Numerical Methods to Analyze Precession in an Elliptical Pendulum
(Abstract and Techniques)

Harshit Chhabra

The precession of the major axis of an elliptical orbit is an important phenomena in many different physical systems (like the planetary orbits). A simple pendulum shows precession as well when flicked in a 2 dimensional motion instead of a simple 1D motion. We simulate a 2 dimensional pendulum and its precession and try to study it numerically. The simulation is based on the numerical solution of 2 orthogonal ordinary linear wave equations' solutions being superimposed and plotted so that the resulting figure is one of the Lissajous figures, i.e., an ellipse, only with precession. The rate of precession is then calculated and plotted against various other quantities to find its correlations and what it depends on. We hope to match the graphs with an analytical solution to get an analytical solution of the precession, or the rate of precession that agrees with the numerical methods

---

Introduction: Setting up the System

We start by setting up a system of equations that we can solve numerically using either one of the Runge-Kutta methods or by using Scipy's inbuilt functions. The equation we are solving is:

$$\frac{d^2\theta}{dt^2} + \lambda\frac{d\theta}{dt} + \omega^2 \sin\theta = 0$$

where λ is the damping coefficient. Then I am using the leapfrog method, or the RK-2 method to numerically solve the equation and produce a result, which looks like the following (with damping),

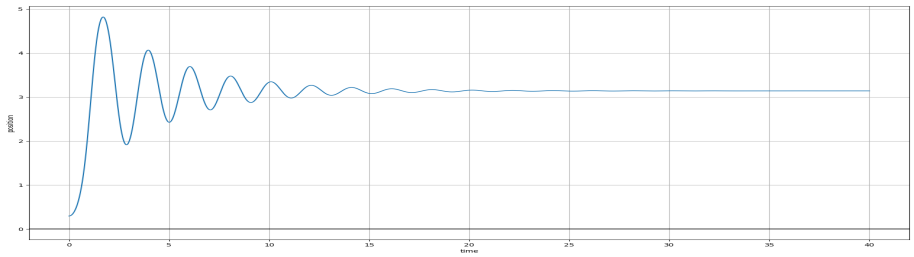


Figure 1: Simple Pendulum

Now we add another simple pendulum in the system with a phase (or initial amplitude) difference. In other words, solve the same equation again but for different initial parameters. The plot of the solution looks like the following (with damping),

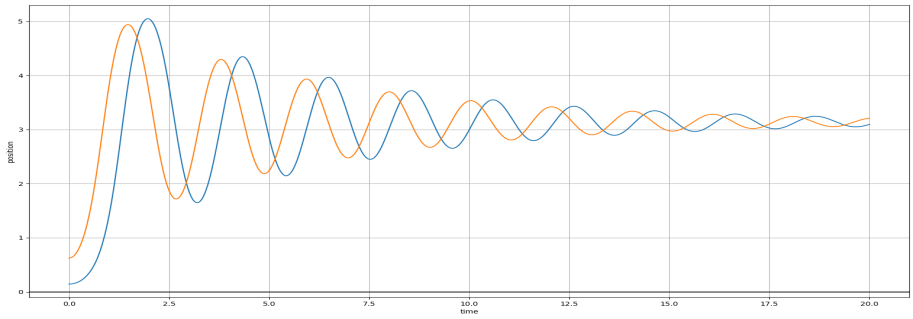


Figure 2: 2 Simple Pendulums with Phase Difference

To get to the desired lissajous figure, I will now plot the solutions of the two equations against each other, in this way, I am superimposing the solutions numerically.

