PHYS340 Report: Computing the Motion of the Pendulum

Gil Garcia

March 28, 2019

1 Introduction

Often times, especially in physics, we run across a problem which has no analytic solution so we must use a numerical method to get an approximate answer to the problem. One such case is when dealing with the motions of a pendulum. Here, we present computational approximations to the undamped and damped pendulum. More specifically, we will present the motion/period of the undamped and damped pendulum as a function of initial angle. And while the undamped pendulum does, in fact, have an analytic solution, the undamped pendulum does not have an analytic solution, so it is necessary to approximate it as presented here. First, however, we must manipulate the equations of the pendulum, so that we can computationally compute them.

1.1 Nondimensionalizing the Undamped Pendulum

To be able to generate a numerical answer to the undamped pendulum, we must nondimensionalize the equation of the undamped pendulum. That is, we want to make our equation independent of dimensions, so that we can generate a general solution to the equation. To do this, consider the equation of the undamped pendulum:

$$ml\frac{d^2\theta}{dt^2} = -mg\sin\theta.$$

We can divide by mg on both sides, and since $\omega_0 = \sqrt{\frac{g}{l}}$, we get

$$\frac{1}{\omega_0^2}\frac{d^2\theta}{dt^2} = -\sin\theta.$$

The nondimensionalizing occurs when we redefine t such that t is now equal to $t\omega_0^2$. So, we have now gotten rid of our dimensions. Hence,

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

Breaking this equation into two first order differential equations, we produce our desired result:

$$\frac{d\theta}{dt} = \omega \qquad \text{and} \qquad \frac{d\omega}{dt} = -\sin\theta.$$

1.2 Nondimensionalizing the Damped Pendulum

We now perform the same nondimensionalization for the damped pendulum equation. That is, we nondimensionalize

$$ml\frac{d^2\theta}{dt^2} = -mg\sin\theta - \gamma\frac{d\theta}{dt}.$$

Similar to the undamped equation, we divide by mg and then we let $1/\omega_0^2 = \frac{l}{g}$ be consumed by the t in the second order differential. Thus,

$$\frac{d^2\theta}{dt^2} = -\sin\theta - \frac{\gamma}{mg}\frac{d\theta}{dt}.$$

Now, we will define the damping factor, $\frac{\gamma}{mg}$ simply as $\frac{1}{q}$. Therefore, our complete nondimensionalized equation is

$$\frac{d^2\theta}{dt^2} = -\sin\theta - \frac{1}{q}\frac{d\theta}{dt}.$$

Breaking this equation into two first order differential equations, we get our desired result for the damped pendulum:

$$\frac{d\theta}{dt} = \omega \qquad \text{and} \qquad \frac{d\omega}{dt} = -\sin\theta - \frac{1}{q}\omega.$$

2 Computational Methodology

Once we have prepared the equations as we did above, we can use numerical techniques in programming to obtain numerical solutions. Thus, we create the program `garcia_damped.c`. In the program, we first define a step size of, `DT`, 0.000001, which is where the accuracy of the step size levels off such that any step size smaller than that will not contribute more accuracy. We now want to iterate for different values of angles. Specifically, we will iterate from angles $[0, \pi]$ in increments of 0.1, since this will produce sufficient data points for our purposes. Therefore, we create a for loop that does as such. For each angle, we want to calculate the period of the pendulum. Here is where the nondimensionalized equations become important. For the undamped equations, we program the equations as `x1+=x2*DT` and `x2-=sin(oldx1)*DT`. Here, `x1` is our angle and `x2` is our velocity. We are updating them by changing them with respect to the defined step size. We are looking for the first two roots, so that we can subtract them and find our period. Therefore, we must call our interpolation function so that when there is a sign switch in our position (when it crosses the x-axis), we will be able to approximate the x value at which it does so. As mentioned, we calculate this for the first two roots and subtract to get the period. Since the undamped equation has an analytic solution, namely `Elliptic[K]`, we also code that into out program so that we generate `Elliptic[K]` values for each angle given. Since period is given by $\frac{4}{\omega_0}$Elliptic[K], and since we defined $\omega_0 = 1$, we simply multiply each Elliptic[K] value by 4 so that we can then compare our computed results.

