PH4419/PAP723: Computational Physics

Assignment 1

INSTRUCTIONS

- *Submission:* Assignment is due: **1st March 2019**.

- *Submission format:* All submission are to be done through NTULearn. There are 2 (two) questions in this assignment. Answer each question in a separate **python .py** file. Label your files using the following convention:

  Matric Number_Assignment 1_Question X.py

  For example: U1234567A_Assignment 1_Question 0.py

  Each file should begin with the lines

```
from scipy import *
import matplotlib.pyplot as plt
```

  Any other Python libraries used must be listed directly below these two lines.

- *Grading:* For full credit, code must follow good programming style. Key code blocks must be clearly commented. Function names must be properly named for ease of understanding your codes. The program structure should be modular; avoid unnecessary code duplication, and group numerical constant definitions neatly together. The output of your programs should be clear. Each generated plot should be well labeled, and if multiple curves are shown, they should be well-labeled.

- *PAP723:* Parts of questions marked with a $(*)$ are for students taking the course PAP723, they must be attempted to receive credit. Undergraduate students who can produce a working code with correct output for these parts will receive 1 (one) bonus mark per question.

- *Documentation:* Whenever the assignment mentions a Scipy function you might need, consult the Scipy/Numpy online documentation to learn what the function does and how to use it.

- *Techniques:* The core techniques you will need to apply in this problem set are: (i) ODE Solver, (ii) Finite Difference method for ODE, (iii) Shooting Method and (iv) Eigenvalue Problem.

00. Classical Oscillator [9 marks]

A damped harmonic oscillator is described by the following equation:

$$F = -kx - D\dot{x} + f(t)$$

where,
F is the net force on a mass m,
k is a positive real constant called the spring constant,
D is a real constant called the damping constant, and
$f(t)$ is an external friving force on the mass m, which may vary with time t.

(a) **Phase Portrait and Trajectories.** Consider the homogeneous equation of the above differential equation:

$$m\ddot{x} + D\dot{x} + kx = 0$$

We denote $\vec{u}$ as the following vector

$$\vec{u}(t) = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \Rightarrow \frac{d\vec{u}}{dt} = \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix}$$

(i) **[2 marks]** Write a function that computes the time derivative of $x(t)$ and $\dot{x}(t)$ at t:

`def oscillator(u, t):`	
Input	
`u`	A tuple of real numbers specifying the position and velocity of the harmonic oscillator at time t. The element of the tuple describes the position $x(t)$ and the second element describes the velocity $v(t)$ of the oscillator.
`t`	A real number specifying the time. Note that this is just a placeholder variable in the case of **autonomous equations**
Output	
`udot`	A tuple of real numbers specifying the position and velocity of the harmonic oscillator at time t. The first element of the tuple describes the velocity $\dot{x}(t)$ and the second element describes the acceleration $\dot{v}(t)$ of the oscillator. The shape and size of `udot` must be the same as `u`.

(ii) **[4 marks]** For this part of the question, take the following initial conditions:

$$\vec{u}(0) = \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix}$$

Write a function that uses the function in (i) to plot properties of an underdamped oscillator:

`def oscillator_plot(t, m, k, D, u0):`	
Input	
`t`	An array describing time, in this problem, you may decide the span on t that shows the properties of the oscillation.
`t, m, k, D`	As defined above.
`u0`	A tuple $[x_0, \dot{x}_0]$ given by the initial conditions.

There are no return values for this function, but the function should generate two plots:

(1) A phase portrait showing how the velocity varies with position with a trajectory describing the oscillator with the given initial conditions.

(2) A $x(t)$ versus t graph, with the given initial condition.

Briefly describe (as comments in your code) the relationship between (1) and (2). Note: You should not solve the equation analytically and plot the resulting function. Part (a) is meant to be a simple exercise to help you for the next part of this problem. [Hint: You may use the plotting functions `quiver` and `meshgrid` to plot the phase portrait and use `odeint` to solve the differential equation numerically.]

