Physics 305 – Computational Physics, Fall 2020
Term Project
Full project submission Due Date: Tuesday December 15, 5pm
Presentation Phase: November 30 - December 11

The program in your term project can be either submitted as a python program or ipython notebook, where the latter is preferred. The program, an explanation of what the program does, along with answers to **all** questions asked should be uploaded to d2l.

You are expected to write a term paper (in word or Latex) on your project that discusses the problem you are trying to solve, the basic equations that govern the problem, includes plots that show the solutions, and describes the solution and the numerical method involved. In addition, you must demonstrate that your solution is correct by showing that the code converges at the expected order. If your code does not converge at the expected order you should try to identify potential reasons for why this is the case. You are expected to work on your term project by yourself..

Your term project will receive full credit **only** if: (a) the program runs successfully without errors using python 3, (b) the programs have explanatory comments and variable names that identify with the problem equations you are trying to solve, (c) give the correct output, and (d) demonstrate the validity of the solution through convergence plots. No credit will be given to late term projects.

The term paper is as important as the code (50% of the term project credit will go to the code and the other 50% to the paper). Answers to the questions and analysis requested below should be elaborated in the report. Plots should be clearly labeled and be properly described in the report, and not just shown. You will need to explain what each and every plot demonstrates. A polished paper written in word or LaTex (preferred, e.g. please try overleaf) is expected to get full credit.

Note: Before you present results from numerical integrations that answer the questions in the project, it is critical to *first* perform the convergence tests for one case, and to estimate errors. This will tell you how small a step size is necessary for accurate solutions. Only after errors are estimated, does it make sense to run your code for producing results that help you learn more about the system you study.

I. SOLITON SOLUTIONS OF THE KDV EQUATION

You will be examining properties of the Korteweg–de Vries (KdV) equation. The KdV equation is a mathematical model of waves on shallow water surfaces. It is a non-linear partial differential equation, solutions of which can be derived by turning it to an ordinary differential equation (ODE). In this case, the solutions are solitons, i.e., profiles that maintains their shape as they travel. The precise form of the KdV equation is

$$\frac{\partial \phi}{\partial t} + \frac{\partial^2 \phi}{\partial x^3} - 6\phi \frac{\partial \phi}{\partial t} = 0. \tag{1}$$

This equation can be cast to an ODE by simply seeking a wave-like solution of the form $\phi(x,t) = f(x - ct - a) = f(X)$. If we plug this ansatz in Eq. (1) we arrive at the ODE

$$\frac{d^3 f}{dX^3} = (c + 6f)\frac{df}{dX}. \tag{2}$$

You will be solving this equation. Notice that the transformation $X \Rightarrow -X$ leaves the equation invariant. Therefore, the function $f(x)$ must be even, i.e., $f(-X) = f(X)$. Thus, if we start with initial conditions at $X = 0$ and integrate forward we can always find the solution for negative X by using $f(-X) = f(X)$.

The equation has an integral (constant) of the motion,

$$C = \frac{d^2 f}{dx^2} - (c + 3f)f, \tag{3}$$

which is determined by the initial conditions. The constant C will allow you to validate the quality of the numerical integration of Eq. (2) (think of it as conservation of total energy). For simplicity you will be setting $c = 1$. To solve equation (2), you will need to cast it to first-order form. Show your work in the report. You will also need to specify initial conditions. These are $f(0) = -1/2$, $df/dX|_{X=0} = 0$, $d^2 f/dX^2|_{X=0} = 1/4$. You will be integrating the equation forwards (for $X > 0$).

1. Use RK4 to integrate numerically the first-order reduction of the ODE (2) from $t = 0$ forward $X_n = n * h, n = 0, 1, \ldots N$ and for sufficiently large X (i.e. large N). You will need to be plotting f vs X to see if the solution is settling to determine when to stop the integration.

   Use your judgement as to how small a step size you need to solve this system accurately. If you cannot figure this out from pure thought, experiment with different step sizes and use $\delta C = |(C(t) - C(t = 0))/C(t = 0)|$ to determine the accuracy. If δC is smaller than 10^{-3} for all integration times, then you have decent accuracy.

   Show in a plot your solution for $f(X)$, for $X \in [-20, 20]$.

2. Recall that $\phi(x, t) = f(x - t)$. Plot $\phi(x, t)$ for $x \in [-10, 10]$ at $t = 0$, $t = 1$, $t = 2$, $t = 3$, $t = 4$, $t = 5$. Note that you may need to perform interpolation using your data for $f(X)$ to achieve this this (use a third-order Lagrange interpolation to achieve this). What happens to $\phi(x, t)$ as t increases? This is exactly what a soliton looks like.

3. **Convergence:** Demonstrate that as you decrease the step size h, $\delta C(X = 5)$ converges to 0. Make a plot to determine the order of convergence of δC, and discuss why or why not this matches your expectation.

4. **Self-convergence**: Use a number of step sizes and make a plot to demonstrate that the code solution for f at $X = 5$ the final time self-converges for fixed initial conditions. Does the order of convergence match your expectation? If not, try to explain why.

5. Using the order of convergence you determined, employ Richardson extrapolation to determine an error for the solution for $f(X)$ at a time of your choosing.