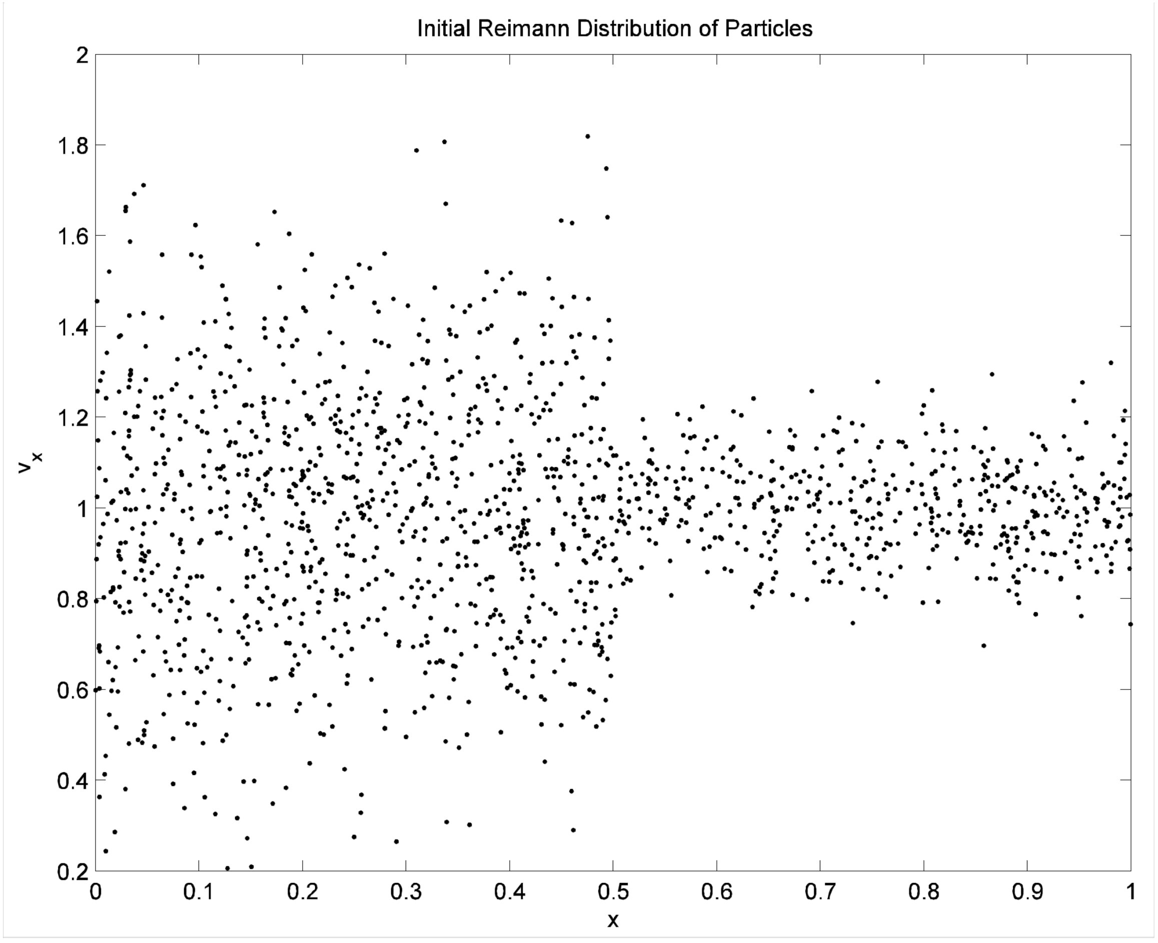
Chad Smith
10 September 2006

Outline of Summer Work for Professor Edgar Choueiri

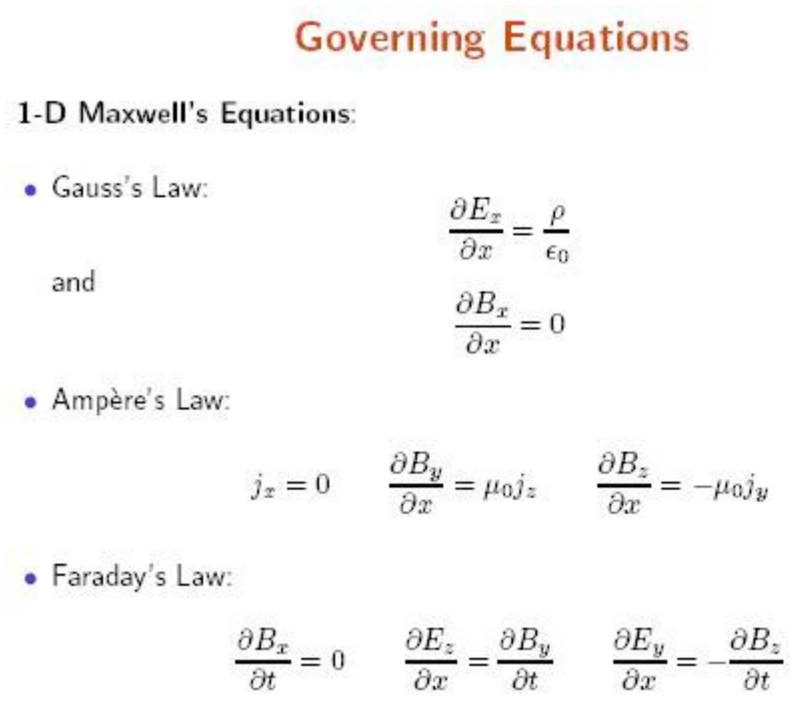
This summer I concentrated on a 1-D particle micro-simulator for exploring the boundary interactions in a plasma finite volume code. One rather substantial difficulty encountered in computational plasma physics is the vast difference in size scales encountered in the governing equations. Plasma behavior is partially described by fluid and particle equations, but the two fields make fundamentally different assumptions; for instance, a single fluid 'element' may comprise millions of particles. The real difficulty comes in the self-consistency of the equations—particle motions create fields which induce their motions. Also, electromagnetic forces travel on the time scale of the speed of light whereas particle collisions occur at an electron and ion thermal energy scales, and then fluid physics are much slower, based on shock waves and initial behaviors. Obviously Debye lengths are much smaller than thermal gradients. The goal of my project was to attempt a somewhat innovative type of simulation, called the 'Equation Free' method that was proposed by Ioannis G. Kevrekidis, C. William Gear, J.M. Hyman, P.G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Their proposal: Why not simulate finite-volume fluid code and use particle simulation at the boundaries? Or, essentially take a snapshot of the particle behavior at various points in the fluid and use it to extrapolate large scale behavior. Additionally, one won't necessarily need a global governing equation. My goal was to write a Particle-in-Cell, (PIC), simulation that would handle the interactions between finite-volume cells. This simulation could yield important insight into the efficacy of agent based methods for simulation of non-ideal effects in MHD systems—this would be a critical component in studying how hybrid methods might be used to incorporate such phenomena as finite-rate ionization, sheath physics, and plasma-wall interaction into fluid-based plasma simulation.

