WPI | Department of Physics

Modeling a pendulum beyond the small angle approximation

114X

Last Edited September 20, 2022

Lab Objectives

- Intermediate modeling skills – Write code to numerically solve the actual pendulum oscillator
- Explore the limits (or constraints or variables or options but whatever sounds best to you) of friction, initial angle, time steps
- Differential Equations in coding –Write code to compare this actual solution to the small angle approximation

Lab Equipment

- Computer.

Theory

Originally written by Edward Jarvis. Any mistakes within are due to Dana – so please email her if you have comments or corrections.

You've been taught that the solution to the pendulum oscillator is:

$$\Theta(t) = \Theta_0 e^{-\gamma t} cos(\omega t + \delta) \tag{1}$$

However, this solution is for the simplified differential equation:

$$\ddot{\Theta}(t) = -\mu\dot{\Theta} - \frac{g}{l}\Theta \tag{2}$$

This simplification is from the Taylor series approximation of the sine function, the full differential equation is as follows:

$$\ddot{\Theta}(t) = -\mu\dot{\Theta} - \frac{g}{l} sin\Theta \tag{3}$$

This equation is solvable only by using functions called elliptic integrals. Rather than go through the laborious mathematics to solve this equation analytically, the goal of this lab is to solve the equation numerically.

Procedure

To solve the pendulum oscillator use python, with the packages numpy and matplotlib.pyplot. You will want to initialize the various parameters of your system: gravity, length, friction, initial angle, and initial velocity. You will want to establish the total time, time step (start small!), as well as define various other parameters (like the angular frequency) in terms of these parameters.

You will then want to define your differential equation as such in python:

```
def get_theta _double _dot(theta, theta _dot):
    return -mu * theta _dot - (g/l) * np.sin(theta)
```

Next, you will want to solve the equation. Think about what it means to 'solve' a differential equation; you want to know all of the angles at any particular time. Since the initial angle was defined, you just need to update that value for every time step. The simplest way to update that angle is to add the amount it would have moved by in that time step. That amount is simply the angular velocity multiplied by that time step.

In addition, you'll need to update the angular velocity in the exact same way, by adding on the angular acceleration multiplied by the time step. You will calculate the angular acceleration using the differential equation. Then you will append the theta value to a list so you can construct a list of numbers for plotting.

Finally, you will plot your data. You will then plot the theoretical equation you learned in class. You will want to run your code and vary several different parameters. What happens when friction is very high, very low, or zero? What happens when the initial angle is xvvery large, very small, or exactly pi radians? What happens if your time step is too large?

Lab Objectives

- Several graphs of angle vs time for each case (see end of procedure for more details)

- Discussion of differences between small angle approximation and the numeric solution
- Discussion of timesteps and errors that can arise
- Blooper graphs: trials that did not work or seemed to make graphs that were nonsense

If you'd like to try to model a double pendulum for some extra credit, see the github page: https://github.com/wojciechmo/double-pendulum

Acknowledgement

Thank you to Ed Jarvis for writing this lab. Any good things in this lab are his, any mistakes are from re-writes.