Quiz 3 - Computational Physics I

NAME: Alan Israel Palma Travez **SCORE:** 15.6/20

Date: Monday 18 November 2024 **Duration:** 45 minutes

Credits: 20 points (5 questions) **Type of evaluation:** LAB

Provide short and concise answers to the following items: 7.8/10

1. **(2 points) Numerical differentiation**
   Explain how the finite-difference methods for calculating derivatives work.
   Provide the mathematical definition of a forward-difference scheme.

Finite-difference methods use finite diferences (Δx) to aproximate the derivative of a function that could be not solved analitically. Therefore, it is used that the derivative can be aproximated to the variation of the dependent variable over the variation of the independent variable $f'(x) \simeq \frac{\Delta y}{\Delta x}$. There are three aproaches learned: forward, backward, and central differences methods.

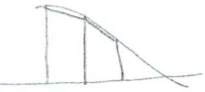
-1 Central differencest method ? → $f'(x) = \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}} + O(\Delta x^2)$. ↳ Second order method.

What about the forward diff. definition?

2. **(2 points) Numerical integration**
   Indicate 2 numerical methods used for calculating 1D integrals numerically.
   Briefly explain how each method works.

1. Rieman integrals: Rieman integrals are sums that aproximate the area under a curve by filling with rectangles equally-spaced, calculating their area and summing all. The height of the rectangles can be the upper limit, the lower limit or a central point.

2. Trapezoid rule: This method is essentially the same as Rieman integrals but instead of rectagles it uses trapezoids. Therefore the upper line with fit better to the curve than a straigh line.



3. **(2 points) Optimisation problems**
   Provide an example of an optimisation problem.
   List the main steps for solving optimisation problems in python.

-1 • An optimization problem is finding the maxima and minima of a function or data array or also fit some data to a trial function (physically meaning usually). For example: we can fit observational data from solar spectrum to the black body radiation model.

1. Inspect the data of the problem
2. Evaluate if it is needed smooth (Interpolate) the data to get a interpretable result. ?? Is this essential for optimisation? What about ansatsz?
3. Use third party function (e.g. scipy) or work with our own implementation (In the case of maxima and minima we can use numerical differentiation to get the local maxima and minima points, and second derivative to see wich of it is actually a maxima or a minima. In the case of fitting data is more suitable to use a third party library).
4. Check if the result is correct by plotting. ?

But, What about the objective function? bounds? constraints?

4. **(2 points) Errors in numerical differentiation**
Indicate the sources of errors when computing the curl of a 2D velocity field: $\vec{\nabla} \times \vec{v}$
What defines the order of accuracy of a differentiation method?

- The sources of errors can be:
  1. The error produced from the numerical aproach. There is always be an error in numerical methods, but it can be minimized by increasing the order of the method. In numerical differentiation there are problems due to shifting arrays, speatially at the edges where we does not have "enough" information to compute it properly. ?
  2. Error related with the machine epsilon → especially when we are leading with values too small or too large.

