InSight: RIVIER ACADEMIC JOURNAL, VOLUME 9, NUMBER 2, FALL 2013

THE LORENZ SYSTEM SIMULATION

Maxim Sukharev-Chuyan*
Junior, Souhegan High School, Amherst, NH

Abstract

The Lorenz system is discussed as a simple basic model of chaotic behavior. Java code for an animation of the waterwheel model of the Lorenz system is included. The simulation demonstrates chaotic behavior of the numerical solution of the Lorenz' system of nonlinear ordinary differential equations.

Edward Lorenz (1917–2008) was an MIT meteorologist and mathematician best known for his pioneering work in chaos theory. His first paper on unpredictable chaotic behavior, *Deterministic Nonperiodic Flow*, was published in 1963. In this paper, he introduced a system of three nonlinear ordinary differential equations modeling chaotic behavior, which became known as the Lorenz system:

$$\frac{dx}{dt} = \sigma\,(y - x)$$

$$\frac{dy}{dt} = x\,(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

The three parameters, σ, ρ, and β, determine the behavior of the system. Some sets of values of the parameters result in a repeating pattern, while other sets result in chaotic behavior. The values Lorenz used in *Deterministic Nonperiodic Flow*, and the values used throughout this project, are $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$.

The Lorenz system is a simplification of a system of twelve equations that Lorenz had developed to model atmospheric processes. The solutions of this system of twelve equations exhibited what Lorenz described as the 'butterfly effect', later termed 'sensitive dependence on initial conditions'. While the simpler system of three equations was not intended to model any natural phenomenon, it was later found to accurately describe a simplified model of atmospheric convection. In this model, a torus-shaped tube forming a closed loop is filled with a fluid and heated from below, causing the heated fluid to rotate to the top of the tube. The rate of heating and the rate of cooling can be adjusted so that the fluid will occasionally change direction, never settling into any predictable pattern. This motion is accurately modeled by the Lorenz equations. An equivalent model, simulated in the included Java program, replaces the circular tube with a waterwheel: cups are arranged around the wheel and are filled with water when they pass under a faucet located above the wheel (which corresponds to the heating of the fluid at the bottom of the tube). The water leaks from the cups as they move around the wheel (which corresponds to the cooling of the fluid). The heavier cups will rotate to the bottom of the wheel, just as the heated fluid rises to the top of the tube.

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

In the Java program simulating the waterwheel, the angular acceleration of the wheel is calculated according to the laws of physics based on the changing mass of each cup, and this acceleration modifies the angular velocity on each iteration of the code.

The acceleration of the simulated waterwheel over time is shown in Fig. 1 below.

