chaotic Behavior of Autonomous Nonlinear Circuits

Andrew Garmon

Christopher Newport University

Spring 2013

Abstract

The purpose of this project was to construct, observe, and analyze data from an electric circuit that experiences chaotic behavior. Part of the project was conducting research in order to develop a further understanding of circuitry and chaos. The engineering portion of this project was working with an automated data collection system in order to gather data from a circuit. Over the course this project, I developed practical skills of experimentation and circuitry design as well as skills in stability analysis and systems modeling.

Introduction to Chaos

Chaos is defined by the behavior of dynamical systems which are highly sensitive on initial conditions [7]. This means two points which are arbitrarily close initially, diverge to completely different trajectories in a relatively short amount of time. Most systems which exhibit chaos are extremely peculiar in terms of predictability. They are deterministic, meaning the present determines the future, but because of the sensitivity on initial conditions, the approximate present does not determine the approximate future. This makes long term prediction of an overall system impossible, even though the system is deterministic.

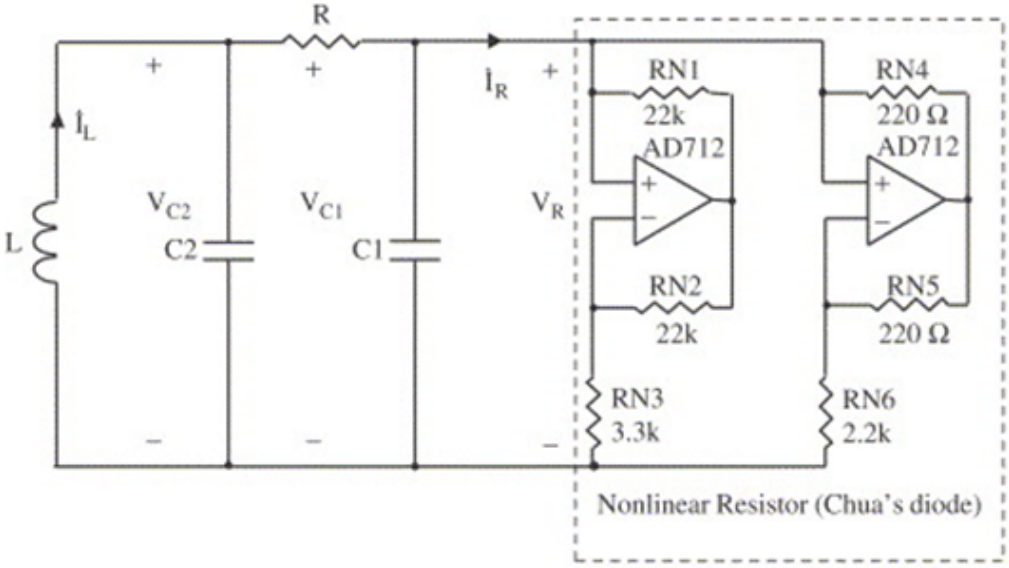
Chaos Theory all began when a physicist named Edward Lorentz created a computer generated system to model the weather; It was a very intricate program. The results never seemed to fully repeat, just like the actual weather. Everyone who was working on the project with him was hoping that he had created the ultimate weather forecasting system, able to predict weather conditions far into the future. One day, as the legend has it, Dr. Lorentz started up another computer running the same model that he had been already running. He then paused the already running system, and printed out its current state conditions. Then he typed them all into the new system, started it, and resumed the already running system. He then went down the hall to get some coffee, and when he returned, both systems were predicting totally different weather. The old system was forecasting sunshine and blue skies, whereas the new system had tornado conditions! He quickly stopped both systems in order to figure out what had gone so terribly wrong. After three months of studying both systems he found the error. When he had paused the first system and printed its current state variables, the system truncated them all to three decimal places. This very slight change in initial conditions, one millionth of a decimal, resulted in drastically different behavior between the two systems.

This is what Dr. Lorentz referred to as the Butterfly Effect. The general idea is: A butterfly flaps its wings in the rainforest and two years later there is a tornado in Kansas. No flap, no tor-

nado. Even Dr. Lorentz admitted that this is quite a farfetched example, but the underlying idea does remain. Any small perturbation from the initial conditions will result in drastically different behavior over time.