-0.25

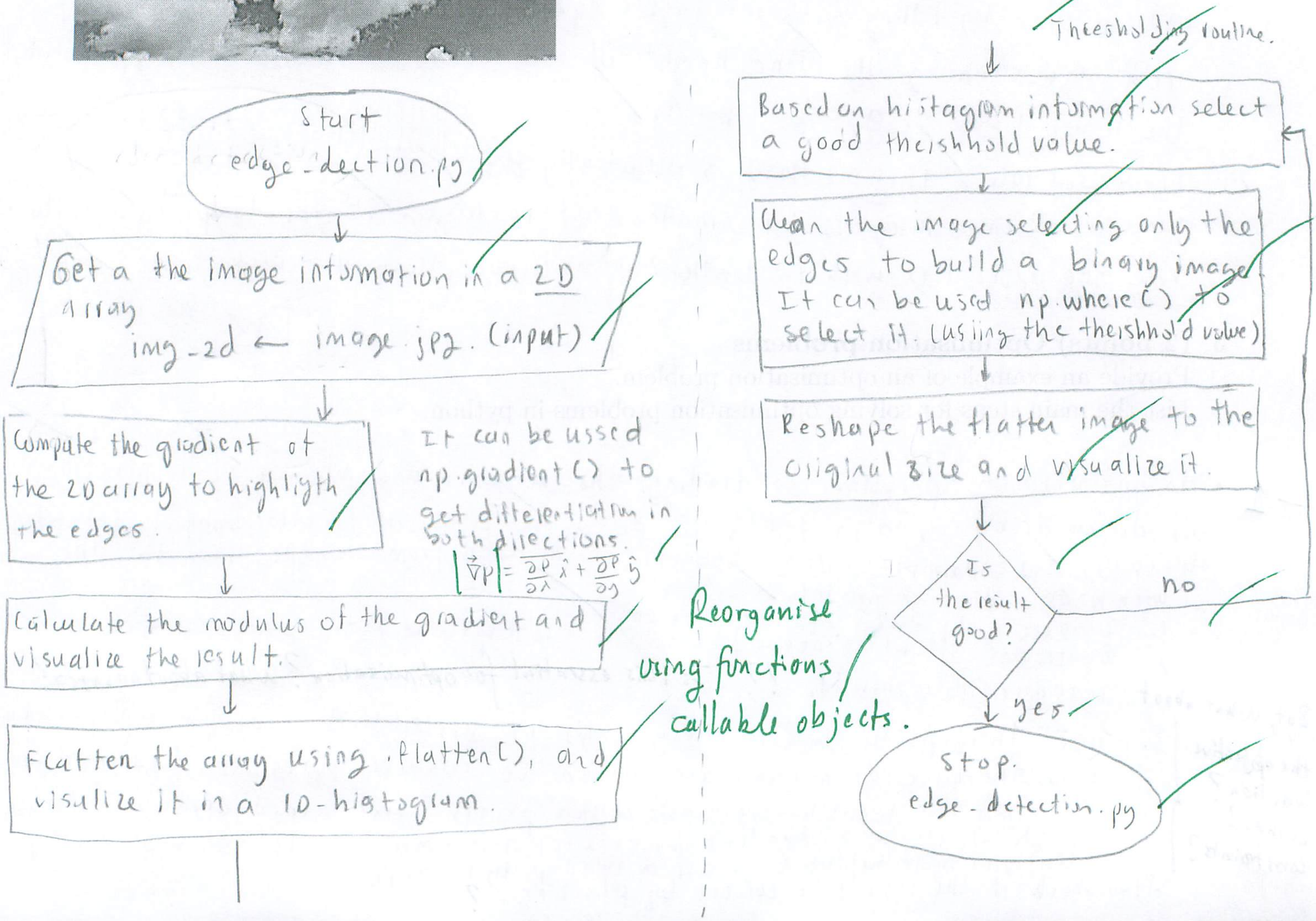
- The order of occuracy is related with the number of points that we use in the method. ? One-point aproach has first-order (Δx) (backword, fordward), while two-point aproach has second order (Δx^2) (central).

It comes from a Taylor series.

5. **(2 points) Image processing: edge detection**

The gradiend works as an edge detector because the variation $\frac{\partial P}{\partial x}$ or $\frac{\partial P}{\partial x}$ is bigger at the edeges than in other part.

Imagine you obtain the photograph (shown on the left-hand side) of atmospheric clouds, and you are asked to find the edges of the clouds. Design and sketch a suitable algorithm workflow to achieve this goal in python.



Start
edge-dection.py

Get a the image information in a 2D array
img-2d ← image.jpg. (input)

Compute the gradient of the 2D array to highligth the edges.

It can be ussed np.gradient() to get differentiation in both directions.
$|\vec{\nabla} p| = \frac{\partial p}{\partial x}\hat{x} + \frac{\partial p}{\partial y}\hat{y}$

Calculate the modulus of the gradient and visualize the result.

Flatten the array using .flatten(), and visulize it in a 1D-histogram.

Theesholding routine.

Based on histagram information select a good theishhold value.

Clean the image selecting only the edges to build a binary image It can be used np.where() to select it. (usiing the theishhold value)

Reshape the flatten image to the original size and visualize it.

Is the result good?

no

yes

Stop.
edge-detection.py

Reorganise using functions callable objects.