IBSimu Particle Diagnostic

Working Draft

Duccio Marco Gasparri

2020-11-30

1 variables

- m [kg] particle mass (provided in u)
- q [J] charge of beam particle (provided in multiples of e)
- J [A/m$\hat{2}$] beam current density
- E [J] mean energy (provided in eV)
- Tp [J] parallel temperature (provided in eV)
- Tt [J] transverse temperature (provided in eV)
- $(x_1, r_1), (x_2, r_2)$ [m] beam emission line vectors
- N number of particles
- IQ [A] (A/m?) particle current
- v [m/s] from E and m

2 Setup

2.1 Geometry

Relevant IBSimu files:

- geometry.hpp
- geometry.cpp
- mesh.hpp

- mesh.cpp

Relevant client files:

- configuration-setup.hpp
- configuration-setup.cpp

Parameters:

- mesh-cell-size-h [eg. 0.5e-4]
- origin-x
- origin-y
- origin-z
- geometry-start-x [UNUSED]
- geometry-start-y [UNUSED]
- geometry-start-z [UNUSED]
- geometry-size-x
- geometry-size-y
- geometry-size-z

The class Geometry creates the mesh of the simulation starting from the CAD files and/or function definitions.

The parameters origin-x, origin-y and origin-z are used to move the reference frame relative to the bbox in the DXF file [see mesh.hpp line 70-73, mesh.cpp line 60].

The physical length/height/depth of the simulation is size-x/y/z * mesh-cell-size. Specifically:

$$size - x = (int)floor(geometry - size - x/mesh - cell - size - h) + 1$$

$$x_{max} = origin - x + mesh - cell - size - h * size - x$$

3 ParticleDataBaseCylImp::add_2d_beam_with_energy

Function ParticleDataBaseCylImp::add_2d_beam_with_energy (file: *particle-databaseimp.cpp*, line: 968) is used to add a beam of N particles with average

energy E to a cylindrical geometry.

The charge q is provided by the user and is set constant for all the particles.

The beam emission line norm s [m] is defined:

$$s = \sqrt{(x_2 - x_1)^2 + (r_2 - r_1)^2} \tag{1}$$

The current IQ [A] is set for each particle as follows:

$$IQ = \frac{2\pi s J}{N}(r_1 + \frac{(r_2 - r_1)}{N}(n + 0.5)) \tag{2}$$

where $n \in [0, 1, ..., N-1]$.

The particles are distributed evenly spaced along the emission line defined by the vectors $(x_1, r_1), (x_2, r_2)$. The particle velocities v_x, v_r [m/s] are:

$$v_x = \frac{(x_2 - x_1)}{s}\sqrt{\frac{Tt}{m}}rnd_0 + \frac{(r_2 - r_1)}{s}\sqrt{\frac{2E}{m} + (\sqrt{\frac{Tp}{m}}rnd_1)^2} \tag{3}$$

$$v_r = \frac{(r_2 - r_1)}{s}\sqrt{\frac{Tt}{m}}rnd_0 + \frac{-(x_2 - x_1)}{s}\sqrt{\frac{2E}{m} + (\sqrt{\frac{Tp}{m}}rnd_1)^2} \tag{4}$$

and

$$w = \frac{d\theta}{dt} = \frac{\sqrt{\frac{Tt}{m}}rnd_2}{r_1 + \frac{(r_2 - r_1)}{N}(n + 0.5)} \tag{5}$$

with rnd_0, rnd_1 and rnd_2 normally distributed random variables.

4 Iteration

The method ParticleDataBasePPImp::iterate_trajectories (inherits ParticleDataBaseImp, file: particledatabaseimp.hpp::641) outputs the message "Using non-relativistic iterator", clears the scharge, creates a iterators vector of size equal to the set thread, populates it with new instances of ParticleIterator¡PP¿ and waits for the iteration to be finished. Then calls scharge_finalize_linear

(file:scharge.cpp:149) Finally it publishes the particle hisories ("Particle histories").

scharge_finalize_liear prints "Finalizing space charge density map (LINEAR method)"

The core of the iteration isinside the ParticleIterator¡PP¿

5 Particle Diagnostic

The relevant functions are in files gtkparticlediagdialog.cpp (the GTK dialog file) and particlediagplot.cpp (does the actual plotting). It has the following methods:

- ParticleDiagPlot::build_data(): the function extracts the data from the ParticleDatabase
- ParticleDiagPlot::build_plot(): calls build_data(). the function extracts the data from the ParticleDatabase, set the decorations and add the graph to the _frame

Trajectories are obtained from ParticleDataBase::trajectories_at_plane() that calls ParticleDataBaseImp::trajectories_at_plane(). It requires the axis (X, Y, Z, R) and the distance from origin. It returns a vector of particle point objects ParticleP¡PP¿ (either ParticleP2D, ParticlePCyl, or ParticleP3D), each particle carrying its information about position and momentum

q is the direction normal to the diagnostic plane

DIAG_X: x position [m] DIAG_R: r position [m]

DIAG_RP: $\frac{v_r}{v_q}$

DIAG_AP: $\frac{v_\theta}{v_q}$

$$v_\theta = r\frac{d\theta}{dt}$$

Emittance:

- X axis, r-r': DIAG_R (r position), DIAG_RP ($\frac{v_r}{v_q}$)
- X axis, r-a': DIAG_R (r position), DIAG_AP ($\frac{v_\theta}{v_q}$)
- X axis, z-z': using DIAG_R, DIAG_RP, DIAG_AP and EmittanceConv class instead

The diagnostic data are obtained from the Emittance class. For Cylindrical coordinates, the class EmittanceConv converts to (x,x') equivalent emittance from the following data:

- r (radius)
- rp (radial angle)
- ap (skew angle)
- I (current)

It builds (x,x') data in a grid array of size n by m. Here the skew angle is $r\omega/v_z$, where v_z is the velocity to the direction of beam propagation.

This description from class reference list is not clear:

The conversion is done by rotating each trajectory diagnostic point around the axis in rotn steps (defaults to 100). The output grid size can be forced by setting (xmin,xpmin,xmax,xpmax) variables, otherwise the grid is autotomatically sized to fit all data.

The emittance statistics is built using original data and not the gridded data for maximized precision.

6 ToDo

Particle Types

template¡class PP¿ class Particle { std::vector¡PP¿ _trajectory; ¡- trajectories PP _x; ¡- current position } Typedef Particle¡ParticleP2D¿ Particle2D; Typedef Particle¡ParticlePCyl¿ ParticleCyl; Typedef Particle¡ParticleP3D¿ Particle3D;

Particle Types in Beam

ParticleDataBaseCylImp::add_*_beam -¿ ParticlePCyl-¿ParticleCyl ParticleDataBase2DImp::add_*_beam-¿ParticleP2D-¿Particle2D ParticleDataBase3DImp::add_*_beam-¿ParticleP3D-¿Particle3D

ParticleDataBasePPImp¡ PP ¿::trajectories_at_plane

Int ParticleP2D::trajectory_intersections_at_plane (NON CONST std::vector¡ ParticleP2D ¿ & intsc) Return the number of trajectory intersections with plane Intersection points are appended to vector intsc. Int TrajectoryRep1D::solve() Returns solutions found [Linear [0,1], Quadratic[0,1,2], Cubic [0,1,2,3]]

IBSimu DXF bug in mydxffile.cpp #define CODE_STRING(x) code (x) == 101 is not included as it is a later standard in DXF file.