A 3D HYBRID CODE TO STUDY ELECTRIC THRUSTER PLUMES

Filippo Cichocki (1), Adrián Domínguez (2), Mario Merino (3), Eduardo Ahedo (4)

(1) Equipo de Propulsión Espacial y Plasmas (EP2), UC3M, Madrid, Spain, fcichock@ing.uc3m.es
(2) Equipo de Propulsión Espacial y Plasmas (EP2), UC3M, Madrid, Spain, addoming@ing.uc3m.es
(3) Equipo de Propulsión Espacial y Plasmas (EP2), UC3M, Madrid, Spain, marmerin@ing.uc3m.es
(4) Equipo de Propulsión Espacial y Plasmas (EP2), UC3M, Madrid, Spain, eahedo@ing.uc3m.es

KEYWORDS: plasma plumes, hybrid codes, PIC, structured meshes, IBS, LEOSWEEP

ABSTRACT

A new 3D hybrid particle code is being developed at the Universidad Carlos III de Madrid, with the major goal of studying the expansion of an electric thruster plasma plume into vacuum and its interaction with either the spacecraft or an object in front of it. It is a hybrid particle/fluid code, thus representing a good compromise between a full particle code (numerically expensive) and a simplified fluid model approach. The development status, some of its innovative features and applications and, finally, the validation and benchmarking campaign of a preliminary code version, are here presented and described.

1. INTRODUCTION

Nowadays, the simulation of electric thruster plumes onboard modern satellites is becoming extremely important and demanding. Firstly, from a system engineering point of view, it is necessary to evaluate the plume interaction with sensitive S/C surfaces, such as onboard sensors and solar panels. In fact, the main plume ions and the secondary ions (generated by charge-exchange collisions, CEX) can hit such surfaces and damage them through sputtering or deposition. Secondly, a refined plume expansion model is also necessary in the design of active debris removal missions based on the ion beam shepherd technique (IBS), whose feasibility is currently being studied with a European Union funded project, named LEOSWEEP [1]. In this context, a debris object is progressively relocated to a different orbit by means of the ion push of a plasma plume directed towards it, as illustrated in Figure 1.

For what concerns the plume modeling, this can be achieved by means of either fluid or kinetic approaches. The former do not account for the complexity of the expansion of a near-collisionless medium. Notwithstanding this, a simplified fluid treatment can be applied with good results in the far region of the plasma plume, sufficiently far away from the thruster exit, as shown in [3] and [4], where their major limitation consists in the difficulty of modelling the lateral plume and the detailed energy distribution functions (strongly influenced by charge-exchange collisions). Kinetic models, on the other hand, do not present such limitations but generally require a much larger computational power.

In order to study the plume expansion with reasonable computational power, while covering the above described phenomena and the particularities of a rapidly-expanding plasma into vacuum, we are developing EP2PLUS (Extensible Parallel Plasma PLume Simulator). Such code is a 3D hybrid PIC/fluid code and it represents an intermediate solution between fluid and fully kinetic simulators. An arbitrary number of heavy species populations (including multiply charged ions or neutrals of different species) can be simulated as in conventional PIC codes, while the “neutralizing” electrons are treated as a fluid, thus avoiding the need to resolve the electron timescales in the simulation. Heavy species particles are moved with a conventional particle mover and collisions are simulated with dedicated Monte Carlo techniques. The electron fluid properties and the electromagnetic fields necessary to move the heavy particles and close the loop are obtained, on the other hand, by imposing quasi-neutrality and by either assuming unmagnetized electrons with simplified thermodynamics (numerical fittings from kinetic studies, Boltzmann or polytropic laws) or by solving the electron conservation equations for the magnetized case.

The 3D architectural choice enables the study of...
interesting non-axially symmetric problems, such as plumes with a neutralizer (typically on one side of the thruster and emitting neutral atoms), impact of the plume on an arbitrary target object, different S/C shapes and also the effects of an oblique magnetic field (such as the geomagnetic field) on the plume expansion.