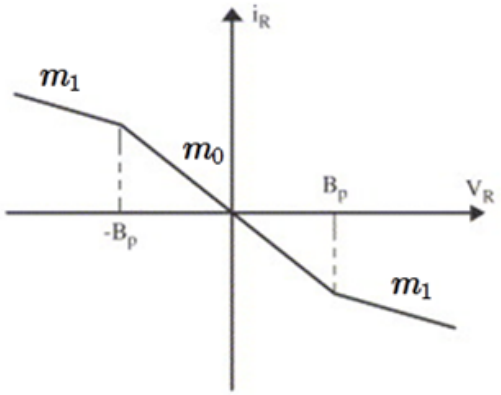
Chua's Circuit

An autonomous circuit consisting of resistors, capacitors, and inductors must satisfy three criteria in order to exhibit chaotic behavior [5]. The circuit must contain (1) at least one active resistor, (2) at least three energy-storage elements, and (3) at least one nonlinear element. The simplest electronic circuit which satisfies these criteria is Chua's circuit. It contains the minimum number of elements to satisfy each condition, minimizing the complexity of the circuit. For the choice of the energy storing elements one inductor and two capacitors were chosen. The simplicity of Chua's circuit is most noticeable by the choice of the nonlinear element. The circuit uses what is known as Chua's diode shown below.



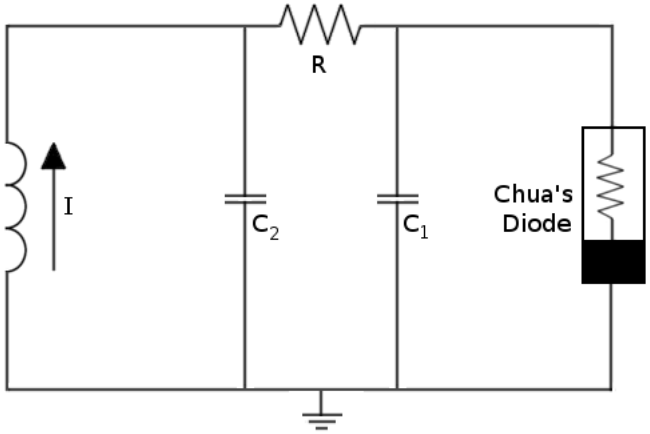
It consists of two operational amplifiers (op-amps) in junction with some carefully selected

resistors. Chuas diode acts as a negative resistance and operates as a function of the voltage across the diode.



$$\phi(v_R) = \begin{cases} m_1 v_R + m_1 - m_0, & \text{if } v_R \leq -1 \\ m_0 v_R, & \text{if } -1 \leq v_R \leq 1 \\ m_1 v_R + m_0 - m_1, & \text{if } 1 \leq v_R \end{cases}$$

The diode consists of a piecewise-linear nonlinearity, arguably the simplest type of nonlinearity. Most forms of nonlinearity contain higher order terms which are much harder to analyze and would complicate to overall system greatly. All together, the components are wired according to the schematic shown below to produce Chua's circuit.



It is clear to see Chua's circuit is the simplest autonomous circuit able to exhibit chaotic behavior. This is due to its satisfaction of the criteria with the fewest number of elements as well as its very simple choice for the nonlinear element. For this reason, we chose to focus on Chua's circuit as it would serve as a basic representation for all autonomous chaotic circuits.

Theoretical Analysis

The elements we are most interested in are the energy-storage elements because it is through these elements that the chaotic behavior is observed. Therefore, we wish to consider how the state of these elements changes in time. We know from the electromagnetic laws of physics how inductors and capacitors change sate as a function of time.

$$\begin{aligned} C\frac{dv_c(t)}{dt} &= i_c(t) \\ L\frac{di_l(t)}{dt} &= v_l(t) \end{aligned}$$

These equations then give us a set of three differential equations describing the state of the first capacitor, the second capacitor, and the inductor. Starting to analyze the circuit as described without simplifications would be incredibly difficult.

