NOMADS: DEVELOPMENT OF A VERSATILE PLASMA DISCHARGE SIMULATION PLATFORM FOR ELECTRIC PROPULSION

A. Domínguez [(1)*], D. Pérez-Grande [(2)*], P. Fajardo [(3)*], E. Ahedo [(4)*]

[(1)] Corresponding author: *addoming@ing.uc3m.es*
[(2)] *daperezg@ing.uc3m.es*
[(3)] *pfajardo@ing.uc3m.es*
[(4)] *eahedo@ing.uc3m.es*
** Equipo de Propulsión Espacial y Plasmas (EP2), Universidad Carlos III de Madrid, Spain*

KEYWORDS: plasma thrusters simulation, hybrid codes, PIC, HET

ABSTRACT:

A new, 2D axisymmetric, multi-thruster, hybrid code for simulating plasma discharges is being developed by EP2 group over the coming years. This paper intends to present the overall methodologies and structure of the code together with the main innovations that will be added on the treatment of the heavy species (ions and neutrals with a Particle-In-Cell (PIC) approach) and on the electron macroscopic fluid model. The key aspects of the new code are oriented toward reducing PIC-related numerical noise and expanding our capabilities for modelling a variety of different magnetic field topologies.

1. INTRODUCTION AND MOTIVATION

The Electric Propulsion (EP) landscape is changing fast: next generation thrusters belonging to proven technologies, such as the Ion Thruster or the Hall-Effect Thruster (HET), are being designed for larger on board power (~20-100kW), different control schemes (such as "direct-drive" [1]) and new mission scenarios. This is true both in the private and public sectors, with applications ranging from Station Keeping to "Space Tugs" [2] or Planetary Exploration. In addition, new thruster technologies, such as the Helicon Plasma Thruster (HPT) [3] or the Electron-Cyclotron-Resonance Accelerator (ECRA), are strong candidates for future missions and currently enjoy dedicated development efforts.

Focus on improving European competitiveness in Space has cast the spotlight on EP technology, as part of the Horizon2020 framework (EPIC [4]); thus, the opportunity for complementing new developments with simulation tools that will allow us to accurately model plasma discharges in a variety of different EP thrusters, is well based on the current climate. These tools are essential in order to reduce development time and costs, reveal optimization opportunities and predict operational parameters throughout the thrusters' lifetime.

To this end, EP2 has decided to develop NOMADS (Non-structured Magnetically Aligned Discharge Simulator), a versatile, multi-thruster simulation platform.

NOMADS is based on the group's broad expertise with previous simulation codes such as HallMA [5] and HPHall2 [6], hybrid codes for HET simulations based on the original HPHall by Fife [7]. The various versions of this code are well known within the EP community and have been used in the development and characterization of several HETs. In the same manner, the limitations and shortcomings of the original and derived codes are also well known; these range from "stiffness" in the code structure to physical limitations due to the assumptions made on the magnetic field.

Even with advances made in areas such as Plasma-Wall interaction, Particle-In-Cell (PIC) method noise reduction, etc., new development trends in the Space sector call for a more flexible and capable environment. Thus, NOMADS is built with much of the heritage of previous codes in mind but with the intention of expanding their functionality.

NOMADS is going to be a hybrid PIC (heavy species) / fluid (electron population) code purposed for solving an axisymmetric description of the plasma in the discharge chamber and "near-plume" regions. It will be applicable to thrusters which share the commonalities of producing low density, low collisionality plasmas, with strong magnetization of the electron population and negligible self-field production. In principle, this includes various electromagnetic-type thrusters such as HET, HPT, ECRA and High-Efficiency Multistage Plasma Thruster (HEMPT).

This intent on versatility will be achieved through well-proven methodologies such as module-based construction (e.g., the future inclusion of the plasma-wave interaction module) and the capability for modelling complex magnetic topologies and non-uniform magnetic field effects,

through a bi-Maxwellian, two-temperature, description of the electron population.

In addition, updates to the heavy-species, PIC segment are aimed towards improvement of PIC-derived statistics and numerical optimization. New capabilities such as the simulation of inner active surfaces in the simulation domain will be developed too.

The goal of this paper is to offer an update on the development process of the NOMADS platform. Section 2 will introduce the methodologies used for code development and give a concise overview of the intended code structure; the remainder of the manuscript will be dedicated to describing innovations on the PIC side and the particularities of the proposed new electron-fluid model.

2. CODE DEVELOPMENT METHODOLOGIES AND GENERAL STRUCTURE

2.1. Methodologies and standards

The new, versatile plasma discharge simulator is based on modularity and thus will potentially be extensible to Helicon, ECR sources or HEMPT as well as HET, aiming to be a more flexible and capable platform. Modularization strategy also facilitates code development and debugging, the integration of different subroutines and the addition of new capabilities such as dedicated modules for turbulence simulation or plasma-wave interaction in Helicon sources. Some of the principal code modules will be the PIC module and the electron fluid module, as well as the sheath module or the mesh-interpolation module, which are further described later on.