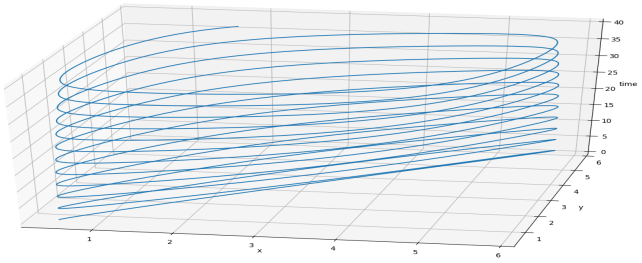


Figure 3: Elliptical Path in Time (without Damping)

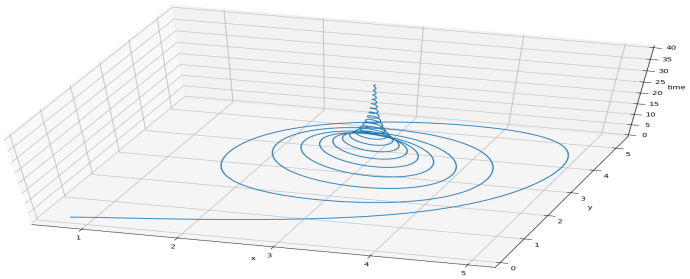


Figure 4: Elliptical Path in Time (with Damping)

We can see in Figure 3 that the figure precesses quite a lot in time but that precession decreases when damping is introduced (Figure 4). Now we shall calculate the precession rate (in radian per second).

Calculating the Precession Rate

Here, I will clarify that by precession and precession rate (rad/s), I mean the precession of the major axis of the ellipses formed by the revolution of the pendulum. I numerically calculate the precession rate of the of the pendulum using the following steps:

1. Pick 2 ellipses, preferably near the start and the end of the revolution of the pendulum
2. Find the major axes of the ellipses, by first finding out the near center of the ellipse (using Center of mass method, as mass of the bob is identity) then finding out the point farthest away from the center, that point will be on the major axis, along with the center, you will have an equation for the axis.
3. Then finding the angle between the 2 lines will give you the precession in radians.
4. Dividing it by the time passed between the 2 revolutions (you can find the time passed numerically by using the index of the x and y lists and passing them in the time list in your code) will give you the precession rate in rad/s.

The equation for finding the angle between the lines is,

$$\tau = \arctan \frac{m_1 - m_2}{1 + m_1 m_2}$$

where τ is the angle in radians and m_1 and m_2 are the slopes the major axes used.

The custom functions written for finding the slope of the major axes directly are as follows (they follow the steps as written above 1-4),

```
def center(list1, list2):
    x_com = st.mean(list1)
    y_com = st.mean(list2)
    return x_com, y_com
def maximum(list1, list2, x_com, y_com):
    distance = []
    i_max = list1[0]
    m_max = list2[0]
    for i,m in zip(list1, list2):
        dist = np.sqrt((i-x_com)**2 + (m-y_com)**2)
        distance.append(dist)
        if i_max > list1[distance.index(dist)]:
            i_max = i
            m_max = m
    return i_max, m_max
def slope_major_axis(i_max, m_max, x_com, y_com):
    slope = (m_max-y_com)/(i_max-x_com)
    return slope
```

Figure 5: Custom Methods

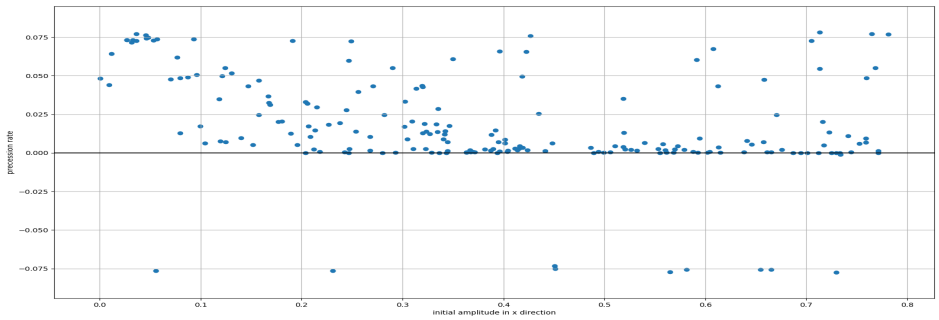
Now we will plot the precession rate against some other quantities to see the relations and find some correlations.

