UCSB, Physics 129AL, Computational Physics: Section Worksheet, Week 8A

Zihang Wang (UCSB), zihangwang@ucsb.edu

February 27, 2025

Section Participation and Submission Guidelines

Section attendance is required, but you do not need to complete all the work during the section. At each section, the TA will answer any questions that you might have, and you are encouraged to work with others and look for online resources during the section and outside of sections. Unless otherwise stated, the work will be due one week from the time of assignment. The TA will give you 1 point for each task completed. You can see your grades on Canvas.

We will use GitHub for section worksheet submissions. By the due date, you should have a single public repository on GitHub containing all the work you have done for the section work. Finally, upload a screenshot or a .txt file to Canvas with your GitHub username and repository name so the TA knows who you are and which repository you are using for the section.

Remember: talk to your fellow students, work together, and use GPTs. You will find it much easier than working alone. Good luck! All work should be done in the Docker container, and don't forget to commit it to Git!

Task 1: Quadrature

Let's look at different quadrature methods and rules.

Midpoint Rule

$$\int_a^b f(x)\,dx \approx (b-a)\cdot f\left(\frac{a+b}{2}\right)$$

Trapezoidal Rule

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left[f(a)+f(b)\right]$$

Simpson's Rule

$$\int_a^b f(x)\,dx \approx \frac{b-a}{3}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

0.1 A)

Define one class that contains all quadrature techniques.

0.2 B)

Write the above three quadrature rules as class methods without any pre-defined function from any package 1) Midpoint rule 2) Trapezoidal Rule 3) Simpson's Rule. You need to figure out what additional inputs each rule needs.

C) Gauss-Legendre Quadrature

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{N} w_i \cdot f(x_i). \tag{1}$$

Since the range is from a to b, let's make a variable change,

$$x_i \to \frac{b-a}{2}x_i + \frac{a+b}{2}$$

Calculate the above condition **analytically**, You should get something like this:

$$\int_a^b f(x)dx = \frac{b-a}{2}\sum_i w_i f\left(\frac{b-a}{2}x_i+\frac{a+b}{2}\right)$$

D) Legendre Polynomials

To find the position and weights of an order M Gauss-Legendre Quadrature, we need to calculate the roots of an order-M Legendre polynomial,

$$(1-x^2)\frac{d^2P_M(x)}{dx^2} - 2x\frac{dP_M(x)}{dx} + M(M+1)P_M(x) = 0,$$

and the solutions are given by the following:

$$P_M(x) = \frac{1}{2^n n!}\frac{d^M}{dx^M}\left[(x^2-1)^M\right]$$

Write a child class named Gauss_Quad_ inherits methods from its parent class Quad_. This class takes an additional input, called order. Define a new method that outputs an order-M Legendre polynomial. Plot the following Legendre polynomials: M=[1,2,3,4,5].

E) Newton's Method

The positions of an order M Gauss-Legendre Quadrature are calculated by finding the roots of an order-M Legendre polynomial,

$$P_M(x) = 0.$$

To numerically find those roots (M of them), we can use Newton's method:

$$x_{n+1} = x_n - \frac{P_M(x_n)}{P'_M(x_n)}.$$

You should be careful with the initial guess.

The weights w_i for Gauss-Legendre Quadrature are calculated as:

$$w_i = \frac{2}{(1 - x_i^2)[P'_M(x_i)]^2},$$

where $P'_M(x_i)$ is the derivative of the Legendre polynomial of degree M evaluated at each root x_i.

Calculate 'M' position and weights for Gaussian quadrature integration between 'a' and 'b' with the Newton's method. Returns a tuple of 2 arrays, the first array is the position of points and second array is the corresponding weights. Output an text file that contains the roots and weights for M=[1,2,3,4,5]. Using the the following information, https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.roots_legendr calculate the roots and weights for M=[1,2,3,4,5]. Compare your results.

Task 2: Quadrature on test functions

In the following question, we want to use the above four methods, `mid_quad`, `trapz_quad`, `simpson_quad`, `gauss_quad`, to numerically calculate the quadrature on two test functions ($k \geq 0$).

A general k-th order polynomial with a quadrature:

$$I_{\text{true}}^{\text{T=A}} = \int_a^b x^k \, dx = \frac{1}{k+1}(b^{k+1} - a^{k+1})$$

A Fermi–Dirac distribution with a quadrature:

$$I_{\text{true}}^{\text{T=B}} = \int_a^b \frac{1}{1 + e^{-kx}} \, dx = \frac{1}{k} \left(\log(e^{kb} + 1) - \log(e^{ka} + 1) \right)$$

Let's define the following: for each order k and N, we have the quadrature value and the relative error:

1. Midpoint Rule (`mid_quad`): $M(k, N)$,

$$\Delta M(k, N) = 2\frac{I_{\text{true}} - M(k, N)}{I_{\text{true}} + M(k, N)}$$

2. Trapezoidal Rule (`trapz_quad`): $T(k, N)$,

$$\Delta T(k, N) = 2\frac{I_{\text{true}} - T(k, N)}{I_{\text{true}} + T(k, N)}$$

3. Simpson's Rule (`simpson_quad`): $S(k, N)$,

$$\Delta S(k, N) = 2\frac{I_{\text{true}} - S(k, N)}{I_{\text{true}} + S(k, N)}$$

4. Gauss-Legendre Quadrature (`gauss_quad`): $G(k, N)$,

$$\Delta G(k, N) = 2\frac{I_{\text{true}} - G(k, N)}{I_{\text{true}} + G(k, N)}$$

Make the following heatmaps for each quadrature method above (1–4), over the range $[0, 1]$:

- y-axis: k from 0 to 10
- x-axis: N from 10 to 10^5
- Value: Relative error

A) Polynomial

B) Fermi-Dirac

Task 3: harmonic oscillator

The total energy of a harmonic oscillator is given by:

$$E = \frac{1}{2}m\left(\frac{dx}{dt}\right)^2 + V(x)$$

Assuming that the potential $V(x)$ is symmetric about $x = 0$ and the amplitude of the oscillator is a, the equation for the time period is given by:

$$T = \sqrt{8m}\int_0^a \frac{dx}{\sqrt{V(a) - V(x)}}$$

A)

Suppose the potential is $V(x) = x^4$ and mass of the particle $m = 1$, write a function that calculates the period for a given amplitude.

B)

Let $a = 2$. Use inbuilt 'fixed_quad' (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.integrate
function to calculate the time period for different values of 'N' (number of integration points). Calculate the error in the integral by estimating the difference for 'N' & '2N'. Approximately, at what 'N' is the absolute error less than 10^{-4} for 'a = 2'?

C)

Use inbuilt 'quad' (https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quad.html) function that returns an error estimate and compare your answer for 'a = 2' (quad uses a more advanced integration technique)

D)

Calculate the time period by using the inbuilt romberg function (https://docs.scipy.org/doc/scipy/reference/gen
for Romberg integration. A simplistic usage with romberg(func, 0, a), where a is the amplitude, will probably give error or 'nan'. Why?

E)

Assume that we can tolerate the uncertainitiy of 10^{-5} in the position. Show and output of 'keyword' show = True for 'a = 2'. Use this to estimate error for divmax = 10.

F)

Change divmax to change the number of divisions. How does the accuracy change on going from 10 to 15 divisions.

G)

Use the function to make a graph of the period for amplitude ranging from a=0 to a=2.