Currently, there are eight different state variables, but only three of which were relevant (the voltage in the first capacitor, the voltage in the second capacitor, and the current in the inductor). Thanks to Kirchhoffs laws of electrical circuits we are able to simplify the circuit into just three variables. Kirchhoffs Voltage Law (KVL) states: The directed sum of the electrical potential differences (voltage) around any closed network is zero. Kirchhoffs Current Law (KCL) states: At any node in an electrical circuit, the sum of currents flowing into that node is equal to the sum

of currents flowing out of that node. These two laws provide equations which allow for us to put certain state variables in terms of other state variables. With the use of KVL and KCL we are able to write the set of differential equations in terms of only the three variables in which we were interested in measuring. The nonlinear function can be written as a function of the voltage in the first capacitor rather than the voltage in the nonlinear element because they are in parallel with each other, thus the voltage across both elements is the same.

After the previous calculations we then made a change of variables for aesthetic pleasure. We represent the voltage in the first capacitor, the voltage in the second capacitor, and the current in the inductor as x, y, and z respectively. Then, after consolidating the constants, we are left with the set of differential equations show below.

$$\begin{aligned} \dot{x} &= \alpha(y - x - \phi(x)) \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y \end{aligned}$$

The first step when analyzing a system of differential equations is to find the equilibrium points. An equilibrium point is a point from which the system will not move once there, its rate of change at that point is zero. We find these equilibrium points by setting the inhomogeneous term (right side of differential equation) of each equation equal to zero. Immediately the third equation gives us y equal to zero. Therefore, y drops out of the other two equations, which then results in x equal to negative z from the second equations and negative x equal to the nonlinear phi function. Therefore, x and z will carry a negative symmetry and the equilibrium positions will be where negative x is equal to the nonlinear phi function. Upon solving we find that there are three equilibrium positions. This is caused by the fact that the nonlinear phi function is piece-wise linear into three segments. The three equilibrium points are located at $(0, 0, 0)$, $(3/2, 0, -3/2)$, and $(-3/2, 0, 3/2)$.

The first equilibrium point seems obvious when thought of practically. If there is no voltage in

either capacitor and no current in the inductor then the system will not oscillate. This equivalently says, a dead circuit will remain dead. The other two equilibrium points are a result of the symmetry between x and negative z, and they come from the difference of the slopes in the nonlinear phi function.

After finding the equilibrium points we began to take a deeper look by determining the stability of each equilibrium point. Equilibrium points can be classified into several different subtypes, but overall each point is either stable or unstable. The stability of an equilibrium point is found by finding the eigenvalues of a specific system nearby that equilibrium point. For nonlinear systems, like the system we are analyzing, one must use a Taylor series expansion in what is referred to as linearization. After a linearization we then construct the Jacobian matrix for the specific equilibrium point. The Jacobian is a matrix of all first order partial derivatives with respect to all vectors.

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}.$$

After constructing the Jacobian, one must find the eigenvalues and eigenvectors of the matrix.If all eigenvalues are negative then the equilibrium point is asymptotically stable. This means a small perturbation from the equilibrium point will return directly to the equilibrium point. It is similar to a ball in the bottom of a valley, if the ball is slightly disturbed it will roll back to the bottom of the valley. If all eigenvalues are positive then the equilibrium point is called unstable. Any small perturbation from the equilibrium point will leave and it will never come back. This is similar to a ball resting on the top of a hill, while it will rest peacefully on top of the hill any slight disturbance will start the ball rolling down the hill, and it will not return. If there is a positive eigenvalue and a negative value then the equilibrium is called a saddle point. This is a type of unstable equilibrium, but it is a special case. It is both a local maximum like the hill and a local minimum like the valley.

It is unstable, meaning any small perturbation from the equilibrium point will not return to the equilibrium point. What makes it different from a normal unstable point is that it has an attractive nature as well as a repulsive. From one path it is a stable node, whereas from another path it is an unstable node. This is a unique equilibrium because it can create bounded trajectories due to its 'pushing' and 'pulling' nature.

