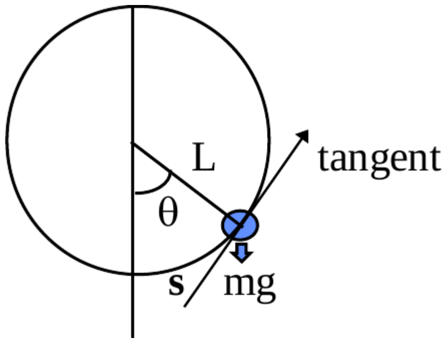
Assignment #2: The Pendulum problem (**total 10 points**), due by 3:00 pm Monday, 13 February 2023

The purpose of this laboratory is to investigate the dynamics of systems which have a nonlinear force law as well as how to use Python to solve coupled ordinary differential equations (ODEs). The equation of motion for the pendulum is found by equating the mass times acceleration of the pendulum bob to the component of the force acting on the bob - its weight - along the direction of motion. The bob moves on the arc of a circle of radius L, and the distance travelled along the tangent to the arc is denoted s (see the figure below). L is also the length of the pendulum string. In the figure, the pendulum bob appears as a blue circle in the figure below and its mass is given by m. g is the acceleration of gravity.



1 Mathematical derivation of equations of motion

From the figure above, the distance travelled along the tangent to the arc traced out by the pendulum motion is

$$s = L\,\theta \tag{1}$$

where θ is the angle subtended by the pendulum and the vertical line cutting the circle in half. θ is zero when the pendulum is at rest. The acceleration of the pendulum bob along the tangent is given by d^2s/dt^2 where t is the time. Since the length of the pendulum, L, is constant, the acceleration can be written as

$$\frac{d^2s}{dt^2} = L\,\frac{d^2\theta}{dt^2} \tag{2}$$

The force which acts on the bob is its weight, mg. The component of the weight along the tangent is the force which accelerates the bob, $-mg\sin\theta$. The component of the weight along the pendulum string, $mg\cos\theta$, (or light rod) holding the bob is balanced by the tension in the string or rod and there is no acceleration in that direction. The equation of motion for the pendulum, according to Newton's second law, is obtained by equating the mass times acceleration to the force acting along the tangent

$$mL\frac{d^2\theta}{dt^2} = -mg\sin\theta \tag{3}$$

which gives

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta \tag{4}$$

For sufficiently small oscillation angles, $\sin\theta \sim \theta$, therefore, the equation above may be approximated by

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\theta \tag{5}$$

This is a "linearised" version of the pendulum equation, since the right hand side is linear in θ. When the pendulum equation of motion is approximated in this way, it is referred to as the linear or simple pendulum. Equation (4) is nonlinear as it carries a $\sin\theta$ dependency on the right hand side. Equation (5) can be solved analytically whereas equation (4) cannot. The solution to equation (5) is

$$\theta(t) = \theta_0 \sin(\nu t + \delta) \tag{6}$$

where ν is the natural frequency of oscillation and is equal to $\sqrt{g/L}$ and θ_0 and δ are the amplitude of motion and the initial phase, which both depend on the initial angle and angular velocity. Note that the natural frequency does not depend on the amplitude of motion for the simple pendulum. This is not so, however, for the nonlinear pendulum where the $\sin\theta$ term in the force law is retained.

To solve a second order differential equation numerically, a new variable is introduced which transforms the second order problem into two first order problems. ω is the angular velocity of the bob. In terms of the angular velocity, the second order differential equation (5) becomes a pair of first order equations

$$\frac{d\theta}{dt} = \omega \tag{7}$$

$$\frac{d\omega}{dt} = -\frac{g}{L}\theta \tag{8}$$

The reason for making this transformation is that it can be simpler to solve first order equations than it is to solve second order equations numerically.

