Modeling Chaos and Different ODEs with Mathematica

Edbert Handjaja

Abstract

The year 1961 was a turning point in mechanics and mathematics. This year, American mathematician and meteorologist Edward Norton Lorenz discovered chaos theory: a field of mathematics which asserts that non-linear dynamical systems are highly sensitive to changes in initial conditions. Lorenz was able to depict his findings through a model, the Lorenz attractor. The goal of this paper is to provide an introduction to chaos theory as well as the procedure for modeling it.

1. Introduction

As a child, you probably heard about the saying, “one thing leads to another”. This is known as the butterfly effect. The expression encapsulates the more technical notion of sensitive dependence on initial conditions in chaos theory. Fundamentally, the expression is a metaphor coined by scientist Edward Lorenz denoting the unpredictability of chaotic systems (Palmer et al, 2014).

Chaos theory is a scientific concept and mathematical division that focuses on the hidden patterns and predetermined principles of dynamical systems that are extremely sensitive to initial conditions and were previously assumed to have completely random states of disorder and irregularity. Long-term predictions of the theory’s behavior are still impossible to comprehend due to its sensitivity.

Physicists, astronomers, biologists, and scientists from a variety of fields have constructed a new perspective for complexity in nature over the last few decades: mathematical modeling. It is unarguable that this concept has grown and improved dramatically over the last few decades. A model is made up of time-dependent equations, which are typically written as differential equations for continuous models and difference equations for discrete models. These models can depict real-life situations and have proven to be extremely useful in the scientific world, including the study of physical and natural sciences, economics, and engineering. Weather forecasting, ocean stream patterns, disease spread, and population analytics are just a few of the countless examples.

Historically, scientists have believed in the possibility of long-term weather forecasting. However, Henri Poincare, a French mathematician, astronomer, and dynamist, was the first to recognize that (Duplantier et al, 2014):

"it may happen that small differences in the initial conditions produce very great ones in the final phenomena. A small error in the former will produce an enormous error in the latter. Prediction becomes impossible, and we have the fortuitous phenomenon."

Technology was far more primitive in the early twentieth century than it is today. There was no mathematical modeling available to depict the potential error gap caused by changes in initial conditions.

In 1961, MIT meteorologist Edward Lorenz expanded his research on this phenomenon now that computers could comprehend and investigate the behaviors of continuous models over a longer time period. He worked at the period when measurements, theory, and simulation models were revealing the nature of the general circulation of the atmosphere and the technology of numerical weather prediction was being developed (McWillians et al, 2019). Lorenz, like Poincare, observed that minor changes in a dynamic system such as the atmosphere could have massive and often unexpected consequences. Lorenz investigated the phenomenon in-depth and discovered three non-linear differential equations that exemplified the complex chaotic behavior of the theory. His finding has inspired others in various fields to investigate more about chaotic systems. Subsequently chaos theory has played a major role in nonlinear dynamics, contemporary physics, engineering, and mathematics. Because of the potential applications in different domains of science and technology, studies of chaos have grown in popularity (Munmuangsaen et al, 2018).

1.1 Attractors

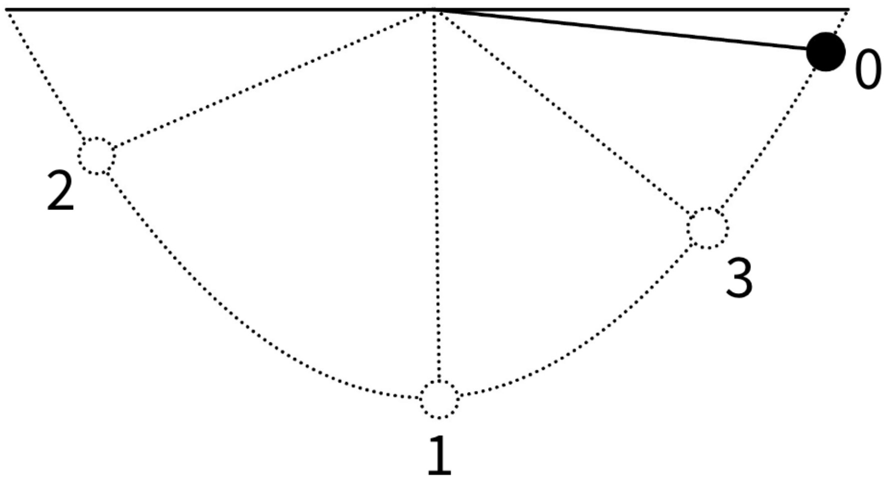
Before diving into this theory, let us take a step back. Dynamical system is a central concept in chaos theory. What exactly is a dynamical system? Simply put, dynamical systems is any system whose evolution due to changes to their initial condition could be predicted and modeled by a few rules, often as equations. Similarly, a differentiable dynamical system is a dynamical system defined by an evolution equation.