NOMADS will make use of several programming languages such as Python/Matlab for data pre and post-processing and results analysis and Fortran for the main numerical core. Additionally, industry-level standards such as HDF5 technology for data management and Open-MP for code parallelization will be considered. In order to maximize code sharing and standardization, NOMADS is being designed with the same overall architecture, data structure and interfaces as those of EP2-PLUS [8], including common baseline modules and dedicated subroutines whenever possible. Likewise, both codes benefit from use of strict development and validation standards as Test Driven Design (TDD) philosophy and wide specific documentation (development document and user's manual are written and updated in parallel). Therefore, modular and integrated tests will be designed to check each of the code functionalities during code development process. Moreover, NOMADS tool development will be carried out under version control from the start by setting a standard Mercurial online repository which can be cloned to any number of local machines.

2.2. Overall tool architecture

As depicted in Fig. 1, the NOMADS tool will consist of three independent program units:

- SET: Coded in Python/Matlab, this unit will be in charge of the pre-processing tasks, generating the necessary input files for the core (both HDF5 and text format) from a user defined text file specifying the simulation characteristics.
- CORE: Written in Fortran, it will correspond to the simulation core unit, which will carry out the plasma discharge simulation thus producing a set of outputs files in HDF5 format.
- POST: Coded in Python/Matlab, the post-processing unit will produce as outputs the different results (plots and diagrams) required by the user.

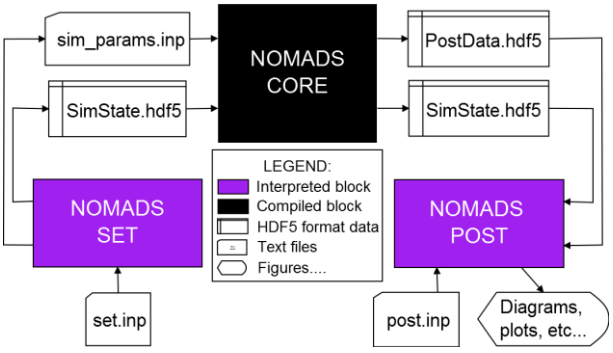

Figure 1. NOMADS tool overall architecture

Therefore, the whole simulation process will start with the SET unit, which will read the user defined text file *set.inp* containing the parameters defining the simulation and will produce the CORE input files *sim_params.inp* and *SimState.hdf5* containing the minimum set of variables and parameters needed to start the simulation, including the mesh data for both PIC and electron fluid modules. The CORE unit will then read those input files and will run the simulation generating the output file *PostData.hdf5* containing required computed variables at given time steps and an updated version of the *SimState.hdf5* file. Those files, along with the user edited file *post.inp* will be taken as inputs by the POST unit, thus producing the user required simulation results.

2.3. Main CORE modules description

The CORE unit will be composed of different dedicated modules, each of them in charge of performing predefined tasks and containing the necessary functions and subroutines with the different numerical algorithms implemented. The two central modules of the hybrid simulator will be the PIC module and the electron module. The former, taking as inputs the electric potential and the electron temperature, will propagate the heavy species (neutrals and ions) one PIC time step

forward obtaining the plasma density and particle fluxes. The latter, taking those values from the PIC module, will solve a magnetized electron fluid model computing the electric potential and the electron temperature, thus closing the loop. However, each of those central modules will operate on a different mesh of the simulation domain: a structured mesh for the PIC module, and a non-structured magnetically aligned mesh for the electron fluid module. Therefore, both modules will communicate each other through a bidirectional interpolation module. Here a brief description of the main CORE modules is provided:

- Input reading module: this module will contain the subroutines in charge of reading and initializing all CORE variables at the beginning of the simulation with the information contained in the input files *sim_params.inp* and *SimState.hdf5*.
- Interpolation module: it will be responsible for performing the bidirectional interpolation of different CORE variables from the PIC mesh to the electron fluid mesh and vice-versa.
- PIC module: it will perform the simulation of the heavy species. It will contain the subroutines in charge of injecting, propagating, sorting, weighting and removing particles from domain, as well as performing the different particle collisions modelled.
- Boundary correction module: it will apply Bohm's condition whenever necessary, changing the weighted particle density computed by the PIC module at the corresponding nodes.
- Electron fluid module: this module will be in charge of solving the electron fluid model obtaining, mainly, the electric potential and the electron temperatures using the PIC module output variables such as the plasma density and particle fluxes as input.
- Sheath module: this will be a dedicated module for the sheath regions which will relate sheath potential, and deposited particle and energy flows.
- Post module: this module will contain all functions and subroutines dedicated to post-process the CORE results thus computing important statistics of the CORE simulations and to write them out to the output files *PostData.hdf5* and *SimState.hdf5*.