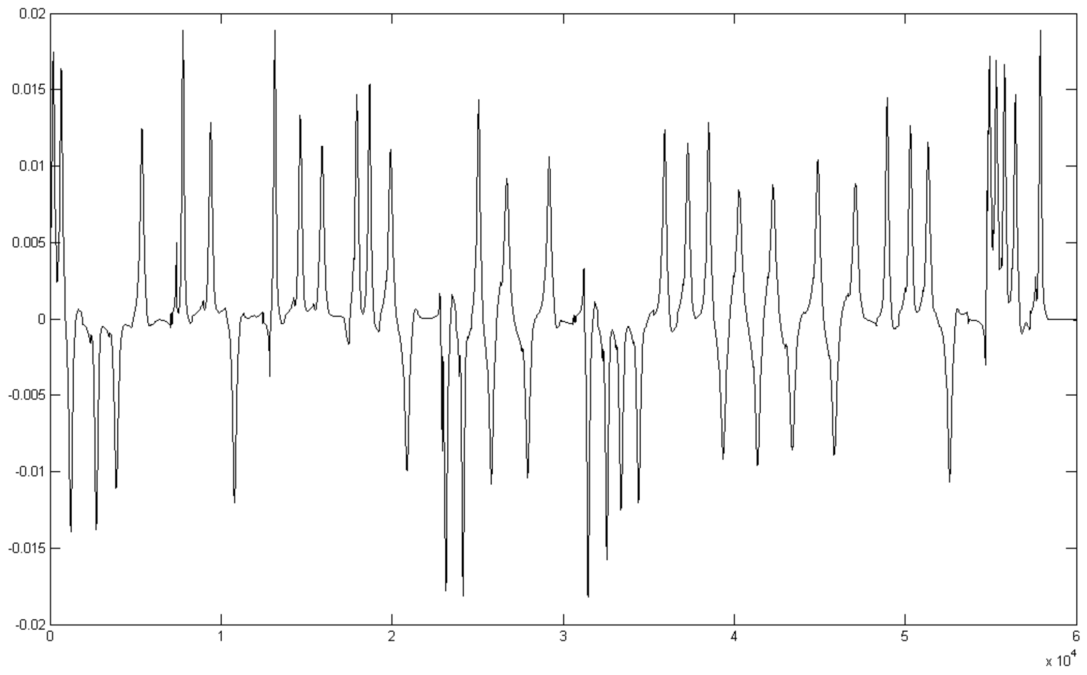

Figure 1. Angular velocity as a function of time.

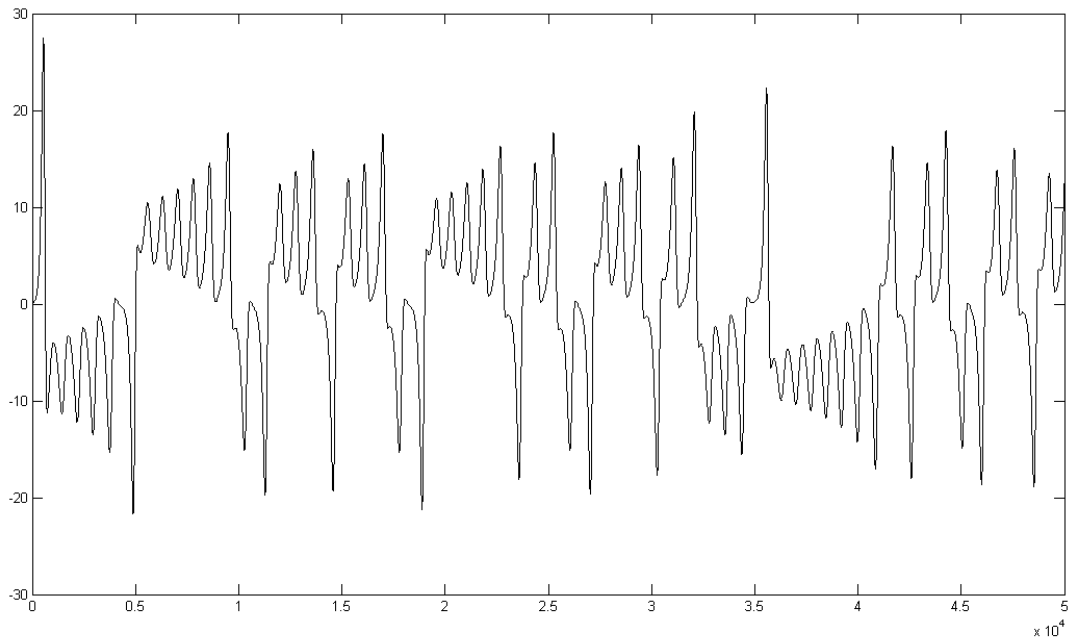

Figure 2. Variable *y* of the Lorenz system as a function of time.

The graph in Fig. 2 shows variable *y* of the Lorenz system as it changes over time. Figure 3 shows the same process as one in Fig. 2 with *y* on the horizontal axis and *z* on the vertical axis.