$$\frac{dx}{dt} = f(x) \tag{1}$$

As continuous-time cases, such equations highlight the presence of the philosophy that changes produce effects. (Eckmann, 1985). Many objects and instances obey the rules of this system; they are known as attractors: a set of states that a system tends to evolve to, which varies depending on the system's initial conditions.

1.1.1 Non-chaotic attractors

Consider a pendulum of mass m which is attached to a string so that it is able to swing back and forth from some initial position. We will consider two components of the pendulum in this example: the velocity of the pendulum, v, and the position of the pendulum, measured as the displacement from its initial position in degrees, x. Consider the figure below with some point (position 0) as the pendulum's initial position. With an initial velocity of 0, the pendulum is released from the initial position. As it approaches point 1, the velocity increases due to gravity (-9.8 m/s). After passing point 1, the pendulum will continue moving—but directed at an upward angle. Consequently, the speed will continually decrease to some point (position 2). It will then swing back passing point 1 with less speed than its first passage towards some point (position 3). This will continue until the pendulum returns to point 1 assuming that its energy is dissipated by friction or air resistance as it moves (Tsonis et al, 1989). As demonstrated by this illustration, a simple system such as the complete evolution of a pendulum can be shown by just 2 variables—in this case, velocity, v, and position, x.



1.1.2 Chaotic/Strange attractors

As opposed to chaotic attractors, whose behavior is practically unpredictable, the non-chaotic attractors are predictable and “well-behaved”. This is because, while non-chaotic attractors are integer-dimensional attractors whose final point in phase space is predictable, strange attractors are quite the contrary. One main attribute that differentiates these attractors from chaotic attractors is that its complete evolution is not sensitive to changes in their initial conditions: chaotic attractors are ones whose behaviors are simply unpredictable. Unlike non-chaotic, well behaved attractors, chaotic attractors are much harder to illustrate: they must consist of both Lyapunov exponents—a quantity denoting the rate of separation of infinitely close trajectories—and dissipation criteria (Zarei, 2015).

In 1963, Edward Lorenz discovered the first system of this type: the Lorenz Attractor. The system showed the behavior of a model which illustrated a gas in a solid rectangular box with a heat source at the bottom. Simplifying a few fluid dynamics equations, Lorenz came up with 3 differential non-linear equations and 3 initial conditions:

$$\frac{dx}{dt} = P y - P x \tag{2}$$

$$\frac{dy}{dt} = R x - x z - y \tag{3}$$

$$\frac{dz}{dt} = x y - B z \tag{4}$$

$$x(0) = x_o \tag{5}$$

$$y(0) = y_o \tag{6}$$

$$z(0) = z_o \tag{7}$$

P representing the ratio of fluid viscosity to its conductivity, R representing the temperature difference between the top of the model and the bottom of the model, and B representing the ratio of the model’s box’s length to width. Lorenz’ values for these equations were such that P = 10, R = 28, and B = 8/3 (Lorenz, 1963).

Through this attractor, he showed that even the slightest changes in initial conditions can lead to a very different result in the long-term. In fact, American physicist Edward Ott once said (Ott, 1996):

> “the key observation is that a chaotic attractor typically has embedded within it an infinite number of unstable periodic orbits”

It is this very thing, sensitivity to initial conditions, that prevents us from predicting weather at a 100% accuracy. Though the model appears simple and straightforward when depicted as three equations, it represents a complex dynamical system. Despite this, this paper will go in-depth on the modeling of such attractors using a software known as Wolfram Mathematica.

