The Lorenz Model

Luke Timmons

December 2, 2016

Contents

1 Description of Physical Setting

The Lorenz system is a system of ordinary differential equation originally studied by the atmospheric scientist, Edward Lorenz. At the time, Lorenz was studying the basic equations of fluid mechanics, or Navier-Stokes equations, which are a complicated set of differential equations that describe the various parameters, such as temperature, density, velocity, of the fluid as functions of position and time. Due to their complicated nature, they are difficult to solve analytically in cases of practical interest and as a result Lorenz sought to solve the Navier-Stokes equations using a computational approach. He investigated the Rayleigh-Bénard problem, in which the top and bottom of a fluid in a container are held at different temperatures. Due to the fact that Lorenz did his work more than 40 years ago, he had to consider a greatly simplified version of the Navier-Stokes equations when applied to the Rayleigh-Bénard problem. He was able to reduce the problem to only three equations, now known as the Lorenz model, as follows:

$$\frac{dx}{dt} = \sigma(y - x) \tag{1}$$

$$\frac{dy}{dt} = rx - xz - y \tag{2}$$

$$\frac{dz}{dt} = xy - bz \tag{3}$$

where r is the ratio of the Rayleigh number to the critical Rayleigh number, σ is the Prandtl number and b is related to the horizontal wave number of the system.

2 Description of Numerical Solution

Through the use of the Euler method, the following solution to the Lorenz model can be derived:

$$x(t + \Delta t) = x(t) + \sigma\Delta t(y - x) \tag{4}$$

$$y(t + \Delta t) = y(t) + \Delta t(rx - xz - y) \tag{5}$$

$$z(t + \Delta t) = z(t) + \Delta t(xy - bz) \tag{6}$$

However, the Euler method may not be appropriate for the Lorenz model as the Euler method is only an approximation and therefore the error terms associated with the method are not zero. As a result, in oscillatory problems, such as the Lorenz model, the errors introduced by the Euler method generally tend to accumulate such that the results generated are subject to instability. As a result, the Euler method is in general not suitable for oscillatory problems. Instead the second-order Runge-Kutta method can be used such that the results generated are more stable.

Using the Runge-Kutta method the following solution was derived for the Lorenz model:

$$x_{i+1} = x_i + \sigma\Delta t(y' - x') \tag{7}$$

$$y_{i+1} = y_i + \Delta t(rx' - x'z' - y') \tag{8}$$

$$z_{i+1} = z_i + \Delta t(x'y' - bz') \tag{9}$$

where:

$$x' = x_i + \frac{\sigma\Delta t}{2}(y_i - x_i)$$

$$y' = y_i + \frac{\Delta t}{2}(rx_i - x_i z_i - y_i)$$

$$z' = z_i + \frac{\Delta t}{2}(x_i y_i - bz_i)$$

The next problem is to plot the variation of the Lorenz variable z as a function of time for three different values of r, 25, 10, 5; with a time step of 0.0001 seconds and initial condition $x = 1$, $y = z = 0$.

In order to plot the variation of the variable z over time, a for loop can be used to insert data into a set of arrays for the variables x, y, z and the time t for each value of r. Using gnuplot, the data from the z array could then be plotted against the data from the t array. However, the standard gnuplot C++ file does not have a function for plotting three functions on the same space so the standard file will have to be edited to contain a function that plots three functions on the same graph.

Following on from that, the next step is to plot the trajectory of the Lorenz model onto the x-z plane for the value of r of 25 with a time step of 0.0001 seconds and initial condition $x = 1$, $y = z = 0$.

To plot the trajectory of the Lorenz model on to the x-z plane, the existing data from the previous step can be inputted into gnuplot to plot the values for the z and x variable onto the x-z plane.

The final step for the minimum goals is to plot Phase-space plots for the Lorenz model with $r = 25$ for a time step of 0.0001. Two phase plots must be plotted, the first for z against y with points plotted only when $x = 0$ and the second for z against x only when $y = 0$.

In order to plot the phase plot of z against y only when $x = 0$, the program must first determine the point at which $x = 0$. An if statement, nested within a for

loop, that will check if the value of x stored in an array is equal to zero could be used. If it equal to zero, the corresponding values for y and z will be stored into separate arrays (e.g phase_y, phase_z). If not, then no values will be entered into the arrays. The for loop will then run through the same section of code until all the values for x have been checked. The data from the arrays will then be inputed into gnuplot so that the phase plot can be obtained.