This paper presents the status of the EP2PLUS code development. In Section 2, the code goals and methodologies to reach them are presented. In Section 3, the general code structure and its key algorithms are introduced. Section 4 focuses on some of the major code innovative features, with an application example in the context of the LEOSWEEP project, while Section 5 provides the code validation results, by benchmarking it against our 2D axisymmetric fluid code EASYPLUME. Finally, the conclusions are drawn in Section 6.

2. CODE GOALS AND METHODOLOGIES

2.1. Code goals

EP2PLUS has the major goals of:

- Simulating the near region plume physics, which requires detailed modeling of collisional effects, as these can affect greatly this plume region, as shown in [3] and [4].
- Simulating the interaction of the plume with arbitrary objects, a requirement coming directly from the LEOSWEEP project [1]. Such objects include the target debris of an IBS mission and the satellite itself (solar panels, sensors, thruster etc.), affected by the ion deposition due to the ion backflow (charge-exchange in the near plume region) and back-sputtering (erosion of the target debris).
- Simulating the 3D effects of an oblique magnetic field like the Earth’s magnetic field, which might distort in non-trivial ways a plume containing electrons, as shown in [5].
- Obtaining an accurate description of the evolution of the properties of a plume that is rapidly expanding into the vacuum, or in a vacuum chamber. In this context, EP2PLUS shall allow extrapolating experimental plume data, obtained in vacuum chambers with a known background pressure, to real flight conditions.

In order to reach the first goal of simulating the near region plume, EP2PLUS shall simulate the most relevant collisions between heavy particles. Such collisions shall be modeled with well-known approaches like the Monte Carlo Collision (MCC) or the Direct Simulation Monte Carlo (DSMC) methods, as well as hybrid approaches, not belonging to any of the above mentioned families.

For the simulation of the plume interaction with external objects, EP2PLUS features a structured mesh, which permits to use much quicker algorithms than an unstructured mesh, although being less flexible in terms of simulation geometries. Such structured mesh is tailored to the problem at hand, by assigning specific material flags to some of its surface elements. In this way, different object geometries can be added to the simulation domain by simply setting a 3D surface elements IDs matrix, without even touching the structured mesh nodes. More details on this can be found in Section 4.

With the goal of studying the effects of an oblique magnetic field, it is foreseen to develop a fluid solver for the electron conservation equations (continuity and momentum) coupled with Ampere’s law to study the effect of self-induced electron currents. For other study purposes, EP2PLUS does however feature simplified electron fluid approaches. One is that of assuming polytropic electrons, a model which can capture pretty well the major plume expansion features, as shown in [6]. The other approach is to use numerical fittings for the electron thermodynamics, coming from ongoing kinetic studies, like that described in [7] and [8], for the expansion of a magnetized plasma plume.

Finally, since EP2PLUS is primarily dedicated to study plasma plumes, it is specifically optimized to achieve this goal. In this context, an innovative expanding structured mesh can be used in order to reduce the numerical PIC noise and obtain better plume results, as shown in Section 4.

2.2. Development Methodology

EP2PLUS is being developed following strict development and validation standards. A test driven design (TDD) development permits to validate new modules by designing, in the first place, dedicated unit and integration tests. In this way, new code functions are added only when validated, and the functionality of old functions is continuously checked throughout the development phase by running the existing tests suite. At the same time, a sound documentation is being generated, and a version control software is being used.

The code has been developed in such a way that it is scalable and highly modular, with industry level input/output formats, like HDF5. The physical unit is called the CORE and is coded in Fortran, while the pre-processing (PRE) and post-processing (POST) units are coded in Python. As clearly stated by the code acronym, parallelization is foreseen with Open-MP, to take the advantage of large shared memory workstations.

In order to maximize the code sharing and standardization within our research group, EP2PLUS also shares the same code structure,
interfaces and modules of a hybrid plasma source axisymmetric code currently being developed (NOMADS) and presented in [9].