1.2 Goals and Thesis

I believe that more research into chaotic systems, particularly those modelled by three Ordinary Differential Equations (ODEs), should be conducted because there may be other systems in engineering science that are governed by a system of these three ODEs that also exhibit chaotic properties. In this paper, I will not only investigate the process of modeling Lorenz’ original equations, but will also experiment on other differential equations that may exhibit chaotic behavior.

2. Methodology

2.1 Mathematica

Mathematica is a high-computational software system which includes over 6000 built-in functions and options. It is mostly used in many engineering, scientific, mathematical, and computing fields to solve, calculate, and model mathematical concepts and problems. This theory’s modeling will only use a subset of the many functions (NDSolve, WorkingPrecision, and ParameticPlot3D) and options (SetPrecision and PrecisionGoal). Though these functions and options may appear to be perplexing, they are not. Listed below are the functions and options that will be utilized.

2.1.1 NDSolve and NDSolveValue

The first function is NDSolve. This function is built-in in the Wolfram Mathematica software. Mostly used to solve general numerical differential equation, the function has the ability to solve nonlinear boundary-value problems and a second-order partial differential equation subjected to inconsistent boundary and initial conditions on its own. To denote and utilize this function in Mathematica, the format to be used is as follows: NDSolve[eqns, u, {x, xmin, xmax}]. This will program the system to find a numerical solution to the differential equations eqns for the function u with the independent variable x, ranging from xmin to xmax.

Another similar built-in function, NDSolveValue, will also be used in the modeling of chaos. The major difference between these two functions is quite recognizable: NDSolve returns a list of solutions to the given general numerical differential equations as rules whereas NDSolveValue returns a specific solution to the equations (Mikhailov, 2007). To denote and utilize this function in Mathematica, the format to be used is as follows: NDSolveValue[eqns, expr, {*x*, xmin, xmax}]. This will program the system to instead give a specific value to the expression expr determined by a numerical solution to the differential equations eqns for the function u with the independent variable x, ranging from xmin to xmax.

2.1.2 ParameticPlot3D

A parametric plot compares one function to another that uses a same independent variable. The ParametricPlot3D function generates a three-dimensional space curve (Wolfram). When plotted over a 1D domain, the plot is known as a parametric curve, and when plotted over a 2D domain, it is known as a parametric surface. To denote and utilize this function in Mathematica, the format is as follows: ParametricPlot3D[{fx,fy,fz},{u,umin,umax}]. This will program the system to plot a three-dimensional space curve of the three equations, fx, fy, and fz, parametrized by the variable u, ranging from umin to umax.

2.1.3 SetPrecision

The SetPrecision option for NDSolve is used to increase the precision of a certain number or value. When this function is used to increase the precision of a value, that value is augmented with zeroes. The augmented zeroes are always taken in base 2 form. To denote and utilize this function in Mathematica, the format to be used is as follows: SetPrecision[expr,p]. This will program the system to return a version of expr whose numbers are already set to precision p.

2.1.4 PrecisionGoal

Like SetPrecision, PrecisionGoal is also an option for NDSolve. It is a numerical operation option that defines how many effective digits of accuracy should be sought in the final result. To denote and utilize this function in Mathematica, the format to be used is as follows: PrecisionGoal ⟶ x, where x is the number of digits precision the number or value will be set to. To further demonstrate this option, shown below is the format in which the option is used whilst operating the function NDSolve.

```
sol = NDSolve[{x'[t] == 0, x[0] == 10}, x, {t, 0, 5}, PrecisionGoal → 10];
```

In this example, the option PrecisionGoal approximates the integral to at least 10 digits of precision. (Wolfram)

2.1.5 WorkingPrecision

Similar to PrecisionGoal, WorkingPrecision is also a built-in option in Wolfram Mathematica. However, while PrecisionGoal defines how many digits of accuracy should be sought as a minimum, WorkingPrecision does the exact same work but defines how many digits of accuracy should be maintained in computations. To denote and utilize this function in Mathematica, the format to be used is as follows: WorkingPrecision ⟶ x. Setting WorkingPrecision to x will cause all internal computations to be done to at most x-digit precision (Wolfram).

