Kinetic Plasma Simulations using PIC Method

Anatoly Spitkovsky

July 21, 2016

1 Lecture 1

There are many systems in high energy astrophysics where the mean free path is very large. This leads to nonthermal distributions of the particles and maybe acceleration of high energy particles. Typical particle acceleration processes include magnetic reconnection and shocks.

The typical way to simulate these plasmas is using the particle-in-cell (PIC) technique. We will first introduce how PIC works, then introduce gradually electrostatic codes, and then full electrodynamic codes. Finally we will introduce applications and examples.

Most of the techniques that we will talk about are as old as 40 years, but only recently there is a huge development of computing power, and it allows us to see big enough systems and find interesting things. That is why there is a resurgence in interest in this kind of simulations.

There are various length scales in the plasma of interest. We have the (typically shortest shortest scale) the plasma frequency, Debye length and the skin depth $\lambda_{\rm skin} = c/\omega_p$, and there is the Larmor frequency $\omega_c = eB/mc$. Typical time scale ordering is roughly

$$\tau_{pe} < \tau_{ce} < \tau_{pi} < \tau_{ci} < \tau_{\rm Alfven} < \tau_{cs} < \tau_{ei} \tag{1.1}$$

Kinetic simulations sits between the time scale of plasma time and cyclotron time scales of the electron and ions.

Why are we interested in this kind of kinetic simulations? As far as a single fluid is concerned, every element has only one velocity, and two counter streaming fluid has a zero net velocity. Even when we use two streams, when some streams reverse we need to describe more velocities. This is when a kinetic description becomes useful.

Debye length is the length of screening in a plasma. The more particles there are in a Debye cube, the more electric field is screened, and the more collisionless is the plasma, since they don't interact with Coulomb forces anymore.

1.1 How PIC Works

What we are trying to solve is the full Vlasov-Maxwell system for the distribution function $f(\mathbf{x}, \mathbf{v}, t)$. Since this equation is 6 dimensional in nature, a direction solution is very expensive. Instead we can solve an equation that involves many individual particles. Here there are two philosophies people use to think about these particles. We could think of them as real particles smeared around an area, or we could think of each particle as a characteristic of this set of equations. The shape of the particles is a way to couple the particles to the electromagnetic field, sampling the force on the particle as well as giving moments to the source of the Maxwell equations.

We discretize the distribution function to individual particles

$$f = \sum_p f_p(\mathbf{r}_p, \mathbf{p}_p, t) \tag{1.2}$$

The particles experience electromagnetic force which can be sampled from a grid where the electric and magnetic fields live on. We solve the particle equations directly, and solve the Maxwell equation on a grid.

There is a subtlety here. There are two interpolations to be made. One needs to interpolate from the grid to the particles to get the force on the particles, and from the particles to the grid to provide the moments as sources to the Maxwell equations. These operations need to be done in the same way, to avoid spurious self-force on the particles.

Since Coulomb forces is short ranged, we want to avoid large kicks between the particles when they come very close. So we want to use finite-sized particles, which considerably reduces Coulomb interaction.

The typical flow of PIC is as follows: One loads the particle distribution and solve the particle equations of motion. Then one extrapolate to the grid, solve the Maxwell equations on the grid, then interpolate the fields onto the particles to solve the next step of the particle equations.

The first PIC method dates back to 1950s when John Dawson began 1D electrostatic sheet experiments in Princeton, and later in UCLA. In 1D one can solve for the electric field directly at all particle positions so it was easy to calculate the dynamics. Later in 1965 Hockney and Buneman introduced grids and direct Poisson solve on the grid for the electric field. In the 1970s Langdon developed the theory of electrostatic PIC, and first electromagnetic codes were developed. Then in 1980s-90s 3D EM PIC takes off.

1.2 Electrostatic Codes

We want to solve the Maxwell equations. When the timescale of the system is much larger than the light crossing time, magnetic fields are static, so we can simply solve the Poisson equation for the static electric field. One way to do this is to use FFT solvers.

The major criteria to choose an algorithm for integration are convergence, accuracy, stability, and efficiency. We also need to be aware of numerical/artificial dissipation in the code, and how well it conserves conserved quantities like energy.