3. GENERAL CODE STRUCTURE AND KEY ALGORITHMS

3.1. General Code Structure

As mentioned in the previous section, EP2PLUS presents 3 independent units, as depicted in Figure 2 and described below:

- **SET**: coded in Python, it is in charge of the pre-processing tasks, including the generation of the necessary input files for the CORE. The simulation settings are specified by a dedicated input file, set.inp, editable by the user. The outputs are sim_params.inp, (text file) containing the simulation parameters, and SimState.hdf5 (HDF5 format), which contains the minimum set of variables to start or re-start the simulation (particle lists data, mesh data, etc.).
- **CORE**: coded in Fortran, it represents the simulation core unit, which carries out the plasma physics simulation. Taking as inputs both sim_params.inp and SimState.hdf5, it runs the simulation and generates the output file PostData.hdf5, containing the plasma plume properties at given time steps, and an updated version of the SimState.hdf5 file, at the last simulation time step.
- **POST**: coded in Python, it reads the CORE output files and produces different graphical results (plots and diagrams) as required by the user, through a dedicated POST input file, named post.inp.

3.2. The CORE unit algorithms

The CORE unit is composed of different dedicated modules, each of them in charge of performing specific simulation tasks and containing the necessary functions.

The two central modules of the hybrid simulator are the PIC and the electron fluid modules, as highlighted in green in Figure 3. The former, which takes as inputs the electric potential and the magnetic field at the PIC mesh nodes, is responsible for propagating the heavy species (neutrals and ions) one time step forward (through a leap frog algorithm), colliding them, checking whether they hit some special domain surface (external boundary, material surface etc.) and weighting them to the mesh nodes, thus obtaining the updated ion/neutral density and particle fluxes. These values are the inputs for the electron fluid model, which solves for the new electric and magnetic field, according to the selected solver type. The available solution models shall be described later on.

![Figure 3. Hybrid particle code loop](image)

In general, EP2PLUS has been designed to work with two different meshes for the PIC and for the electron fluid, in order to share the same structure of NOMADS [9], which requires a magnetic aligned mesh for solving the electron equations. Currently, for EP2PLUS, the two meshes are the same, so that the interpolation from one to the other is a simple identity function.

The main CORE modules are then detailed below:

- Interpolation module: responsible of the bidirectional interpolation of a large number of plasma variables from the PIC mesh to the electron fluid mesh and vice-versa. As previously mentioned, this is a trivial step in EP2PLUS.
- PIC module: it advances the heavy species particles by one PIC time step. In order to do so, it features both “particle-wise” and “cell-wise” algorithms. The former are algorithms applied to each of the heavy species particle, one by one, and include:
  - Interpolation of electric and magnetic fields to the particle position.
  - Moving particle with a leap-frog algorithm.
  - Sorting particles efficiently to the PIC mesh volume elements.
  - Checking whether a particle crosses some special PIC surface elements (external boundaries, material surfaces with different properties etc.). Particles crossing “special surfaces” are stored in dedicated hit list arrays.
The latter algorithms are carried out for each PIC volume element and include:

- Population control to ensure that the number of macro-particles per cell be within a specified interval. This algorithm (part of future work) shall renormalize the velocity distribution function within those cells featuring a number of particles out of this interval.
- Injection of particles from specific PIC surface elements, following arbitrary initial profiles for both the particle density and velocity (provided by the user).
- Collision simulations between heavy particle species or with the electron fluid. These generally follow either an MCC or a DSMC method.
- Surface interaction of the heavy species particles (absorption, emission, reflection etc.).
- Particle weighting, through a trilinear interpolation, based on the computational coordinates of the particles contained in each PIC volume element.

- Boundary correction module: it forces Bohm’s condition in quasi-neutral simulations by changing the weighted particle density at the PIC mesh nodes of a material surface. In fact, as shown in [10], the ion velocity at these nodes is not constrained naturally to satisfy the Bohm’s condition (normal velocity greater than the ion acoustic velocity). Therefore, in order to guarantee that a stable sheath solution exists, it is necessary to impose, in hybrid PIC codes, that these ions be accelerated to this minimum velocity.