The results of the stability analysis of the three equilibrium points from the circuit were as follows. All three yielded both a positive and a negative eigenvalue, thus each equilibrium point was found to be an unstable saddle point. This implies that any small perturbation from the equilibrium point will result in a trajectory that breaks free from the equilibrium point. It will not return to the equilibrium point.

Model and Simulation

Because the engineering portion of the project was taking longer than expected I went ahead and programmed a simulation to model Chua's Circuit. The simulation consisted of a little over 1,500 lines of Java code written from scratch. This was done with the help of a friend of mine who had recently worked on GUI programming and still recalled the syntax for the commands. The purpose of the simulation was to produce results that could be plotted to the screen in real time. This data would be able to confirm or deny the theoretical results mentioned above as well as display the overall existence of chaos.

The simulation works in an iterative process. It takes points which are stored in a Points-array and plots them to an *x*-*y* plane one at a time. After each successive point, a straight line is drawn connecting the previous point to the current point. This continuous process is iterated over, giving the simulation the ability to map out a trajectory. To increase efficiency there were changes made to the time step of the draw function as well as the number of points added to the Points-array before

painting. This is just background GUI functionality which is not critical to the overall simulation but is worth mentioning because of the countless hours put into creating such functionality.

The most important feature of the simulation is the Pulse method. This is the method that actually models the physical system. Originally, I had no idea how to model a system computationally given its set of differential equations. Through research I learned the technique of modeling differential equations as difference equations. This technique changes the problem from studying a continuous system to studying a discrete system. It is important to note that this method only works for small time steps. For large time steps, the approximation will differ greatly from the exact solution. Modeling a differential equation as a difference equation is equivalent to representing a curved line as several straight lines pieced together. The difference equation model works by calculating the next point given the current point. Given an initial condition, the computer can iterate through the entire system several times very quickly.

The simplest method to transform a differential equation into a difference equation is Eulers simple method.

$$\frac{dy}{dt} = f(x, y, z, ...) \rightarrow \frac{y_{n+1} - y_n}{\triangle t} = f(x_n, y_n, z_n, ...)$$

Which allows you to write y_{n+1} as

$$y_{n+1} = \triangle t(f(x_n, y_n, z_n, ...)) + y_n.$$

I started with this approach which allowed me to transform our set of differential equations into

$$\begin{aligned} x_{n+1} &= \alpha(y_n - x_n - \phi(x_n)) \\ y_{n+1} &= x_n - y_n + z_n \\ y_{n+1} &= -\beta y_n \end{aligned}$$

The simple Eulers method worked well in modeling the equations and plotting the trajectory, but through more research I had learned that the most accurate method was the Runge-Kutta method. This is a much more complicated, much more computationally intensive method, but it gives the most accurate results. It works by using four different values from the previous point whereas the simple Eulers method only used the value at the beginning of the interval. The Runge-Kutta method takes a weighted average of the slope at the beginning of the interval, two different slopes at the midpoint of the interval, and one last value of the slope at the endpoint of the interval. For example, given $\dot{x} = f(x, y, z)$

$$\begin{aligned} k_1 &= f(x_n, y_n, z_n) \\ k_2 &= f(x_n + \triangle t\frac{1}{2}k_1, y_n, z_n) \\ k_3 &= f(x_n + \triangle t\frac{1}{2}k_2, y_n, z_n) \\ k_4 &= f(x_n + \triangle t k_3, y_n, z_n) \\ x_{n+1} &= x_n + \triangle t\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

This weighted average gives a much more accurate model of the overall system. When I ran the code with the implemented Runge-Kutta method the trajectories were much sharper and tighter. This correction to my model made for a very accurate simulation.

Once I had constructed the simulation of Chua's circuit it was time to test my hypotheses. My first goal was to implement a way for the user to witness the chaos. To do this I needed to double the entire simulation so I could run two concurrently. My goal was that after a relatively small amount of time two trajectories, both with arbitrarily close initial conditions, would completely diverge. I ran an experiment choosing my initial conditions of $(0.7, 0, 0)$ and $(0.701, 0, 0)$. These two initial conditions were only off by one one-thousandth of a decimal in the x-coordinate.