3. Results

3.1 Lorenz Attractor with Mathematica’s NDSolveValue and ParametricPlot3D (1)

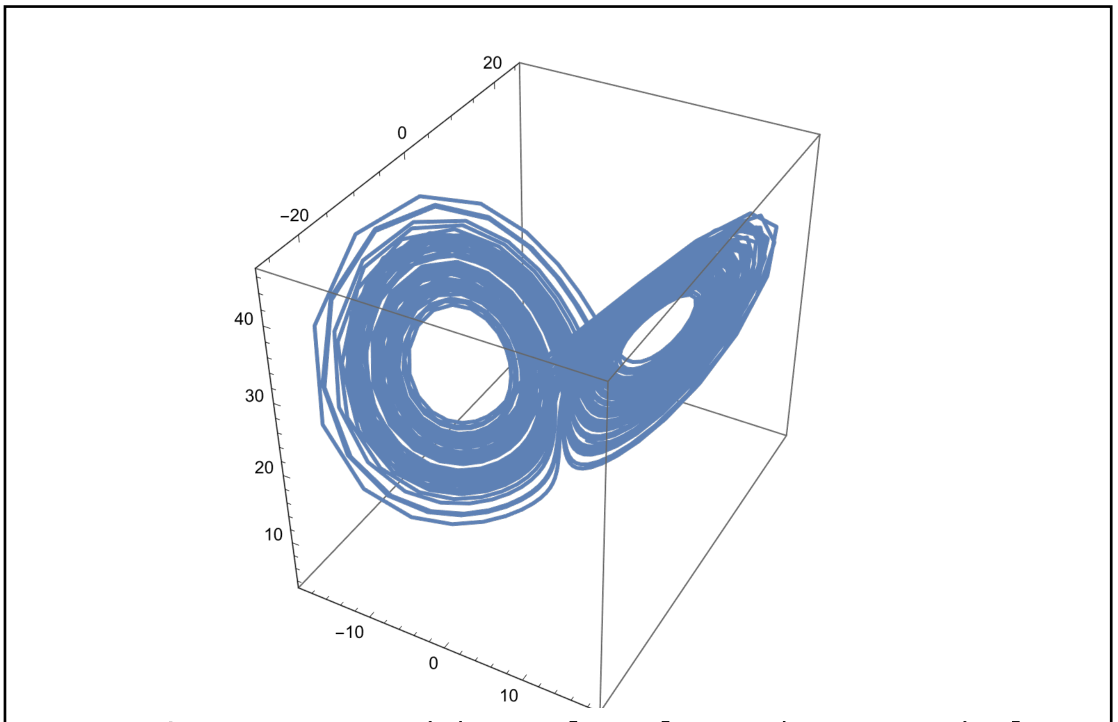
3.1.1 Procedure

1. Find or derive Lorenz’ three non-linear differential equations and three initial conditions with respect to time, t.
2. Find or make parameters for unknown values in these 6 equations and conditions.
3. Create a Mathematica notebook.
4. Input the assigned parameters for the unknown values as the first line of input.
5. Assign both the three differential equations and the three initial conditions to an equation variable eqns as shown in the example below.
6. Use the NDSolveValue function to solve for x, y, and z from the equations assigned to eqns, while also setting a parameter for time, t. These solutions—in the form of an interpolating function—will be assigned to a variable, s.
7. To plot the solutions, s, use the ParametricPlot3D taking each of the solutions to the differential equations with respect to time, as shown in the example below.

3.1.2 Model

```
In[104]:= P = 10; R = 28; B = 8 / 3; xo = 1; yo = 5; zo = 10;
          eqns = {x'[t] == P y[t] - P x[t],
              y'[t] == R x[t] - x[t] × z[t] - y[t],
              z'[t] == x[t] × y[t] - B z[t],
              x[0] == xo, y[0] == yo, z[0] == zo
            };
          s = NDSolveValue[
              eqns,
              {x, y, z},
              {t, 0, 100}
            ];
          g1 = ParametricPlot3D[{s[[1]][t], s[[2]][t], s[[3]][t]}, {t, 0, 100}]
          Framed@Labeled[g1,
            Style["1.Lorenz's attractor with NDSolveValue and ParametricPlot3D",
             Black, 16], Alignment → {Bottom, Center}]
```

