OSIRIS Collision Package Overview

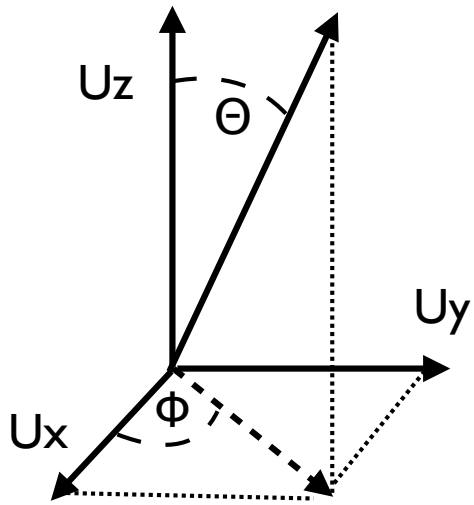
Josh May
September 2017
OSIRIS Workshop
UCLA, Los Angeles CA

Why collisions are needed

- PIC makes simulating plasmas computationally possible by restricting field to a grid and using finite-size macroparticles.
- Effects smaller than a cell size (rarely smaller than a Debye length) can't be resolved.
- Pure PIC has no concept of real number density; one simulation represents a range of physical systems, up to a scaling factor.
- Discrete particles effects are simulated using Monte Carlo (stochastic) methods, assuming the information which was discarded was essentially random
- (Note PIC *DOES* have collisions, but they're distinct in scale and behavior from physical collisions.)

Monte Carlo Binary Coulomb Collision Model

- Particles are randomly paired, and collided off each other using fluid and particle data
- In center-of-momentum frame, collision is a rotation in p-space
- Θ is a random variable with variance given by collision frequency (1); Φ is random and uniform
- By randomly sampling many pairs, approximates an integral over the distribution function, and so solves the collision term in the Landau form (2).
- Assumes Θ << 1



(1) $$< tan(\Theta) >^2 = \frac{2\pi e_a^2 e_b^2 \, n \, ln(\lambda)}{\mu_{ab}^2 u^3} \Delta t$$

(2) $$\frac{\partial f}{\partial t}\Big)_{\text{col}} = -\sum_\beta \frac{\partial}{\partial v_j} \frac{2\pi e_a^2 e_b^2 \, ln(\lambda)}{m_a} \int d\mathbf{v}' \left[\frac{\delta_{jk}}{u} - \frac{u_j u_k}{u^3}\right] \left[\frac{f_a}{m_b} \frac{\partial f_b(\mathbf{v}')}{\partial v_k'} - \frac{f_b(\mathbf{v}')}{m_a} \frac{\partial f_a}{\partial v_k}\right]$$

T. Takizuka and H. Abe. "A Binary Collision Model for Plasma Simulation with a Particle Code" Journal of Computational Physics, 25:205-219, 1977.

Nanbu extension to large collision events

- Collision model of Nanbu
- Question is how to model many small-angle scatterings from Coulomb Collisions in a single simulated collision timestep
- Ansatz assumption that the distribution of the scattering angle is given by equation 1
- Equations 3, 4 then follow analytically; though 4 is non-invertible, various approximations in different regimes can be used to efficiently numerically solve these

(1) $$\left\langle \sin^2 \frac{\chi_N}{2} \right\rangle = \frac{1}{2}(1 - e^{-s})$$

(2) $$s = 4\pi \ln\Lambda \left(\frac{q_1 q_2}{\mu_{12}} \right)^2 n_2 v_{rel}^{-3} \Delta t \quad (s \sim \nu_{coll} \Delta t)$$

(3) $$cos\chi = \frac{1}{A} \ln\left(e^{-A} + 2U \sinh A\right)$$

(4) $$\coth A - A^{-1} = e^{-s}$$

K Nanbu. “Theory of cumulative small-angle collisions in plasmas.” Physical Review E, 55(4):4642–4652, April 1997.