2 The damped, driven, nonlinear pendulum

So far, we have only considered a pendulum bob moving under the force due to gravity, with or without the simplifying approximation of $\sin\theta \sim \theta$. The dynamics of the pendulum become more complex (and interesting) when we keep the nonlinear $\sin\theta$ term in the weight of the pendulum bob and add terms which account for friction and an external driving force. If we add only a friction term, then the pendulum can quickly come to rest. However, adding both friction and driving terms means that the pendulum does not come to rest and complex motion such as period doubling or chaos can be found. The first order equations of motion which include friction and driving forces are

$$\frac{d\theta}{dt} = \omega \tag{9}$$

$$\frac{d\omega}{dt} = f(\theta, \omega, t) \tag{10}$$

$$f(\theta, \omega, t) = -\frac{g}{L}\sin\theta - k\omega + A\cos(\phi t) \tag{11}$$

We will set all our scripts to contain the complete form of the function $f(\theta, \omega, t)$ as in equation (11). We will choose the ratio g/L in the inertial term to be unity. The damping term is $-k\omega$. This resistive force is proportional to the pendulum velocity and is in the opposite sense to the motion. The amplitude of the driving force is A and its functional form is sinusoidal in time and has an angular frequency of ϕ. For the purpose of initial programming implementation, we will set k and A to be zero.

3 Solving ordinary differential equations by numerical methods

To solve these equations, we need appropriate numerical methods that can solve system of ODEs. A way to think about such problem is to choose initial values for $\theta(t = 0)$ and $\omega(t = 0)$ and step both variables forward in time using a Taylor series expansion. The Taylor series for expansion of a function f about the point $x = x_0 + h$ is given by

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + O(h^3) \tag{12}$$

where h is a variable increment. To solve our first order equations of motion for the linear pendulum, we need to expand functions of time following the same procedure as in equation (12):

$$\theta(t + \Delta t) = \theta(t) + \Delta t\,\frac{d\theta(t)}{dt} + O(\Delta t^2) \tag{13}$$

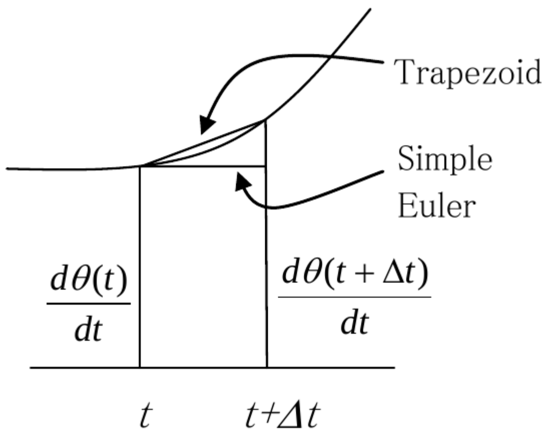
$$\omega(t + \Delta t) = \omega(t) + \Delta t\,\frac{d\omega(t)}{dt} + O(\Delta t^2) \tag{14}$$

Δt here is the time step. Note that terms of order Δt^2 and higher are neglected in this initial approach. In order to get a sufficiently accurate solution to the equations of motion, we must use a sufficiently small time step. A more convenient notation for algorithmic purposes can be adopted for writing the equations of motion. Since time steps can be varied discretely, they can be labeled by an integer subscript n with $t_{n+1} - t_n = \Delta t$. The equations of motion become

$$\theta_{n+1} = \theta_n + \omega_n \Delta t \tag{15}$$

$$\omega_{n+1} = \omega_n + f(\theta_n, \omega_n, t_n)\,\Delta t \tag{16}$$

where $d\theta(t)/dt = \omega(t) \equiv \omega_n$ and $d\omega(t)/dt \equiv f(\theta_n, \omega_n, t)$ as defined in equations (9,10,11). This method is called the simple *Euler* method. It provides a simple introduction to numerical methods for solving ODEs, but we need to secure a more accurate method before tackling the pendulum problem. The trapezoid rule is an improved method for solving ODEs as the figure below shows