The method of integration of the equation of motion is typically leap-frog. Positions are usually defined at integer steps and velocities at half-integer steps. The precision is second order in time. It is a simplectic method so it is stable and can be evolved backwards in time as well. However there is a stability criterion which is $\omega_p \Delta t \leq 2$, where ω_p is the frequency of some hypothetical periodic motion of the particle. Since the fastest frequency in a plasma is the electron plasma frequency, this limitation requires us to resolve it.

As we talked about, charge assignment and force interpolation should be symmetric. The simplest way to do both is to use "nearest grid point" approach, where we deposit all of the charge of a particle to the nearest grid point. However this will introduce significant noise since there will be discontinuous changes when particle moves from cell to cell. A better way is to use "cloud in cell", which linearly interpolates the particle charge between two adjacent cells. Higher order shape function typically filters out high-frequency noises in the deposited distribution. This can be shown when deriving the full dispersion relation of the plasma on a grid.

High-frequency noises are important especially when we do FFT on the field equations. High k modes in the charge distribution will directly feed into the potential, and eventually affect our solution. This is because we have a finite grid, which looks like a crystal. A crystal has Brillouin zones, which show up on a grid too. It turns out that higher k modes will leak into lower k modes (aka "first Brillouin zone") due

to the periodic nature of the k space. This is called aliasing, and it usually leads to extra heating in the problem and in the worst case destabilize the system.

Major noise in PIC mainly comes from the aliasing effect, which can be reduced in two ways: increasing number of particles, or increase the order of shape function. The extent of using more particles only reduces noises at a rate of $1/\sqrt{N}$. However increasing the order of shape functions increases the number of floating point operations on all particle calculations. It is not always obvious which one wins.

If Debye length is unresolved on the grid, aliasing will heat up the plasma until Debye length is marginally resolved. This is called numerical heating. We don't really need to reduce the fluctuations to ther correct values, but merely to levels which does not cause significant artificial heating in the plasma.

When extending to 2D, usually area weighting is used. There are also many other choices of shape factors which could work equally well.

1.3 Electromagnetic Codes

We will use TRISTAN-MP as an example. The interpolation/extrapolation schemes are the same. We now have a few more steps to take because we need to solve full Maxwell equation. We use the so called Finite-Difference-Time-Domain (FDTD) method, on a Yee (1966) staggered mesh. We decenter all components of the electromagnetic field, so that magnetic fields live on the centers of the faces of the grid, while electric fields live on the edges of the grid. This makes sense because if we take the integral of $\Delta\mathbf{B}$ over a surface it is equal to to the circulation of the electric field around the area.

Due to the leap-frog nature, the E and B fields are decentered in time too. Usually this poses no problem in PIC. Another good thing about decentering is that it preserves $\nabla \cdot \mathbf{B}$ to machine precision.

We can find the numerical dispersion relation for a plane wave on a grid, we have

$$\sin^2(\omega\Delta t/2) = \frac{\Delta t^2}{\Delta x^2}\sin^2(k\Delta x/2) \tag{1.3}$$

therefore at higher wave number there will be phase error on the wave, and since the phase velocity/group velocity can be slower than speed of light, particles traveling faster than that will emit numerical Cherenkov radiation.

When we introduce special relativity to the equation of motion to the leap-frog scheme, the $\mathbf{v}\times\mathbf{B}$ term poses a difficulty of time-centering

$$\frac{\mathbf{u}^{t+\Delta t/2} - \mathbf{u}^{t-\Delta t/2}}{\Delta t} = \frac{q}{m}\left(\mathbf{E}^t + \frac{\mathbf{u}^{t+\Delta t/2} + \mathbf{u}^{t-\Delta/2}}{2\gamma^t} \times \mathbf{B}^t\right) \tag{1.4}$$

The Boris scheme solves this issue and will be explored in the homework. One good thing about the Boris scheme is that we can overstep the magnetic rotation without stability issues. Larmor radius will maybe get screwed up, and particles will bounce around the magnetic field, but the numerical scheme will not blow up.

1.4 Current Deposition

How about the Poisson equation? On parallel computers this won't scale very well. In fact if we are very careful about the way we deposit current, we can keep Gauss's law at all times if it is satisfied at the first moment and we are not violating charge conservation on the way. By "careful way of depositting current", we mean satisfying the continuity equation.

There are several ways of depositing current that guarantees charge conservation. If we just use volume-weighting on the current the continuity equation is not observed. One method we use is the so-called Buneman method due to Villasenor & Buneman (1992). The idea is to count what is the "volume current" through the appropriate faces. In 2D one needs to know if the particle crosses 4 or 7 boundaries, and in 3D one needs to account for more possibilities. Higher order schemes are also possible (Esirkepov 2001, Umeda 2004).