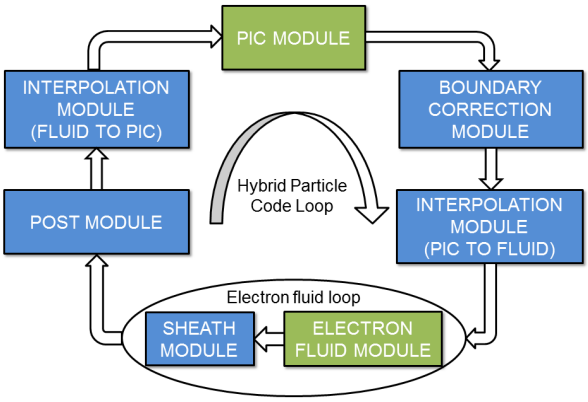

Figure 2. Hybrid particle code loop

Fig. 2 shows the general, hybrid particle code loop. The main simulation loop is the PIC loop in charge of advancing the heavy species one PIC time step. On each simulation step, a dedicated electron fluid loop is run with a much lower time step up to the general simulation time is achieved.

3. INNOVATIONS

3.1. PIC module innovative characteristics

Innovations on the treatment of the heavy species considered on the hybrid code will be focused on improving the population control and PIC statistics in order to reduce numerical noise and add new capabilities such as the introduction of inner, active surfaces in the simulation domain.

Each of the various ion and neutral species which can be simulated will be treated in separated particle lists containing all the necessary particle related information such as current and previous position and velocity, elementary particle mass and charge status, and number of elementary particles represented by each simulation particle. Such organization of different species facilitates the population control during the simulation and the treatment of the various particle collisions to be modelled between different particle lists, thus contributing to reduce the PIC associated numerical noise. A remarkable fact is that, although NOMADS is a 2D (axisymmetric) plasma discharge simulator, it will make use of a 3D Cartesian particle mover following a leap frog method similar to Boris' CYLRAD algorithm [9]. Therefore, on each PIC time step the 3D position (x, y, z) and velocity (v_x, v_y, v_z) of the particle is updated and then the particle is projected to the 2D axisymmetric plane (z, r) in order to weight the particle to the PIC mesh nodes. This strategy is justified in terms of code sharing and standardization with EP2-Plus code, which uses the same particle mover algorithm [8]. Besides, it avoids the singularity problem at $r = 0$ of a 2D cylindrical particle mover [10].

Given the 2D, structured PIC mesh of the physical simulation domain, a uniform computational mesh with squared elements can be defined in such a way that each mesh node in physical domain (z,r) corresponds to a computational node (ξ,η), where $\xi \in [0, N_\xi - 1]$ and $\eta \in [0, N_\eta - 1]$ are the computational coordinates taking integer values at the nodes and N_ξ and N_η are the corresponding number of nodes along each coordinate (see Fig. 3). After propagating particles a PIC time step forward, they will be weighted to the PIC mesh nodes so as to obtain the resulting macroscopic magnitudes for the various species simulated. The weighting process will be carried out using the computational coordinates of the particles computed from the 2D particle coordinates on the physical domain through a Newton-Raphson iterative algorithm.

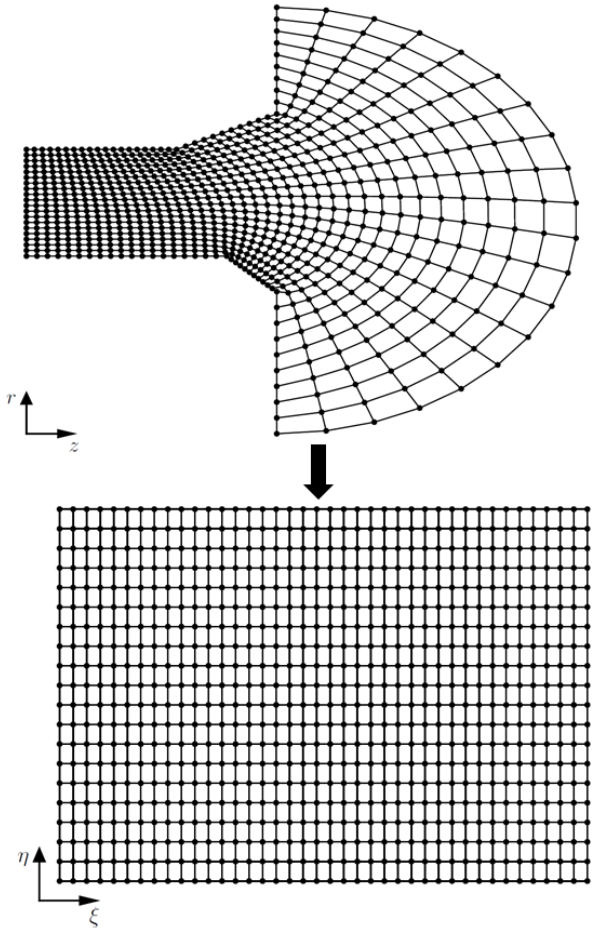

Figure 3. Physical mesh (top) and corresponding computational mesh (bottom) of the simulation domain