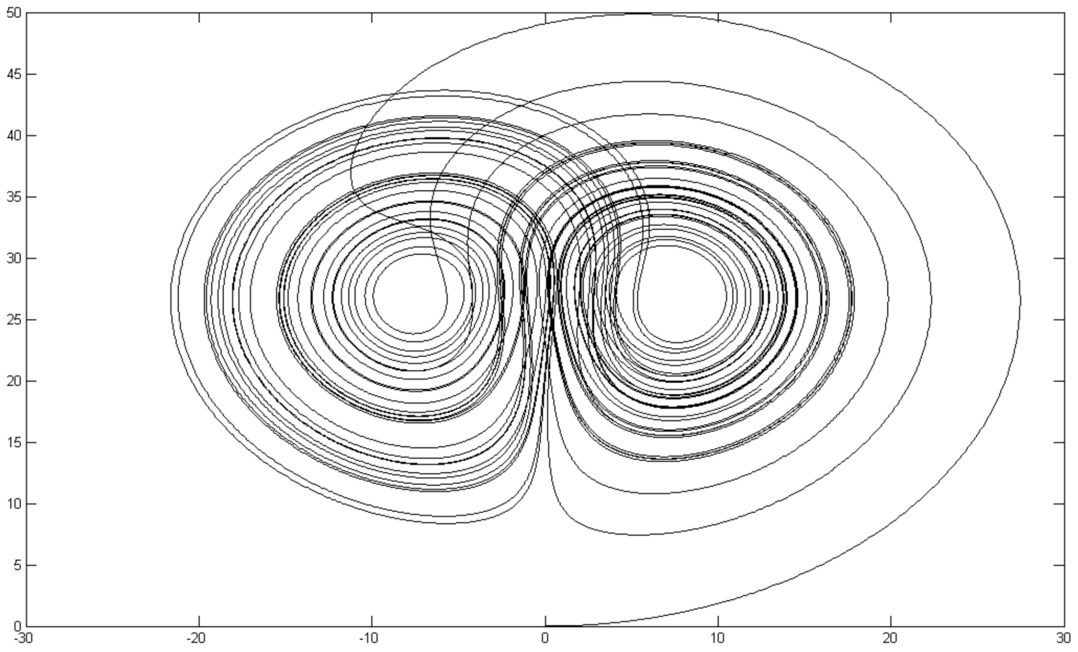

Figure 3. Relation between *y* and *z* coordinates in the Lorenz system.

In Fig. 1, positive values of angular velocity represent rotation of the wheel in the positive direction while negative values represent rotation in the negative direction. Consequently, changes between positive and negative values represent changes in the direction of the wheel's rotation. Positive values of *y* in Fig. 2 correspond to the right attractor in Fig. 3, while negative values of *y* in Fig. 2 correspond to the left attractor in Fig. 3. Changes between positive and negative values of *y* in Fig. 2 represent transfers between the two attractors. Thus, the two attractors in Fig. 3 correspond to the two directions of rotation of the waterwheel. The chaotic nature of the Lorenz system makes the transfers between the two attractors, and, consequently, the changes in direction of the waterwheel it models, unpredictable.

The Java code included below (see *Appendix*) creates an animation of the waterwheel model of the Lorenz system. The program also outputs the values of the position and velocity of the waterwheel as a text `(.txt)` file. This animation demonstrates chaotic behavior of the numerical solution of the Lorenz' system of nonlinear ordinary differential equations.

I would like to thank Dr. Vladimir Riabov, my computer science professor at Rivier University, for help and inspiration.

Bibliography

Bourke, P., The Lorenz Attractor, April 1997. Online: http://paulbourke.net/fractals/lorenz
Gleick, J., *Chaos: Making a New Science*, New York: Penguin Books, 2008.
Lorenz, E. N., *The Essence of Chaos*, Seattle: University of Washington Press, 1993.
Lorenz, E. N., Deterministic nonperiodic flow, *Journal of Atmospheric Sciences*, 20 (2), 130–141, 1963.

