PHY407 Formal Lab Report 2

Fall 2023

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 to Gaussian quadrature. 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.

Questions

1. **Central Differences for Numerical Differentiation** Consider the function $f(x) = e^{2x}$, whose first derivative at $x = 0$ is $f'(0) = 2$.

   (a) Using the central difference approximation, find this function's derivative numerically for a range of h values from $10^{-16} \rightarrow 10^{0}$, increasing by a factor of 10 each step (so you should have 17 h values with respective values of the derivative for each h value). **Hand in your code and printed output of the answers.**

   (b) Calculate the absolute value of the error in each derivative (i.e. the magnitude of the difference between your result and exactly 2). Plot the error as a function of step size. Does the minimum in error correspond to the expected value from Section 5.10, with $C = 10^{-16}$? **Hand in your plot, and explanatory notes.**

   (c) Write a program to calculate the first 10 derivatives of $f(x) = e^{2x}$ at $x = 0$ using central differences (Equation 1). Print the results. (Hint: the mth derivative of f is 2^m, so it should be easy to check the results.) **Hand in your code and output.**

2. **Integrating the Dawson Function Again**

   In the previous lab, we evaluated the Dawson function $D(x) = \exp(-x^2) \int_0^x \exp(t^2) dt$ at $x = 4$ using the Trapezoidal Rule and Simpson's Rule numerical methods, as well as SciPy's implementation (assumed to give the "true" value). We'll now try Gaussian Quadrature as another numerical method.

   (a) Calculate the same integral with all three numerical methods for a range of `N` slices/sample points between `N=8` and `N=2048`. (For Trapezoidal and Simpson's, you can re-use your code from the previous lab.) **Hand in plots or tables, it's your choice how to present the results.**

   (b) Estimate the error of the Gaussian Quadrature using Equation (2), which requires you to do the calculation at `2N` as well as at `N`, for each of the `N` values you used in the previous part. Plot the magnitude of the error as a function of `N` (you may want to use log axis), and comment on the results. **Hand in plot and comments.**

   (c) Calculate the relative error of the Gaussian Quadrature result compared to the true value of $D(4)$, for each of the `N` values you used in the previous part. Plot the magnitude of the error as a function of `N` (you may want to use log axis), and comment on the results. **Hand in plot and comments.**

3. **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.**