2 Lecture 2

Last time we covered a lot of the basis of PIC code, up to electromagnetic codes and how we integrate particles using Boris pusher and deposit current.

Since with charge-conserved deposition one can have Poisson equation satisfied at all times, we only need to ensure it is satisfied in the initial condition. One can either start with $E = 0$ and all electrons on top of ions, which is the most common way. We could also have more elaborate states if we insert an initial Poisson solve, especially if we only need to do it once.

If we really initialize with charge imbalance but don't add electric field, for example only electrons in the initial state, then the code will assume there is an equal number of ions in the background grid, which is not moving and provides a static background charge density. This is an easy way to initialize plasma oscillations.

An improvement to the Buneman method is the zig-zag method proposed by Umeda (2003). The essence of this method is simply to replace the particle trajectory when it goes through cell boundaries by it going through the cell corner, or mid-points of the cell edge. This is an optimization over the Buneman method but it has the same problem when going to higher order. Esirkepov scheme works well for higher order but it is much more expensive. The alternative is to give up charge conservation but just solve Poisson equation every timestep to correct the electric field.

One trick to reduce noise is to filter current over the grid to smooth out the high frequency aliases. In Fourier code one can simply truncate the wave in k space, but in grid-space code one can't do that. In finite-difference code one does "digital filtering" where we replace ϕ_j with

$$\frac{W\phi_{j-1} + \phi_j + W\phi_{j+1}}{1 + 2W} \tag{2.1}$$

With $W = 0.5$ the digital filter approaches zero at $k\Delta x = \pi$. This is equivalent to convoluting the result with $\cos^2(\theta/2)$ where $\theta = k\Delta x$. One can apply this N times. Another thing one can do is to use $W = -1/6$ to "compensate" which takes some of the power we attenuated at high k and puts them back.

In relativistic codes there is a nasty instability called the numerical Cherenkov instability. Remember our numerical dispersion relation has smaller phase/group speed at high k. Normally the deviation is not large, however in a relativistic simulation the particles can fly faster than the speed of light in the medium. What one sees when one simulates a relativistic neutral beam is that the beam heats up and stops on the grid. Things that help is to use higher order FDTD schemes (e.g. 4th spatial order), and it postpones the onset of the instability quite a lot. Filtering also helps.

Lets talk a bit about boundary conditions. Simplest boundary condition is periodic boundary where we just copy the field from one end to the ghost zones of the other end. The other simple condition is perfectly conducting walls, which kills tangential E and perpendicular B. Another boundary condition which is very helpful is the so-called open boundary or absorbing layer. One wants particles to freely stream out, and waves not to reflect. Usually we simply designate a region of the grid to have a finite

conductivity. Another way is to use the so-called perfectly matched layer, where we imagine a medium with separate conductivities for electric and magnetic field. However this only works with vacuum and if particles escape through the layer it will charge up. One might need to neutralize this charge density in some way. Another way was proposed by Lindman (1975) called transmitting wall. It adjusts the E and B fields at the wall to match outgoing waves.

Another thing people implement in codes is called "moving window". This means the whole simulation box is moving at a high speed (usually at c). This is useful in beam-plasma interaction study and collisionless shocks. One could also inject particles at the boundary, with injector possibly moving.

A quick note about parallelization. We use MPI to decompose the domain into subdomains and communicate between them. One might run into trouble with load balancing, especially when particles are mostly in one subdomain. In uniform plasma problems one is usually okay, but in shocks or reconnection problems this can be a big problem. One thing we can do is to have dynamic reconfiguration of decomposition. Or one could use some scheme to vary the particle weights in dense regions so that one particle represents many.

Beyond the standard electromagnetic PIC, one can add curvilinear coordinates. One can add an extra term in the evolution to add radiation loss and emission of non-thermal radiation to the particles, e.g. radiation reaction force given by Landau-Lifschitz, inverse Compton over some external source background, adding pair creation and QED effects, GR effects, etc.

Another variation of the theme is hybrid PIC simulations. Since electron scales are typically much smaller than ion scales, one can treat electron as a fluid and model ions using PIC. See e.g. Lipatov (2002) a book on the method, also Kunz's paper (2014) on this topic.