Binary collisions in SMILEI

F. Pérez

March 23, 2015

Relativistic binary collisions between particles have been implemented in SMILEI with the same scheme as the one developed for the code CALDER. The following references describe the physics and numerics of this implementation.

Ref. [1] gives an overview of the technique.
Refs. [2] and [3] give the original technique from which Ref. [1] was developed.
Following Refs. provide additional information.

[1] F. Pérez *et al.*, Phys. Plasmas **19**, 083104 (2012)

[2] K. Nanbu, Phys. Rev. E **55**, 4642 (1997)

[3] K. Nanbu and S. Yonemura, J. Comput. Phys. **145**, 639 (1998)

[4] Y. Sentoku and A. J. Kemp, J. Comput. Phys. **227**, 6846 (2008)

[5] Y. T. Lee and R. M. More, Phys. Fluids **27**, 1273 (1984)

[6] N. E. Frankel, K. C. Hines, and R. L. Dewar, Phys. Rev. A **20**, 2120 (1979)

Contents

1 How to add collisions in the input file

To have binary collisions in SMILEI, add one or several copies of the following block in the input file:

```
Collisions(
        species1 = ["species_name", "species_name", ...],
        species2 = ["species_name", "species_name", ...],
        coulomb_log = ln Λ,
        debug_every = every
)
```

Each *species_name* must be the name of an existing species. This name can be found in the input file inside any block `Species(...)` under the keyword `species_type`.

The collisions will occur between (1) all species under the group `species1` and (2) all species under the group `species2`. For instance, to have collisions between `electrons1` and `ions1`, use

```
        species1 = ["electrons1"],
        species2 = ["ions1"]
```

Other example:

```
        species1 = ["electrons1", "electrons2"],
        species2 = ["ions"]
```

will collide all electrons with ions.
WARNING: this does not make `electrons1` collide with `electrons2`.

The two group of species have to be completely different OR exactly equal.
In other words, if `species1` is not equal to `species2`, then they cannot have any common species. If the two groups are exactly equal, we call this situation "intra-collisions".

The optional value $ln\Lambda$ must be a float.

- If $ln\Lambda = 0$, the Coulomb logarithm is automatically computed for each collision (default).
- If $ln\Lambda > 0$, the Coulomb logarithm is equal to this value.

The optional value *debug_every* must be an integer: the number of timesteps between each output of information about collisions. If absent or zero, there will be no outputs. See section 4

2 Description of the binary collision scheme

Collisions are calculated at each timestep.

For each collision block (given in the input file):

- For each particle cluster:

  - If *intra-collisions*

    Create one array containing the indices pointing to all particles of the species group.
    Shuffle the array.
    Split the array in two halves.

  - Otherwise

    Create two arrays containing the indices pointing to all particles of each species group.
    Shuffle the largest array. The other array is not shuffled.

  => The two resulting arrays represent pairs of particles (see algorithm in Ref. [3]).

  - Calculate a few intermediate quantities:

    Particle density n_1 of group 1.
    Particle density n_2 of group 2.
    "crossed" particle density n_{12} (see Ref. [1]).
    Other constants.

  - For each pair of particles:

    - Calculate the momenta in the center-of-mass (COM) frame.
    - Calculate the coulomb log if requested (see Ref. [1]).
    - Calculate the collision parameter s and its correction at low temperature (see Ref. [1]).
    - Pick the deflection angle (see Ref. [2]).
    - Deflect particles in the COM frame and go back to the laboratory frame.

3 Test cases

3.1 Beam relaxation

An electron beam with narrow energy spread enters an ion background with $T_i = 10$ eV.
The ions are of very small mass $m_i = 10 m_e$ to speed-up the calculation.
Only e-i collisions are calculated.
The beam gets strong isotropization => the average velocity relaxes to zero.

Three figures show the time-evolution of the longitudinal $\langle v_\parallel \rangle$ and transverse velocity $\sqrt{\langle v_\perp^2 \rangle}$

- Figure 1: initial velocity = 0.05, ion charge = 1
- Figure 2 : initial velocity = 0.01, ion charge = 1
- Figure 3 : initial velocity = 0.01, ion charge = 3

Each of these figures show 3 different blue and red curves which correspond to different ratios of particle weights: 0.1, 1, and 10.

