GPU-accelerated Electromagnetic Particle-in-Cell Program

Beverly Lowell & Gabriel Casabona

Introduction

In this project, we will optimize an existing 2D particle-in-cell (PIC) code, which calculates the trajectories of interacting particles from electromagnetic forces. This program considers the motion and interaction of electrons, calculating the total kinetic energy and transverse magnetic and electric field energies. Using PIC models is the most robust way to simulate plasmas. This problem is computationally intensive because it requires tracking and calculating the self-consistent electromagnetic field from many interacting particles in a fixed grid, solving coupled partial differential equations (PDEs) to do so.

Description

The program initializes a 2D grid with a uniform distribution of electrons. The main loop of the program updates the time step and does the following:

1. In position space, it calculates charge and current densities.
2. Densities are then Fourier transformed into k-space.
3. Using these densities, solve Maxwell's equations to calculate electric and magnetic fields, and then fields are updated.
4. The new fields are Fourier transformed back into position space.
5. The momentum and position of the particles are calculated using the new fields.
6. This process is looped over all cells in the grid.

This program requires 2D FFTs.

Objective and Deliverables

Several versions of this code are available **[1]**, with varying dimensions (1-3) and levels of parallelization using combinations of MPI, OpenMP, and CUDA. We are choosing to use the serial version of the code with no parallelization in 2D, considering electromagnetic forces, and optional relativistic effects. This project will parallelize the most computationally-intensive calculations by developing kernels to run on a Wilkinson lab GPU. We suspect that the heaviest calculations are the FFT of the fields and the respective grid updates of the fields and particle trajectories, but we can identify them by simply timing each calculation in the code. Updating the particle positions will be an easy way to optimize the code, because each particle is independent. We will also attempt to use Eclipse as an extra timing diagnostic. If memory allocation becomes an issue, then we will implement tiling techniques to coalesce the memory access.

Background

Both members in our group have the necessary physics background in electromagnetism (EM) and general computational techniques in solving numerical PDEs. We will both need to develop a deep understanding of PIC codes, but senior members of our research group have strong PIC code backgrounds and will be available to help us if needed. We will need to develop an understanding of 2D fast Fourier transforms (FFT) and the time-stepping techniques, i.e. leap-frog method. Overall, we have experience in computational physics through our research group that has allowed to learn techniques that are transferable to this project.

Resources

The code that we are using for this project [1] was developed by the UCLA Plasma Simulation Group and has sufficient documentation on how to run the code. Specifically, we will be focusing on the directory which contains the serial versions of the code called *bpic2*. Vast amounts of scientific literature exists that explains the origins and implementation of PIC codes and how beneficial they are in solving these kinds of problems.

We will use their documentation on generally accelerating code for GPUs, but we will only implement techniques learned in lecture.

[1] https://github.com/UCLA-Plasma-Simulation-Group/PIC-skeleton-codes
[2] https://github.com/UCLA-Plasma-Simulation-Group/PIC-skeleton-codes/blob/master/gpu/gpubpic2/GPU-PIC.pdf

Contact Information

Beverly Lowell: beverlylowell2023@u.northwestern.edu
Gabriel Casabona: gabrielcasabona2025@u.northwestern.edu

Ian Christie (post doc in group)
Sasha Tchekhovskoy (group P.I.)