Out[108]=



1.Lorenz's attractor with NDSolveValue and ParametricPlot3D

3.2 Lorenz Attractor with Mathematica's NDSolve and ParametricPlot3D (2)

3.2.1 Procedure

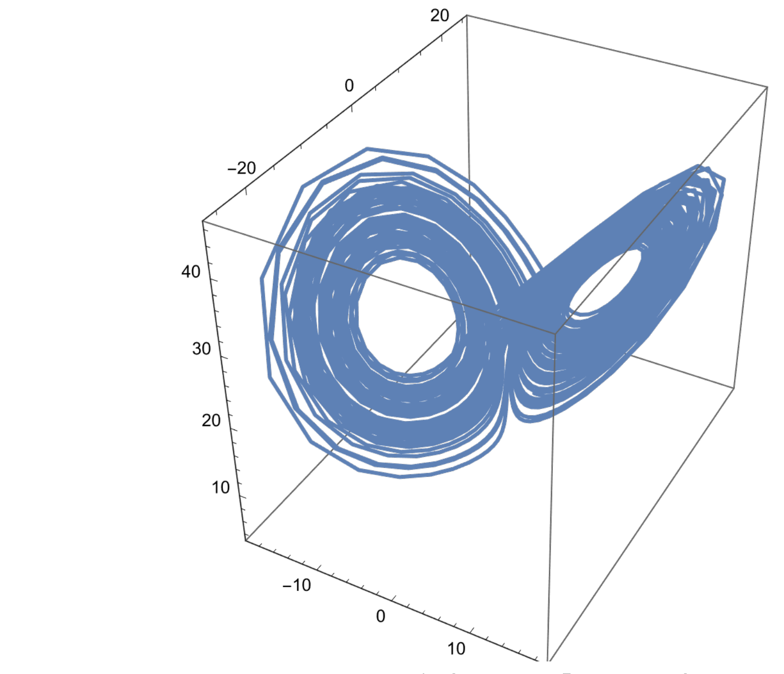
1. Find or derive Lorenz' three non-linear differential equations and three initial conditions with respect to time, t.
2. Find or make parameters for unknown values in these 6 equations and conditions.
3. Create a Mathematica notebook.

4. Input the assigned parameters for the unknown values as the first line of input.
5. Assign both the three differential equations and the three initial conditions to an equation variable eqns as shown in the example below.
6. Use the NDSolve function to return a list of solutions to the given differential equations, assigning them to a variable, s1.
7. To plot the solutions, s1, use the ParametricPlot3D evaluating functions x, y, and z replacing all the functions with s1 with respect to time, as shown in the example below.

3.2.2 Model

In[124]:=
```
P = 10; R = 28; B = 8 / 3;
eqns = {x'[t] == P y[t] - P x[t],
    y'[t] == R x[t] - x[t] × z[t] - y[t],
    z'[t] == x[t] × y[t] - B z[t],
    x[0] == xo, y[0] == yo, z[0] == zo
  };
s1 = NDSolve[eqns, {x[t], y[t], z[t]}, {t, 0, 100}];
g2 = ParametricPlot3D[Evaluate[{x[t], y[t], z[t]} /. s1], {t, 0, 100}]
Framed@Labeled[g2,
  Style["2.Lorenz's attractor with NDSolve and ParametricPlot3D", Black, 16],
  Alignment → {Bottom, Center}]
```