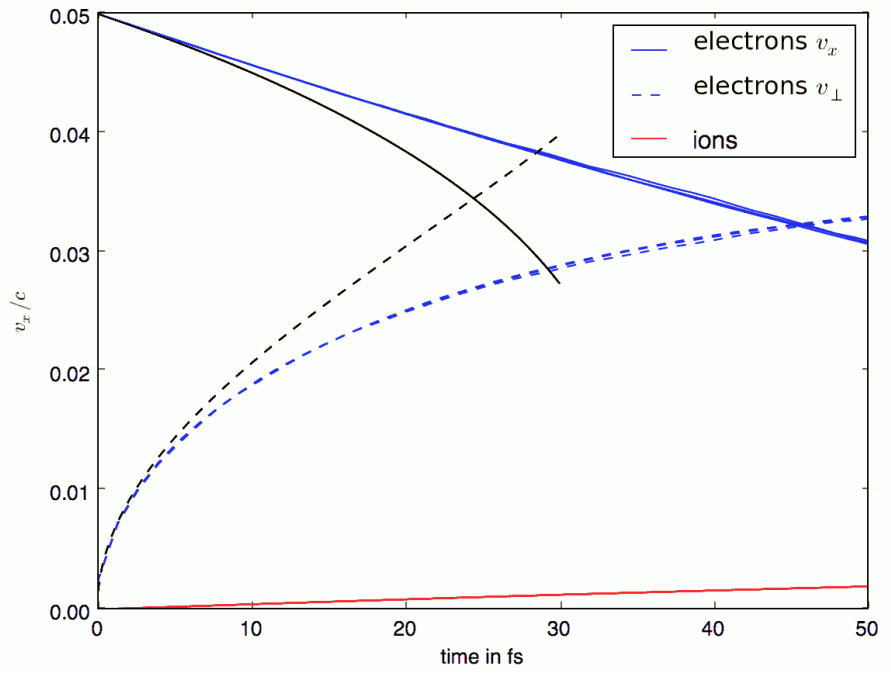

Figure 1: Relaxation of an electron beam. Initial velocity = 0.05, ion charge = 1.

The black lines correspond to the theoretical rates taken from the NRL formulary:

$$\nu_\parallel = -\left(1 + \frac{m_e}{m_i}\right)\nu_0 \quad \text{and} \quad \nu_\perp = 2\,\nu_0 \quad \text{where} \quad \nu_0 = \frac{e^4\, Z^{\star 2}\, n_i \ln\Lambda}{4\pi\epsilon_0^2\, m_e^2\, v_e^3}$$

The distribution is quickly non-Maxwellian so that theory is valid only at the beginning.