Similarly for the phase plot of z against x only when $y = 0$, the points at which $y = 0$ must first be determined using an if statement nested within a for loop. If the value of y is zero then the corresponding values for x and z will be inputted into separate arrays (e.g. phase_x, phase_z). The for loop will then check each value of y until all have been checked. The data from the loop can then be inputted into gnuplot so that the phase plot of z against x can be obtained.

The first of the stretch goals is concerned with showing that the results generated from the final step of the minimum goals is not dependent on the initial conditions of the system. This goal could be achieved by simple comparison between different sets of initial conditions for the variables x, y and z.

The second and final stretch goal is the determination of the Lyapunov exponent for a number of trajectories of the Lorenz model near its transition to chaos at $r = 24.74$. As well as this, observations of the exponents change from negative in the non-chaotic regime to positive in the chaotic regime must be made. The Lyapunov exponent is a quantity that characterizes the rate of separation of infinitesimally close trajectories. To achieve this goal the program will generate three trajectories for the Lorenz system, each infinitesimally separated from each other and calculate the separation between the trajectories for each value of x, y and z.

3 Pseudocode

3.1 Minimum Goals

Method for generation of x,y and z values
Input initial conditions for system, x_0, y_0 and z_0. Input values for constants r_1, r_2, r_3, b and σ.

Input values for time step (dt) and the total time (T) and calculate size for arrays by the equation:

$$Size = \frac{T}{dt}$$

Create arrays for x-values, y-values, z-values and time-values with first data points set to x_0, y_0, z_0 and 0.0 respectively.

Open for loop that will loop through each data point in array.

Input data into array for x-values where each data point can be defined by the equation:

$$x_{i+1} = x_i + \sigma\Delta t(y' - x')$$

And:

$$x' = x_i + \frac{\sigma\Delta t}{2}(y_i - x_i)$$

$$y' = y_i + \frac{\Delta t}{2}(r_1 x_i - x_i z_i - y_i)$$

Input data into array for y-values where each data point can be defined by the equation:

$$y_{i+1} = y_i + \Delta t(rx' - x'z' - y')$$

And:

$$z' = z_i + \frac{\Delta t}{2}(x_i y_i - bz_i)$$

Input data into array for z-values where each data point can be defined by the equation:

$$z_{i+1} = z_i + \Delta t(x'y' - bz')$$

Input data into array for time-values where each data point can be defined by the equation:

$$t_{i+1} = i(dt)$$

Close for loop.

Method for plotting z variable as a function of time
Create a new gnuplot function that plots three functions on the sam graph from existing function that plots two functions on the same graph.

Input data for z-values and time values from their respective arrays into the newly created gnuplot function i.e. gnuplot_three_functions.

Method for plotting trajectory of x & z variable on x-z plane
Input data for x-values and z-values from their respective arrays into the gnuplot function for plotting one function i.e. gnuplot_one_function.

Method for plotting phase plot of z against x only when y = 0
Create arrays for x-values, y-values and z-values to be inputted into phase plot.

Open for loop that will loop through each data point in array.

Create if statement that inputs values into arrays for x and z phase values if the y-values are between a certain range such that it is approximately zero i.e. if -0.0001 < y < 0.0001.

Close for loop.

Input data from arrays for x and z values for phase plot into the gnuplot function for plotting one function i.e. gnuplot_one_function.

Open for loop that will loop through each data point in array

Set value of datapoint in x, y and z phase arrays to zero.

Close for loop

Method for plotting phase plot of z against y only when x = 0
Open for loop that will loop through each data point in array.

Create if statement that inputs values into arrays for y and z phase values if the x-values are between a certain range such that it is approximately zero i.e. if -0.0001 < x < 0.0001.

Close for loop.

Input data from arrays for x and z values for phase plot into the gnuplot function for plotting one function i.e. gnuplot_one_function.

Open for loop that will loop through each data point in array

Set value of datapoint in x, y and z phase arrays to zero.

Close for loop

3.2 Stretch Goals

Method for plotting phase plot of z against x only when y = 0
Set initial conditions such that $x_0 = 0$, $y_0 = z_0 = 1$

Open for loop that will loop through each data point in array.

Create if statement that inputs values into arrays for x and z phase values if the y-values are between a certain range such that it is approximately zero i.e. if -0.0001 < y < 0.0001.

Close for loop.

Input data from arrays for x and z values for phase plot into the gnuplot function for plotting one function i.e. gnuplot_one_function.

Open for loop that will loop through each data point in array

Set value of datapoint in x, y and z phase arrays to zero.

Close for loop

Method for plotting phase plot of z against y only when x = 0
Open for loop that will loop through each data point in array.