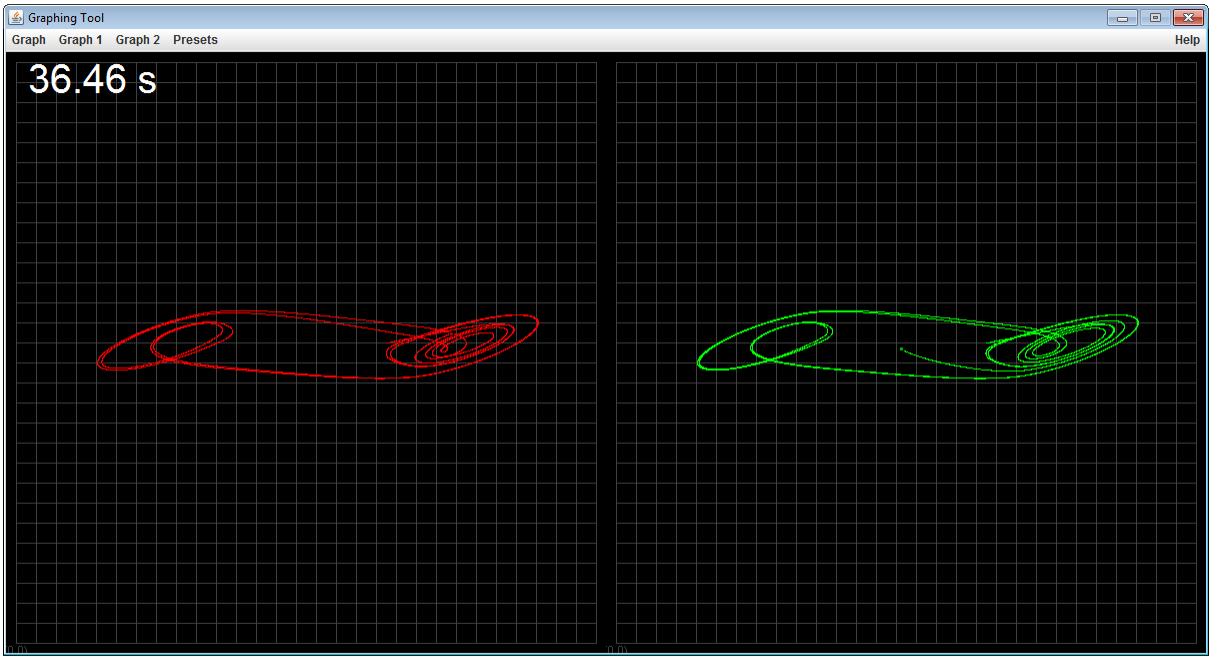


Figure 1: Initial conditions of $(.7, 0, 0)$ on left and $(.701, 0, 0)$ on right.

Just after 36 seconds, the two trajectories with arbitrarily close initial conditions completely diverge. The trajectory on the left continues in a loop whereas the trajectory on the right begins to return to the other scroll. This result was a huge success for the validity of the simulation, and supplied visual evidence of Chua's circuit being a chaotic circuit.

As you can see above, there were points where the two trajectories followed each other closely, but one had a tighter loop than the other. This does not imply chaos. The chaos is illustrated when

the two trajectories completely diverge, not just linearly separating over time. This is the reason why chaotic behavior cannot be attributed to simple computer round-off error over long amounts of time. Computer round-off error would linearly diverge from the trajectory.

After successfully displaying the chaotic behavior by use of the circuit I decided to test the equilibrium points calculated in the theoretical analysis. Recall that all three equilibrium points were saddle-node equilibrium points, meaning they were unstable. The theory then suggests that any small perturbation from the equilibrium point will not return to the equilibrium point. First, to test that the equilibrium points were indeed equilibrium points I ran the simulation with the starting initial conditions at an equilibrium point. The simulation produced a small dot on the screen, and that dot did not move. I tested this for all three points and got the same results. This was evidence that all three points we found were indeed equilibrium points.