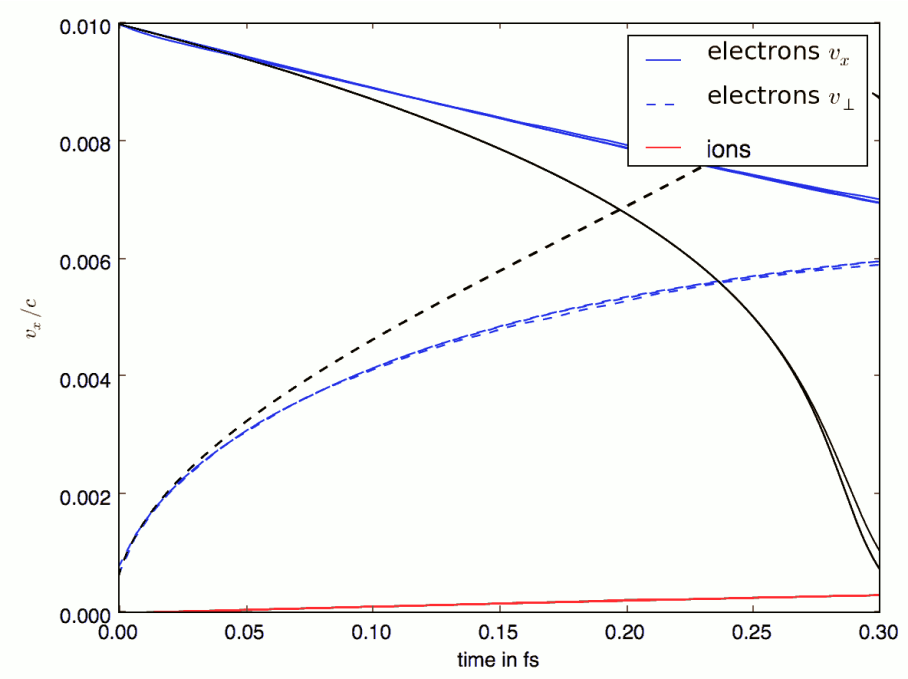

Figure 2: Relaxation of an electron beam. Initial velocity = 0.01, ion charge = 1.

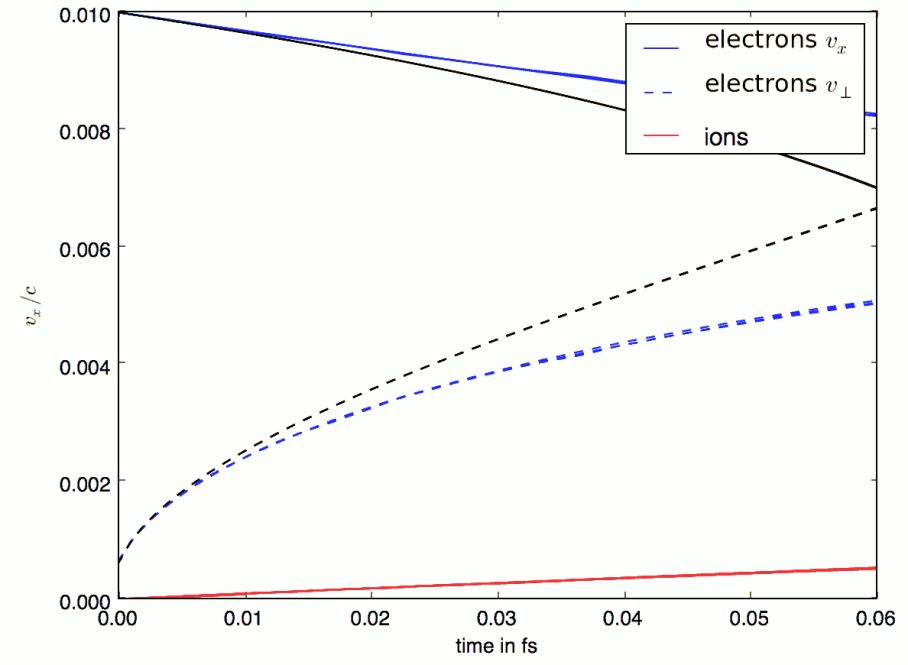

Figure 3: Relaxation of an electron beam. Initial velocity = 0.01, ion charge = 3.

3.2 Thermalization

A population of electrons has a different temperature from that of the ion population.
Through e-i collisions, the two temperatures become equal.
The ions are of very small mass $m_i = 10 m_e$ to speed-up the calculation.
Three cases are simulated, corresponding to different ratios of weights : 0.2, 1 and 5.
They are plotted in Figure 4.

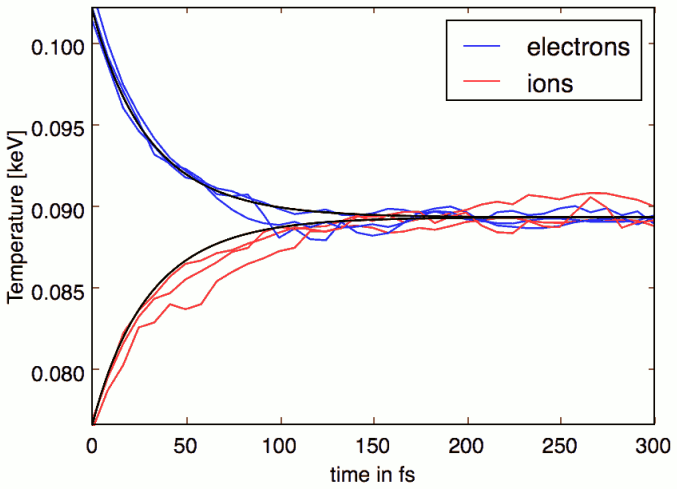

Figure 4: Thermalization between two species.