Pérez Relativistic and Low Temperature Corrections

- Equation 2 is not however relativistically correct (no distinction is made between p and mv, for instance)
- Pérez et al. solved this by using a relativistic invariant to arrive at equation 5
- Working in momentum space the collision remains a rotation, and so equations 3, 4 remain valid with the new s

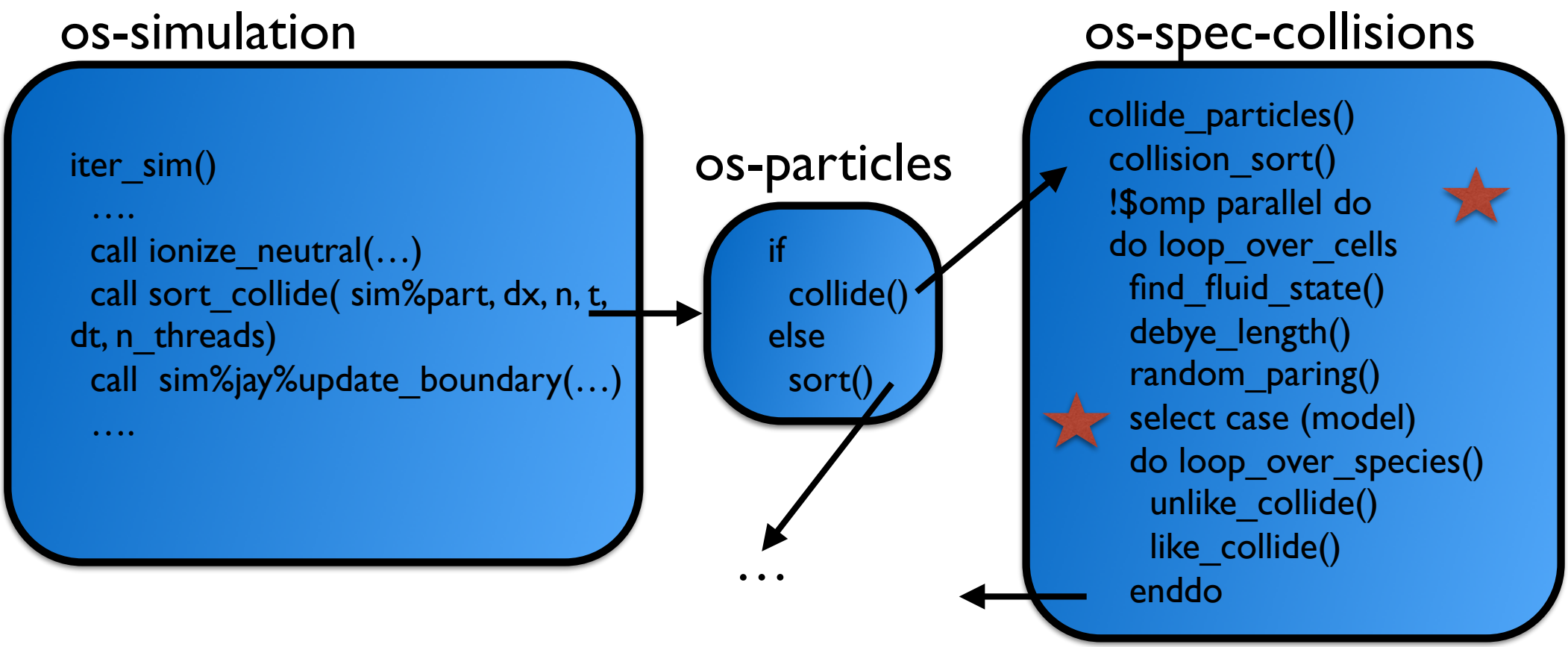
(5) $$s_{12} = \frac{4\pi n_2 \Delta t \ln\Lambda (q_1 q_2)^2}{c^4 m_1 \gamma_1 m_2 \gamma_2} \frac{\gamma_C p_1^\star}{m_1\gamma_1 + m_2\gamma_2} \left(\frac{m_1\gamma_1^\star m_2 \gamma_2^\star}{p_1^{\star 2}} c^2 + 1 \right)^2$$