The goal of my program was essentially to take as inputs the boundary conditions of two neighboring cells, run a simulation on particle behavior, and then output the flux of various quantities between the cells. My micro-simulator region was initialized with Riemann initial conditions where the bulk velocity, bulk density, and temperature of the left cell are implemented on the left half of the simulation region, and likewise the right half of the simulation region uses quantities from the cell on the right. This diagram helps to explain how the initial conditions work:



Here you can see that the left cell has a greater standard deviation of velocities (because of higher temperature) and the left half actually has twice as many particles (resulting from a greater bulk density), and both cells here have the same bulk velocity.

The governing equations in 1-D can be simplified to:



Governing Equations

1-D Maxwell's Equations:

- Gauss's Law:

$$\frac{\partial E_x}{\partial x} = \frac{\rho}{\epsilon_0}$$

  and

$$\frac{\partial B_x}{\partial x} = 0$$

- Ampère's Law:

$$j_x = 0 \qquad \frac{\partial B_y}{\partial x} = \mu_0 j_z \qquad \frac{\partial B_z}{\partial x} = -\mu_0 j_y$$

- Faraday's Law:

$$\frac{\partial B_x}{\partial t} = 0 \qquad \frac{\partial E_z}{\partial x} = \frac{\partial B_y}{\partial t} \qquad \frac{\partial E_y}{\partial x} = -\frac{\partial B_z}{\partial t}$$

along with which are the kinematic equations and Poisson's equation and that is all the equations I dealt with.

The general scheme of the program is as follows:

1. Add particles $(x,v)_i$ to the simulation region based on the boundary conditions.
2. The particle positions and velocities give the charge and current densities at gridded points.

3. The charge and current densities yield the electromagnetic fields at the gridded points.
4. Move the particles according to forces imparted by the electromagnetic fields.
5. Remove particles that have left the simulation region.
6. Calculate statistics based on the particle locations and velocities, etc.
7. Add particles based on the bulk density conditions.
8. Go to Step 2.

Although this program may seem like a typical Particle-In-Cell code, there were several differences between one and my specific problem that I had to account for. One complication arises from the fact that the boundary conditions were not periodic; in order to get around this problem and to ultimately solve the partial differential Poisson's equation, I decided that the neighboring fluid cells exhibit no net charge and therefore the gradient of the electric field at the boundary of the simulation region should be zero. This allowed me enough equations and few enough variables to solve it by implementing a tridiagonal matrix inverter. This is different from a PIC code which often uses Fourier analysis to solve the periodic equation.

Another difference between my code and a PIC code arises from how I determined to add new particles to the simulation region. In a periodic region, a particle that moves off one end is put at the other side of the simulation, however, in my simulation a particle that leaves through one boundary never returns. In order to generate new particles at the end of each step, then, I devised side simulation regions outside of the region I was working with. Essentially, these micro-micro-simulation regions took the bulk velocity (and four standard deviations of that) and times *dt* one gets both units of distance and the maximum distance that a particle could travel in a time step, and so these simulations would probabilistically add particles into the simulation region that naturally would have come in from the boundaries (an acceptance/rejection scheme). This creates a constant influx of particles in both halves of the simulation region to even out with the particles leaving. This allows the simulation to really be focused on neighboring cells and what sort of flux of particles and statistical characteristics are creating between the cells.

The simulation runs well with the exception that an error was found in the equations I was using in the magnetic field solver. I had enough time to write code around the problem and so a pretty straightforward calculation of particle velocities should complete the code and correctly determine the magnetic fields. This will enable the graduate student I was working with, Peter Norgaard, to incorporate my micro-simulation into a larger-scale finite volume code to determine what effects this method can show that a standard PIC code cannot.

Lastly, I'd like to thank PPPL for its support this summer, Professor Choueiri for his help and allowing me to work in his lab, and Peter Norgaard for his guidance on the entire project—I found it very interesting and in addition to giving me a fine understanding of plasma physics, I learned a lot about how research at a graduate level is conducted.