PHY407 Formal Lab Report 2

Fall 2024

Background

Numerical Derivatives of Discrete Functions : Section 5.10 of the textbook deals with various methods for solving derivatives numerically. They assume you have a function which you can calculate at any point. If instead, you just have function values at different discrete points, then the `h` term in the formulas has to be the distance between your data points. So for example, with a central difference scheme, to calculate derivatives you could use something like:

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

where `deltax` is the spacing between your points. However, notice that there will be an issue for the first and last `i` values. At `i=0, f[i-1]` doesn't exist. Similarly, at `i=xlen-1, f[i+1]` doesn't exist. So you can't use this formula for the endpoints. Instead, you need to use a forward difference scheme for the first point and a backward difference scheme for the last point.

Forward difference:

```
dfdx[i]=(f[i+1]-f[i])/deltax
```

Backward difference:

```
dfdx[i]=(f[i]-f[i-1])/deltax
```

Higher derivatives with central differences : The central difference approximation to the first derivative of $f = f(x)$ is

$$f'(x) \approx \frac{f(x+h/2) - f(x-h/2)}{h} \equiv \Delta_h f|_x,$$

where we have defined $\Delta_h f|_x$ as a central difference operator applied to f at x for a given interval h. Section 5.10 in the text points out that there is an optimum step size h for calculating derivatives using finite differences, including central differences. If h is too large the truncated Taylor series expansion becomes inaccurate. If h is too small the impact of numerical roundoff becomes large. So the optimum h balances numerical roundoff with algorithmic accuracy.

The central difference approximation to the second derivative of f is

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h/2)}{h^2} = \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h} = \Delta_h(\Delta_h f|_x).$$

This means that repeated application of the central difference operator gives higher order derivatives: the mth derivative of f can be approximated as

$$f^{(m)}(x) \approx \Delta_h^m(f|_x), \tag{1}$$

where the superscript m indicates m applications of the operator on $f|_x$. For each higher order derivative, samples of f further from $x = 0$ are required. This effect, together with cancellation errors related to taking differences between numbers with similar values, serious affect the accuracy of this approximation.

This method can be implemented iteratively (i.e. using loops). It can also be implemented recursively; while you are not required to use recursion in this course, here's the recursive code if you're interested:

```
# define function f(x) here
def my_f(x):
    return ... # return f given x

# calculate the m-th derivative of f at x using recursion.
def delta(f, x, m, h):
    if m > 1:
        return (delta(f, x + h/2, m - 1, h) - delta(f, x - h/2, m-1, h))/(h)
    else:
        return (f(x + h/2) - f(x - h/2))/(h)

# E.g. third derivative of my_f at x = 1.5 using h = 0.1.
# Notice how you can pass any function into this
d3fdx3 = delta(my_f, 1.5, 3, 0.1)
```

Gaussian Quadrature : Section 5.6 of Newman provides a nice introduction. 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), or to know how to fetch them in different directories.

The line `x, w = gaussxw(N)` returns the `N` sample points `x[0]`, ..., `[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`, which you can use if you wish.

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:

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

Quantum harmonic oscillator : In units where all the constants are 1, the wavefunction of the nth energy level of the one-dimensional quantum harmonic oscillator —i.e., a spinless point particle in a quadratic potential well— is given by

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} \exp(-x^2/2)\, H_n(x), \tag{3}$$

for $n = 0 \ldots \infty$, where $H_n(x)$ is the nth Hermite polynomial. Hermite polynomials satisfy a recursion relation,

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x). \tag{4}$$

The first two Hermite polynomials are $H_0(x) = 1$ and $H_1(x) = 2x$.

The quantum uncertainty of a particle in the n^{th} level of a quantum harmonic oscillator can be quantified by its root-mean-square position $\sqrt{\langle x^2 \rangle}$, where

$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 |\psi_n(x)|^2 dx. \tag{5}$$

This is also a measure of twice the potential energy of the oscillator.

Relativistic Particle on a Spring : The energy of a relativistic particle, which is conserved, is given by

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

which can be rearranged to show that

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

Assume the particle started from rest, from an initial position x_0. Then $E = mc^2 + \frac{1}{2}kx_0^2$ and after rearranging terms we can write the following expression for the positive root:

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

where $g(x)$ is a function of x.

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:

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

Notice that for $k\left(x_0^2 - x^2\right)/2 \ll mc^2$ (the small-amplitude, "classical" limit) we find $v \approx \sqrt{k\left(x_0^2 - x^2\right)}$, as expected for an energy conserving linear harmonic oscillator. We expect T to approach $2\pi\sqrt{m/k}$ as x_0 approaches 0.