Create if statement that inputs values into arrays for y and z phase values if the x-values are between a certain range such that it is approximately zero i.e. if -0.0001 < x < 0.0001.

Close for loop.

Input data from arrays for x and z values for phase plot into the gnuplot function for plotting one function i.e. gnuplot_one_function.

Open for loop that will loop through each data point in array

Set value of data point in x, y and z phase arrays to zero.

Close for loop.

4 Code implemented

Code for generating x, y, z and t values

```
//for loop to calculate x, y ,z and t values for each plot point when r = r1
for (nr = 0; nr < number_steps-1; nr++)
{
x_prime = value_x1[nr] + ((sigma*dt)/2)*(value_y1[nr] - value_x1[nr]);
y_prime = value_y1[nr] + (dt/2)*(r1*value_x1[nr] - value_x1[nr]*value_z1[nr]
          - value_y1[nr]);
z_prime = value_z1[nr] + (dt/2)*(value_x1[nr]*value_y1[nr] - b*value_z1[nr]);

value_t[nr + 1] = dt*(nr + 1);

value_x1[nr + 1] = value_x1[nr] + dt*(sigma*(y_prime - x_prime));
value_y1[nr + 1] = value_y1[nr] + dt*(r1*x_prime - y_prime - x_prime*z_prime);
value_z1[nr + 1] = value_z1[nr] + dt*(x_prime*y_prime - b*z_prime);
}
```

This is the implementation of code for the generation of the x, y, z and t values for the Lorenz model using the second order Runge Kutta method, for $r = r_1$.
The initial data points for the arrays value_x1, value_y1, value_z1 and value_t are set outside of the for loop with the values, x_0, y_0, z_0 and 0.0 respectively.
The same method is used to generate the values for the x, y and z variables of the Lorenz model, when $r = r_2$ or $r = r_3$ with the arrays changing to value_x2, value_y2, value_z2 and value_x3, value_y3, value_z3 respectively.

Code for plotting z as a function of time for each value of r

```
 //call newly created gnuplot function for plotting three functions
 on the same space

gnuplot_three_functions ("Z over time", "lines", "t", "z", value_t, value_z1,
number_steps, "r=25", value_t, value_z2, number_steps, "r=10",value_t,
value_z3, number_steps, "r=5");
```

This section of code passes the data from the arrays for the variable z and the time in to the gnuplot function "gnuplot_three_functions", labelling each function with its respective r value. "gnuplot_three_functions" was adapted from the existing functions for plotting two functions on the same space. As such the code for gnuplot_three_functions is as follows:

```
void gnuplot_three_functions (
  string   title,
  string   style,
  string   label_x,
  string   label_y,
  double  *   x,
```

```
  double  *   y,
  int         n,
  string   dataname,
  double  *   x2,
  double  *   y2,
  int         n2,
  string   dataname2,
  double  *   x3,
  double  *   y3,
  int         n3,
  string   dataname3
)
{
  gnuplot_ctrl    *   handle ;

  if (x==NULL || y==NULL || x2==NULL || y2==NULL || x3==NULL || y3==NULL
      || n<1 || n2<1 || n3<1)
return ;

  if ((handle = gnuplot_init()) == NULL)
return ;

  if (style != "")
gnuplot_setstyle(handle, style.c_str());
  else
gnuplot_setstyle(handle, (char *)"lines");

gnuplot_set_xlabel(handle, label_x.c_str());

    gnuplot_set_ylabel(handle, label_y.c_str());

  gnuplot_cmd(handle, "set title \"%s\"", title.c_str()) ;

  gnuplot_plot_xy_three (handle, x, y, n, dataname.c_str(), x2, y2, n2,
   dataname2.c_str(), x3, y3, n3, dataname3.c_str());

  printf("press 2 x ENTER to continue\n");
  while (getchar()!='\n') {}
  while (getchar()!='\n') {}

  gnuplot_close(handle);
  return ;
}
```

Code for plotting the trajectory of x and z on the x-z plane

```
//Print one function on graph with set range for x and y axes using
```

```
newly created gnuplot function

gnuplot_one_function_range ("r=25", "lines", "X", "Z", value_x1, value_z1,
                            number_steps, xrange, yrange);
```