Out[128]=



2.Lorenz's attractor with NDSolve and ParametricPlot3D

3.3 Lorenz Attractor with Mathematica’s NDSolve, ParameticPlot3D, and Precision options (3)

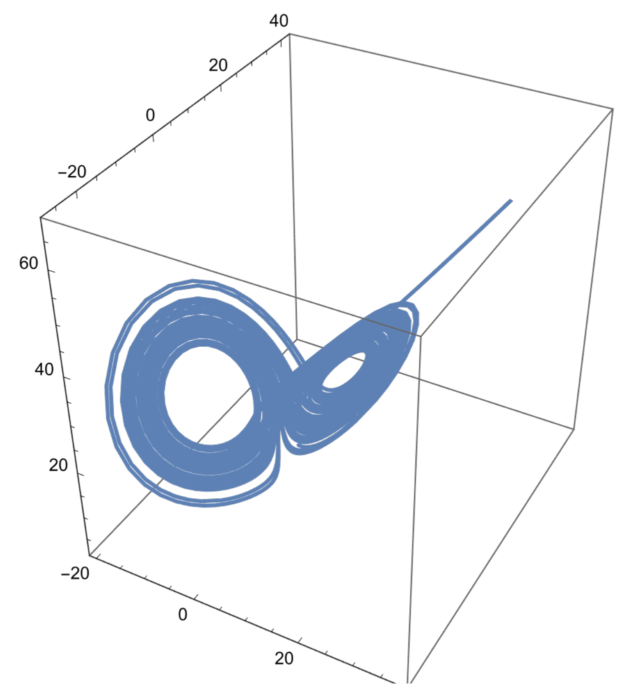
3.3.1 Procedure

1. Find or derive Lorenz’ three non-linear differential equations and three initial conditions with respect to time, t.
2. Find or make parameters for unknown values in these 6 equations and conditions.
3. Create a Mathematica notebook.
4. Input the assigned parameters for the unknown values as the first line of input.
5. Assign both the three differential equations and the three initial conditions to an equation variable eqns as shown in the example below.
6. Use the NDSolve function to return a list of solutions to the given differential equations, assigning them to a variable, s2. Subsequently, use the SetPrecision option to set the eqns variable to have a precision of infinity and use both PrecisionGoal and WorkingPrecision as demonstrated below, adjusting the values to your desired precision.
7. To plot the solutions, s2, use the ParametricPlot3D evaluating functions x, y, and z replacing all the functions with s2 with respect to time, as shown in the example below.

3.3.2 Model

```
In[119]:= P = 10; R = 28; B = 8 / 3;
eqns = {x'[t] == P y[t] - P x[t],
    y'[t] == R x[t] - x[t] × z[t] - y[t],
    z'[t] == x[t] × y[t] - B z[t],
    x[0] == xo, y[0] == yo, z[0] == zo
   };
s2 = NDSolve[SetPrecision[eqns, Infinity],
    {x[t], y[t], z[t]}, {t, 0, 50}, PrecisionGoal → ControlActive[0, 10],
    WorkingPrecision → ControlActive[MachinePrecision, 100]];
g3 = ParametricPlot3D[Evaluate[{x[t], y[t], z[t]} /. s2], {t, 0, 50}]
Framed@
 Labeled[g3, Style["3.Lorenz's attractor with NDSolve, ParametricPlot3D, and
     Precision options", Black, 16], Alignment → {Bottom, Center}]
```

Out[123]=



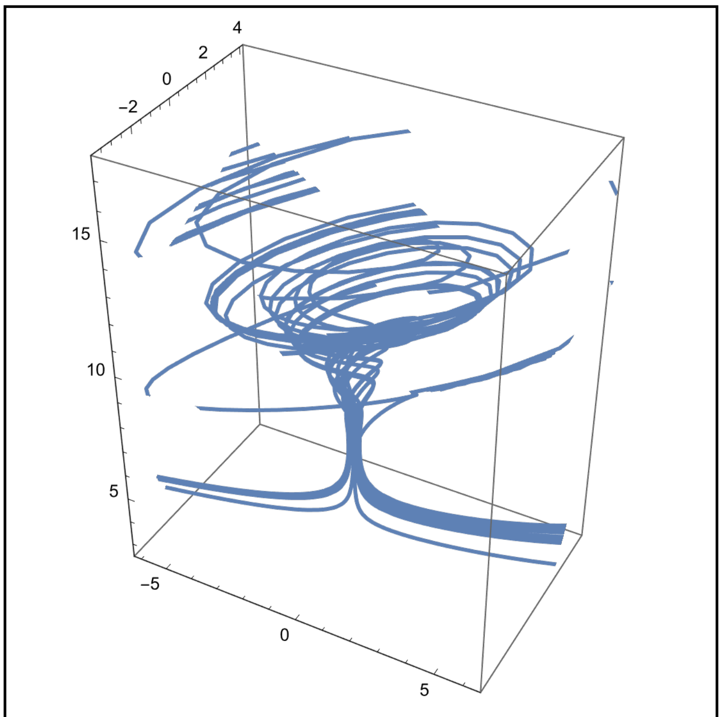
3.Lorenz's attractor with NDSolve,
ParametricPlot3D, and Precision options

