PHZ4151C: Exercise 8

Computational Physics Lab
Due Friday April 5

For each problem you will write one or more python programs. These programs should follow the Python 2.7.x coding and formatting conventions outlined for our course. You must hand in copies of the programs and outputs as prescribed in each problem.

In addition, you must submit all Python programs as an archive tgz file via email to phz4151c@hadron.physics.fsu.edu. Place copies of only your Python programs in a directory called <last_name>-exercise8/where <last_name> is your last name and use similar command as provided in earlier exercises for submission.

1. Inheritance of Vector3D: Using the class **Vector2D** defined in the lecture notes, finish the implementation of the inherited **class Vector3D(Vector2D)** also presented in the lecture notes. Define additional **Vector3D** member functions: **mag()** which returns the vector magnitude, **r()** which return the value of the polar **r** value, **cos_theta()** which returns the cos angle that the vector makes with the z-axis, **phi()** which returns the projected angle in the x-y plane, **__add__()** which preforms vector addition as "A + B", **__sub__()** which preforms vector subtraction as "A - B", **__mul__()** which performs a vector dot product as "A * B", **__truediv__()** which we define to perform the vector cross product as "A / B", and **print()** which print outs the values from the member functions: x(), y(), z(), r(), theta(), and phi().

Write a program which creates a Vector3D object with (x, y, z) values of (2, 3, 4). Print the vector. Create a second Vector3D object with (x, y, z) values of (3, 4, 5) and have it print out its values. Next print out the dot product of the two vectors and the resulting vector for: the vector addition of the two vectors, the subtraction of the first vector from the second vector, and cross product of the first vector with the second vector.

For full credit turn in a printout of your finished program together with your output.

2. Differential Calculator: Create a differential calculator object class which uses the central difference to calculate the derivative of a given function f(x) at a given value of x. The **__init__** constructor should include arguments for the referencing function **func(x)** and central difference step size **h** which should have a default value of 1E-08. Define a __call__(self, x) member function which implements the central difference of the function at the value x. Within the main part of your program, define a function f(x) that returns the value $1 + \frac{1}{2} \tanh 2x$. Declare a differential calculator object for the numerical derivative of f(x). For comparison, define a function for the analytic derivative of f(x) and make a graph with your numerical result and the analytic results on the same plot for the range $-2 \le x \le 2$.

For full credit turn in a printout of your finished program and your graph.

3. Difference Errors: Even when we can find the value of f (x) for any value of x the forward difference can still be more accurate than the central difference for sufficiently large h. For what values of h will the approximation error on the forward difference be smaller than on the central difference? Hint: See discussion on errors in lecture notes.

For full credit turn in the derivation for the values of h which satisfy the above requirements.

4. Radioactive Decays: This problem is a straightforward application of a forward difference to numerically solve a differential equation. Given N(t) radioactive nuclei, they will decay randomly according to the following equation:

$$\frac{dN}{dt} = \frac{-N(t)}{\tau}$$

By replacing `dN/dt` with the forward difference `(N(t+h)-N(t))/h`, one obtains a numerical solution for `N(t+h)`. Given initial conditions (i.e. `N(t=0)`), one can obtain numerical values of `N(t)` for all later times. This differential equation can be solved analytically as `N(t) = N(0) exp(-t/τ)`, where `N(0)` is the initial number (or fraction) of radioactive nuclei. This solution allows one to compare our numerical results with the exact solution.

a) Write a program to numerically solve for the time dependence N(t) from 0.0 s ≤ t ≤ 15.0 s assuming N(0) = 100% and τ = 2 s. Do this for the following values of h: 1.0 s, 0.1 s, and 0.01s. Graphically compare your results to the exact solution as a function of time. Also study the accuracy by plotting the fractional error vs. t. Overlay on one figure the graphs for the different values of h.

b) Using h = 0.01 s, write another program to plot the time dependence of N(t) for τ = 5.0s, 3.0s, 1.0s, 0.1s, and 0.01 s.

- Briefly discuss the accuracy of your results. If there are any problems explain.

c) Consider a system of a parent nucleus, ***P***, and a daughter nucleus, ***D*** both radioactive. The coupled equations which describe their decays are as follows:

$$\frac{dN_P}{dt} = \frac{-N_P(t)}{\tau_P}$$

$$\frac{dN_D}{dt} = \frac{N_P(t)}{\tau_P} - \frac{N_D(t)}{\tau_D}$$

Write a program to numerically solve the above coupled equations and plot the time dependence of N_P and N_D for τ_P = 2.0s and τ_D = 0.02s, 2.0s and 200.0 s. Assume you start out with 100% parent nuclei and no daughter nuclei. Graph all results overlaid on one figure.

- Qualitatively explain the behavior of N_D for situations in which $\tau_P \gg \tau_D$, $\tau_P \approx \tau_D$, and $\tau_P \ll \tau_D$.

For full credit turn in a printout of your three finished programs, your graphs from parts a, b and c, and the discussions for part b & c.