PHY224H1F

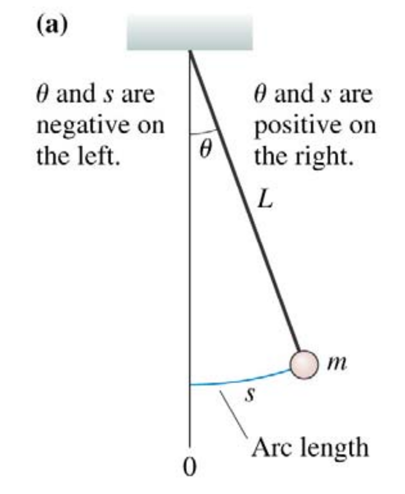
Exercise 1: *The Pendulum (I)*

Pendulums are simple systems, ideal to build quite interesting computational exercises around them
Starting with small angles of oscillation, you will get experimental data and write a program to solve the equation of motion and determine the position (angle) as a function of time. You will plot the graph to visualize the solution. You will also solve the energy equation and plot is. You will have to discuss the output and eventually optimize the code.

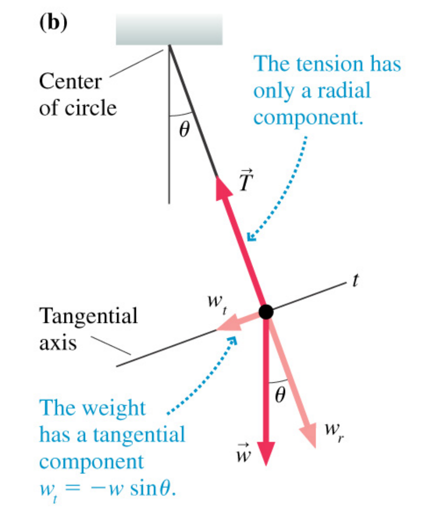
Background knowledge for Exercises 1-3:
Python: lists, arrays, numerical integration, scipy, pylab, leastsq.
Error analysis: chi squared, goodness of the fit
R.Knight: Physics for Scientists and Engineers, 2nd ed., 2008, 14.6: The Pendulum

1.1 Introduction

Physics of pendulum at small angles is based on applying Newton's second law to derive the equation of motion.



The angle from the vertical is θ, the distance from the pivot point is L; $g = 9.8 m/s^2$.



As it can be observed from Fig. a) and b), applying Newton's Law for the tangential force component results into: $-mg\sin\theta = ma_t$

The tangential acceleration is given by: $a_t = \frac{d^2 s}{dt^2}$ where the arc length displacement s is related to the angle θ by: $s = L\theta$. Bringing together all needed quantities, we can write the equation of motion of a pendulum at small angles of oscillation as:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0 \tag{1}$$

➔ **Q1**. Show how equation (1) is demonstrated. Point out the main approximations involved.

The most common method used to finding the solutions to equations of motion is setting up a pair of *coupled ordinary differential equations*.
Given: m = mass, q = coordinate, p = momentum and F = force, we can write:

$$\frac{d\vec{p}}{dt} = \vec{F} \quad and: \quad \frac{d\vec{q}}{dt} = \frac{\vec{p}}{m} \tag{2}$$

Considering $\mathbf{p}_0$ and $\mathbf{q}_0$ to be the initial values, we shall try to find solutions $\mathbf{p}(t)$ and $\mathbf{q}(t)$.

1.2 Numerical methods
To simplify the equation of motion, we approximate the derivatives to:

$$\frac{dy}{dt} \cong \frac{y(t+\Delta t) - y(t)}{\Delta t} \tag{3}$$

Therefore, we can write:

$$\begin{aligned} p(t+\Delta t) &= p(t) + F(q(t))\Delta t \\ q(t+\Delta t) &= q(t) + \frac{p(t)}{m}\Delta t \end{aligned} \tag{4}$$

(4) is a set of *update formulae* that allow us to determine the numerical solution (position and momentum) at Δt, $2\Delta t$, etc., given the starting time $t = t_0$. The numerical solution will approach the actual solution as $\Delta t \rightarrow 0$.
We can re-write (4) in a way closer to our Python code:

$$\begin{aligned} p_{i+1} &= p_i + F_i \Delta t \\ q_{i+1} &= q_i + \frac{p_i}{m}\Delta t \\ i &= 0, 1, 2, \ldots.., \quad t_i = i\Delta t \end{aligned} \tag{5}$$

1.3 Numerical methods and the simple pendulum
The equation of motion for the simple pendulum (1) can be written in the coupled form:

$$\frac{d\theta}{dt} = \omega \ and \ \frac{d\omega}{dt} = -\Omega_0^2\,\theta \tag{6}$$

where: $\Omega_0^2 = g/L$; θ is angle from the vertical; ω is angular velocity.
Our initial conditions will be: $\theta_0 = 5^0$ (small angle approximation) and $\omega_0 = 0$.
Using (6), the numerical approximation can be written as:

$$\begin{aligned} \omega_{i+1} &= \omega_i - \Omega_0^2\,\theta_i\,\Delta t \\ \theta_{i+1} &= \theta_i + \omega_i \Delta t \quad i = 1, 2, 3\ldots. \end{aligned} \tag{7}$$

In equation (7), $\theta_i = q_i$ and ω_i is angular velocity.