The area under the trapezoid in the figure above is

$$(t + \Delta t - t)\left\{\frac{\frac{d\theta(t)}{dt} + \frac{d\theta(t+\Delta t)}{dt}}{2}\right\} \cong \int_t^{t+\Delta t} \frac{d\theta}{dt}\,dt = \theta_{n+1} - \theta_n$$

$$\theta_{n+1} \cong \theta_n + \frac{\Delta t}{2}\left(\frac{d\theta(t)}{dt} + \frac{d\theta(t+\Delta t)}{dt}\right) \tag{17}$$

According to the trapezoid rule, the change in θ over one time step is the time step, Δt, times the average value of the angular velocity at the beginning and the end of the time step. The angular velocity at the end of the time step is not known *a priori* and it must be estimated using a Taylor series:

$$\begin{aligned}\frac{d\theta(t+\Delta t)}{dt} &\cong \frac{d\theta(t)}{dt} + \Delta t\frac{d^2\theta(t)}{dt^2} + O(\Delta t^2)\\ \frac{d\theta(t+\Delta t)}{dt} &\cong \omega_n + \Delta t\, f(\theta_n, \omega_n, t_n) \qquad (18)\end{aligned}$$

Substituting equation (18) into (17),

$$\theta_{n+1} \cong \theta_n + \frac{\Delta t}{2}\left[\omega_n + (\omega_n + \Delta t\, f(\theta_n, \omega_n, t_n))\right] \qquad (19)$$

A similar calculation for the angular velocity gives

$$\begin{aligned}\omega_{n+1} &\cong \omega_n + \frac{\Delta t}{2}\left(\frac{d\omega(t)}{dt} + \frac{d\omega(t+\Delta t)}{dt}\right)\\ \omega_{n+1} &\cong \omega_n + \frac{\Delta t}{2}\left[f(\theta_n, \omega_n, t_n) + f(\theta_{n+1}, \omega_n + \Delta t\, f(\theta_n, \omega_n, t_n), t_{n+1})\right] \qquad (20)\end{aligned}$$

The trapezoid rule equations are

$$\begin{aligned}\theta_{n+1} &= \theta_n + \frac{\Delta t}{2}\omega_n + \frac{\Delta t}{2}\left[\omega_n + \Delta t\, f(\theta_n, \omega_n, t_n))\right] \qquad (21)\\ \omega_{n+1} &= \omega_n + \frac{\Delta t}{2} f(\theta_n, \omega_n, t_n) + \frac{\Delta t}{2} f(\theta_{n+1}, \omega_n + \Delta t\, f(\theta_n, \omega_n, t_n), t_{n+1}) \qquad (22)\end{aligned}$$

A piece of code which implements these equations is given below

```
k1a = dt * omega
k1b = dt * f(theta, omega, t)
k2a = dt * (omega + k1b)
k2b = dt * f(theta + k1a, omega + k1b, t + dt)
theta = theta + (k1a + k2a) / 2
omega = omega + (k1b + k2b ) / 2
t = t + dt
```

4 (Exercise) Python script to solve the linear pendulum equation

(a) Begin a Python script which imports `matplotlib`, `numpy` and `math` and contains a function definition, f, corresponding to equation (11). Remember however that for the linear pendulum case, $\sin\theta \sim \theta$. Note that your function definition for f should have three arguments: θ, ω and t. Add a comment line at the beginning of your file which explains the purpose of the script and add a comment inside the function to describe its purpose.

(b) Set the value of k to 0.0, ϕ to 0.66667 and A to 0.0 for this first exercise. Replace $\sin\theta$ by θ to solve a linear pendulum equation. This will be used to solve the equation of motion for a linear pendulum with no driving force or damping.

(c) Initialise the values of θ, ω, t, and Δt (named `dt` in the code) using the following code after your function definition for f:

```
theta = 0.2
omega = 0.0
t = 0.0
dt = 0.01
```

(d) Add the trapezoid rule equations (21) and (22) by copying the trapezoid piece of code to your script. Be careful that you put brackets in the correct places. Do not use more brackets than are absolutely necessary. Adding unnecessary brackets makes the code hard to read and to debug.