(b) **Double Potential.** In classical mechanics, a conservative force can be represented by a potential energy function

$$\vec{F} = -\nabla U$$

in the simplified case where $\vec{F}$ depends only on x, we can write

$$\vec{F}(x) = -\frac{dU}{dx}\hat{x}$$

Furthermore, the total energy of a system ($m = 1$) is

$$E = \frac{1}{2}\dot{x}^2 + U(x)$$

where, $U(x)$ is the potential energy. A particle is set in motion in a double potential well. The potential energy of a double well is

$$U(x) = -\frac{1}{2}x^2 + \frac{1}{4}x^4$$

(i) **[3 marks]** Write a function `double_potential_plot` that does the following:

(1) Plot the phase portrait of the double potential well system.

(2) Plot at least 3 **closed** trajectories and briefly describe (in code comments) how different initial conditions affect the harmonic motion of the particle in this potential well.

[Hint: The code for the phase portrait is similar to the one you have for (a)]

01. 1D Schrödinger Equation [11+5 marks]

Consider the 1D Time Independent Schrödinger Equation

$$\mathcal{H}\psi(x) = \Big[-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\Big]\psi(x) = E\psi(x) \tag{1}$$

where $\mathcal{H}$ is the Hamiltonian. If we were to use natural units, where the *energy* is measured in units of $\hbar\omega$ and *length* is measured in units of $\sqrt{\hbar/m\omega}$, then the Schrödinger Equation becomes

$$\mathcal{H}\psi(x) = \Big[-\frac{1}{2}\frac{d^2}{dx^2} + \nu(x)\Big]\psi(x) = \epsilon\psi(x) \tag{2}$$

(a) **Predicting Energy Eigenvales for the Finite Potential Well.** Recall the Finite Potential Well (FPW) defined by the following potential:

$$\nu(x) = \begin{cases} 0 & -1/2 \le x \le 1/2 \\ \nu_0 & \text{otherwise} \end{cases} \tag{3}$$

In this problem, we shall set $\nu_0 = 50$.

(i) **[3 marks]** We now want to write a function that plots the first 4 (four) wavefunctions for the Finite Potential Well for $\nu_0 = 50$.

`def FPW_demo(psi0, nu0, x, epsilon):`	
Input	
`psi0`	A tuple. Elements `psi0[0]` and `psi0[1]` are the initial conditions ψ_0 and ψ_1.
`nu0`	A positive real number. The height of the potential well given by $\nu(x)$.
`x`	An array of length N in the domain $x \in [x_{min}, x_{max}]$. The domain of x is free for you to choose.
`epsilon`	A positive real number. This is the energy eigenvalue of the Potential Well.

This function has no output value. But it should plot the probability density $|\psi_n(x)|^2$ versus x corresponding to the first 4 energy eigenvalues (i.e $n = 1, 2, 3, 4$). You have to trial-an-error for the various values of ϵ, hence the name, **shooting method**. Plot each graph on a separate subplot. [Note: You should ensure that $|\psi(x)|^2$ is normalized.]

(ii) **[1 mark]** Set $\epsilon > 50$, briefly explain your result.

(b) **Finding Energy Eigenvales for the Finite Potential Well.** Now, it would be ridiculous for you to perform the shooting method to find the energies for the different eigenstates, especially if you have no idea how many there would be and what numbers to guess. Therefore, the shooting method is often complemented with first finding the energy eigenvalues.