This section of code passes the data from the arrays for the variable x and z in to the gnuplot function "gnuplot_one_function_range", with the x and y axes of the graph set by two arrays, one for the x-axis and one for the y-axis, each consisting of only two data points. The newly created function "gnuplot_one_function_range" is as follows:

```
void gnuplot_one_function_range (
  string title,
  string style,
  string label_x,
  string label_y,
  double  *   x,
  double  *   y,
  int         n,
  double xrange[],
  double yrange[]
)
{
  gnuplot_ctrl    *   handle ;

  if (x==NULL || n<1) return ;

  if ((handle = gnuplot_init()) == NULL) return ;
  if (style!="") {
      gnuplot_setstyle(handle, style.c_str());
  } else {
      gnuplot_setstyle(handle, (char *)"lines");
  }
      gnuplot_set_xlabel(handle, label_x.c_str());
      gnuplot_set_ylabel(handle, label_y.c_str());
      gnuplot_set_xrange(handle, xrange[0], xrange[1]);
      gnuplot_set_yrange(handle, yrange[0], yrange[1]);
  if (y==NULL) {
      gnuplot_plot_x(handle, x, n, title.c_str());
  } else {
      gnuplot_plot_xy(handle, x, y, n, title.c_str());
  }

  printf("press 2 x ENTER to continue\n");
  while (getchar()!='\n') {}
  while (getchar()!='\n') {}
  gnuplot_close(handle);
```

```
  return ;
}
```

This section of code passes the data from the arrays for the variable x and z in to the gnuplot function "gnuplot_one_function_range", with the x and y axes of the graph set by two arrays, one for the x-axis and one for the y-axis, each consisting of only two data points.

Code for plotting the phase-space plot of z against y only when $x = 0$

```
//for loop that inputs values into phase arrays, if the value of x is between
-0.0001 and 0.0001

int j = 0;

for (nr = 0; nr < number_steps-1; nr++)
{
if(value_x1[nr] < 0.0001 && value_x1[nr] > -0.0001){
phase_y[j] = value_y1[nr+300000];
phase_z[j] = value_z1[nr+300000];
j = j + 1;
}

}

//set values for range of x axis and range of y axis on graph
xrange[0]=-10;
yrange[0]=0;
xrange[1]=10;
yrange[1]=30;

//Print one function on graph with set range for x and y axes using newly
 created gnuplot function

gnuplot_one_function_range ("r=25", "points", "Y", "Z", phase_y, phase_z,
                            phase_steps, xrange, yrange);
```

This section of code inputs the $(nr + 300000)$-th values for y and z into j-th of their respective arrays for the phase plots if the the value of x is between -0.0001 and 0.0001. The ranges for the x and y axes for the plot are then inputted into their arrays and the values for the variables y, z and the ranges for the x and y axes are passed into into the gnuplot function "gnuplot_one_function_range".

Code for plotting the phase-space plot of z against x only when $y = 0$

```
//for loop that inputs values into phase arrays, if the value of y is between
-0.0001 and 0.0001
```

```
j = 0;

for (nr = 300000; nr < number_steps-1; nr++)
{

if(value_y1[nr+300000] < 0.0001 && value_y1[nr+300000] > -0.0001){
phase_x[j] = value_x1[nr+300000];
phase_z[j] = value_z1[nr+300000];
j = j + 1;
  }
}

//set values for range of x axis and range of y axis on graph
xrange[0]=-20;
yrange[0]=0;
xrange[1]=20;
yrange[1]=40;

//Print one function on graph with set range for x and y axes using
newly created gnuplot function

gnuplot_one_function_range ("r=25", "points", "X", "Z", phase_x, phase_z,
                            phase_steps, xrange, yrange);
```

This section of code inputs the $(nr + 300000)$-th values for x and z into the j-th their respective arrays for the phase plots if the the value of y is between -0.0001 and 0.0001. The ranges for the x and y axes for the plot are then inputted into their arrays and the values for the variables x, z and the ranges for the x and y axes are passed into into the gnuplot function "gnuplot_one_function_range".

5 Discussion of results

The Lorenz system is a system of first order ordinary differential equations which provide an approximation of the basic equations of fluid mechanics, known as the Navier-Stokes equations, in relation to the Rayleigh-Bénard problem, which describes a scenario in which the top and bottom of a fluid in a container are held at different temperatures.

