Parham Keshavarzi - 670820642

**Purpose of the experiment:**

The purpose of this lab is to express an integral that is impossible or difficult to evaluate in closed form.

**Problem Description:**

In lab 5, the period of a non-linear pendulum will be determined via numerical integration. The equation of motion of a non-linear pendulum is described below.



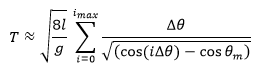
*(Equation 1 – equation of motion of a non – linear pendulum)*

Where g is acceleration due to gravity, l is the length of the pendulum and theta is oscillation angle. The period of the pendulum is described by equation 2 below:



*(Equation 2 – Period of pendulum)*

Where theta\_m is the largest angle, the pendulum makes with the vertical. Solving with analytical terms of elementary functions is impossible. Calculating the integral will be done by discretizing theta and calculating the area under the curve as shown in equation 2.1 below.



*(Equation 2.1 – Period of pendulum discretization)*

Delta theta describes the grid size, i\_max is equivalent to theta\_m = i\_max \* delta\_theta.

**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.

d\_theta = 0.001

l = 1

gravity = 9.8

theta\_m = np.linspace(np.pi/30, np.pi, 777)

Period = np.zeros(theta\_m.size)

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

D\_theta is delta theta is delta\_theta from equation 2.1, l is length of pendulm in equations 1, 2, and 2.1, theta\_m is an array that represents the maximum angle with the respect to vertical from equations 2 and 2.1, g is gravity, Period is an array that holds the time values for each theta\_m and i\_max is an array that is described by i\_max in equation 2.1.

for j in range(len(Period)):

i\_max[j] = (theta\_m[j]/d\_theta)

dT = [] #place holder

for i in range(i\_max-1):

dT.append(d\_theta/ ( np.sqrt(np.cos(i\*d\_theta) - np.cos(theta\_m[j]) )))

Period[j] = np.sum(dT)\*np.sqrt((8\*l)/gravity)

*(Figure 2 – Nested for loops for my calculations)*

I then calculated equation 2.1 with a nested for loop. The outer loop calculates i\_m an array of max angle and initializes dT to store increment additions. The outer loop has the same number of iterations as the length of the array in Period, which stores the period at theta\_m. The inner loop then calculates the period by calculating dT for each grid point I and multiplying it by square root of ((8\*l)/gravity). It is repeated the same number of times as the size of i\_max minus 1.

**Equations Solved & Algorithms Used:**

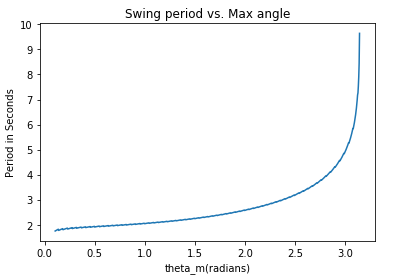
In this lab, I solved for equations 1 and 2. The algorithm used for solving and plotting the equations was newton-cotes method. This was especially useful for solving integrals in python that was difficult or impossible to evaluate in close form. Newton-Cotes form for an integral is described as the formula below



*(Equation 3 – Newton-Cotes method of an evaluation of an integral)*

Where x\_0 is a and and x\_n is b which are the bounds of the integral, f(x) is the function. i is the number of equally spaced point between x\_0 and x\_n. x\_i = h\*i +x\_0, where h is the called step size equal to (x\_n – x\_0) / n = (b − a) / n. w\_i are called weights. The weights are found in a way similar to the closed formula. Using this method on equation 1 would evolve to equation 2 for evaluation.

**Results & Analysis**



*(Figure 3 – The Period of a pendulum graphed plotted with respect to maximum angles)*

The period in figure 3 increases as the maximum angles increases and seems to approach infinity when the angle approaches pi, but then comes back down after a long amount of time. This is because pi is the largest angle that is possible.