Plotting Relations

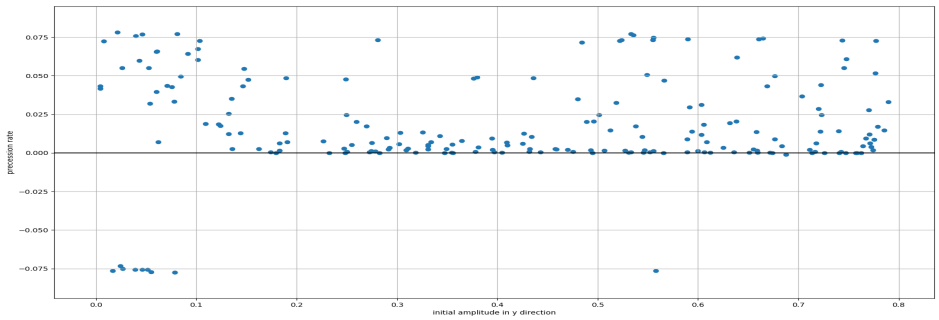
As we solved two different equations to get the Lissajous curve, we will plot two different graphs for initial amplitudes in x direction and in y direction and then some combination of the two. I have taken a sampling of 200 randomly generated values between 0 and $\pi/4$ for initial amplitudes so that I do not have to best fit and yet I can see the general trend, instead of doing a best fit for the curve since I do not have an analytical solution as of yet so a best fit would not give anything physical and even mislead.

Preliminary Plots

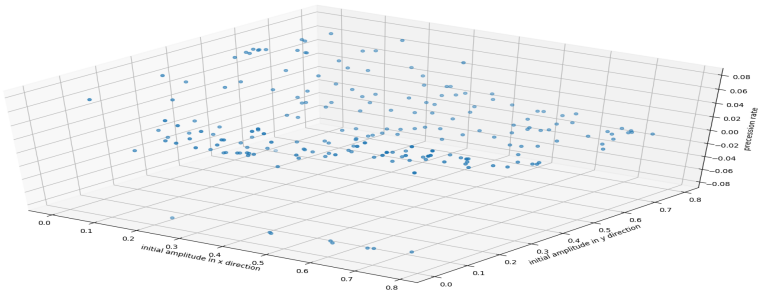
1. Initial amplitude in x direction vs Precession Rate



2. Initial amplitude in y direction vs Precession Rate



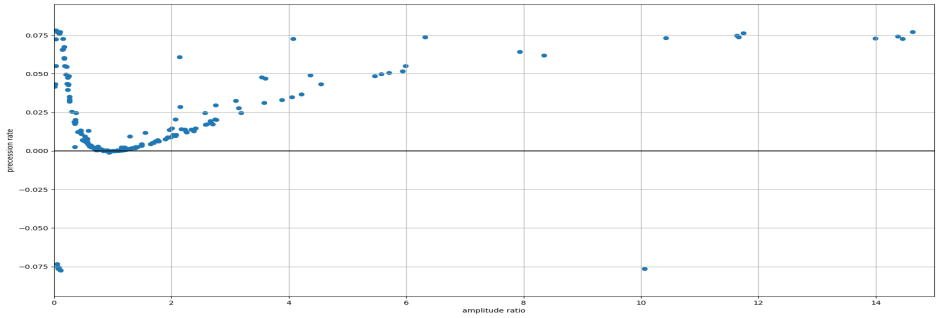
3. Initial amplitude in x direction vs initial amplitude in y direction vs Precession Rate