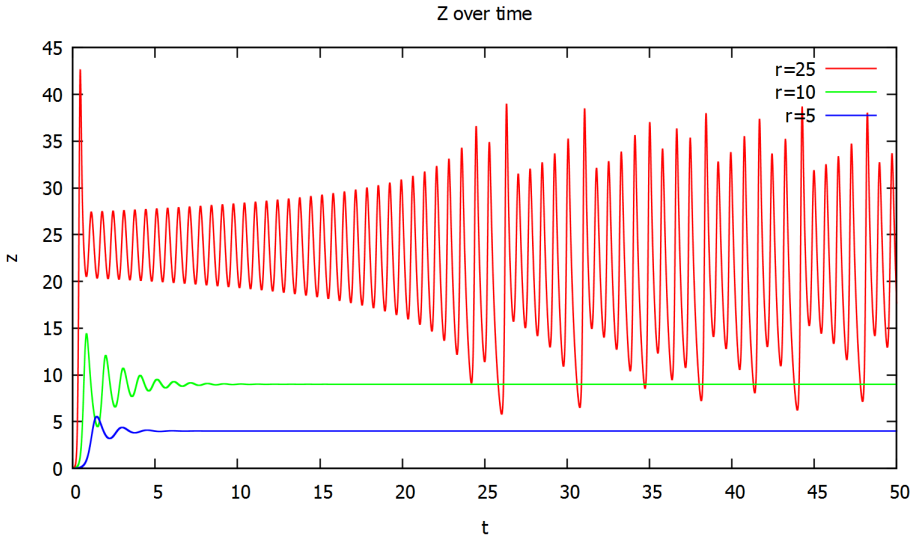

Figure 1: Variation of the Lorenz variable z as a function of time. The calculation was performed using the second order Runge-Kutta method with a time step of 0.0001. The initial conditions were $x = 1$, $y = z = 0$.

From *figure 1*, it can be seen that the system has both a chaotic and non-chaotic regime. In the non-chaotic regime, the variable z decays to zero exponentially over time, as can be seen from the plots when $r = 10$ and $r = 5$. While, in the chaotic regime, the amplitude of the oscillations of the z variable increases exponentially for as certain amount of time, showing that the initial transient of the system is roughly periodic. However after $t = 20$, the initial transient decays to give way to an irregular chaotic time dependence. While the chaotic behaviour seen when $r = 25$ from the plot of z as a function of time, can be seen to be random and unpredictable in nature, this is not quite so as can be seen from *figure 2*.

Figure 2 is a plot of the trajectory of the Lorenz model projected onto the x-z plane, where $r = 25$. This plot shows the Lorenz attractor, which is a set of chaotic solutions of the Lorenz model which, when plotted, resemble a butterfly or figure eight. This phase plot also shows that for the Lorenz model in the chaotic regime there is still some element of regularity. However, this phase plot does little more than hint at these regularities. To get further indications to the regularities that are faintly apparent in *figure 2*, Poincaré sections of the trajectory of the Lorenz

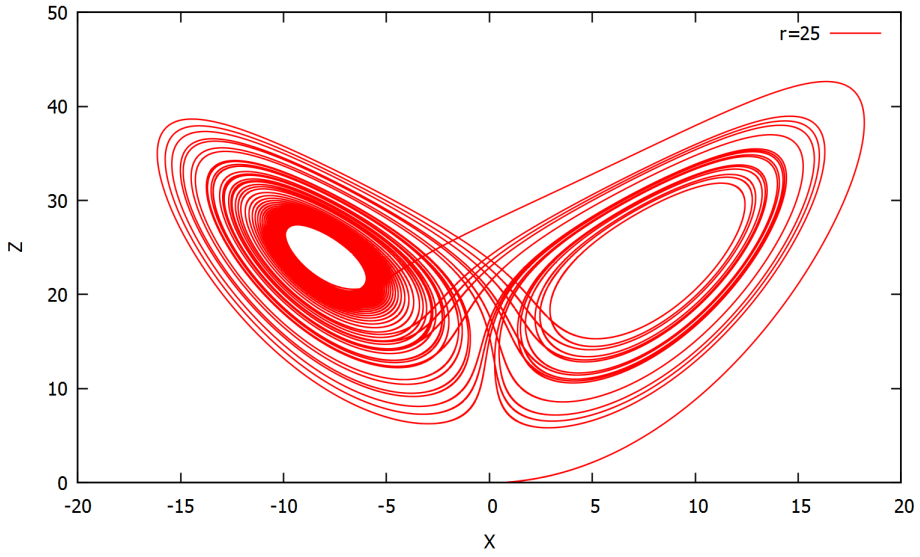

Figure 2: Trajectory of the Lorenz model projected onto the x-z plane, with $r = 25$. This was calculated using the second order Runge-Kutta method with a time step of 0.0001 and the initial conditions $x = 1$, $y = z = 0$.

model is required.