The next test for the equilibrium points was to confirm their type of stability (Recall, the analysis found them to be unstable saddle-nodes). To do this I started the simulation with the initial conditions just a small perturbation from the equilibrium points. The image below is a snapshot of the simulation with the left side starting with initial condition at an equilibrium point and the right side starting at just a small perturbation from that equilibrium point.

Figure 2: Initial conditions of $(1.5, 0, -1.5)$ on left and $(1.501, 0, -1.5)$ on right.

This was exactly the result I expected. The theoretical analysis of the equilibrium points was correct. I continued to test each equilibrium point with small perturbations in each dimension and achieved the same result from each run. They were all unstable with the trajectories which were not purely repelled from the equilibrium point. They were all saddle-points, exactly as the theoretical analysis predicted.

The last result I was hoping to produce from the simulation was a long term picture of the attractor. An attractor is defined as a set towards which a variable, moving according to the dictates of a dynamical system, evolves over time. Points that get close enough to the attractor remain close even if slightly disturbed [1]. The attractor of Chua's circuit is a special attractor called Chua's double-scroll attractor [2] It is a called a strange attractor because the trajectory of the attractor is unpredictable but contains an overall pattern. What makes the double-scroll attractor special is that it has two centers about which it unpredictably orbits moving back and forth between center points. It is a recognized chaotic attractor by of its overall pattern and its never repeating.

To produce an image of this strange attractor I ran the simulation from arbitrary initial conditions for an extended period of time. The overall shape of the trajectory is what is recognized as Chua's double-scroll attractor.

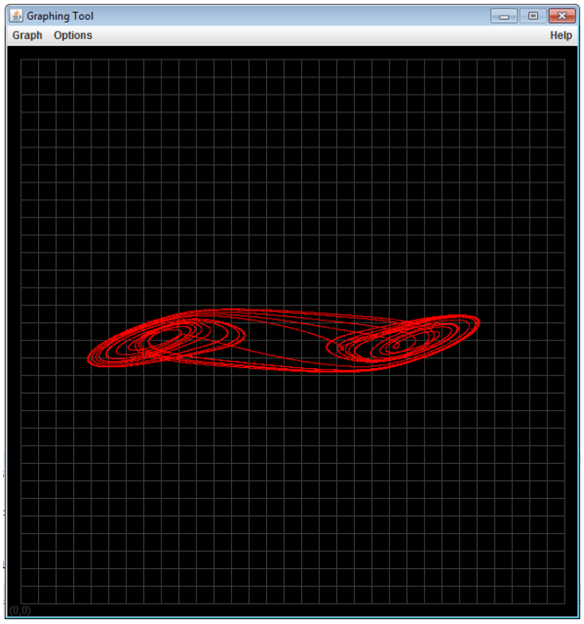


Figure 3: Chaotic attractor from simulation

This is exactly the image expected as it matches the well-known image of Chua's double-scroll attractor [6].

Experimental Analysis

The experimental portion of this project did not come easy. The original course of the project involved acquiring and storing data from the physical circuit, then with that data we would be able to analyze the circuit. The construction of the physical circuit took much longer than expected due to the extreme sensitivity of the overall circuit. We started with all gold-band resistors and circuit components that were within five percent of the required value. This caused the error to amplify and we were not able to find the bifurcation point at all. The bifurcation point is the point

at which the circuit goes from periodic to chaotic. We find this bifurcation point by slowly varying the resistance in the variable resistor. We had a very sensitive potentiometer that we moved very carefully in order to find the bifurcation point. Once we discovered that gold-band components would not satisfy the constraints of the sensitivity we ordered and replaced those components with more accurate ones.

With more accurate components it was still incredibly hard to find the bifurcation point and keep the system in its chaotic state. Far too often it went into its chaotic state then immediately crashed down to a single point. We believe this point is where the resistance on the potentiometer was too high, and it killed the circuit. As the year went on we tried several different techniques of conducting the transition to the bifurcation point, all with little success. It seemed like all hope was lost for acquiring data from the physical circuit. It was just too sensitive.