This is called the Forward Euler Method, because the right-hand side of (7) is evaluated at the initial point of the iteration step

➔ **Q2**. Prove equations (7).

1.4 Python programming (preliminary)

The basic steps you have to take are the following:

- define constants
- write initial conditions
- use numerical approximation (7) to step forward in time
- loop until done
- plot the graph
- interpret the result

Remember that comments start with # and the Python code is case-sensitive.

1.5 Lab exercise

A pendulum consisting of a steel wire and a bob is attached to a rotational motion sensor, connected to a National Instruments interface. The output of the NI interface is analyzed by a LabView application.

Level the horizontal arm of the stand by using the level provided and the knobs at the base. Measure the length of the pendulum and weigh the bob.

Open the 2nd Yr Lab Files folder from the Desktop. Double click on the RMS.vi shortcut to open the LabView application. Before you begin the data acquisition, get familiar with cursors' positions and the use of the Graph Palette to resize the graph.

Click on 'Acquire' to start the program. In order to stop the acquisition, click on 'Acquire' again.

The STOP button exits the program.

When taking the pendulum out of equilibrium, rotate slowly the motion sensor wheel (do not touch the bob) to avoid wobbling.

Take the pendulum out of equilibrium by ~5^o (this is the upper limit of the linear approximation leading to equation (1)). Use the display to setup the initial angle.

Remember that the rotational motion sensor sets up its starting point where you initially bring the pendulum and the LabView application reads angles in degrees.

Start the acquisition and let the pendulum swing for ~ 10 seconds.

In order to do the Python exercise, you will need to determine Ω_0. Take the average of 5-6 oscillations and use the cursors to obtain the period of oscillation T.

1.6 Python programming (plan)

Get through this program carefully because you may use it as a template for future applications.

- Import the needed modules: pylab will be needed for mathematical libraries and also for submodules (matplotlib.pyplot) used to plotting the graphs:

- Define the needed constants: time step Δt and Ω_0.

- We need to calculate all the values for the plot. You took the experimental data for a total time of 10 seconds, so it is not a bad idea to do the calculation up to t = 10.0 s. Given deltat = 0.01 and t_0 = 0, we'll have 1001 values. The best way to manipulate all the

calculated values is to place them in an array (see the Compwiki tutorial). Arrays are indexed starting at zero. Therefore an array with N elements has an index running from 0 to N-1. Our array will be initially set to zero.
- Known initial conditions for our problem are: initial angular velocity is zero. We have to setup the angle and the momentum initial values. Angles have to be expressed in radians
- Next step will be writing the loop that increases the time by Δt at each iteration. At step i, the corresponding time will be iΔt. The last time value will be 10.0 s
- The last step will be to plot the calculated data. We shall plot the angle on y-axis and time on x-axis. We shall label the axes and put the plot on the screen.

Write and run the program. Save the program on your memory stick

Note: Sometimes, IDLE and Pylab do not work well together. You may need to open few windows in order to keep the program stable.

What happens? Are you confused? Can you interpret the graph?

To get more information, try two other plots:

i) **'angular velocity vs. time' (plot time on x-axis and ω on y-axis)**
ii) **'angular velocity vs. angle' – the phase plot (plot θ on x-axis and ω on y-axis). To do this, you have to change the last lines of code.**

Save each version under a different name.

It could be very useful to analyze the total energy of the pendulum. All first-year textbooks claim that energy of a simple pendulum is conserved:

$E_{tot} = K(\theta,t) + U(\theta)$ where K is kinetic energy ($K = \frac{mv^2}{2} = \frac{1}{2}mL^2\omega^2$) and U is potential energy ($U = mgh \cong \frac{1}{2}mgL\theta^2$). The energy expression you have to use is:

$$E = \frac{1}{2}mL^2\left(\frac{d\theta}{dt}\right)^2 + \frac{1}{2}mgL\,\theta^2 \qquad (8)$$

Modify your program to calculate the energy at each step. Note that energy is not zero at t = 0. You have to include the length of the pendulum (L), the mass (m) and the gravitational constant (g) to the constants section of your program. **Plot:**

iii) 'energy vs. time'

What does the energy plot suggest? Does it explain the strange appearance of the i) and ii) plots?

➔ **Q3**. For a simple pendulum, the phase plot should be an ellipse. Using energy conservation, explain why. Now try to give an explanation for your phase plot.

➔ **Q4 (Bonus question)**. Determine the leading error in our numerical method: perform a Taylor expansion of y(t + Δt) and find the terms we have ignored in Equation (4).

Answer all the questions and submit all the code files and plots to your demonstrator.

Written by Ruxandra Serbanescu (v.6, 2009-2011)