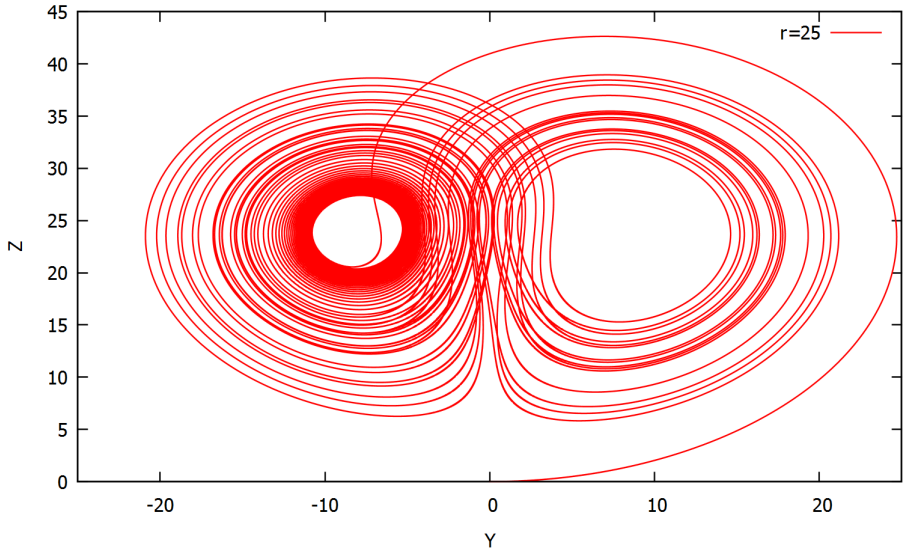

Figure 3: Trajectory of the Lorenz model projected onto the y-z plane, with $r = 25$. This was calculated using the second order Runge-Kutta method with a time step of 0.0001 and the initial conditions $x = 1$, $y = z = 0$.

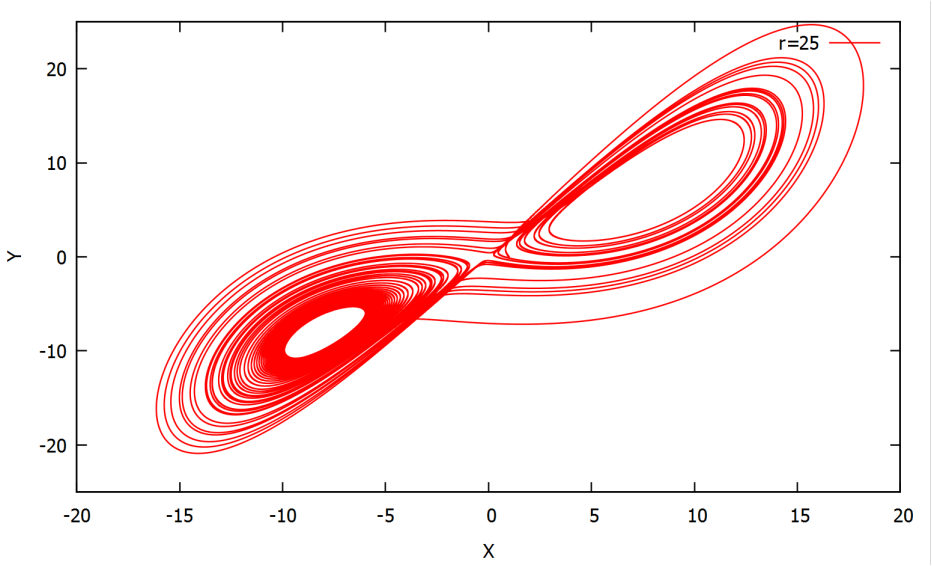

Figure 4: Trajectory of the Lorenz model projected onto the x-y plane, with $r = 25$. This was calculated using the second order Runge-Kutta method with a time step of 0.0001 and the initial conditions $x = 1$, $y = z = 0$.

To construct the Poincaré sections for the Lorenz model, a phase plot of z against y whenever $x = 0$ as in *figure 5*. More simply put, it is a plot of the points at which the trajectory of the y-z plane. *Figure 5* was obtained when $r = 25$, therefore it can be said with certainty that the Lorenz model was in the chaotic regime. However, while the plot of z against y still exhibits chaotic behaviour, a higher degree of regularity is visible than what was gleamed from *figure 2*.