The black lines correspond to the theoretical rates taken from the NRL formulary:

$$\nu_\epsilon = \frac{e^4 \, Z^{\star 2} \sqrt{m_e m_i} \, n_i \ln \Lambda}{8 \epsilon_0^2 \, (m_e T_e + m_i T_i)^{3/2}}$$

3.3 Temperature isotropization

Electrons have a longitudinal temperature different from their transverse temperature.
They collide only with themselves (intra-collisions) and the anisotropy disappears as shown in Figure 5.

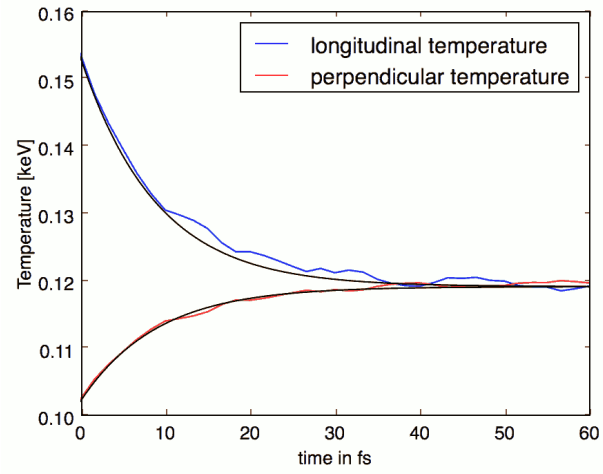

Figure 5: Temperature isotropization of an electron population.

The black lines correspond to the theoretical rates taken from the NRL formulary:

$$\nu_T = \frac{e^4\, n_e \ln\Lambda}{8\pi^{3/2}\epsilon_0^2\, m_e^{1/2}\, T_\parallel^{3/2}} A^{-2}\left[-3+(3-A)\frac{\operatorname{arctanh}(\sqrt{A})}{\sqrt{A}}\right] \quad \text{where} \quad A = 1-\frac{T_\perp}{T_\parallel}$$

3.4 Maxwellianization

Electrons start with zero temperature along y and z. Their velocity distribution along x is a boxcar.
They collide only with themselves and the boxcar becomes a maxwellian as shown in Figure 6.

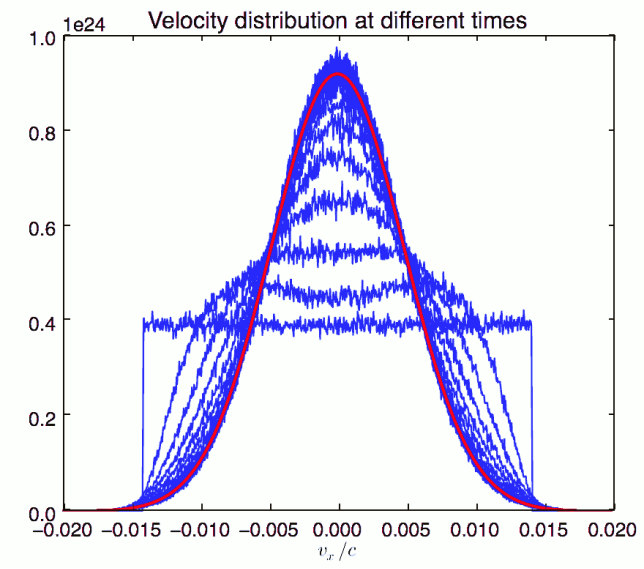

Figure 6: Maxwellianization of an electron population. Each blue curve is the distribution at a given time. The red curve is an example of a gaussian function.

