**Purpose of the experiment:**

The purpose of this lab is to use Euler Comer method to solve for theta of a motion of a pendulum without friction or driving force acting on it. We will not assume that the amplitude of the oscillation is small. This assumption is useful for solving mass swings, large angles, and the pendulums that go around the pivot point.

**Problem Description:**

In lab 4, equations like describing a harmonic oscillator will be used to solve for the equations given in this lab:



*(Equation 1 – angular velocity)*



*(Equation 2)*

I then plot the theta as a function of time where L was the length of a string of 1 meter, gravity (g) was 9.8 m/(s^2), initial displacement angle (theta\_0) was pi/12 and initial angular velocity (w\_0) is zero. I then plotted for 10 seconds. I then adjusted my code to change theta\_0 to be pi/6 and then pi/3.



*(Equation 3 – Energy of the system is conserved)*

I then calculated energy since I already knew w and theta at each time step and solved for equation (3). I then modified my script and plot Energy.

I then modified my script to adjust to show the relationship between the amplitude and period. When the amplitude is large, the pendulum spends a longer time at its turning pints, making it larger than ones with smaller amplitudes. In the modification, I could find time that it took to reach the minimum displacement of the pendulum (which is initial displacement angle times -1). The time represents one half the period of oscillation. I then plotted the period against different values of the initial displacement angle. I also found the minimum and maximum of array with built in python functions. I changed the commands to make sure the they returned values specifically for my own plot.

**Code:**

In the beginning of my code, I imported useful functions numpy and and matplotlib.pyplot for mathematical and plotting uses. I then declared the initial values that would be used to solve for the equation 1 and 2.

t = 10

dt = 0.01

n = int(t/dt)

mass = 1

l = 1

gravity = 9.8

*(Figure 1 – These are my initializing statements)*

t holds the value of 10, which is the time length we will examine the behavior of the ball for, and dt is the number of steps. mass is mass in kilograms. n represents the number of points that will be plotted and g represents gravity in m/s^2. L represents the length of the of the string in meters

def calc(theta):

theta\_f = np.zeros(n); w = np.zeros(n); t = np.zeros(n)

w[0] = 0; t[0] = 0; theta\_f[0] = theta

for i in range(n-1):

w[i+1] = w[i] - (gravity/l)\*np.sin(theta\_f[i])\*dt

theta\_f[i+1] = theta\_f[i] + w[i+1]\*dt

t[i+1] = t[i] + dt

Energy = ((1/2)\*mass\*(l\*\*2)\*(w\*\*2)) + (l\*mass\*gravity\*(1-np.cos(theta\_f)))

min\_f = np.min(theta\_f[0:int(3.0/dt)])

Period = 2\*np.argmin(theta\_f[0:int(3.0/dt)])

return(theta\_f, Energy, t, min\_f, Period)

*(Figure 2 – The initial values in calc, energy calculation, and fucntions used for the limits)*

I then created my function calc similarly to figure (2) to create the values for a graph. In my function, I passed the variable theta. In the beginning of the function, I initialized arrays to store values at each step for theta, angular velocity (w), and time (t). I then had a for loop to calculate equations (1) and (2) in the for loop and to increment time variable (t). I then returned the array of thetas, energies, minimums, and periods with respect to time.

def plot2d(t, y):

plt.plot(t, y, label = labels)

plt.legend()

plt.title(titleP)

plt.xlabel(labelx)

plt.ylabel(labely)

*(Figure 3 – plot2d function I use to plot my values)*

I have a plot2d function that I used to graph my results since it would have been repetitive to hard code for plotting. For the function I passed the time array and theta array returned from figure (3).

#Initialize constants and title names

#Block of codes#

#initialized more constants depending what is needed to be calculated

#call calc\_f function and assign it

#plot the results

#call the function plot2d() to set parameters and pass the constants and titles declared before

…(Repeat blocks of codes until everything has been graphed)…

*(Figure 4 – structure of my function calls and plotting in ‘main’ portion)*

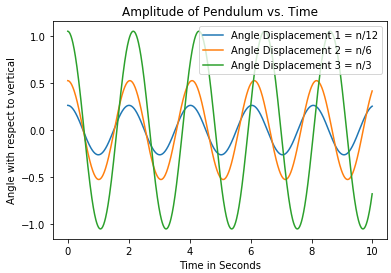
I also have a “main portion” of the lab that does all the function calls and some changes to the variables such as the name of axes for the plot2d function.

**Equations Solved & Algorithms Used:**

In this lab, I solved for equations 1 and 2. We know that the pendulum is a nonlinear system and treat it as a simple harmonic oscillator for calculation of small angles but not larger ones. When I solved for both equation 1 and 2, I knew that the tension of the string is -mgsin(theta) where m is mass, g is gravity, and theta is displacement. Since it’s in a closed system, the tension is equivalent to mass \* acceleration where acceleration in this type of system is angular acceleration which is described as -g\*sin(theta). With the angular velocity is described in equation 1 and the derivative of the angular velocity to describe for acceleration, we have angular acceleration equivalent to –(g/l) \*sin(theta) and dtheta = w \* dt.

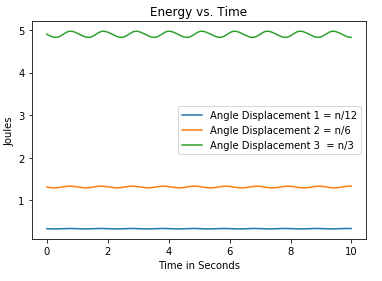
The reason why didn’t use just Euler’s method and used Euler – Cromer method was because the values being calculated were in polar coordinates. If Euler’s formula was used, there would be infinite energies since it would not be conserved.

**Results & Analysis**



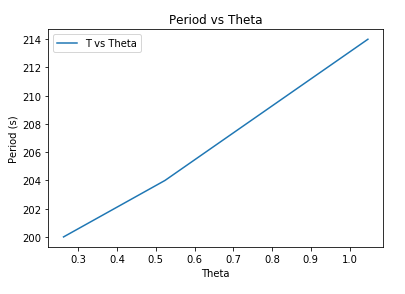
*(Figure 5 – Amplitude of pendulum vs. time)*

In figure (5), there are three lines that describe the nonlinear pendulum with different angle displacements. The period of constant length is changing compared to its initial displacement and increases when angle displacement increases.



*(Figure 6 – Energies of nonlinear pendulum with different angle displacements)*

In figure (6), we observe that the system is closed. That means there are no added forces or energies onto the pendulum. Therefore, energy is conserved even though it oscillates because it will have the same amplitude.



*(Figure 7 – Each period plotted against initial angle displacements)*

In figure 7, it can be concluded that period is directly proportional to the initial angles for displacements base off the linear shape of the graph despite the a small ‘slant’ in the middle of the graph.