3.4 Experimenting on different ODEs

3.4.1 Chen’s attractor (Zhou et al, 2004)

In[129]:=
```
α = 5; β = -10; δ = -0.38; x1 = 5; y1 = 10; z1 = 10;
eqns1 = {x'[t] == α x[t] - y[t] × z[t],
    y'[t] == β y[t] + x[t] × z[t],
    z'[t] == δ z[t] + (x[t] × y[t])/3,
    x[1] == x1, y[1] == y1, z[1] == z1
  };
s3 = NDSolveValue[
    eqns1,
    {x, y, z},
    {t, 1, 100}
  ];
g4 = ParametricPlot3D[{s3[[1]][t], s3[[2]][t], s3[[3]][t]}, {t, 1, 100}]
Framed@Labeled[g4,
  Style["4.Chen's Attractor", Black, 16], Alignment → {Bottom, Center}]
```

Out[133]=



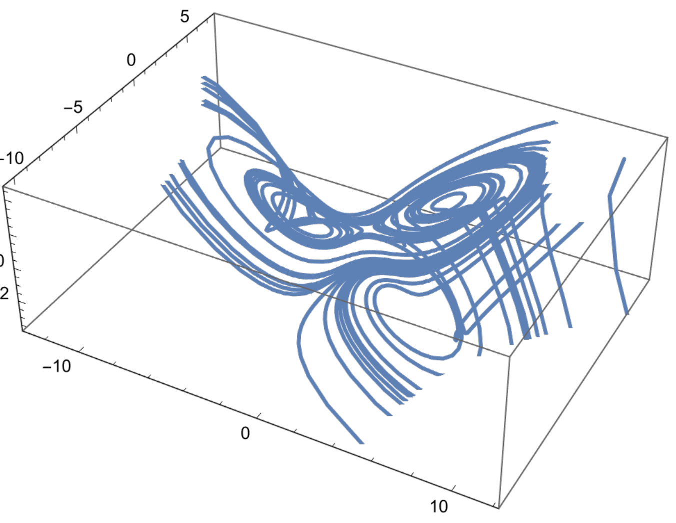
4.Chen's Attractor

3.4.2 Dadras' attractor (Saberi Nik et al, 2015)

```
In[134]:= a = 3; b = 2.7; c = 1.7; d = 2; e = 9; x1 = 1.1; y1 = 2.1; z1 = -2;
eqns2 = {x'[t] == y[t] - a x[t] + b y[t] × z[t],
    y'[t] == c y[t] - x[t] × z[t] + z[t],
    z'[t] == d x[t] × y[t] - e z[t],
    x[1] == x1, y[1] == y1, z[1] == z1
  };
s4 = NDSolveValue[
    eqns2,
    {x, y, z},
    {t, 1, 100}
  ];

g5 = ParametricPlot3D[{s4[[1]][t], s4[[2]][t], s4[[3]][t]}, {t, 1, 100}]
Framed@Labeled[g5,
  Style["5.Dadras' Attractor", Black, 16], Alignment → {Bottom, Center}]
```

Out[138]=



5.Dadras' Attractor

4. Discussions

In this paper, we investigated ways to model strange attractors exhibited by Ordinary Differential Equations (ODEs). Even though it might take decades for scientists to model such attractors, with today's technological advancements and software, it would take as little as 30 minutes to model it. Through the experiments done in this paper, it shows concrete evidence of how simple and effortless it is to model chaotic attractors using softwares such as Mathematica. With its countless functions and options, it provides individuals the ability to do various modeling and solving of equations very easily as shown in the results section of this paper (Sections 3.1.2, 3.2.2, 3.3.2, 3.4.1, and 3.4.1).