(e) Enclose the trapezoid rule code in a `for` loop where the total number of steps taken is 1000. The time, t, should be incremented by a constant time step, `dt`, in the loop, as already used in the piece of code (line `t = t + dt`).

(f) Add `matplotlib` commands to plot θ and ω as a function of time. θ and ω should appear in the same panel but colour them differently, e.g. θ can appear as a red curve whereas ω can appear as a blue curve. Add statements to your script which place axis labels and a title on your graph. Set the range on the y-axis to $[-\pi, +\pi]$.

(g) Make plots of $\theta \times t$ and $\omega \times t$ for the following sets of initial conditions (see table below).

$\theta(t=0)$	$\omega(t=0)$
1.0	0.0
3.1	0.0
0.0	1.0

(h) **IMPORTANT:** Upload the `.ipynb` files you generated for item 4(g) in addition to your report **(0.5 points)**.

(i) **IMPORTANT:** Add the plot for $\theta(t=0) = 3.1$ and $\omega(t=0) = 0.0$ in your report and explain what the plot shows. Note that you will have to compare this result for the linear pendulum with the nonlinear pendulum case [see item 5(d)]. **(0.5 points)**

5 (Exercise) Python script to solve the nonlinear pendulum equation

(a) Copy your script for the linear pendulum to another file which you will use to simulate the nonlinear pendulum. Choose a file name which clearly indicates what the purpose of the script is.

(b) Put a statement at the beginning of the script which describes what the script does.

(c) Change the function f to describe the nonlinear pendulum or define a new function named `f_nonlin` that will contain equation (11) in which `theta` is replaced by `math.sin(theta)` or `np.sin(theta)` (in case you are working with `numpy` programming).

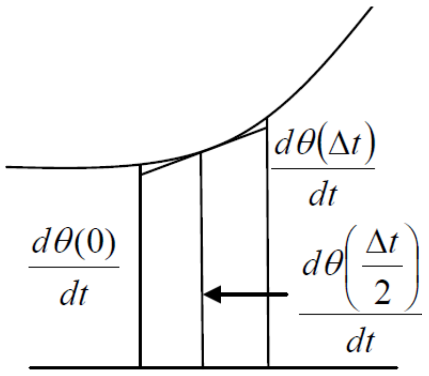
(d) Your script should run simulations for both linear and nonlinear pendulums using the initial conditions on the table above. Plot graphs in which $\theta \times t$ and $\omega \times t$ are compared for linear and nonlinear pendulums.

(e) **IMPORTANT:** Upload the `.ipynb` files you generated for item 5(d) in addition to your report **(0.5 points)**.

(f) **IMPORTANT:** Include the plot for $\theta(t=0) = 3.1$ and $\omega(t=0) = 0.0$ for the nonlinear pendulum case in your report and compare it with the linear case studied in the previous section. **(0.5 points)**

6 Change the numerical integration algorithm

The second order Runge-Kutta algorithm for solving ODEs is similar to the trapezoid rule except that the Taylor series expansion is carried out about the midpoint of the time step, rather than at the beginning (see figure below). This new choice has the advantage that it is accurate to the order of Δt^2 rather than Δt.


$\frac{d\theta(\Delta t)}{dt}$
$\frac{d\theta(0)}{dt}$
$\frac{d\theta\left(\frac{\Delta t}{2}\right)}{dt}$

