Assignment 3 - Derivative types & Carts

Start Assignment

- Due No due date
- Points 20
- Submitting a text entry box or a file upload

Exercise 1: Game of Life Simulation:

The Game of Life, also known simply as Life, is a fascinating cellular automaton devised by the British mathematician John Horton Conway in 1970. Unlike typical computer games, it's a zero-player game, meaning its evolution is determined solely by its initial state, requiring no further input. Let me delve into the intriguing details:

- What Is Conway's Game of Life?
  - The Game of Life is not your ordinary video game. It operates as a cellular automaton, a grid of cells that can live, die, or multiply based on a few simple mathematical rules.
  - John Conway introduced this game, and it gained widespread recognition after being featured in a Scientific American article in 1970.
  - The game consists of a grid where each cell can be alive or dead. These cells evolve according to specific rules.
  - Rules of the Game:
- For a space that is populated:
  - Each cell with one or no neighbors dies, as if by solitude.
  - Each cell with four or more neighbors dies, as if by overpopulation.
  - Each cell with two or three neighbors survives.
  - Each cell with three neighbors becomes populated.
- Patterns and Evolution:
  - Depending on the initial conditions, the cells form various intricate patterns throughout the course of the game.
  - These patterns can include oscillators, gliders, and even structures that seem to move indefinitely.
  - The Game of Life has captured the imagination of mathematicians, computer scientists, and enthusiasts alike.
- What you should implement:
  - Implement Conway's Game of Life on a distributed 2D grid topology using MPI_Cart_create.
  - Each process should manage a sub-section of the grid, calculating the next state of cells based on the current state's rules.
  - Use MPI_Cart_shift to exchange boundary information with neighboring processes to accurately compute the state transitions at the edges.
  - Simulate the game for a specified number of generations, with an option to visualize the grid at each step.

Exercise 2:

- Solve the problem defined in poisson.pdf using the poisson_empty.py file.

[poisson.pdf (https://um6p.instructure.com/courses/5047/files/278728?wrap=1)](https://um6p.instructure.com/courses/5047/files/278728?wrap=1) (https://um6p.instructure.com/courses/5047/files/278728/download?download_frd=1)

[utils.py (https://um6p.instructure.com/courses/5047/files/278727?wrap=1)](https://um6p.instructure.com/courses/5047/files/278727?wrap=1) (https://um6p.instructure.com/courses/5047/files/278727/download?download_frd=1)

[poisson_empty.py (https://um6p.instructure.com/courses/5047/files/278726?wrap=1)](https://um6p.instructure.com/courses/5047/files/278726?wrap=1) (https://um6p.instructure.com/courses/5047/files/278726/download?download_frd=1)