5. Conclusions

This research led to the following conclusions:

1. Decades of work in modelling is simplified as hours, maybe less, of work through the use of advanced mathematical software.
2. The built in options that could vary the model’s precision affects the model only to some extent.
3. Mathematica can not only model Lorenz’ attractor, but also other attractors such as Chen’s attractor and Dadras’ attractor.
4. Mathematica could be a software used to create new strange attractors.

6. Future Research

In this paper, we did not have time to further explore the concept of modelling chaos theory by creating our own differential equations, initial conditions, and parameters. Essentially such research would take a significant amount of time as a lot of trials must be done to come up with the specific variables that works.

7. Acknowledgements

Writing this paper was not just typing words in my keyboard; it was an adventure. I would like to acknowledge the following individuals to have helped me throughout my research journey:

1. My professor, Zbigniew J. Kabala
2. My PCA, Omar Dacres
3. My parents, Anny Gorat and Perry Handjaja
4. My math teacher, Geoffrey Popadiuk
5. Pioneer Writing Center

References

1. Duplantier, B., & Rivasseau, V. (2014). *Henri poincare, 1912–2012: Poincaré seminar 2012*. Springer.
2. Eckmann, J., & Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *The Theory of Chaotic Attractors*, 273-312. https://doi.org/10.1007/978-0-387-21830-4_17
3. Gleick, J. (2008). *Chaos: Making a new science*. Penguin.
4. Krishnamurthy, V. (2015). Edward Norton Lorenz. *Resonance*, 20(3), 191-197. https://doi.org/10.1007/s12045-015-0169-4
5. Lorenz, E. N. (1963). Deterministic Nonperiodic flow. *Journal of the Atmospheric Sciences, 20*(2), 130-141. https://doi.org/10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2
6. McWilliams, J. C. (2019). A perspective on the legacy of Edward Lorenz. *Earth and Space Science, 6*(3), 336-350. https://doi.org/10.1029/2018ea000434
7. Mikhailov, M. D. (2007). Modelling unsaturated flow by using Mathematica. *Communications in Numerical Methods in Engineering, 24*(11), 947-959. https://doi.org/10.1002/cnm.1005
8. Munmuangsaen, B., & Srisuchinwong, B. (2018). A hidden chaotic attractor in the classical Lorenz system. *Chaos, Solitons & Fractals, 107*, 61-66. https://doi.org/10.1016/j.chaos.2017.12.017
9. Ott, E., Grebogi, C., & Yorke, J. A. (1996). Controlling chaos. *Controlling Chaos*, 77-80. https://doi.org/10.1016/b978-012396840-1/50032-1
10. Palmer, T. N., Döring, A., & Seregin, G. (2014). The real butterfly effect. *Nonlinearity, 27*(9), R123-R141. https://doi.org/10.1088/0951-7715/27/9/r123
11. Ruelle, D. (1995). Differentiable dynamical systems and the problem of turbulence. *Turbulence, Strange Attractors and Chaos*, 233-246. https://doi.org/10.1142/9789812833709_0014
12. Saberi Nik, H., Van Gorder, R. A., & Gambino, G. (2015). The chaotic dadras–momeni system: Control and hyperchaotification. *IMA Journal of Mathematical Control and Information, 33*(2), 497-518. https://doi.org/10.1093/imamci/dnu050
13. Stavrinides, S. G., & Ozer, M. (2020). *Chaos and complex systems: Proceedings of the 5th international interdisciplinary chaos symposium.* Springer Nature.
14. Tsonis, A. A., & Elsner, J. B. (1989). Chaos, strange attractors, and weather. *Bulletin of the American Meteorological Society, 70*(1), 14-23. https://doi.org/10.1175/1520-0477(1989)070<0014:csaaw>2.0.co;2
15. Zarei, A. (2015). Complex dynamics in a 5-D hyper-chaotic attractor with four-wing, one equilibrium and multiple chaotic attractors. *Nonlinear Dynamics, 81*(1-2), 585-605. https://doi.org/10.1007/s11071-015-2013-5
16. Zhou, T., Tang, Y., & Chen, G. (2004). Chen's attractor exists. *International Journal of Bifurcation and Chaos, 14*(09), 3167-3177. https://doi.org/10.1142/s0218127404011296
17. Wolfram Mathematica
18. Clarivate Web of Science