We eventually were able to overcome all the intangibles and get the circuit to remain in its chaotic state. It was so incredibly sensitive that touching a capacitor would knock it out of the chaotic state. Even small vibrations, like the shaking of a table, would disrupt the circuit. For reasons explained in the engineering portion of this paper, data acquisition was not able to accurately be performed. What was attainable were images from the oscilloscope measuring the data directly from the circuit. The physical circuit moved with such a high frequency that a single image displayed enough data to make a clear representation of the double-scroll attractor.

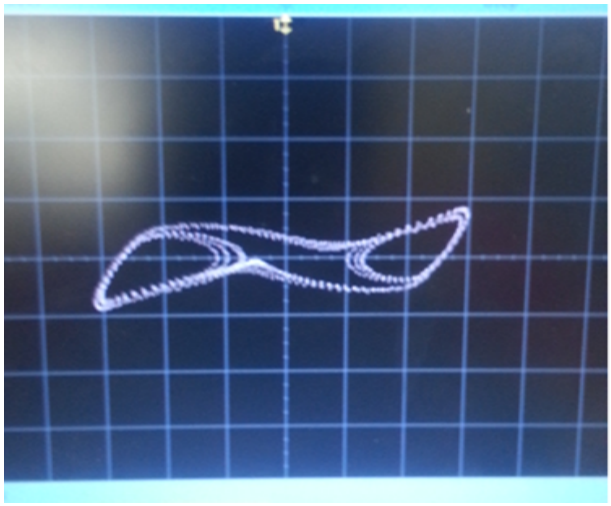


Figure 4: Chaotic attractor from oscilloscope

As you can see, there is a clear connection between the attractor from the simulation and the attractor from the physical circuit (Figure 3). This was a major accomplishment for the project. Being able to connect the simulation to the physical circuit was quite remarkable.

Conclusion

Throughout this project we constructed, observed, and analyzed data from an electric circuit that experienced chaotic behavior. We were able to do this with the selection of Chua's circuit as it was a simple case of such a circuit. I derived the equations of Chua's circuit from the very basic differential equations of electrical circuits and Kirchhoffs fundamental laws of circuitry. I performed theoretical analysis on the circuit, discovering its equilibrium points and determining their type of stability. I wrote a simulation modeling Chua's circuit in order to confirm the theoretical results. From the simulation I was able to prove Chua's circuit to be chaotic as well as present an image of Chua's double-scroll attractor. We were able to transition the circuit through the bifurcation point and into its chaotic state. From its chaotic state we were able to capture an image of the chaotic double-scroll attractor, successfully linking the theoretical analysis to the experimental data.

There were several limitations in constructing the circuit. The most troublesome of which was the sensitivity. The sensitivity could be contributed to several factors, but most specifically, the quality of the circuit. In future research, I recommend using a circuit board rather than a bread-board when constructing the circuit. I would also recommend using components which are within one percent of required value. Expansion of this project would include data acquisition of all three state variables as well as expanding the simulation to three dimensions.

References

[1] Bronshtein, I. N. Handbook of Mathematics. *Handbook of Mathematics*. Berlin: Springer, 2004. Print.,

[2] Chua, Komuro, and Matsumoto. *"The Double Scroll Family."* IEEE Xplore. IEEE, 1986. Web. 29 Apr. 2013.

[3] Hirsch, Morris W., Stephen Smale, Robert L. Devaney, and Morris W. Hirsch. Differential Equations, Dynamical Systems, and an Introduction to Chaos. San Diego, CA: Academic, 2004. Print.

[4] Hirsch, Morris W., Stephen Smale, Robert L. Devaney, and Morris W. Hirsch. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. San Diego, CA: Academic, 2004. Print.

[5] Kennedy, M. P. *"Three Steps to Chaos. II. A Chua's Circuit Primer."* IEEE Xplore. IEEE, Oct. 1993. Web. 29 Apr. 2013.

[6] Letellier, Christophe. *A Chua Electronic Circuit. Digital image*. Atomosyd.net. N.p., 13 Oct. 2009. Web. 20 Apr. 2013.

