This is a base code for a 2D PIC (particle in cell) scheme parallelized by MPI (Message Passing Interface). As of now, it is a field solver (does not account for particles yet).

The code divides the workload of the spatial domain into different CPU cores and solves for electric and magnetic fields. Three examples of electromagnetic waves were left for the user to try.

To run the main code in the terminal:

The user should have HDF5 installed in their computer. All the files should be in the same folder, where the user will run the following commands:

```
mpic++ -std=c++11 PIC_2D_final_dt.cpp Auxiliar_functions.cpp Field_update.cpp HDF5_output.cpp  -o PIC_2D_final_dt.o -I/opt/homebrew/include -L/opt/homebrew/lib -lhdf5_cpp -lhdf5

mpirun --prefix /opt/openmpi-5.0.7 -np 4 PIC_2D_final_dt.o
```

Note: in the second command, “np” stands for number of processors. In this example, the code was executed with 4 CPU cores, but the user may change it to however many they please.

Below is a brief explanation of the files and what they do.

Main code: PIC_2D.cpp

Works with Field_update.cpp, Auxiliar_functions.cpp and HDF5_output.cpp

This is the bulk of the code. It sets up for the simulation with space discretization into cells and tiles (which are made up of cells), sets up how the MPI communication is going to work, initializes the electromagnetic waves and evolves them in time in the time loop.

To execute their own simulation, the user may alter “Simulation parameters” for different spatial resolutions and simulation time. Then, they may also alter “Tile initialization” where they will find the different wave equations to begin the simulation.

WARNING: the code only evolves the fields correctly if nx is an integer multiple of tileCols and ny is an integer multiple of tileRows.

The authors advise not to change anything else as it may disrupt the functionality of the code.

Auxiliar_functions.cpp

This code supplies functions that will be used in the main code, mostly for what concerns the MPI part of the code: locating tiles among ranks, sending and receiving a tile etc.

Field_update.cpp

This is the electromagnetic field updater. In a comment in the very first lines, the user will find how the physical spatial coordinates (i,j) were rearranged in a Yee scheme staggered grid.

The field updater works with two functions: one that updates the B field in half time steps and one that updates the E field in full time steps (this is done to follow Yee’s algorithm). They are used in the main code in the time loop to evolve the fields in time.

HDF5_output.cpp

This reads the results of the simulation and stores them in hdf5 files. It creates a folder called “Fields” inside another folder called “Simulation” and saves the files there. The user can alter the save frequency in “Simulation parameters” in the main code.

It also creates a text file named params.txt containing all the relevant simulation parameters that will then be read by the file reader to create plots.

File_reader.py

This is a python code that reads the hdf5 saved files from the simulation folder and makes plots and video animations.

For the figure plots of Ex, Ey, Ez, Bx, By, Bz, the user should look for a big arrow in the main function that points to the line that should be changed: “step_to_load”. This refers to the time step that will be plotted.

For video animations, the user should have a folder in their own computer named “Simulation_Videos” and after running the code, the animations should be saved there.