In the case for the damped pendulum, we replace the equations with the following:

`x1+=x2*DT` and `x2-=sin(oldx1)*DT + 1/q*DT`.

These equations follow directly from the nondimensionalization of the damped pendulum. We now run through the same algorithm as the undamped case to get our periods as a function of angle. Here, however, we can adjust our `q` variable as needed. That is, we define a large `q` for a soft damping and a small `q` for a heavy damping. In this case, however, we do not have analytic solution to compare our computed values with.

Using this program, we can, as mentioned, generate plots of period of the different pendulum cases as a function of angles.

3 Results

With `garcia_ damped.c`, we first generate a plot that looks at the period as a function of angle for the undamped pendulum. We also plot the first 5 terms of the 4*Elliptic[K] function to compare our computed values:

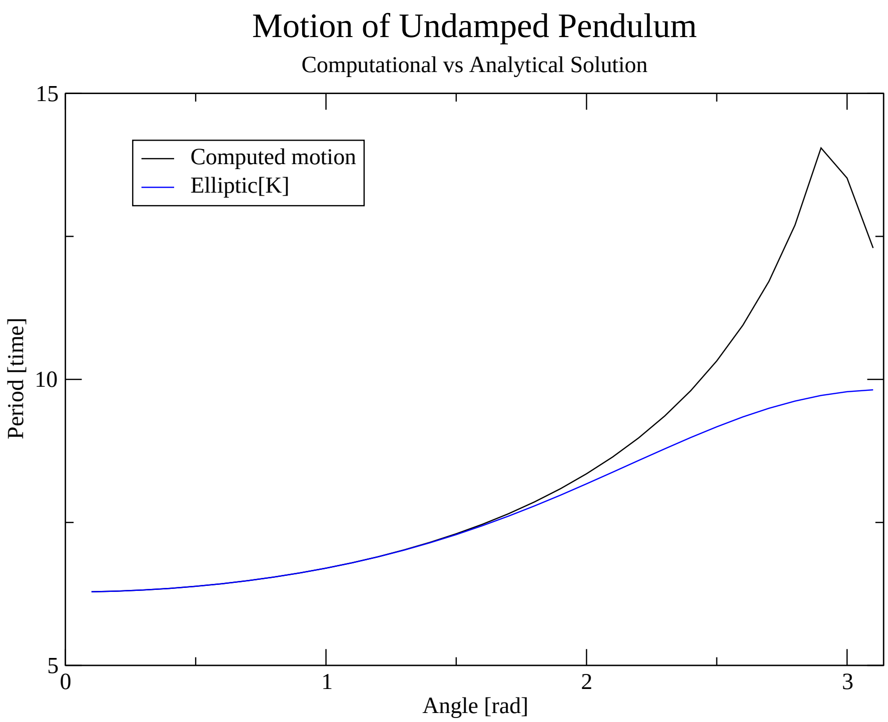

Figure 1 – Period vs angle of the computed values and the analytic solution(4*Elliptic[K]).

To get a better grasp of the difference, we also produce the following plot:

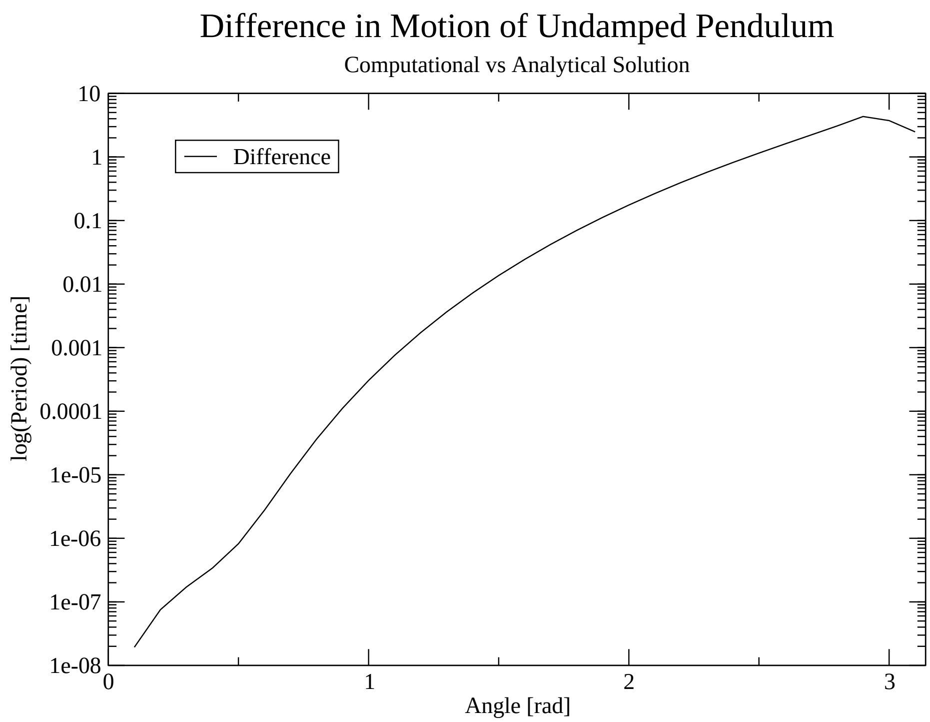

Figure 2 – The log of the period vs the angle of the absolute value of (computed values)-4*Elliptic[K].

We then generate similar plots for the damped pendulum case:

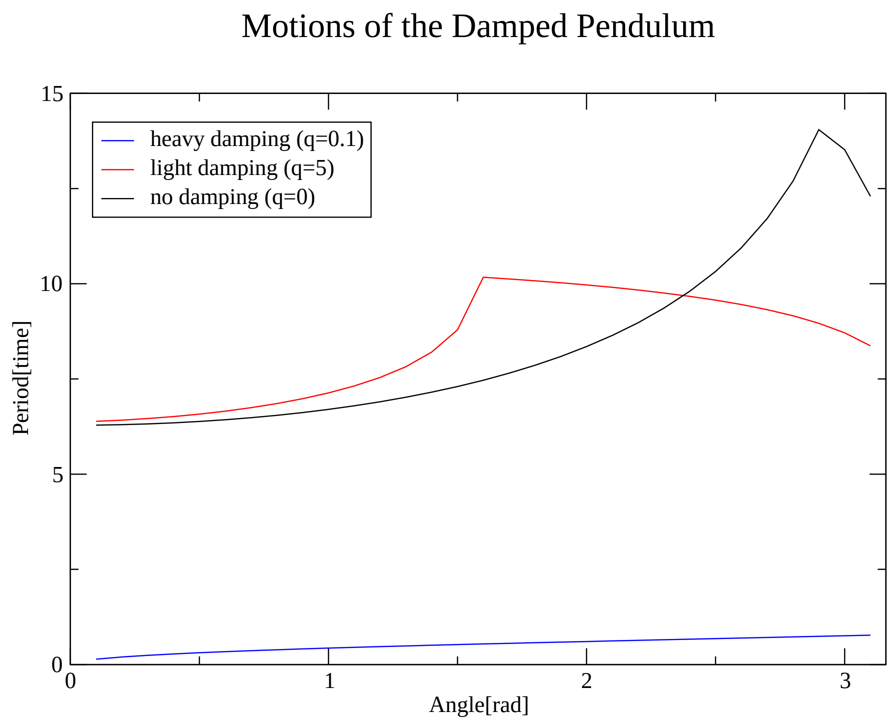

Figure 3 – Period as a function of angle of an undamped, lightly damped, and heavily damped pendulum.

4 Discussion

Using the figure 1 and 2, we can determine the degree of accuracy of our program. We see that until about 1.5 radians, our difference is less than one. However, the general trend is that our program becomes less accurate as we increase in angle. One reason for this difference is that we only use the first five terms of the Elliptic[K] function. If we wanted to be more exact, we will need to include more terms in the series. We can look at out how far off we get at higher values of angles with the following plot:

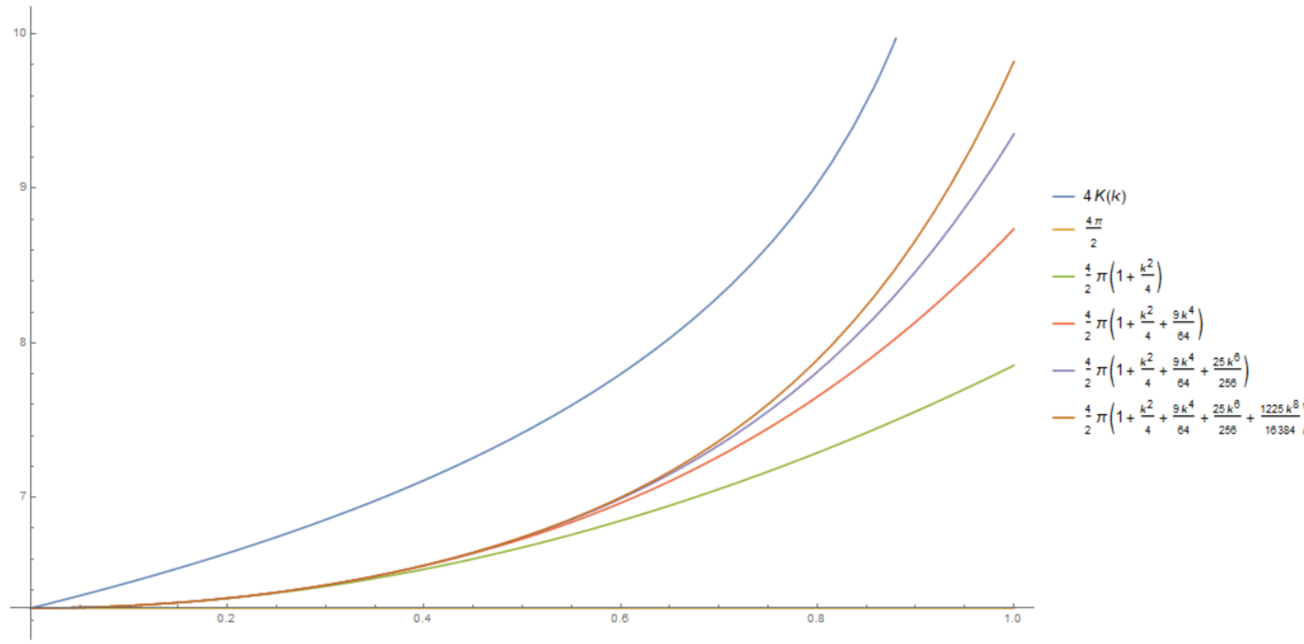

Figure 4 – 4*Elliptic[K] values using different number of terms from its expansion.

We see that the more terms, the more accurate we are, and since we have five terms, we have a fair approximation. However, we are still not as accurate as can be since we do not the analytic form of Elliptic[k].

In the case for the damped pendulum, we see that can manipulate the periods as a function of angle by adding different amounts of damping. In figure 3, we presented three different levels of damping and wee see that the smaller `q` value that we have (bigger $\frac{1}{q}$), the more intense we damp our pendulum. More specifically, a higher amount of damping will the periods to become very small as a function of angle. Thus, for $q = 5$, we see that the periods are significantly smaller than the the lightly damped or the undamped pendulum. In the case for the lightly damped pendulum, we still see similar features to the undamped pendulum. However, since there is damping, it will experience those features a lot faster. That is, we will reach the point of where our program becomes less accurate faster since it reaches a peak sooner. After that peak, our program is not capable of calculating proper values for the period of the damped pendulum.

5 Conclusion

Through this process, a variety of skills and tools were used to get a complete picture of the motion of the pendulum problem. A big skill that was implemented here was the nondimensionalization of a differential equation with physical dimensions, which is what allows us to access the solution to the undamped/damped pendulum computationally. From there, the coding of the program also presented learning opportunities with working with nested for loops and if-else statements. Lasty, error analysis continues to be a big part of what we must do when we choose to attack a problem computationally. That is, we needed to know how to analyze our errors when there is no analytic solution to compare our answers to.

Going forwards, there are a couple things we can do to our program still. The big one is that we must figure out how to get more accurate answers as we increase in angle. This could possible be done by having a dynamic step size as we move up through angles. We then want to also be able to accurately compute values after an angle of π. Right now, the program seems to break down after that angle, but it is possible that we will want our program to calculate the period at higher angles. All in all, there is always ways in which we can minimize errors in our program, so that we produce more accurate results.