- This model also breaks down for very low speeds/temperatures - the collisionality diverges to infinity
- Equations 6 gives a mean free path equal to the particle spacing; by using this as a lower bound the divergence is avoided

(6) $$s' = \left(\frac{4\pi}{3}\right)^{1/3} \frac{n_1 n_2}{n_{12}} \bar{\Delta t} \frac{m_1 + m_2}{\max\left(m_1 n_1^{2/3}, m_2 n_2^{2/3}\right)} v_{rel} \qquad s = \min(s, s')$$

F Pérez, L Gremillet, A Decoster, M Drouin, and E Lefebvre. "Improved modeling of relativistic collisions and collisional ionization in particle-in-cell codes." Physics of Plasmas, 19(8):083104, 2012

Collision step in main iteration loop



OpenMP parallelization for all collision cells

Collision loop in each cell

```
unlike_collide_perez()
  <find correction for uneven particle weights>
  <find (weighted) average relative velocity>
  log_lambda = …
  s_first = … !shared part of collision frequency
  do loop_over_pairs
    relative_velocity = … ! one-particle-at-rest frame
    <boost momenta to center-of-momentum frame>
    s = … ! collision frequency for pair
    <invert Nanbu equation to get collision angle>
    <calculate rotation in CoM>
    <update momenta stochastically based on weighting>
    <boost momenta back to lab frame>
```

Five input parameters are required

- Needed input parameters:
  - Actual charge density for n=1 (cgs)
  - Which species to collide
    - Self-collide?
  - Charge state of ion
  - Collision timestep (0, the default, doubles as flag to turn collisions off)

```
1   simulation
2   {
3     n0 = 1.1d21,
4   }
5
6   species
7   {
8     if_collide = .true.,
9     if_like_collide = .false.,
10    q_real = 1,
11  }
12
13  collisions
14  {
15    n_collide = 1,
16    ! Defaults
17    collision_model = "Perez",
18    nx_collision_cells(1:2) = 1, 1,
19    coulomb_logarithm_automatic = .true.,
20    ! Optional (not default)
21    perez_low_temp_correction = .true.,
22  }
23
```

Other inputs to know

- Defaults to be aware of
  - Perez collision model
  - Collision cell size equals PIC cell size
  - Calculate Coulomb log in each cell
- One option one might want: Low temperature correction

```
1   simulation
2   {
3     n0 = 1.1d21,
4   }
5
6   species
7   {
8     if_collide = .true.,
9     if_like_collide = .false.,
10    q_real = 1,
11  }
12
13  collisions
14  {
15    n_collide = 1,
16    ! Defaults
17    collision_model = "Perez",
18    nx_collision_cells(1:2) = 1, 1,
19    coulomb_logarithm_automatic = .true.,
20    ! Optional (not default)
21    perez_low_temp_correction = .true.,
22  }
23
```

Things you have to worry about a little

- Numerical heating (see Paulo Alves)
- Ratio of collision cell to MPI node
- Good resolution
  - Depends on the problem, no hard rules
  - Size of collision gradients (nx_collision_cells)
  - Resolving the collision frequency (n_collide)
  - Representing the distribution function (nx_collision_cells, ppc)
  - That said, the algorithm is fairly forgiving

Things you don't have to worry about

- Over-resolving the collision frequency
- Relative size of macro-particles
- Multiple equivalent species
- Highly different densities
- Relativistic effects
- Conservation of energy and momentum *on average* (except for heating)

Computational Cost

- It is fairly expensive
- Depends on system, but roughly 2X cost for two species colliding at each timestep
- Currently only vectorization is OpenMP (which works quite well!); adopting for GPGPU or KNL seems promising, but future work
- SIMD within each collision cell is possible too, but is challenging

Take Away Message

- For the general user, just a few parameters need to be input and the collision package will take care of the rest. Be careful of numerical heating, but otherwise you can't really mess up.
- With a little more care, you may improve the computational performance without losing physics.
- For the developer, both the package and the individual models are well isolated; easy (and encouraged) to add to or modify.
- Better vectorization, impact ionization are obvious goals