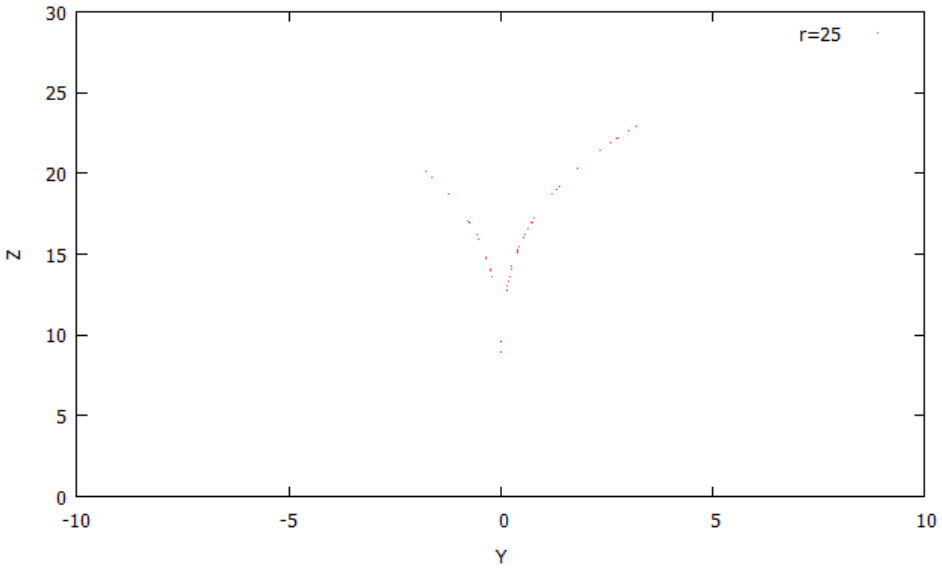

Figure 5: Phase-space plot of z versus y with points plotted only when $x = 0$ with $r = 25$, calculated with a time step 0.0001 and total time of 500. Initial conditions are $x = 1$, $y = z = 0$

This is shown even further through the a phase plot of z against x whenever $y = 0$ (*Figure 6*). Like *figure 5*, there is a degree of regularity present despite the fact that the behaviour exhibited by the trajectory of the Lorenz system is highly chaotic. From this an observation can be made that even thought time-dependent behaviour of the Lorenz system is unpredictable and highly irregular, it is possible to predict with certainty that the system can be seen somewhere on the attractor surface in phase-space.

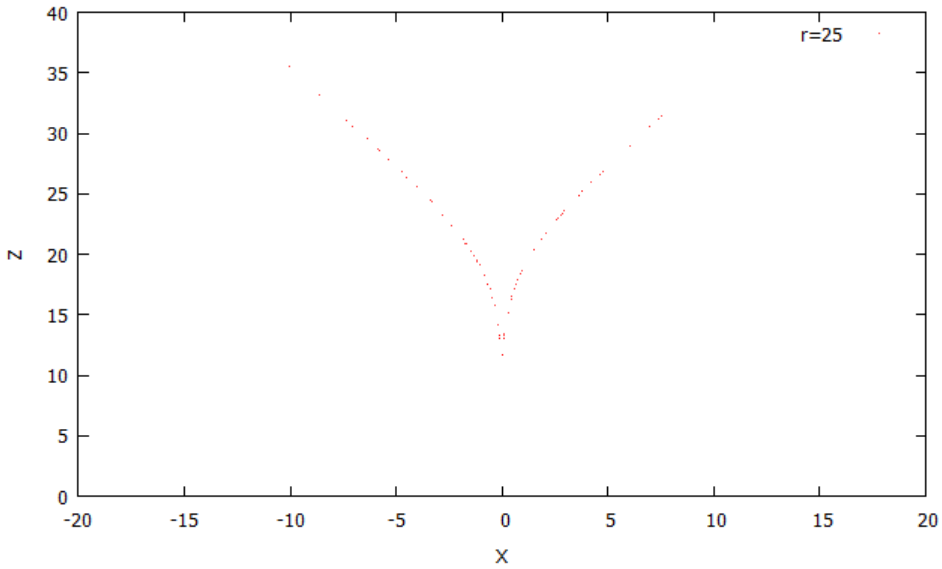

Figure 6: Phase-space plot of z versus x with points plotted only when $y = 0$ with $r = 25$, calculated with a time step 0.0001 and total time of 500. Initial conditions are $x = 1$, $y = z = 0$

In *figures 7 and 8*, the initial conditions of the system have been changed from the standard initial conditions of $x = 1$ and $y = z = 0$ to the initial conditions $x = 0$, $y = z = 1$. What is interesting to note is that the Poincaré exceptions generated by these initial conditions are very similar to those produced under the standard conditions. This suggests that the Poincaré sections of the Lorenz system are independent of initial conditions. This furthers the idea there is a degree of regularity present in the chaotic regime of the Lorenz system and that the system can be seen somewhere on the attractor surface in phase-space.