- Electron fluid module: this module is in charge of solving the electron fluid model obtaining the new electric and magnetic fields to advance to the next PIC time step, self-consistently. Different electron fluid models can be selected as shown below and all of them assume quasi-neutrality:
  - Accurate numerical fittings for the electron thermodynamics such as $T_e = f(n_e)$, coming from dedicated kinetic studies for the unmagnetized plume case. References [7] and [8] show one of such studies, carried out for magnetized plumes.
  - Polytropic unmagnetized electrons: in this case, we assume a polytropic relation for the electrons like the one shown in equation 1, in which $T_{e0}$ and $n_{e0}$ are the electron temperature and density at a reference node of the domain and $\gamma$ is the polytropic exponent (1 for isothermal electrons):
    \[
    \frac{T_e}{T_{e0}} = \left( \frac{n_e}{n_{e0}} \right)^{\gamma - 1}
    \]  
    (1)

With this assumption, as shown in [3] and [4], the electric potential can be obtained from the electron density knowledge as:

\[
\phi = \begin{cases} 
\frac{T_{e0}}{e} \ln \left( \frac{n_e}{n_{e0}} \right) & \text{for } \gamma = 1 \\
\frac{T_{e0}}{e} \left( \frac{n_e}{n_{e0}} \right)^{-1} - 1 \cdot \frac{\gamma}{\gamma - 1} & \text{for } \gamma > 1 
\end{cases}
\]  

(2)

- Solution of the stationary electron continuity and momentum conservation equations, coupled with Ampere’s law. In this electron model, both the electric and magnetic field are computed self-consistently at each time step.

- Sheath module: this module is generally run to compute the sheath potential drop from the electron temperature knowledge, thus updating the impacting ion energies, for appropriate post-processing of the energy fluxes to material surfaces.

- Post module: this module contains all the functions dedicated to write the simulation output files PostData.hdf5 and SimState.hdf5 and update some simulation variables (like the average values of the plasma properties over a given number of time steps).

4. INNOVATIVE FEATURES AND EXAMPLES

The EP2PLUS code introduces some innovative features for the PIC module. First of all, the PIC uses a structured mesh in which it is possible to define inner material surfaces that do not coincide with the external simulation domain. This is a fundamental capability to enable the simulation of the plume interaction with inner objects and with the spacecraft, while maintaining a simple structured mesh, easier to treat and subject to lower numerical issues than an unstructured mesh.

Secondly, EP2PLUS can also make use of an expanding PIC mesh, which naturally follows the plume expansion and allows obtaining better results in terms of computational time and results noise.

These innovative features are further detailed in the following sub-sections.

4.1. Inner material surfaces

4.1.1. The 3D surface elements IDs matrix

As mentioned before, EP2PLUS has the capability of including inner material surfaces in the simulation domain, which are sinks for the ions, which recombine with electrons into neutral atoms, and which are specular/diffuse reflection or source surfaces for neutrals.

Before proceeding with the description of the approach followed by EP2PLUS, let us first
introduce some key concepts.

First of all, each particle species is represented by one or more dedicated particle lists (e.g. to differentiate between slow and fast ions), which are 2D matrices containing the particle properties along the different columns and a row for each particle. So the row number specifies the particle ID in the given particle list. Moreover, as already mentioned, the tool makes use of a structured PIC mesh, which means that the mesh nodes can be clearly identified with integer computational coordinates, ranging from 0 to the number of nodes minus 1 along each coordinate direction. So a particle at the physical coordinates \((x, y, z)\) can also be represented by the computational coordinates \((\xi, \eta, \zeta)\) with \(\xi \in [0, N_x - 1]\), \(\eta \in [0, N_y - 1]\) and \(\zeta \in [0, N_z - 1]\) with \(N_x, N_y, N_z\) the number of nodes along each coordinate axis. Of course such computational coordinates assume integer values at the PIC mesh nodes. In a structured mesh, a coordinate surface can be identified by the four neighbouring nodes with a constant value of one of the 3 computational coordinates. In order to quickly access such surface elements, we then define the integer coordinates \((\xi_{surf}, \eta_{surf}, \zeta_{surf})\) of a coordinate surface element as shown in Equation 3, where \((\xi_c, \eta_c, \zeta_c)\) are the computational coordinates of the surface element centers:

\[
(\xi_{surf}, \eta_{surf}, \zeta_{surf}) = 2 \cdot (\xi_c, \eta_c, \zeta_c)
\]

(3)

The computational coordinates of some surface elements of an arbitrary mesh are shown in red in Figure 4, which is 2D for the sake of clarity (so that a surface element is represented by a line).