The area under the trapezoid which approximates the function $d\theta(t)/dt$ is the time step, Δt, times the average value of $d\theta(t)/dt$ at the beginning and end of the time step. When a Taylor expansion of $d\theta(t)/dt$ is carried out about the midpoint of the time step, the result is

$$\frac{d\theta(t)}{dt} = \frac{d\theta(\Delta t/2)}{dt} + \frac{d^2\theta(\Delta t/2)}{dt^2}\left(t - \frac{\Delta t}{2}\right) + O\left(\frac{\Delta t^2}{2}\right) \tag{23}$$

$$\int_0^{\Delta t} \frac{d\theta(t)}{dt}dt \cong \frac{d\theta(\Delta t/2)}{dt}\Delta t + \frac{d^2\theta(\Delta t/2)}{dt^2}\int_0^{\Delta t}\left(t - \frac{\Delta t}{2}\right)dt = \frac{d\theta(\Delta t/2)}{dt}\Delta t + 0 \tag{24}$$

From the derivation above, the rule to update θ_n in an algorithmic notation is

$$\theta_{n+1} - \theta_n \cong \omega_{n+1/2}\,\Delta t + O\left(\frac{\Delta t^2}{2}\right) \tag{25}$$

$$\omega_{n+1/2} \cong \omega_n + f(\theta_n, \omega_n, t_n)\frac{\Delta t}{2} + O\left(\frac{\Delta t^2}{2}\right) \tag{26}$$

Substituting equation (26) into (25),

$$\theta_{n+1} \cong \theta_n + \left(\omega_n + f(\theta_n, \omega_n, t_n)\frac{\Delta t}{2}\right)\Delta t \tag{27}$$

The rule to update ω_n is obtained similarly,

$$\omega_{n+1} - \omega_n \cong f(\theta_{n+1/2}, \omega_{n+1/2}, t_{n+1/2})\,\Delta t + O\left(\frac{\Delta t^2}{2}\right) \tag{28}$$

$$\omega_{n+1} \cong \omega_n + f(\theta_{n+1/2}, \omega_{n+1/2}, t_{n+1/2})\,\Delta t \tag{29}$$

$$\omega_{n+1/2} \cong \omega_n + f(\theta_n, \omega_n, t_n)\frac{\Delta t}{2} \tag{30}$$

$$\theta_{n+1/2} \cong \theta_n + \omega_n\frac{\Delta t}{2} \tag{31}$$

The method described above is known as the second-order Runga-Kutta method or midpoint method. It improves the Euler method by adding a midpoint in the step which increases the accuracy by one order. The fourth order Runge-Kutta method is based on a similar principle with the addition that the method works with four slopes calculated within a $[t, t + \Delta t]$ interval: two slopes are taken at the midpoint, as well as the end points of the interval, and those are used in a weighted average scheme to determine the new values in time of θ and ω. The rule to update θ and ω for the fourth order Runge-Kutta is not derived here since it is a very extended derivation. Yet, we provide a suggested piece of code for this method as below

```
k1a = dt * omega
k1b = dt * f(theta, omega, t)
k2a = dt * (omega + k1b/2)
k2b = dt * f(theta + k1a/2, omega + k1b/2, t + dt/2)
k3a = dt * (omega + k2b/2)
k3b = dt * f(theta + k2a/2, omega + k2b/2, t + dt/2)
k4a = dt * (omega + k3b)
k4b = dt * f(theta + k3a, omega + k3b, t + dt)

theta = theta + (k1a + 2 * k2a + 2 * k3a + k4a) / 6
omega = omega + (k1b + 2 * k2b + 2 * k3b + k4b) / 6
```

7 (Exercise) Implementing Runge-Kutta method

(a) Copy your script for the nonlinear pendulum to another file in which you will use to simulate the nonlinear pendulum using the fourth order Runge-Kutta algorithm. Choose a file name which clearly indicates what the purpose of the script is.

(b) Put a comment statement at the beginning of the script which describes what the script does.

(c) Add the suggested Runge-Kutta piece of code to your script. Place it within a `for` loop so that the nonlinear pendulum equations are solved by this method instead. Make sure all variables and terms are entered correctly.