[7] Strogatz, Steven H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Reading, MA: Addison-Wesley Pub., 1994. Print.

Appendix

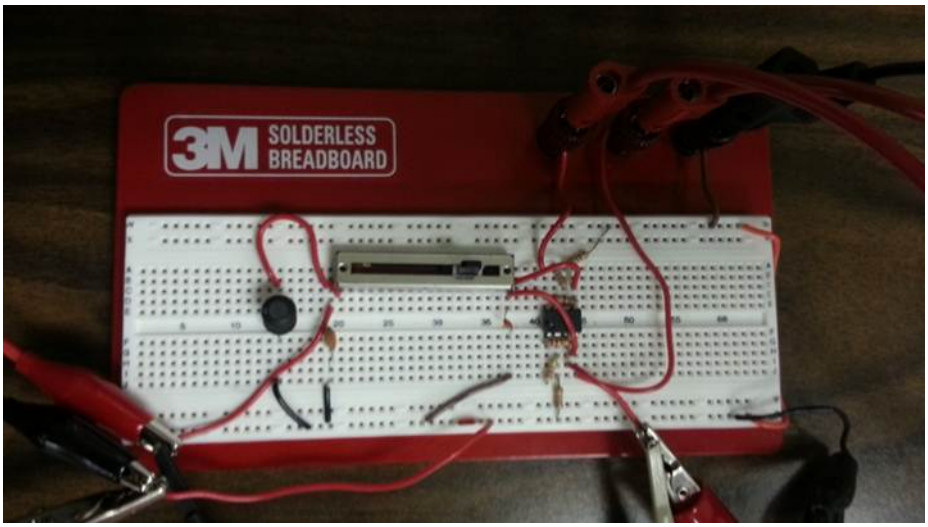


Figure 5: Physical Circuit Layout

```
private double x1;
private double y1;
private double z1;

private double x2;
private double y2;
private double z2;

private double f;
private double x1new;
private double y1new;
private double z1new;
private double x2new;
private double y2new;
private double z2new;

double k1;
double k2;
double k3;
double k4;

private final double a = 15.6;
private final double b = 28;
private final double m0 = -(8.0/7.0);
private final double m1 = -(5.0/7.0);

private final double delta_t = 0.001;
```

```
public double phi(double x) {
    f=m1*x+0.5*(m0-m1)*(Math.abs(x+1)-Math.abs(x-1));

    return f;

}

/**
 * this is the method which is called on each pulse of the si
 */
public void pulse() {

    timerCount +=(double)TIME_STEP/1000.0;
    timerLabel.setText(String.format("  %.2f s", timerCount))


    for(int i = 0; i<10;++i) {

        //calculate new x
        k1=(y1-x1-phi(x1));
        k2=(y1-(x1+k1/2.0)-phi(x1+k1/2.0));
        k3=(y1-(x1+k2/2.0)-phi((x1+k2/2.0)));
        k4=(y1-(x1+k3)-phi((x1+k3)));
        x1new=x1+delta_t*a*(k1/6.0+k2/3.0+k3/3.0+k4/6.0);

        //  xnew=x+delta_t*(a*(y-x-phi(x)));

        //calculate new y
        k1=(x1-y1+z1);
        k2=(x1-(y1+k1/2.0)+z1);
        k3=(x1-(y1+k2/2.0)+z1);
        k4=(x1-(y1+k3)+z1);
        y1new=y1+delta_t*(k1/6.0+k2/3.0+k3/3.0+k4/6.0);

        //   ynew=y+delta_t*(x-y+z);

        //calculate new z
        k1=-b*y1;
        k2=-b*y1;
        k3=-b*y1;
        k4=-b*y1;
        z1new=z1+delta_t*(k1/6.0+k2/3.0+k3/3.0+k4/6.0);

        x1=x1new;
        y1=y1new;
        z1=z1new;

        panel1.addPoint((int) (x1*100+300),(int) (y1*100+300)

    }
```

Figure 6: Relevant Java Simulation Code Implementing Runge-Kutta Method with Simple Euler's Method Commented-out