Therefore, by defining a 3D surface element IDs matrix with dimensions \((2N_x - 1) \times (2N_y - 1) \times (2N_z - 1)\), we can specify different surface elements material properties by simply modifying the corresponding matrix element. For example, an ID equal to 0 means that the surface is transparent to ions, while an ID of -1 means that it is a free loss boundary (particles crossing it are simply removed from the domain, with no further action). Of course, by defining other IDs, different types of material surfaces can be added to the simulation domain, as shown in Figure 4.

Of course, such an approach requires the definition of a dedicated algorithm to check surface elements crossing. This is the particle crossing check algorithm, which checks if the particle has crossed any inner, important surface element along its motion (any surface element with an ID other than 0). As we are using a leap frog algorithm for particle motion, the particle trajectory is a straight line between the initial and final particle positions (blue line in Fig. 4), which are stored in memory. The algorithm then checks the IDs of the surface elements crossed by the particle, one after the other, in the order dictated by the direction of motion. If it happens that the particle crosses a surface element with ID different from zero, it is collected into a dedicated particle hit list, containing the particle ID in the corresponding particle list, the corresponding particle velocity and the exact crossing point on the surface element.

As for the particle lists, there is one or more particle hit list for each ionic or neutral species. These hit lists are used to carry out the necessary actions with the use of other dedicated algorithms, which depend on the surface element ID applying in each particle case: simple removal, removal plus injection of a neutral, reflection etc. Finally, they also represent an extremely useful tool for counting particle fluxes through user’s defined surfaces.

4.1.2. Application to an IBS mission

An example of the above described functionality is shown below. With a simple Cartesian mesh it is possible to easily define a relatively complex geometry like that of an IBS mission scenario. This is shown in Figure 5. Here the IDs for the external boundary of the simulation domain (-1), for the spacecraft (1) and for the object (2) are different, thus presenting a different shade of grey.

Figure 4. PIC mesh surface elements IDs and particle crossing check algorithm in 2D, for the sake of clarity

Figure 5. A 3D surface element IDs matrix display, generated for an IBS mission scenario analysis
With this geometry, by adding an injection surface on the S/C side facing the space debris object, it is possible to study the interaction of the plasma plume with it. In this case this object is cubic with 24 cm side and at a distance of 1 m from the thruster exit plane.

The resulting 3D electric potential is shown in Figure 6, for a collisionless plume simulation.

The formation of a wake behind the object and the corresponding ion number density contours can be better appreciated in Figure 7, representing a cross section at \( y = 0 \). The plume, made of singly charged Xenon ions, is injected with an initial axial velocity of 40 km/s, following the Parks-Katz plume profiles [3] with an initial density of \( 10^{16} \) m\(^{-3} \) and a 5 deg initial divergence angle (for the 95% ion current streamline). Regarding the electron fluid model, polytropic electrons with \( \gamma = 1.05 \) have been assumed with \( T_e = 3 \) eV (electron temperature at the thruster exit center, where the plume density is highest).

The ions are perfectly absorbed by the material surface of the space debris, thus generating a region of near-vacuum behind it (i.e. a plasma wake). However, the density gradients induced by the object, also result, through Equation 2, into large electric potential gradients and hence electric fields directed towards the plume centerline. For this reason, the vacuum region quickly disappears (after approx. 1.5 m in the current example). This can also be clearly observed in Figure 8, providing the 2D contour of the x-velocity, again at the \( y = 0 \) plume cross section.

The ions are perfectly absorbed by the material surface of the space debris, thus generating a region of near-vacuum behind it (i.e. a plasma wake). However, the density gradients induced by the object, also result, through Equation 2, into large electric potential gradients and hence electric fields directed towards the plume centerline. For this reason, the vacuum region quickly disappears (after approx. 1.5 m in the current example). This can also be clearly observed in Figure 8, providing the 2D contour of the x-velocity, again at the \( y = 0 \) plume cross section.

The ions are perfectly absorbed by the material surface of the space debris, thus generating a region of near-vacuum behind it (i.e. a plasma wake). However, the density gradients induced by the object, also result, through Equation 2, into large electric potential gradients and hence electric fields directed towards the plume centerline. For this reason, the vacuum region quickly disappears (after approx. 1.5 m in the current example). This can also be clearly observed in Figure 8, providing the 2D contour of the x-velocity, again at the \( y = 0 \) plume cross section.

