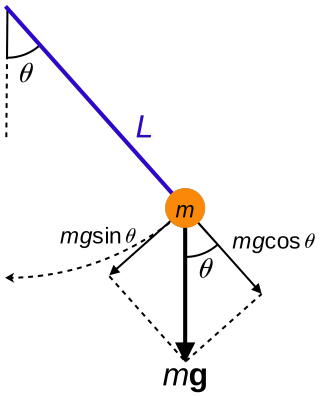
PHYS2300: Assignment 6 - The not so simple pendulum

Objective

This purpose of this exercise is to use numerical integration to solve the equation of motion for a simple pendulum, and extend that to include damping and driving forces.

Prerequisites: Review Chapter 8. This activity uses Visual Python.



Introduction

Our last assignment modeled the motion of particles, but we can also numerically model other systems, including systems that rotate. A good example of this is the simple pendulum shown in the figure. The sum of the torques on the ball is proportional to the angular acceleration of the pendulum,

$$\sum \tau = I\alpha$$
$$\alpha = \frac{d^2\theta}{dt^2} = \frac{\sum \tau}{I}$$

Notice that this equation looks very similar to our Newton's second law from Assignment 4, but uses the rotational angle, angular velocity, and angular acceleration instead of position, velocity, and acceleration in cartesian coordinates. Otherwise, the numerical integration method we used before can be adapted to solve this as well, provided we have an expression for the sum of the torques.

In this case, the torque is provided by the gravitational force, such that

$$\sum \tau = -Lmgsin\theta$$

and since the moment of inertia for a point mass is

$$I = mL^2$$

we have a new second order differential equation

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}sin\theta$$

which, like last time, we can break up into two equations

$$\frac{d\theta}{dt} = \omega$$
$$\frac{d\omega}{dt} = -\frac{g}{L}sin\theta$$

The Task

Given this mathematical framework, do the following:

1. Design a program that uses the computation of the position angle and velocity to animate a simple pendulum of length 1 meter. To do this, you want to make a ceiling, hang a rod from the ceiling that is attached to a ball at the end. Have the program compute the angle of the pendulum at each time step, and convert this into *x, y, z* coordinates so you can update the location of the rod and ball during the simulation.
2. Start by animating this using a method similar to the one used in Assignment 4.
3. To get a more accurate animation, change the calculation method to the 4th order Runga-Kutta integration method described in Chapter 8 of your text (specifically, look at Example 8.5).
4. Next, add in damping (or friction) on the pendulum by adding a term proportional to the angular velocity

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}sin\theta - c\omega$$

   Find a value for *c* that slows your pendulum gradually to a stop.
5. Next, add in a driving term that forces the pendulum at a frequency, *f*

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}sin\theta - c\omega + Asin(ft)$$

6. Test your program with an *A* of 0.5 and a driving frequency equal to 2/3 of the natural frequency of the pendulum.
7. Let's now test the longterm behavior of the pendulum. Create another pendulum in the same view, but give it slightly different initial conditions. Run this for an extended period of time and watch the results.
8. Using VPython's graphing capabilities, plot the angular position of the pendulum as a function angular velocity. This should produce some interesting patterns that show the deviation of the two pendulums over time.
9. Finally, open the file called "doublependulum.py" that came with the installation of VIDLE. Review the code to note any differences or similarities to yours, and run this, just for fun, noting the complex behavior.

To hand in

Your final animation of the two damped, driven pendulums including the graph of their behavior.