3.5 Stopping power

Test electrons (very low density) collide with background electrons of density 10 n_c and T_e = 5 keV.
Depending on their initial velocity, they are slowed down at different rates, as shown in Figure 3.5.

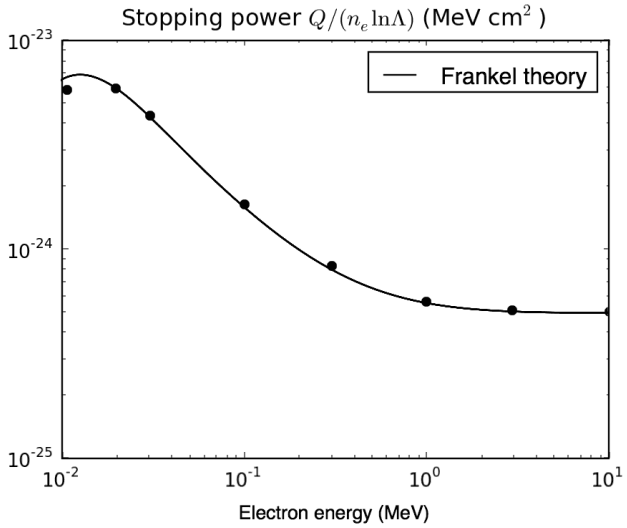

Figure 7: Stopping power of test electrons into a background electron population. Each point is one simulation. The black line is Frankel's theory (Ref. [6]).

3.6 Conductivity

Solid-density Cu is simulated at different temperatures (e-i equilibrium) with only e-i collisions.
An electric field of E = 3.2 GV/m (0.001 in code units) is applied using two charged layers on each side of the solid Cu.
The electron velocity increases until a limit value v_f.
The resulting conductivity $\sigma = en_e v_f / E$ is compared in Figure 8 to the models in Refs. [5] and [1].

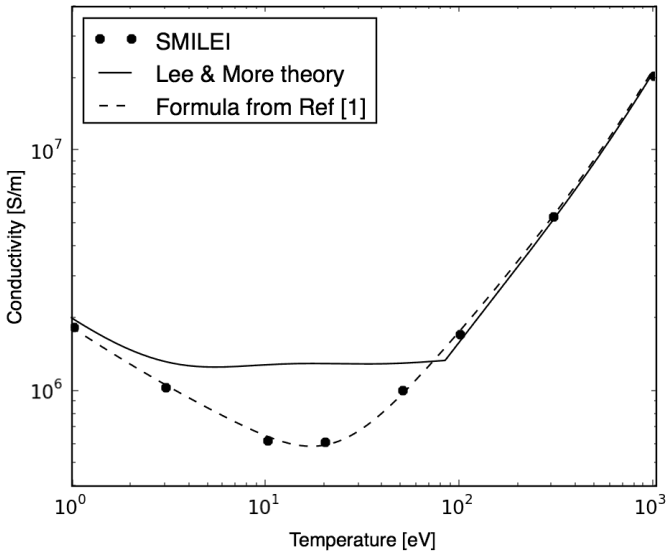

Figure 8: Conductivity of colid-density copper. Each point is one simulation.

4 Collisions debugging

Using the parameter `debug_every` in a `Collisions(..)` group will create a file with info about these collisions. These information are stored in the files "Collisions0.h5", "Collisions1.h5", etc.

The *hdf5* files are structured as follows:
> One HDF5 file contains several groups called "t********" where "********" is the timestep.
> Each group "t********" contains several arrays, which represent quantities *vs.* space.

The available arrays are:

- s: defined in Ref. [1]: $s = N\langle\theta^2\rangle$, where N is the typical number of real collisions during a timestep, and $\langle\theta^2\rangle$ is the average square deviation of individual real collisions. This quantity somewhat represents the typical amount of angular deflection accumulated during one timestep.
  It is recommended that $s < 1$ in order to have realistic collisions.
- *coulomb_log*: average Coulomb logarithm.
- *debyelength*: Debye length (not provided if all Coulomb logs are manually defined).

The arrays are all one-dimensional: they are in the same order as the *clusters* or *patches*.
You have to figure out by yourself how to convert that to (x, y, z) !