The ions are perfectly absorbed by the material surface of the space debris, thus generating a region of near-vacuum behind it (i.e. a plasma wake). However, the density gradients induced by the object, also result, through Equation 2, into large electric potential gradients and hence electric fields directed towards the plume centerline. For this reason, the vacuum region quickly disappears (after approx. 1.5 m in the current example). This can also be clearly observed in Figure 8, providing the 2D contour of the x-velocity, again at the \( y = 0 \) plume cross section.

The ions are perfectly absorbed by the material surface of the space debris, thus generating a region of near-vacuum behind it (i.e. a plasma wake). However, the density gradients induced by the object, also result, through Equation 2, into large electric potential gradients and hence electric fields directed towards the plume centerline. For this reason, the vacuum region quickly disappears (after approx. 1.5 m in the current example). This can also be clearly observed in Figure 8, providing the 2D contour of the x-velocity, again at the \( y = 0 \) plume cross section.
Figure 10 shows the corresponding electric potential 2D field. Once again, the plasma wake is clearly visible. Please consider that, from Equation 2, a polytropic exponent of 1.05 results into a minimum electric potential equal to \(-T_{e0}/(\gamma - 1) = -63\) V (not captured by the contour levels).

Finally, the initial ion density profile at \(z = 0\) m, applied in both target debris cases, is shown in Figure 11.

expansion of the plume. One example of such meshes is referred to as cold beam mesh as it is a conically expanded 3D mesh that follows the ion streamlines of a plume with zero electron thermal pressure. As shown in [3] and [4], if the electrons are cold, or with negligible thermal energy, no ambipolar electric field builds up, so that the ion trajectories are simple straight lines (for a hypersonic plume like that of an electric thruster).

The only parameters required to define the cold beam mesh are therefore only the initial radius \(R_0\) and the divergence angle \(\alpha_0\) of the 95% ion current streamline. An example of such a cold beam mesh with \(R_0 = 10\) cm and \(\alpha_0 = 5\) deg is shown in Figure 12. Once again, the \(x - y\) plane is the plume injection plane and \(z\) is along the plume axis.

In order to better study an expanding plume into vacuum, EP2PLUS also features the possibility of using PIC meshes that naturally follow the expansion of the plume. One example of such meshes is referred to as cold beam mesh as it is a conically expanded 3D mesh that follows the ion streamlines of a plume with zero electron thermal pressure. As shown in [3] and [4], if the electrons are cold, or with negligible thermal energy, no ambipolar electric field builds up, so that the ion trajectories are simple straight lines (for a hypersonic plume like that of an electric thruster).

The only parameters required to define the cold beam mesh are therefore only the initial radius \(R_0\) and the divergence angle \(\alpha_0\) of the 95% ion current streamline. An example of such a cold beam mesh with \(R_0 = 10\) cm and \(\alpha_0 = 5\) deg is shown in Figure 12. Once again, the \(x - y\) plane is the plume injection plane and \(z\) is along the plume axis.

4.2. The expanding PIC mesh

In order to better understand the advantage of using a cold beam mesh, Figure 13 shows a comparison of the ion iso-density lines for a collision-less plasma plume with \(R_0 = 10\) cm and \(\alpha_0 = 5\) deg, obtained with a simple Cartesian mesh (in red) and with the cold beam mesh (in black).

To better understand the advantage of using a cold beam mesh, Figure 13 shows a comparison of the ion iso-density lines for a collision-less plasma plume with \(R_0 = 10\) cm and \(\alpha_0 = 5\) deg, obtained with a simple Cartesian mesh (in red) and with the cold beam mesh (in black).

Figure 12. The 3D cold beam mesh geometry. For the sake of clarity, only 1 mesh node every 10 is shown
The number of particles per cell close to the injection surface ($z = 0$) is the same for both mesh cases (50 macro-particles per cell), however, as the plume expands, the number of particles per cell, in the Cartesian mesh case, drops abruptly due to the geometrical beam expansion, resulting into large fluctuations in the weighted ion density.

