PHY407 Computational Lab 3
Gaussian Quadrature and Numerical Differentiation

General Advice

- Read this document and do its suggested readings to help with the pre-labs and labs.
  - Ask questions if you don't understand something in this background material - maybe we can explain things better, or maybe there are typos.
- Carefully check what you are supposed to hand in for grading in the section "Lab Instructions".
- Whether or not you are asked to hand in pseudocode, you NEED to strategize and pseudocode BEFORE you start coding. Writing code should be your last step, not your first step.
- Test your code as you go, NOT when it is finished. The easiest way to test code is with 'print' statements. Print out values that you set or calculate to make sure they are what you think they are.

Computational Background

- **Gaussian quadrature (for Q1 and Q2):** Section 5.6 of Newman provides a nice introduction to the theory of Gaussian quadrature.[1] To summarize:
  - Gaussian quadrature tweaks the Trapezoidal and Simpson Rule type approach to integration in two important ways.
  - First, instead of sampling a function at regularly spaced points, it finds optimal points to sample the function that will lead to a really good estimate of the integral.
  - Second, instead of aiming for a specific order of errors, it describes an integration rule that is accurate to the highest possible order in a polynomial fit, which turns out to be a polynomial fit of order $2N-1$ for N sample points.
  - The cool thing is that if you happen to be integrating a polynomial function of less than order $2N-1$, Gaussian Quadrature will be exactly correct (to within numerical roundoff). And it also works well with non-polynomial functions.

  Example 5.2 on p.170 gives you the tools you need to get started. The files referenced are `gaussint.py` and `gaussxw.py`. To use this code, you need to make sure that both files are in the same directory (so the import will work). This code approximately calculates an integral according to the following formula:[2]

[1] You might notice that when a Physics prof says "nice" she often means elaborate/detailed/`tl;dr`.

[2] Here is an explanation of all the notation in Equation (1):

- a and b are limits of integration, and u is the (dummy) variable of integration.

$$\int_a^b f(u)du \approx \sum_{k=1}^{N} w_k f(u_k). \tag{1}$$

The line

```
x,w = gaussxw(N)
```

returns the `N` sample points `x[0],...,x[N-1]` and the `N` weights `w[0],...,w[N-1]`. These weights and sample points can be used to calculate any integral on the interval $-1 < x < 1$. To translate this integral into one that approximates an integral on the interval $a < x < b$ you need to implement the change of variables formulas (5.61) and (5.62) of Newman, which are written in the code as

```
xp = 0.5*(b-a)*x+0.5*(b+a)
wp = 0.5*(b-a)*w
```

The loop then sums things up into the summation variable `s`.

On p.171, there's a bit of code that lets you skip the change of variables by using `gaussxwab.py`; this is acceptable but we won't need to use this code.

In Section 5.6.3 there's a discussion of errors in Gaussian Quadrature, which are somewhat harder to quantify than for the previous methods we've seen. Equation (5.66) suggests that by doubling N we can get a pretty good error estimate:

$$\epsilon_N = I_{2N} - I_N. \tag{2}$$

We will use this expression in Q1.

- **Solving derivatives numerically (for Q3):** Section 5.10 deals with various methods for solving derivatives numerically. You can use the central difference scheme to calculate derivatives along the lines of

```
for i in range(xlen):
     dfdx[i]=(f[i+1]-f[i-1])/(2.0*deltax)
```

where deltax is the spacing between your points.

– There are N so-called *sample points* $u_1, ..., u_N$ on the interval $a \leq u \leq b$.

– $f(u_k)$ is the function f measured at the sample point u_k.

– There are N coefficients called *weights*, $w_1, ..., w_N$.

– k is a dummy summation variable.

It's possible to write the previous integration formulas we've used before in this way. For example, for the Trapezoidal rule, the sample points are $u_1 = a, u_2 = a + h, ..., u_N = b$ and the weights are $w_1 = 1/2, w_2 = 1, w_3 = 1, ..., w_{N-1} = 1, w_N = 1/2$.

- **Plotting a vector field in Python (for Q3):** Vectors can be tricky to plot properly because their orientation can depend on both their components and the aspect ratio of the plotting window. Please see `PlotVectorsExample.py` for an example of how to create a vector plot. You can adjust the spacing of the vectors with the variable `vec_spacing`.

- **A filled contour plot in Python (for Q1):** We have so far used density and contour plots to visualize a function depending on two coordinates. Another plotting format is the filled contour plot implemented as `contourf` in `pylab`. This creates a set of contours with the space in between filled with different colors according to a colormap. The lines below are the typical kinds of calls you can make:

```
from pylab import contourf, colorbar
#plot filled contours
contourf(x,y,f)
#create colorbar
colorbar()
```

- **Using codes from previous labs:** You will notice that the questions below build on solutions from previous labs. Feel free to download and modify the solution codes from previous labs as you see fit. Don't forget to tell us which codes you are working from.

Physics Background

- **Near field diffraction theory (Q1):** The result quoted in the text Exercise 5.11 is from the theory of near-field (Fresnel) diffraction, which is described in optics textbooks like Hecht or Pedrotti. Ask Paul for the references if you'd like to explore this topic.