Our goal this time is to solve the Schrödinger Equation for the unknown eigenfunction ψ corresponding to the eigenvalues E. Recall that ψ can be written as the linear combination of a set of orthonormal basis ψ_n

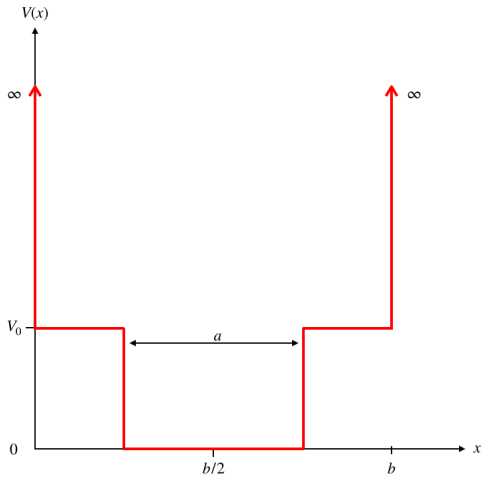
$$\psi = \sum_{n=1}^{\infty} c_n \psi_n \quad c_n \in \mathbb{C} \tag{4}$$

Which can be (easily shown) implies that the Schrödinger Equation will take the form

$$\sum_{n=1}^{\infty} H_{mn} c_n = E c_m \tag{5}$$

To find a numeric way of finding the solution to this Schrödinger Equation, we, consider the following potential

$$V(x) = \begin{cases} 0 & |x - b/2| \leq a/2 \\ \infty & x < 0 \text{ and } x > b \\ V_0 & \text{otherwise} \end{cases} \tag{6}$$



Again, to get a similar result as above, we take $a = 1$ and $V_0 = 50$ (in natural units); we will also take $b = 4$ in our calculation. Here, we have placed the finite well within the infinite potential well because we know exactly what the eigenstates and energy eigenvalues of the infinite potential well is.

$$\psi_n(x) = \sqrt{\frac{2}{b}} \sin\left(\frac{n\pi x}{b}\right) \tag{7}$$

$$E_n = \frac{\pi^2 n^2}{2b^2} \tag{8}$$

We hope that the energy eigenfunctions for the infinite square well is general enough to contain all the eigenfunctions ψ that we are interested to find. In fact, We are now able to express the Hamiltonian as a sum of 2 pieces

$$H = H_0 + \Delta V(x)$$

Here, H_0 is the Hamiltonian whose eigenfunctions belong to that of the infinite potential well and whatever is left is $\Delta V(x)$, is simply, in our case, $V(x)$. Hence, we can express the Hamiltonian and elements of a $N \times N$ matrix,

$$H_{mn} = E_n \delta_{mn} + \int_0^b \psi_m^*(x) V(x) \psi_n(x)\, dx \tag{9}$$

and

$$\begin{pmatrix} H_{11} & H_{12} & \dots \\ H_{21} & \ddots & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix} = E \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix} \tag{10}$$

(i) **[3 marks]** Write a function that takes two values m and n and calculates the integral in Eq. (9).

`def Potential(x, m, n):`	
Input	
`x`	An array of real numbers. The domain of the function.
`m, n`	Integers describing the eigenstate $\psi_m^*(x)$ and $\psi_n(x)$.
Output	
`V_mn`	A real number. This is the solution to the integral in Eq. (9)

(ii) **[4 marks]** Write a function that solves for the eigenvalue of a $N \times N$ matrix, A, and return the lowest k eigenvalues.

`def eigenvalue_solver(A, N, k):`	
Input	
`A`	An $N \times N$ matrix. This is matrix that we want to solve for the eigenvalues.
`N`	An integer. This is the size of a square matrix $N \times N$.
`k`	An integer.
Output	
`energy_k`	An array containing k real numbers. This is k lowest energy eigenvalues

Print the lowest 4 eigenvalues (i.e set `k=4`). Briefly describe how you would make the numeric eigenvalues more accurate.

(iii) **[5 marks]** $(*)$ Modify your code in (b)(ii) to print the c_n's of the wavefunction ψ corresponding to the largest allowable bound state energy eigenvalue, hence, plot the wavefunction corresponding to the largest allowable bound state energy. Briefly describe the difference between the wavefunction you obtain here and the one obtained in (a)(i).