Modeling the Chaotic Pendulum in MATLAB

Elijah Coleman Emily Herndon Andrew Mays
Chris Cutler Chris Larson Bill Bodenhamer

Abstract

We have created a function to model the bevavior of two periodically forced pendulums started very close in their phase space. We have incorporated the existing function ode45 in our code. Our final code is gives a movie as output and is of the form, chaos(initial values 1, initial values 2, maximum time, time map value, file name, speed of movie).

1 Introduction

The chaotic pendulum we will be modelling is the periodically forced pendulum, meaning that a periodic force is applied to an otherwise normal pendulum. This pendulum swing in only two dimensions and can be modeled with the equations

$$\dot{\theta} = \omega \tag{1}$$
$$\dot{\omega} = -\sin\theta + \epsilon\cos t \tag{2}$$

where θ is the angle of the pedulum measured from the lowest position of the pendulum, ω is the angular velocity, and t is time. These equations have their solution in the θ vs. ω phase space, so, in order to create a graph, we will take a "snapshot" at certain intervals in the extended phase space and project those points onto the phase space. This time of graph is called a time x map where x is the interval between the "snapshots". To create these plots we have to numerically integrate the system of equations 1 & 2 by using the Runge-Kutta Method. Because these equations are extremely sensitive to error we have to re-evaluate at much smaller intervals. We want to see the relationship between two pendulums started very close together, so we evaluated for two systems.

2 Methods

Equations marked with a $*$ are evaluated for both pendulum 1 and 2.

In order to solve this system with ode45 we first have to define it as a function in the form

```
function(t,y)
```

This means that we have to express θ and ω as two elements of a vector. So we will define our function as

```
function rr=cps(t,y)
   epsilon = .01;
rr = [y(2); -sin(y(1))+epsilon*cos(t)];
```

We can now insert our system into ode45 as follows

```
[t1,p1] = ode45(@cps,timespan,initial values 1, options*
```

This gives us an output as a vector where t1 is the time and p1 is the vector [theta1,omega1]. To increase the accuracy of this method we reduced the step size, so we set the options as

```
options = odeset('MaxStep',(time(2)-time(1))/1000)
```

We now have a solved system, but it is not in the form that we want it in, so we need to get these at the values corresponding to the intervals defined by the time map. So what we do is interpolate between each point, here we have done this linearly.
We define

```
time_space = 0 :  time map value :  maximum time
```

and with these points we interpolate

```
theta1_space = interp1(t1,p1(:,1),time_space,'linear')*
omega1_space = interp1(t1,p1(:,2),time_space,'linear')*
```

thus producing quantized data for both θ and ω. Now we have the ability to create this time map, but if we graphed it now we would have a plot that extended to θ values beyond $-\pi$ and π. This is not what we want, we want a plot that has been normalized between $-\pi$ and π. To do this we need to subtract the number of full rotations it has made from the total angle. We do this with the code

```
for k = max(size(theta1_space))*
    theta1_space_-pi2pi(k) =
theta1_space(k)-2*pi*(round(theta1_space(k)/2/pi)))*
```

We now have all the data we want, but we need to plot it. So what we will do is plot the function point by point and take a picture each time a point is added. In order to do this we need to pre-define the axis so they won't jump around. To make sure that our graph still accomodates for all of the points we need to set y-axis values that change with the data, our x-axis data has been set to between $-\pi$ and π so those will be our axis limits. For the y-axis we define our max axis value as

```
axis_val = max([-omega1_space,omega1_space])*
```

We can now plot our function, but in order to take a picture of each new point we have to use a for loop

```
figure(1);hold;grid;axis([-pi pi -axis_val axis_val])
          for k = 1:max(size(time_space))
  plot(theta1_space_-pi2pi(k),omega1_space(k),'.g'
    ,theta2_space_-pipi(k),omega2_space(k),'.b')
                H(k) = getframe;
              W(k) = getframe(gcf);
```

We must also label our axes and title our plot

```
          xlabel('\theta (rad)','Fontsize',16)
        ylabel('\omega (rad/s)','Fontsize',16)
title('\it(\bf(Two Chaotic Pendulums))','Fontsize',16)
```

We now define our movies.

```
                    movie(H,1,speed)
movie2avi(W,'filename','compression','None','fps',speed)
```

This allows you to watch the movie as it is being created and later on as a separate file.