# The Lorenz Model

## Final Project Presentation

**Phil Mayer**

Fairfield University

PS215 Computational Physics
May 5, 2016

# Introduction

- For my final project, I chose to study the **Lorenz model** (also known as the Lorenz equations), a system of nonlinear differential equations.
- Originally studied by Edward Norton Lorenz while trying to numerically solve the Navier-Stokes equations.
- Oversimplified the Navier-Stokes equations considerably, but discovered a system with rich dynamics and chaotic behavior.

Figure: E.N. Lorenz, a pioneer in the study of chaos and a meteorologist. Image: Wikimedia Commons.

# Introduction (continued)

- The Lorenz model consists of three equations, given on the right.
- Though often given physical meaning, the model itself is mathematical. We might picture the (deterministic) motion of a particle in three dimensions.
- Uses three parameters: $\sigma$, $r$, and $b$, each positive.
- By convention, we typically choose $\sigma = 10$ and $b = \frac{8}{3} \approx 2.6667$.

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = -xz + rx - y$$
$$\frac{dz}{dt} = xy - bz$$

# Numerical Methods

- Since we have three first-order ODEs, the obvious solution is to employ the Euler method.
- As we will see shortly, many of the solution graphs exhibit oscillatory behavior. While the Euler-Cromer method tends to be more stable for oscillatory problems, the parameters minimize the error (given high precision).
- Recall that the Euler method is easy to derive: Given $n$ points indexed as $x_i$ (over $1 \leq i \leq n-1$), we can estimate $\frac{dx}{dt}$ using backward differencing:

$$\frac{dx}{dt} \approx \frac{x_i - x_{i-1}}{\Delta t}$$

With only a few steps of simple algebraic manipulation, we arrive at the Euler method.

$$x_i \approx x_{i-1} + \frac{dx}{dt} \Delta t$$

# Numerical Methods (continued)

- So applying the Euler method, our solutions are given by:

$$x_i = x_{i-1} + \frac{dx}{dt}\Delta t$$
$$y_i = y_{i-1} + \frac{dy}{dt}\Delta t$$
$$z_i = z_{i-1} + \frac{dz}{dt}\Delta t$$

- Finally, plugging in the definitions of $\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}$, we arrive at:

$$x_i = x_{i-1} + \sigma(y_{i-1} - x_{i-1})\Delta t$$
$$y_i = y_{i-1} + (-x_{i-1}z_{i-1} + rx_{i-1} - y_{i-1})\Delta t$$
$$z_i = z_{i-1} + (x_{i-1}y_{i-1} - bz_{i-1})\Delta t$$

# Results

- Overall, I was able to successfully solve for $x$, $y$, and $z$, under certain situations. Mainly attributed to the stability of the numerical method.
- The program takes in user input ($N$, $\Delta t$, $\sigma$, $r$, $b$, and the initial $x_0$, $y_0$, $z_0$), allocates memory for the $t$, $x$, $y$, and $z$ arrays, sets the initial conditions, solves the system by Euler's method, then graphs.

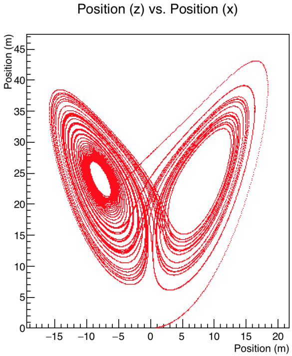


Figure: The Lorenz attractor, seen here as the $x$-$z$ projection of the solution.

# Results (continued)

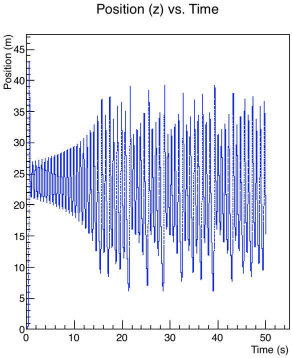


Figure: $z$ vs. $t$ after solving by the Euler method.

- Overall, was able to produce many of the graphics from the textbook, my graphs seemed visually accurate.
- Found that the Lorenz model shows sensitivity to initial conditions.

# Results (demo)

- The six graphs produced in my first program illustrate the overall behavior of the system.
- The second program demonstrates the system's sensitivity to initial conditions.

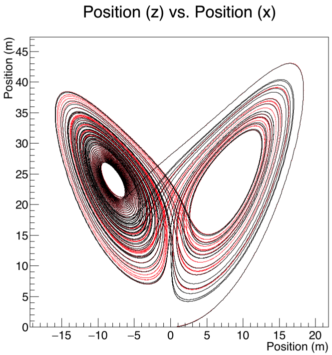


Figure: A sample output from my second program.

# Stability of the Numerical Method

- Ran the program with $\Delta t = 2.0$, 1.0, 0.1, 0.01, 0.001, and 0.0001, but it failed for all $\Delta t \geq 0.01$, crashing while graphing due to *NaN* values.
- At the high precisions required by the numerical method, more points are required. These can quickly exceed the limits of the *int* type in C++, let alone the maximum array size.

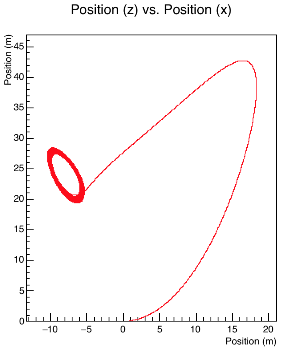


Figure: Not enough points to exhibit chaos: what about adding more points?

# Stability of the Numerical Method (continued)

- Altering the initial conditions and parameter values ends up having little impact on the correctness and accuracy of the Euler method.
- Increasing the parameters, particularly $b$, intensified the graphic. Acted almost like a measure of “speed.”
- Did not experiment with negative values of the parameters $\sigma$, $r$, and $b$ since they are positive by definition.

**Questions?**