(d) **IMPORTANT:** Test your new algorithm by running the calculation with both the trapezoid rule and the Runge-Kutta algorithms using a single script for the initial values of $\theta(t = 0) = 3.0$ and $\omega(t = 0) = 0.0$. The script should plot $\theta \times t$ for both methods on one graph. Include this graph in your report. Remember to label axes, and to add legends identifying each curve to its corresponding method. Compare the methods in your report. **(1 point)**

8 (Exercise) The damped nonlinear pendulum

In this exercise, you will simulate the behaviour of a damped pendulum, i.e. one which is subject to friction forces as well as the force of its own weight. The simplest assumption about forces such as air resistance is that the damping force depends linearly on velocity and acts to oppose the motion. **Note that at this stage you will be able to choose one of the methods learned in this assignment: trapezoid rule or fourth order Runge-Kutta. Which one do you think is the best to proceed?**

(a) Re-use the nonlinear pendulum function defined in past codes and simply change the value of k in the function from 0.0 to 0.5 so that damping (i.e. friction) is turned on.

(b) **IMPORTANT:** Use the initial values of $\theta(t = 0) = 3.0$ and $\omega(t = 0) = 0.0$ and run the script for the case of the damped nonlinear pendulum. Include the generated plot in your report. Explain the shape of the curve for $\theta(t)$ and $\omega(t)$ in your report. **(0.5 points)**

9 (Exercise) The damped-driven nonlinear pendulum

In this exercise, a periodic external force is added to the force acting on the pendulum. This is like the force pushing a swing, except that the force acts continuously and it takes positive and negative values. In this case the motion may be periodic or not. The driving force has a frequency, ϕ, and a period $2\pi/\phi$. This sets a base period for the pendulum. If the pendulum motion remains periodic, the motion must have the period of the driving force or an integer multiple of the driving force. Otherwise, the pendulum motion is non-periodic - in this case its motion is said to be chaotic!

When a parameter such as the amplitude of the driving force, A, is changed, the motion may remain periodic but the actual period may change. In the pendulum, period doubling of the base period occurs many times before non-periodic behaviour (chaotic) is observed. Evidently the motion of the damped, driven nonlinear pendulum is much more complex than even the damped pendulum. A convenient way of representing this motion is to plot the angle $\theta(t)$ versus the angular velocity $\omega(t)$. This is called a phase space plot (phase portrait). In the units we have chosen ($g/L = 1$), the phase portrait for the linear pendulum is a circle (check yourself!). A pendulum swing with a larger amplitude has a phase portrait which is a larger circle.

The phase portrait for the damped, driven, nonlinear pendulum in a periodic motion is a closed loop. This loop is called a limit cycle and represents the behaviour of the pendulum after it has reached steady state/periodic motion. For nearly every set of initial conditions, the initial motion of the pendulum is not on the limit cycle. The pendulum motion, before steady-state/periodic-motion is reached, is called transient motion. In this exercise, you will omit the transient motion from the phase space graph by plotting points on the phase portrait only after steady state motion has been reached. The simplest way to determine when steady state motion has been reached is by plotting the phase portrait. For the first set of parameters you are given for the damped, driven nonlinear pendulum ($A = 0.9$, $k = 0.5$, $\phi = 0.66667$), the phase portrait is a single loop. Any 'tail' on the loop is the transient motion. If your phase portrait has a tail you need to begin plotting points at a higher iteration number.

(a) Copy your script for the nonlinear pendulum to a new file (name it accordingly) and simply change the value of A in the function from 0.0 to 0.9 so that a sinusoidal driving force is turned on.

(b) Initialise a variable named `transient = 5000` in the script before the integration loop starts. This variable establishes the threshold for the transient motion which we wish to eliminate from the phase space graph. Your script should plot a graph of $\theta(t) \times \omega(t)$ only for $t > t[\texttt{transient}]$, being $t[i]$ an array that stores time values sequentially on each of its components i. If `transient = 5000` does not eliminate the transient motion, increase its value.