- **The relativistic particle on a spring – again (for Q2):** Recall the relativistic particle on a spring that we solved for in Lab01. The energy of the particle, which is conserved, is given by

$$E = \frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}} + \frac{1}{2}kx^2$$

which can be rearranged to show that

$$v^2 = c^2\left[1 - \left(\frac{mc^2}{(E-\frac{1}{2}kx^2)}\right)^2\right].$$

In the examples considered in Lab01 Q2, the particle started from rest from an initial position which we will call x_0. In this case $E = mc^2 + \frac{1}{2}kx_0^2$ and after rearranging terms we can write the following expression for the positive root:

$$v = \frac{dx}{dt} = c\left\{\frac{\frac{1}{2}k(x_0^2-x^2)\left[2mc^2+\frac{1}{2}k(x_0^2-x^2)\right]}{\left[mc^2+\frac{1}{2}k(x_0^2-x^2)\right]^2}\right\}^{1/2} = g(x), \tag{3}$$

where $g(x)$ is a function of x. Notice that for $\frac{1}{2}k(x_0^2 - x^2) \ll mc^2$ we find $v \approx \sqrt{k(x_0^2 - x^2)}$ as expected for an energy conserving linear harmonic oscillator. For $\frac{1}{2}k(x_0^2 - x^2) \gg mc^2$, v approaches c but remains less than c.

Given (3), the period for the oscillation is given by four times the time taken for the particle to travel from $x = x_0$ to $x = 0$. Using separation of variables (see Example 5.10 in the text for a somewhat similar example):

$$T = 4 \int_0^{x_0} \frac{dx'}{g(x')}. \tag{4}$$

In the small and large amplitude limits described above, we expect T to approach $2\pi\sqrt{m/k}$ and $4x_0/c$, respectively. In Q2 we will calculate T for a range of x_0 and compare it to these expected limits. Because $g(x) \to 0$ as $x \to x_0^-$, the integral diverges and a fairly large number of points will be required for an accurate calculation.

- **The electric field given an electrostatic potential (for Q3):** In Q3 we will ask you to do something fairly simple, which is to obtain the electric field numerically given the potential distribution $V(r, z)$. In this case the electric field is $\boldsymbol{E} = -\nabla V = -\left(\frac{\partial V}{\partial r}, \frac{\partial V}{\partial z}\right)$.

Lab Instructions

For grading purposes, only hand in solutions to the following parts. Ensure that your codes are well commented and readable by an outsider:

- Q1: Hand in plots, written answers, and code.
- Q2: Hand in plots and written answers.
- Q3: Hand in plots and written answers.

Lab Questions

1. **Near field interference (40% of lab mark):**
   (a) Do Exercise 5.11 on p. 174.
   (b) Plot the estimated error in the integration in Q1a using (2). Do this for x and z as specified in Q1a, for $N = 50$, and then again for $N = 100$. How do the errors compare to numerical roundoff error in each case?
   (c) Now create a two dimensional picture of the intensity pattern. Create an array of values of intensity over the range $-5 < x < 5$ and $1 < z < 5$ in the coordinates of the problem. Use filled color contours - I used a command like

   ```
   contourf(z,x,I, levels = linspace(0.0,1.4,15))
   ```

   to get good resolution of the intensity lines. Plot and describe qualitatively what happens as you go from $\lambda = 1.0\,\text{m}$, to $\lambda = 2.0\,\text{m}$, to $\lambda = 4.0\,\text{m}$.

2. **The period of a relativistic particle on a spring (40% of lab mark).** Using Gaussian Quadrature, we will numerically calculate the period of the spring from Lab01 Q2, with the period given by (4), and see how it transitions from the classical to the relativistic case. Do not redo the numerical simulation of Lab01. The idea is to calculate T multiple times for a range of initial positions x_0.

   If `x0` is the array of initial positions, `T` is the array of periods, and `k`, `m`, `c` are the other constants in the problem, the following command

   ```
   plot(x0,T,x0,2*pi*(m/k)**0.5+0*T, x0, 4*x0/c)
   ```

   provides a plot comparing T to the other asymptotic results.

   (a) The first issue is accuracy of the solution. As x_0 gets smaller, the period should approach $2\pi\sqrt{m/k}$. Calculate the period for $N = 8$ and $N = 16$ for $x_0 = 1\,\text{cm}$ and compare this to the known classical solution. Estimate the fractional error for these two cases. To get a better sense of what affects the calculation, plot the integrand $1/g_k$ and the weighted values w_k/g_k at the sampling points. Describe how these quantities behave as the x_0 limit of integration is approached. How do you think this behaviour might affect accuracy of the calculation?

   (b) For $N = 200$, what is your estimate of the percentage error for the small amplitude case? Now plot T as a function of x_0 for x_0 in the range $1\,\text{m} < x < 10x_c$, where x_c was calculated in Lab01, and compare it to the relativistic and classical limits as suggested at the beginning of the problem.

3. **Calculating the electric field from the gradient of the electric potential (20% of lab mark):** Referring to Lab02, Q3, calculate the electric field associated with the electric potential we have previously calculated. Plot the electric field as a vector plot on top of the contour plot of equipotential surfaces. You can use centred differences to create this plot. Document the spacing you used in the calculation; how did you choose this spacing? Make sure to distinguish the computational domain where you are numerically calculating your gradients from the plotting domain where you are plotting the vectors. Even if you need to calculate the gradients on a fine grid in r and z, you can plot the gradient vectors at the appropriate spacing for clarity.

   This involves multiple steps, and builds on a previous example, so this is a good opportunity to modularize your code. You will not be graded on code modularization at this point.