Quiz 2 (part 1) - Computational Physics I

NAME: Males-Araujo Yorlan **SCORE:** 9/10

Date: Friday 17 May 2024 **Duration:** 45 minutes
Credits: 10 points (5 questions) **Type of evaluation:** LAB

This quiz is individual and has two parts: Part 1 is closed-book, in-class, and contains short-answer questions. Part 2 is take-home and contains long application problems.

Provide short and concise answers to the following items:

1. **(2 points) Numerical differentiation**
   Explain how the finite-difference methods for calculating derivatives work, and provide the mathematical definition of a second-order approach.

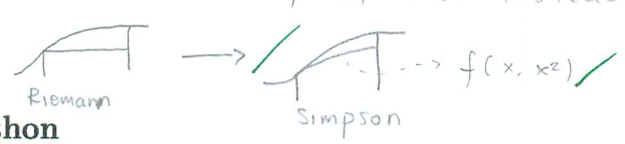
They use a finite difference (eg. $\Delta x = 1$) to get the derivative of a function for which we may not have an expression analytical you mean? ~~(...)~~. In class, we review 3 methods: forward ($(f_{i+1} - f_i)/(x_{i+1} - x_i)$); backward ($(f_i - f_{i-1})/(x_i - x_{i-1})$) and central difference methods. The first two provide a good result but it's shifted a little since the data ~~is~~ are shifted. The last method mentioned is a second-order approach since it uses two shifted points, but the result is quite better as the error is smaller. It's defined as

$$f'(x) \approx \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}} + \underbrace{O(\Delta x^2)}_{\text{error}}.$$

2. **(2 points) Numerical integration**
   Indicate 2 numerical methods used for calculating 1D integrals numerically, and briefly explain how they work.

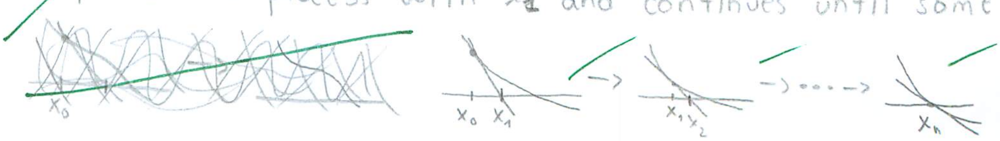
1) Riemann integrals (sums): This methods approximates the area below the curve by fitting rectangles whose height depends on the curve and lenght can be adjusted at will. The narrower the better. It calculates the area of each rectangle and then adds up everything.

2) Simpson's method: It's an improvement of the previous method. It still uses rectangles but also a second-order polynomial at the top of them instead of just a straight line. The result is better.



3. **(2 points) Root finding in python**
   List 2 methods that we can use to find the roots of 1D functions in python, and briefly explain how these python methods work to obtain the roots.

1) Newton-Raphson method: It takes a guess in x_0, takes $f'(x_0)$ and $f(x_0)$, and draws a tangent line at $f(x_0)$, and get's another point, x_1, resulting from the tangent line intersecting the x-axis. It repeats the process with x_1 and continues until some tolerance level is reached.



2) Bisection method: It Takes two inputs for which $f(x_0) > 0$ and $f(x_1) < 0$ (or the other way around) and evaluates $f((x_0 + x_1)/2)$. Based on the sign of it, repeats the process with the point whose image had different sign. It repeats the process until reaching the tolerance level.

4. **(2 points) Errors in numerical differentiation**
Explain the sources of errors when calculating the divergence of a 2D vector field on the grid. What defines the order of accuracy of the method?

Sources of error:

- Since we are performing a numerical calculation, there will always be errors that have to do with the machine epsilon. That is why $\vec{\nabla} \cdot \vec{B}$ does not give us $\vec{0}$, but close values to 0.

- ~~Noise in the data which does not have to be big noise.~~

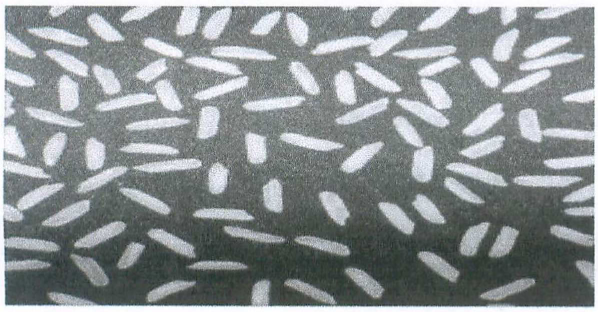
-0.25

- Shifted data will return a shifted result; however, this can be fixed by ~~increasing~~ using a method of higher order since it will make the errors smaller. But there will always be errors in numerical differentiation.

The order of accuracy of the methods is defined by the number of shifted points we may use. If we use just one, the the error is of order $O(\Delta x)$. With two, $O(\Delta x^2)$.

5. **(2 points) Image processing**
Imagine you obtain the following photograph of rice grains, and you are asked to find the edges of the grains. Design and sketch a suitable algorithm workflow to achieve this goal in python.

-0.5



This is not a workflow sketch

- Get the data in 2D.
- Create a function that computes the derivative along each axis (using np.gradient() for ex.).

→ gradient?

- Call the function and store the data in two arrays. Each containing the derivative along an axis.
- Calculate the modulus of the vector with those two arrays.

There's a step missing. You need to filter using a threshold value.

- Now plot that modulus in 2D. What should be obtained is an image where the edges (sections where the image changes) are highlighted.

All is possible because of the gradient works, $\vec{\nabla} f = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y}$. Where the data is repetitive, it's close to zero; but where it changes, it's non-zero. So the gradient work as an edge detector.

Indicate python workflow.

-0.25