(c) Run the script for the following parameter values and generate a phase portrait for each case. The pendulum is nonlinear and damping should be switched on ($k = 0.5$).

$$A = 0.90, 1.07, 1.35, 1.47, 1.5 \text{ and fixed } \phi = 0.66667$$

(d) For some of these parameter values, the motion has a circular profile rather than swinging back and forth, so that θ increases indefinitely. However, a pendulum angle of θ cannot be distinguished from an angle of $\theta + 2\pi$. θ is therefore restricted to lie in the range $[-\pi, +\pi]$ in this case. Use `if` statements to restrict the values of θ within the appropriated range. In some cases, you may need to adjust the $[-\pi, \pi]$ range to $[-\pi + Q, \pi + Q]$, where Q is a shift parameter, to obtain a limit cycle as a single piece and not cut down in two at $+\pi$ or $-\pi$.

(e) **IMPORTANT:** Include the phase portraits for $A = 1.07$ and $A = 1.47$ in your report. Explain what each phase portrait says about the pendulum motion. **(1 point)**

(f) **IMPORTANT:** Attach ALL `.ipynb` files you generated for item 9(d) in addition to your report **(0.5 points)**.

10 Report writing guide (5 points)

After generating your codes and results for the exercises above, you will summarize them in the form of a written report. Please, follow the instructions provided in the report template located in the sub-module of the assignment to write your report. All margins, font style/size, line spacing, and page limits set on the template need to be respected. **We will deduct a significant amount of points from reports that do not align with the template provided or we will not be able to assess the report, meaning we will have to deduct -5/10 points from the total marks.**

There is a `docx` file (generated in MS Word editor) available in the sub-module of the assignment that can, in principle, be used and exported to other text editors. However, some of its pre-defined macros may not be read correctly by other editors. Moreover, inserting text on the editable `docx` file may shift or duplicate some of the original headers. Make sure your written content obeys the format set in the read-only `PHYS 381 Assignment 2 Template.pdf` file. In case you have problems with the `docx` file provided, a suggestion is to start a blank document in the text editor of your choice and set the layout as below:

- Page size: 8.5” $\times$ 11” (letter);
- Margins: 2.54 cm (top, bottom, left, right);
- Font type and size: Times New Roman, 12 pt, non-italic, non-bold;
- Line spacing: between 1.0 and 1.15 pt;
- Figures must be numbered and contain captions. Axes must carry names and their quantities must be expressed with the proper units whenever required. Do not insert too large figures just to cover space. Choose figure sizes typically seen in textbooks and scientific manuscripts.
- Follow the exact page configuration and order as described in the report template, e.g., title page on page 1, abstract on page 2, introduction on page 3, etc. The title page needs to contain the exact same information as in the report template.

10.1 Submission

The due date for the submission of your report is depicted on the first page of the assignment. Once you conclude the writing of your report in accordance with the template provided, generate a `pdf` file and name it as `report_assignment_2.pdf` (since this is report #2). You will upload the `pdf` file of your report plus your Notebook `.ipynb` files requested in the assignment to the Gradescope platform. You can upload multiple files to the Gradescope platform and you can resubmit your work until the due date. We will test-run all your submitted codes to test for errors. We will also check if the results/figures presented in your report match the output of your codes.

For Groups: If you worked in a group, remember to include your group members in Gradescope after you submit all the files on behalf of the group. You will see an option "Add Group Member" after you upload all files to Gradescope. Students can work in groups of up to 4 members.

Autograder option: Ignore the autograder results that may appear after you upload files to Gradescope. There is no autograder configured for this assignment since everything will be evaluated manually by the PHYS 381 Team.

Please, remember to add comments in all your submitted Notebook codes! Pure code lines without explanatory comments or written Markdown cells will have reduced marks.

Standard `.py` Python codes are not accepted in this submission. Work only with Notebook environments. If any member of the group is experiencing problems with their coding environment, please, contact the instructor immediately.

ONLY `pdf` (for the written report) AND `.ipynb` (for codes) FILES ARE ACCEPTED.

*****