5 – Tutorial: Electrostatic Particle-in-Cell simulation

- Difference equations
- Prerequisites
- Project I: Thermal equilibrium
- Project II: Nonlinear plasma waves
- Project III: Landau damping

5 – PIC simulation

The particle-in-cell code currently represents one of the most important plasma simulation tools. It is particularly suited to the study of *kinetic* or *non-Maxwellian* effects. The simplest variation of this technique is a ‘1D1V’-configuration: 1 space coordinate plus 1 velocity, the numerical behavior of which was first considered by John M. Dawson, one of the pioneers of kinetic plasma simulation, some forty years ago. Advances in computing power have enabled the PIC method to be extended to fully electromagnetic, 3D3V simulation. For tutorial purposes, however, we stick to the simplest possible reduced geometry for the sample PIC codes offered here, which offer both an apprenticeship in ‘professional’ plasma simulation, as well as plenty of insight into the behavior of laser-produced plasmas.

5 – Difference equations

The heart of an electrostatic code is based on a cyclic iteration of the following difference equations:

Particle pusher:

$$\begin{aligned} v_i^{n+\frac{1}{2}} &= v_i^{n-\frac{1}{2}} + \frac{q_i}{m_i} E_i^n \Delta t, \\ x_i^{n+1} &= x_i^n + v_i^{n+\frac{1}{2}} \Delta t. \end{aligned} \tag{9}$$

Density gather:

$$\begin{aligned} \rho_j^{n+1} &= \sum_i q_i S(x_i - x_j), \\ & S = 1 - \frac{\mid x_i - x_j \mid}{\Delta x}. \end{aligned} \tag{10}$$

Field integration:

$$E_{j+\frac{1}{2}}^{n+1} = E_{j-\frac{1}{2}}^{n+1} + \rho_j^{n+1} \Delta x. \tag{11}$$

5 – Code structure

The routines for the above three steps — PUSH, DENSITY and FIELD, respectively — are all provided in the script espic.py. The control loop at the bottom of the script calls each of these routines in turn, along with some initialization (LOADX, LOADV) and diagnostic routines (DIAGNOSTICS, PLOTS, HISTORIES).

Normalization:

$$t \rightarrow \omega_p t; \qquad x \rightarrow \omega_p x/c; \qquad v \rightarrow v/c$$
$$E \rightarrow eE/m\omega_p c; \qquad \phi \rightarrow e\phi/mc^2; \qquad n_{e,i} \rightarrow n_{e,i}/n_0$$

5 – Prerequisites: Python version

- Scientific Python download site:
  `http://www.scipy.org/install.html`
- Minimum packages (usually present or post-installable in Linux distribution): python (≥ 2.7), ipython, numpy, scipy, matplotlib ≥ 1.2)
- Recommended package bundle for Windows users: anaconda (all inclusive + Spider IDE)
- Script: espic.py
- Execution: python espic.py (Linux/command shell) or run from Spider GUI (Windows)

5 – Project II: Nonlinear plasma waves

1. A longitudinal plasma wave manifests itself as a disturbance in the electron density: $n_e = n_0 + n_1(x,t)$. We can excite such a wave numerically by displacing the initial positions of the particles. If the density perturbation at $t = 0$ is to have the form $n_1 = A \sin kx$, one can show (Birdsall & Langdon, Section 5-2) that the particles need to be displaced by an amount:

$$\delta x \equiv x(t) - x_0 = \frac{A}{n_0 k} \cos kx.$$

2. Make the necessary modifications to the code (for example in routine `loadx()` in order to initialize such a plasma wave structure.
Suggested parameters are as follows:

```
grid_length=4*pi a0=0.1          vte=0.05
rho0=1.0         dt=0.1          ngrid=100
npart=2000       nsteps=150      igraph=int(pi/dt/4)
```

Note that for large amplitudes A, some particles may be displaced across the simulation boundary, so an additional call to PARTICLE_BC after LOADX may be necessary to avoid 'losing' particles at the edges.

3. How can the statistics or signal:noise ratio in the electron density n_e be improved?

4. (optional) Compare the wave amplitudes and relative phases n_1, E_1, ϕ_1 with those expected from linear theory.

5. How well does the code conserve energy? Examine ΔU_{tot} as a function of Δt.

6. Now try increasing the wave amplitude, e.g.: $A > 0.2$. What happens to the waveforms of density and field? At later times (a few plasma periods, depending on the plasma parameters), one should be able observe some significant wave-particle interaction (trapping and/or wave breaking).

7. Investigate this process further by varying `A, vte, rho0` etc. Hint: it is particularly helpful here to look at the particle phase space (v, x) and velocity distribution $f(v_x)$ (see Project I).

8. For very large amplitudes, the electron velocities will eventually become relativistic ($v \sim c$). Since there are no magnetic fields in the model, this can easily be accomodated in PUSH by substituting the proper velocity, $u = \gamma v$ for v. Care should be taken, however, to incorporate this relativistic upgrade in the code diagnostics, such as the kinetic energy $U_{kin} = mc^2(\gamma - 1)$, phase space (p_x, x), and distribution functions $f(p)$.