It should be finally noted that, although using a cold beam mesh reduces greatly the numerical noise downstream, it also yields a lower spatial resolution.

5. VALIDATION AND BENCHMARKING

In order to validate the code, at least in the collisionless plasma plume scenario, it is possible to take advantage of existing semi-analytical models like the Self Similar Models (SSMs), which provide a solution with a negligible error with respect to the exact fluid solution. As shown in [3] and [4], the normalized collisionless plasma plume depends only on:

- Plasma plume profiles along the initial radius for the ion density, ion axial velocity and streamlines divergence angle.
- Electron polytropic exponent $\gamma$, which models the electron cooling and hence the electron pressure effects.
- Initial divergence angle $\alpha_0$ of the 95% ion current streamline.
- Initial ion Mach number $M_0$, which expresses the ratio between the directed ion kinetic energy and the electron thermal energy, as shown in Equation 4, where $m_i$ is the ion mass and $u_{i0}$ is the initial axial velocity component of the ions:

$$M_0 = \sqrt{\frac{u_{i0}^2 m_i}{\gamma T_{e0}}}$$  \hspace{1cm} (4)

The higher the $M_0$, the lower is the effect of the electron thermal pressure, so that the resulting plume expansion is more and more conical with divergence angle $\alpha_0$. In this case the streamlines radius increases linearly with the distance ($R(z) = R_0 + z \tan(\alpha_0)$) and the centerline density decreases with the square of this streamline radius (to satisfy the mass continuity equation).

On the other hand, the lower the $M_0$ and the closer to 1 is the polytropic exponent $\gamma$ (1 corresponds to the isothermal case), the higher is the effect of the electron thermal pressure, so that the ion streamlines divergence increases with the distance and the ion density decays more rapidly than for the previously considered case.

Figure 14 shows the evolution of such a centerline density with the distance from the thruster for 4 benchmark cases with different values of $M_0$ and $\alpha_0$ and with a polytropic exponent equal to 1.05. The solutions obtained with the generalized (to non-isothermal electrons) Parks-Katz model ([3] and [4]) are shown by solid lines and are plotted against the EP2PLUS (circles) and NOMADS solutions (stars).

Figure 15 shows the evolution of the ion density along the radius at $z = 6$ m, for the same benchmark cases and with the same plume hypotheses.

The EP2PLUS solution clearly follows closely the SSM solution along both the centerline and the radius, with errors never exceeding a few percent points. Such EP2PLUS solutions have been obtained with the aforementioned cold beam mesh.
and with 50 macro-particles per cell close to the injection surface, at \( z = 0 \) m. For the NOMADS code, on the other hand, up to 2000 particles per cell have been considered at \( z = 0 \) m, obtaining much more noisy results. The reason of this behaviour is that this 2-D axisymmetric code is using here a cylindrical uniform mesh, which does not follow the natural beam expansion, so that much larger fluctuations in the number of particles per cell are observed downstream. At the current development stage, neither of the codes uses a particle population control algorithm, although this is planned for the future.

The total computational and simulation time for both the EP2PLUS and NOMADS simulations are shown in Table 1. The computational elapsed time is the required computer time without parallelization, using an INTEL CORE i7 @2.39 GHz. The simulation real elapsed time, on the other hand, has been set to twice the time required by the centerline particles (at \( x = y = r = 0 \)) to reach the other simulation boundary (6 m away), in the cold beam case. The number of particles under a stationary flow, at the simulation end, is displayed as well.