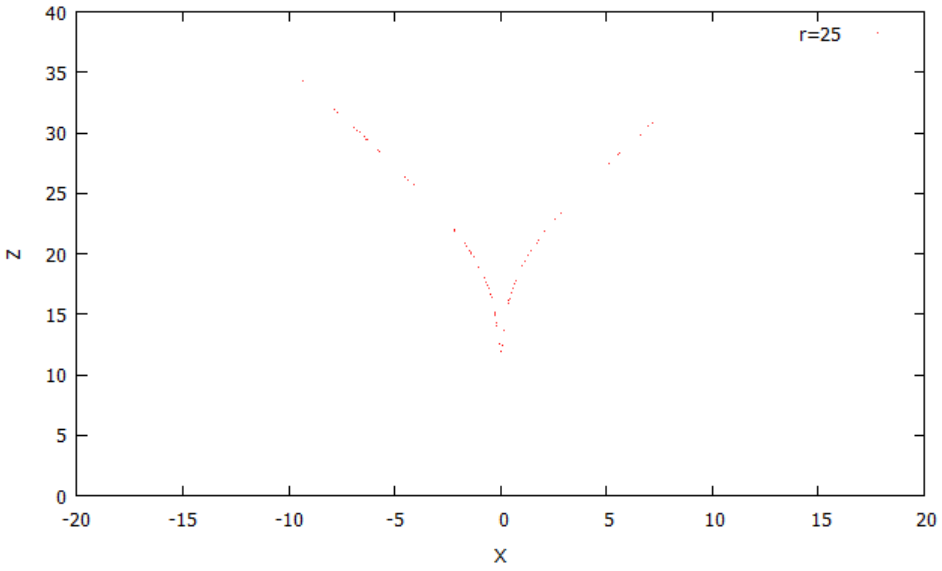

Figure 7: Phase-space plot of z versus x with points plotted only when $y = 0$ with $r = 25$, calculated with a time step 0.0001 and total time of 500. Initial conditions are $x = 0$, $y = z = 1$

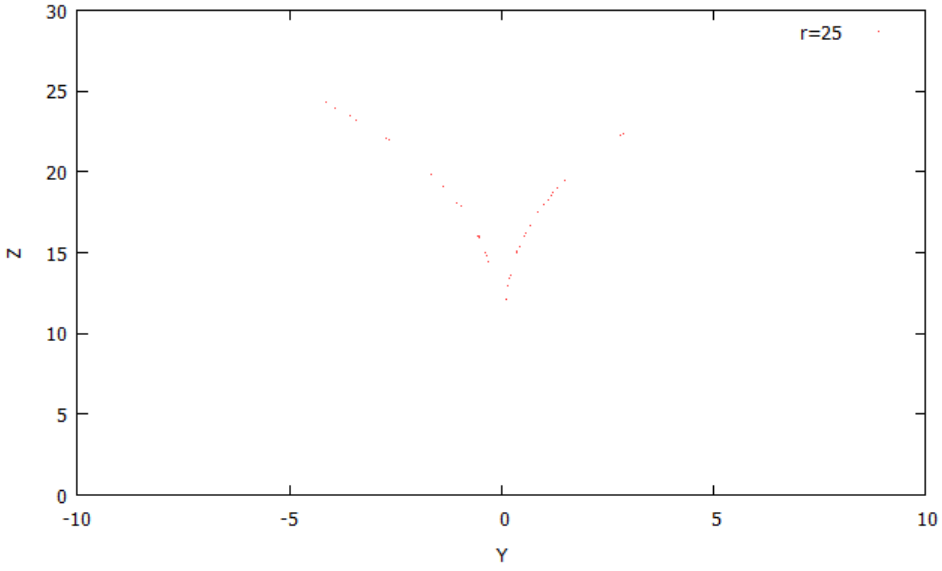

Figure 8: Phase-space plot of z versus y with points plotted only when $x = 0$ with $r = 25$, calculated with a time step 0.0001 and total time of 500. Initial conditions are $x = 0$, $y = z = 1$

For the Lorenz system under the standard initial conditions of $x = 1$ and $y = z = 0$,the Lyapunov exponent was calculated to be 6.80016 for the Lorenz model near the transition to chaos at $r = 24.74$. As the value of r was decreased it was noted that the value of the Lyapunov exponent decreased until it became negative. As well as this, as the value of r was increased the value for the Lyapunov exponent also increased. This indicates that as the Lorenz system passes from the non-chaotic

regime to the chaotic regime, the Lyapunov exponent transitions from a negative value to a positive value.