PPartiC User's Manual

Jorge Gonzalez

October 8, 2020
v0.1

Contents

Chapter 1

Introduction

1.1 About PPartiC

PPartiC (Plasma Particle Code) is a simulation tool that models species in plasmas (ions, electrons and neutrals) following the trajectories of macro-particles as they move and interact between them and the boundaries of the domain. The code is currently in very early steps of development and further improvements are expected very soon.

1.2 Main Guidelines

The PPartiC enviroment aims to be a fully functional tool for the simulation of plasmas from a kinetic point of view. The main guidelines in the creation of the code are:

1. The code is open-source and freely distributed. This means the code can be used and modified by anyone. Nevertheless, the oficial version distributed in the oficial repository will be managed by the developed team, which members have to be given direct permission by the lead developer. PPartiC is distributed with a GNU General Public License v3.0 that covers all files in the repository. Moreover, there should always be an open-source free alternative for external tools required by the program (post-processing, mesh generation, input file generation ...).

2. PPartiC is coded in a *understandable* way. This means that the code is required to be written in a clear way that is easy to understand and maintain. Variables and procedure names need to be self-understanding. This ease the process of fixing bugs and improving the codes by a large team of developers. For more information, please reffer to the PPartiC Coding Style document.

3. PPartiC requires to be ease to use. Input files are required to be in a *human* format, meaning that the different options can be easily understanded without constant reference to the user guide. PPartiC is aimed to be used in a wide range of applications and by a variety of scientist: from very established ones to newcomers to the field and students.

These are foundation stones of the code and code development and should always be followed, at least for the releases in the oficial repository.

1.3 How to collaborate

Right now, development of PPartiC is closed to third parties until a stable version with the basic functionality is released. However, if you have a huge interest in the project please contact jorge.gonzalez@upm.com.

Chapter 2

Operation Scheme

2.1 The Particle Method

PPartiC uses macro-particles to simulate the dynamics of different plasma species (mainly ions, electrons and neutrals). These macro-particles represent a large amount of real particles (usually $> 10^6$ particles per macro-particle). For now own, macro-particles will be referred as just particles by abusing of language. In the evolution of these particle, external forces (as the electromagnetic field), interaction between particles (as collisions) and interaction with the boundaries of the domain.

At each time step, particles are first pushed accounting for possible acceleration by external forces. Then, the cell in which the particle ends up is located. If a boundary is encountered, the interaction between the particle and the wall is calculated. Later, collisions for the particles in the cell are carried on. This may include different collision processes for each particle. Finally, the particles properties are scattered into the mesh nodes. These properties are density, momentum and the stress tensor. Non-dimensional units are used for this, but output files are converted into dimensional units.

More in depth explanation of the different steps are given in the following sections.

2.2 Injection of new particles

PPartiC has the capability of injecting particles at different velocities, temperatures and mass flows in boundary sections defined by the user. Particles are distributed uniformly along the edge and their velocities are based on drifted Maxwellian distributions.

The injection of particles is controlled in the moduleInject module.

2.3 Pushing

Particles are pushed in the selected domain. Velocity and position are updated according to the old particle values and the external forces. All the push routines for the different geometries can be found in moduleSolver

2.3.1 2D Cylindrical

When a 2D cylindrical geometry is used (z, r), a Boris solver[**?**] is used to move particles accounting for the effect of the symmetry axis. with no external force, as it is the case for neutrals, this takes the form:

$$v_z' = v_z \tag{2.1}$$
$$z' = z + v_z'\tau \tag{2.2}$$
$$x = r + v_r\tau \tag{2.3}$$
$$y = v_\theta\tau \tag{2.4}$$
$$r' = \sqrt{x^2 + y^2} \tag{2.5}$$
$$\alpha = \arctan(x, y) \tag{2.6}$$
$$v_r' = \cos(\alpha)v_r + \sin(\alpha)v_\theta \tag{2.7}$$
$$v_\theta' = -\sin(\alpha)v_r + \cos(\alpha)v_\theta \tag{2.8}$$

where primed variables represent the new properties of the particle. Velocity in the θ direction is updated for collision processes, although the dynamic in the angular direction is assumed as symmetric.

Once the position and velocity of the particle are updated, the new cell that contains the particle is searched. This is done by a neighbor search, starting from the previous cell containing the particle. In the process of finding the new cell, it is possible that a particle encounters a boundary. When the particle interacts with the boundary, the particle may continue its life in the simulation (reflected) or might be eliminated from it (absorbed). Once that the new cell is found or that the particle life has been terminated, the pushing is complete. Collisions can change the velocity of the particles involved (elastic), create new particles (ionization-recombination) or change the type of particle (charge-exchange). Right now, only elastic collisions are implemented.

Cross-sections are read from files. These files contains two columns: one for the relative energy (in eV) and another one for the cross-section (in m^{-2})

2.4 Reset of particle array

Once that the pushing is complete, the array of particles that remain inside the domain is copied to a new array. The new array containing only the particles inside the domain will be the one used in the next steps. In this section, particles are assigned to the list of particles inside each individual cell. Unfortunately, this is done right now without parallelisation and is very CPU consuming.

2.5 Interaction between species

For each cell, interaction among the particles in it are carried out. The type of interaction between the different particles is defined by the user. In general, the maximum number of interaction in a cell is computed. For each collision, a pair of particles is selected. A loop over all possible collisions for the pair of particles is performed. If a random number generated is above the probability of collision for the type divided by the maximum one, the collision take place.

2.6 Scattering

The properties of each particle are deposited in the nodes from the containing cell. This process depend on the cell type, but in general, each node receive a proportional part of the particle properties as a function of the particle position inside the cell. Figure 2.1 shows how a particle at a generic position $p(x_1, x_2)$ inside the cell is scattered to the four nodes. Each node receives a proportional part of the area formed by dividing the cell in for rectangles using as an additional vertex the particle position. These properties are dimensionless, but they are converted to the correct units once the output is printed.

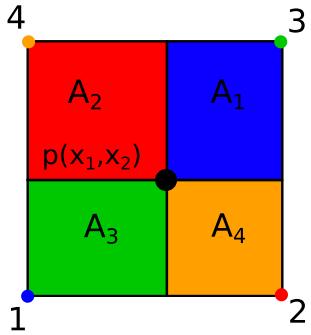

Figure 2.1: Example of how a particle is weighted in a quadrilateral cell.

2.7 Electromagnetic field

WIP.

Chapter 3

Installation

Chapter 4

Input File

4.1 Mesh

4.2 Geometry

4.3 JSON file format

Chapter 5

Future Work