**Table 1: Computational details for the benchmark simulation cases**

<table>
<thead>
<tr>
<th>Case</th>
<th>Code</th>
<th>Comp. and (real) elapsed times [sec]</th>
<th>N. part. at the sim. end</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_0 = 5^\circ ) ( M_0 = 15 )</td>
<td>EP2PLUS</td>
<td>1.1e4 (8.2e-4)</td>
<td>5.462.10^6</td>
</tr>
<tr>
<td></td>
<td>NOMADS</td>
<td>4.9e3 (8.2e-4)</td>
<td>3.062.10^6</td>
</tr>
<tr>
<td>( \alpha_0 = 10^\circ ) ( M_0 = 10 )</td>
<td>EP2PLUS</td>
<td>1.0e4 (8.2e-4)</td>
<td>4.817.10^6</td>
</tr>
<tr>
<td></td>
<td>NOMADS</td>
<td>5.0e3 (8.2e-4)</td>
<td>3.000.10^6</td>
</tr>
<tr>
<td>( \alpha_0 = 5^\circ ) ( M_0 = 40 )</td>
<td>EP2PLUS</td>
<td>2.1e4 (2.1e-4)</td>
<td>1.314.10^7</td>
</tr>
<tr>
<td></td>
<td>NOMADS</td>
<td>6.8e3(2.1e-4)</td>
<td>4.415.10^6</td>
</tr>
<tr>
<td>( \alpha_0 = 10^\circ ) ( M_0 = 40 )</td>
<td>EP2PLUS</td>
<td>2.2e4 (2.1e-4)</td>
<td>1.500.10^7</td>
</tr>
<tr>
<td></td>
<td>NOMADS</td>
<td>7.4e3 (2.1e-4)</td>
<td>4.570.10^6</td>
</tr>
</tbody>
</table>

The red portion includes all the particle-wise algorithms, namely the field interpolation to the particle position, the particle mover, the particle sorting and the particle crossing check (described in Figure 4). The average time per particle for carrying out all the above functions is 0.27 \( \mu \)s. Weighting particles properties in each cell, on the other hand, requires approximately 2/5 of the overall computational time (for a PIC mesh with 61 x 61 x 121 nodes). As the number of particles per cell increases, the particle wise algorithms tend to require a higher fraction of the total computational time. Regarding the electron fluid model (polytropic electrons), it only requires 1.5% of the total time.

Regarding other tool functionalities that have not been validated yet, it is expected that the heavy particles collisions will require a significant fraction of the total computational time.

### 6. CONCLUSIONS AND FUTURE WORK

This paper has presented the current status of the EP2PLUS code, being developed by EP2 at UC3M, in order to study the electric thruster plume expansion and their interaction with either the S/C or other objects, like a space debris in the context of an ion beam shepherd mission.

The overall hybrid particle code structure and the approach followed for the main algorithms have been presented. One innovative feature is the use of structured meshes with 3D surface element IDs matrices that allow simulating inner material surfaces and objects. The dedicated particle crossing check algorithm, required to deal with this new feature, has also been briefly described. Moreover, EP2PLUS permits to choose a dedicated cold beam mesh for pure plume expansion studies, which has proved to be much more efficient than a classical Cartesian mesh.

Finally, the code has been validated in a collisionless simulation scenario, by comparing its results with those (almost exact) of a self-similar plume model, obtaining a very encouraging agreement.

Regarding future work, both particle collisions and surface interaction require significant advances, with a special interest for the modeling of momentum and charge-exchange collisions, the most important collisional phenomena in the near region of plasma plumes. Well-known approaches like Monte Carlo Collisions or Direct Simulation Monte Carlo methods, coupled with ad hoc hybrid approaches, will be used to model these collisions.

Moreover, an independent particles control algorithm to tackle both the geometrical and field-induced dispersion of particles downstream, will be coded to reduce the overall numerical noise and improve the quality of the PIC module results.
An improvement in the allowed simulation geometries is also foreseen, through the use of existing or new algorithms that adjust the structured mesh nodes position to the surfaces of the required objects (with non-rectangular geometries). Besides, the simulation time shall be reduced by parallelizing the overall code with OpenMP.

Finally, regarding the electron fluid module, the development of an electron fluid model, more complex than the existing one based on unmagnetized polytropic electrons, is foreseen in order to study the effects of an oblique magnetic field on the plume expansion. This will require solving, at the PIC mesh nodes, the electron continuity and momentum equations, coupled with Ampere’s law. In this electron fluid model context, the possibility of performing short periods of full PIC simulation (with particle electrons) to improve the existing models shall also be investigated in the future.

7. ACKNOWLEDGEMENTS

This work has been financed by the European Union Seventh Framework Programme (FP7/2007/2013) under grant agreement N.607457 (LEOSWEEP project).

A. Dominguez participation has been funded by the Spanish R&D National Plan, Grant N. ESP2013-41052.

REFERENCES


