**THE SIMPLE PENDULUM, REVISTED**

OBJECTIVES

* To explore numerical solutions of nonlinear ordinary differential equations

THEORY

Previously, we examined a simple pendulum system: a mass suspended at the end of a light string. We found

MATERIALS

data from your Simple Pendulum lab

Excel

Jupyter Notebook

PROCEDURE

Starting from Newton’s II Law for rotational systems and a free-body diagram, work out an expression for the angular acceleration of a simple pendulum in terms of the angle measured from the pendulum’s equilibrium position. Rewrite the expression you found in terms of an ordinary differential equation (ODE) of . Since this is a differential equation with a second-order derivative, we classify it as a *second-order ordinary differential equation*.

**Thinking about relationships between kinematic variables,** discuss with your table how we might rewrite this second-order ODE as two first-order ODE (i.e., as two equations with only first-order derivatives). Once you think you have the two equations, check-in with a TA to check your work.

How could you approximate your equations using what you know about derivatives and calculus? (*Hint:* how do we define a derivative?) Here, we’re making an approximation, just like we did previously with the small angle approximation. What’s different about this approximation? This is called **Euler’s Method**. There are more sophisticated ways to solve ODEs numerically, but this is one of the simplest.

Using Excel, numerically solve the ODE without the small angle approximation using Euler’s Method.

* In a blank spreadsheet, define the following constants: acceleration due to gravity , pendulum length , timestep , initial angle , and initial angular speed .
* Create four columns, and referencing the constants you’ve already inputted into your sheet calculate the angular acceleration , the angular speed , and the angular position from to .
* Plot vs. , vs. and vs. (on the same graph, if you’d like). First, check to see if you’ve coded everything in correctly. What do you expect the behavior of each function to look like? What are the initial values supposed to be? Where should the maximum and minimum values be? Where should the zero values be?  
  Do you see any problems with your graph?
* Now, extend your plotting range out to What do you notice?
* Finally, check to see what happens if we look at larger initial angles. Set , and replot out to . What happens to your graph?
* If anything unusual happened in your graphs above, what could be the cause of it?

Using Python, numerically solve the ODE without the small angle approximation using Euler’s Method.

* Go to your browser and visit . We’ll be looking at Python code using a Jupyter notebook environment which will allow us to run code in blocks. We will use Docker to run the Jupyter notebook through a web browser instead of configuring the software on each individual computer.
* Familiarize yourself with the code by answering the following questions with your table.
  + What Python variables correspond to the variables you used in Excel?
  + What equations are the same? How are they different?
  + Is the variable corresponding with coded in degrees, or radians?
  + What does the function **alpha(x)** do? How is it different from what you defined in Excel?
  + There are two other numerical methods programmed into the template *other* than Euler’s method: the Midpoint method, and Verlet’s method. How are they different than Euler’s method? How are they the same?
* Modify the function **alpha(x)** so that it matches the expression you derived and used in the Excel sheet. If you need to assign a variable a value, add a line in the function that follows **variable = number**. For now, just assign a value of for the length of the pendulum.
* In your own words, describe what the function is doing when you run the following lines of code:

time=ODESolver(omega\_0 = 0, theta\_0 = 45, delta\_t=0.1, n\_iter=300).euler(alpha).time\_  
theta=ODESolver(omega\_0 = 0, theta\_0 = 45, delta\_t=0.1, n\_iter=300).euler(alpha).theta\_

(be as specific as possible!)

* Using the same values as in your Excel activity, plot out the angle of oscillation as a function of time for 30 seconds. On a rough glance, how does your plot compare to the one you found in Excel?
* Let’s remember that in solving this ODE numerically, we’re making an assumption. In Excel, it’s a little annoying to play with this assumption, but in Python it’s pretty easy. Strengthen your assumption made in creating this numerical model and replot your graph from the previous step (making sure you’re still plotting the graph over 30 seconds). What changed?

Compare your large- results from the previous lab to the predictions from one of the numerical models built in this lab.

* Choose three values period measurements from your last lab with initial angles larger than . Inputting the correct pendulum length, and using the Python template, determine the predicted period from the model using Euler’s method. How does the model compare to your measurements?

***Stretch goal:*** What does the analysis look like if you use the other methods programmed into ODESolver()? Is there a method that works better, and why?