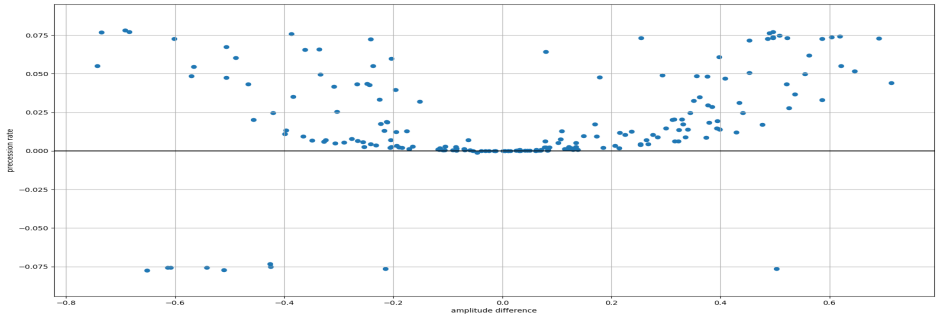
New Ideas

The first three didn't give me much, so I went to Professor Phookun for help, and did what he asked, although I did not get the result he predicted I should get. Furthermore, I have some more ideas branching off of the ideas that I am working on that are not in this report.

1. Ratio of initial amplitudes vs Precession rate

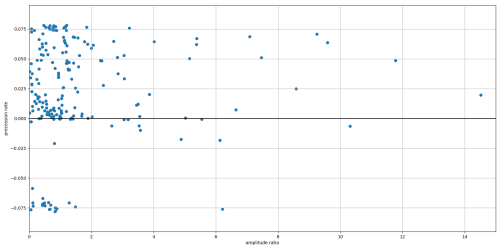


2. Ratio of initial amplitudes vs Precession rate

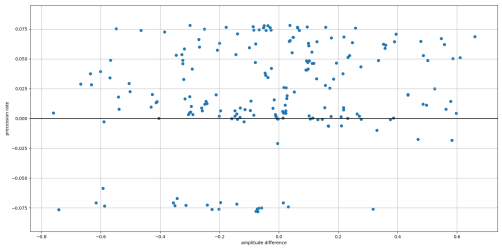


A Discussion on the Possible Critical Role of Damping

I would like to discuss the possible role of damping in these relations. Obviously, damping when applied will have a role but I have come to think that damping has a critical role in bringing about the precession and the correlation between different physical quantities because when I changed damping to 0, the correlations in the last two graphs changed, especially in the graph of amplitude difference vs precession rate, the correlation seemingly went away. Another note, the general curve or the trend on the last curve seemingly depends on the value of the damping coefficient, with the trend going toward linear as the value of the damping coefficient increases.

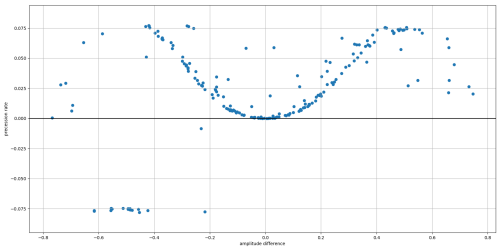


(a) Amplitude ratio vs precession with zero damping

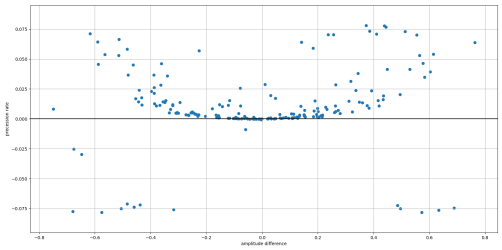


(b) amplitude difference vs precession with zero damping

Figure 6: Difference in General Trend based on just Damping



(a) Damping Coefficient of 0.1 units



(b) Damping Coefficient of 0.9 units

Figure 7: Difference in General Trend based on just Damping

Note that no such relation is found in the amplitude ratio vs precession rate plots, only difference plots. This has led me to explore the direct relation of precession rate and damping coefficient, which is what I am doing right now and hope to complete soon.

Future Work

Now I hope to analyze the Graphs further and as I have said, explore the relation between damping coefficient and precession rate more directly. Furthermore, I hope to have an Analytical solution for the precession and hope to match it with the numerical solution that I am currently working with. I also hope to work out the mathematics and the reason behind the relations given by amplitude ratios and differences graphs as I currently only have little idea.