Notice that for $k\left(x_0^2 - x^2\right)/2 \gg mc^2$ (the large-amplitude, "highly relativistic" limit), v approaches c but remains less than c. We expect T to approach $4x_0/c$.

In the relativistic limit, we can define x_c as the initial displacement x_0 for a particle initially at rest that would lead to a speed c at $x = 0$. The oscillator has initial potential, and therefore total, energy $kx_0^2/2$. For this energy to be entirely converted into maximum kinetic energy $mc^2/2$, we need $kx_0^2 = mc^2$. From this, we can find $x_c = c\sqrt{\frac{m}{k}}$.

Questions

Each of the following questions has equal weight.

1. **Calculating potential energy of QM harmonic oscillator**

   (a) Write a user-defined function `H(n,x)` that calculates $H_n(x)$ for given x and any integer $n \geq 0$, according to Equation 4. **Submit code.**

   (b) Use your user-defined function to make a plot that shows the harmonic oscillator wavefunctions (according to Equation 3) for $n = 0$, 1, 2, and 3, all on the same graph, in the range $-4 \leq x \leq 4$. **Submit code and plot.**

   (c) Write a program that uses your function `H(n,x)` to evaluate potential energy of the oscillator (according to Equation 5), using Gaussian quadrature on 100 points. Use the program to calculate the potential energy for each value of n from 0 through 10. *Note: you will need to use one of the evaluation techniques on p.179 of the textbook to deal with the improper integrals here.* **Submit code and printout of answers.**

2. **Relativistic Particle on a Spring** Using Gaussian quadrature, we will numerically calculate the period of a particle on a spring, and see how it transitions from the classical to the highly relativistic case. The idea is to calculate T multiple times for a range of initial positions x_0, assuming a mass of $m = 1$ kg and a spring constant $k = 12$ N/m.

(a) Use Gaussian Quadrature to numerically calculate the period for $N = 8$ and $N = 16$ for $x_0 =$ 1cm. Estimate the fractional error of the calculation using these two N's. **Submit printout of answers.**

(b) For $N = 8$ and for $N = 16$, plot the integrand values $4/g_k$ and the weighted values $4w_k/g_k$ at the sampling points. Describe how these quantities behave as the x_0 limit (i.e. the upper limit) of integration is approached. How do you think this behaviour might be affecting the accuracy of the integral calculation? **Submit plot and written answers.**

(c) Using Gaussian Quadrature with $N = 16$, plot T as a function of x_0, for x_0 in the range $1\text{m} < x < 10x_c$. (You'll need to calculate x_c for this spring.) Use whatever number of (x_0, T) values seems reasonable to you. Compare your Gaussin Quadrature T vs x_0 result to the classical limit (which you should show as a horizontal line on your plot) and to the highly relativistic limit (which you should show as a diagonal line on your plot). Does it seem reasonable? Why or why not? **Submit plot and written answers.**

3. **Central Differences for Numerical Differentiation** Consider the function $f(x) = \exp(-x^2)$.

(a) Using the central difference approximation, find this function's derivative numerically at $x = \frac{1}{2}$ for a range of h values from $10^{-16} \to 10^{0}$, increasing by a factor of 10 each step. (You should have 17 values of h, with a values of the derivative for each h value). **Submit code and printed output.**

(b) Calculate the function's derivative analytically at $x = \frac{1}{2}$. Calculate the absolute value of the relative error of each numerical derivative value you obtained in the previous part (using the analytical answer as the "truth"). Which h value yields the minimum error? Does this answer match what you would expect from Section 5.10, with $C = 10^{-16}$? **Submit code, printed output, and written answer.**

(c) Repeat the previous two parts using forward difference approximation instead of central difference approximation. **Submit code, printed output, and written answer.**

(d) Plot the absolute value of the relative error of the numerical derivative, as a function of h, for the central difference method and for the forward difference method. (Use logarithmic horizontal and vertical axes. You should have two lines – one for each method – on the same plot.) Does the central difference scheme always clearly beat the forward difference scheme in terms of accuracy? **Submit code, plot, and written answer.**

(e) At high h values, which type of error (round-off or approximation) dominates? What about at low h values? Explain. **Submit written answer.**

(f) Now consider the function $g(x) = \exp(2x)$. Write a program to numerically calculate the first 5 derivatives of $g(x)$ at $x = 0$ using the central difference method, with $h = 10^{-6}$. Print the results. Do they seem reasonable? (Hint: the correct answer can be easily found analytically.) **Submit code, printed output, and written answer.**