Appendix: Java code for waterwheel animation

```
import javax.swing.*;
import java.awt.*;
import java.io.*;

class LorenzSimulation{
   public static void main(String[] args){
      JFrame frame = new JFrame("The Lorenz System");
      frame.setSize(600,760);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Panel pane = new Panel();
      pane.setBackground(Color.black);
      frame.setContentPane(pane);
      frame.setVisible(true);
   }
}

class Panel extends JPanel{
   Waterwheel wheel;
   protected Panel(){
      wheel = new Waterwheel(new Point(300,400), 200, 12);
   }
   protected void paintComponent(Graphics g){
      super.paintComponent(g);
      g.setColor(Color.lightGray);
      g.fillRect(0,20,285,30);      //horizontal pipe
      g.fillRect(285,50,30,40);     //vertical pipe
      g.fillArc(255,20,60,60,0,90); //curve in pipe
      g.setColor(new Color(150,150,250));
      g.fillRect(295,90,10,110);    //stream of water
      wheel.draw(g);
      try{Thread.sleep(2);}
      catch(InterruptedException e){}
      wheel.calculate();
      repaint();
   }
}
```

```
class Waterwheel{
   final double FILL_SPEED = .4;  //.4 for chaotic motion; .9 for stable motion
   final double DRAIN_SPEED = .02;
   final double V_COEFFICIENT = 1/50.0;    //coefficient of rotational velocity
   Point center;                           //added to angle in each iteration
   int r; //radius
   double a; //angle of shift from original position
   double v; //velocity of rotation
   int numCups;
   Cup[] cups;
   PrintWriter outFile;
   public Waterwheel(Point c, int rIn, int n){
      center = c;                              //Waterwheel constructor
      r = rIn;
      a = 0.0;
      v = 0.01;
      numCups = n;
      cups = new Cup[numCups];                          //array of Cups
      for(int i=0; i<numCups; i++) cups[i] = new Cup(i*(2*Math.PI/numCups));
      try{outFile = new PrintWriter("Waterwheel_Output.txt");}
      catch(IOException ignore){}
      outFile.printf("%10s%10s%n","Angle","Velocity");
   }
   public void calculate(){ //recalculate position and conditions of Waterwheel
      double m = 0;
      for(int i=0; i<numCups; i++){
         if(Math.abs(Math.cos(a+cups[i].a)) <.1 && Math.sin(a+cups[i].a) > 0
           && cups[i].volume < 50) cups[i].volume += FILL_SPEED;
         if(cups[i].volume > 0) cups[i].volume -= DRAIN_SPEED;
         v -= (cups[i].volume) * Math.cos(a+cups[i].a) * V_COEFFICIENT;
         m += cups[i].volume;
      }
      v /= m;
      a += v;
      outFile.printf("%10.5f%10.5f;%n",a,v);
   }
   public void draw(Graphics g){                         //draw Waterwheel
      g.setColor(Color.gray);
      g.drawOval(center.x - r, center.y - r, 2*r, 2*r); //draw large circle
      g.setColor(new Color(50,50,200));//new Color(0,0,100));
      for(int i=0; i<numCups; i++) cups[i].draw(center, a, g); //draw cups
   }

   private class Cup{                               //internal Cup class
      private double volume; //radius of cup circle
      private double a;      //position on wheel
      private Cup(double aIn){                          //Cup constructor
         a = aIn;
         volume = 0;
      }
      //draw individual cup:
      private void draw(Point center, double aWheel, Graphics g){
         g.fillOval((int)(center.x + r*Math.cos(a+aWheel) - volume - 3),
                    (int)(center.y - r*Math.sin(a+aWheel) - volume - 3),
                    2*(int)volume + 6, 2*(int)volume + 6);
        }
    }
}
```

Figures 2 and 3 show solutions of the Lorenz system generated by the following Java code (based on C code from http://paulbourke.net/fractals/lorenz).

```
import java.io.*;
class LorenzEquations{
    public static void main(String[] args) throws IOException{
        double h = .001;
        double a = 10;
        double b = 28;
        double c = 8/3;
        double x0 = 0.1;
        double y0 = 0;
        double z0 = 0;
        double x1,y1,z1;
        PrintWriter outFile = new PrintWriter("Equations_Output.txt");
        outFile.printf("%10c%10c%10c%n",'x','y','z');
        for(int i=0; i<50000; i++){
            x1 = x0 + h*a*(y0-x0);
            y1 = y0 + h*(x0*(b-z0) - y0);
            z1 = z0 + h*(x0*y0 -c*z0);
            x0 = x1;
            y0 = y1;
            z0 = z1;
            outFile.printf("%10.5f%10.5f%10.5f;%n",x0,y0,z0);
        }
        outFile.close();
    }
}
```

NOTE: Figures 1, 2, and 3 were generated from output of the included programs using MATLAB.

[*] **MAXIM SUKHAREV-CHUYAN** is a junior at Souhegan High School in Amherst, NH. He has taken a few courses in mathematics and computer science at Rivier University. His other interests include rock climbing and alpine skiing.