Linear weighting functions will be used since they provide the most common compromise between computational expense and smoothness of the representation. The corrected nodal weighting volumes [11] will be computed at pre-processing. For instance, the weighted particle density at a given node (i,j) takes the expression:

$$n_{ij} = \frac{\sum_p N_p S_{ij}(\xi_p,\eta_p)}{\Delta V_{ij}} \qquad (1)$$

where the sum is extended to all particles contained in all cells sharing the node, N_p is the number of elemental particles represented by each simulation particle, $S_{ij}(\xi_p,\eta_p)$ stands for the linear weighting function evaluated at the particle computational coordinates and ΔV_{ij} is the corrected weighting volume associated to the node, computed as:

$$\Delta V_{ij} = \int_{\Omega(\xi,\eta)} 2\pi r(\xi,\eta) S_{ij}(\xi,\eta) \left|\frac{\partial(z,r)}{\partial(\xi,\eta)}\right| d\xi d\eta \qquad (2)$$

A new capability to be included in NOMADS that was not present in our previous simulation codes is the possibility of including active inner surfaces in the simulation domain. Such surfaces can collect ions recombining them into neutrals which will be reinjected in the simulation domain, can diffusively reflect neutrals impinging on them, or can even inject a propellant mass flow into the domain. To enable this capability, surface elements will be defined as the lines between two consecutive nodes along ξ and η. As depicted in Fig. 4, each surface element will be identified from the computational coordinates of the mean point, in such a way that the computational coordinates of the surface elements can be computed as:

$$(\xi_{surf},\eta_{surf}) = 2(\xi_c,\eta_c) \qquad (3)$$

where (ξ_c,η_c) are the computational coordinates of the mean point on the surface element. A surface elements matrix with dimensions $(2N_\xi - 1) \times (2N_\eta - 1)$ will be defined containing a flag for each surface element. This flag will correspond to the surface element ID and will determine the type of surface to be considered (see Fig. 3). Surface elements with ID = 0 will be transparent for the simulation particles, so they can be crossed without consequences. In case of a particle crossing both a free loss surface (ID = -1) or recombination surface (ID = 2), it will be collected by the surface. Injection surface elements will be treated separately, with a dedicated data structure containing the injection-related information.

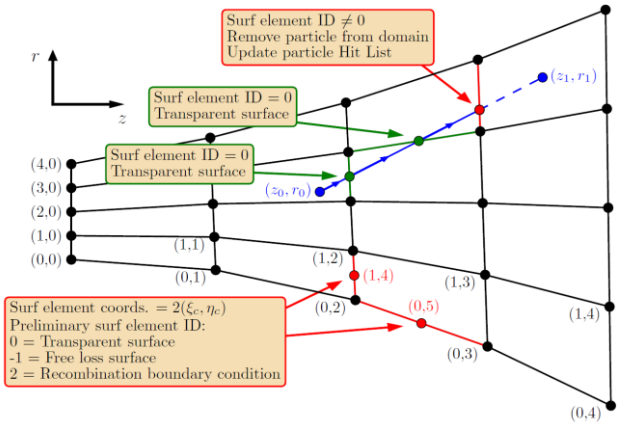

Figure 4. Surface elements and particle loss algorithm.

Computational coordinates of nodes and surface elements are represented on black and red, respectively. Particle straight line trajectory in a PIC time step corresponds to the blue line

After each PIC time step and for each simulation particle of the domain, a dedicated algorithm called *particle loss algorithm* will check if the particle has crossed any inner, important surface element along its motion. Making the reasonable assumption (for low enough PIC time step) of considering the particle trajectory in the 2D (z,r) plane during the time step as a straight line between the initial and final particle positions (blue line in Fig. 4), the algorithm will check the IDs of the surface elements crossed by the particle. The particle will then be collected at the first panel found with ID different from zero, thus updating a particle hit list with the particle ID, the hitting particle velocity and the hitting point on the surface element (red point on the active surface element crossed by the particle on Fig. 4). For each simulated species, a hit list will be defined and updated and will later be used to remove exited particles from the domain and carry out the necessary actions depending on the type of boundary condition applying on each case.

Taking advantage of the surface elements definition exposed above, injection surfaces elements will be identified in a dedicated data structure containing its computational coordinates and the necessary injection properties so that new particles will be injected into the domain from each of those surface elements. This strategy allows for a better control of the number of particles per cell existing at the adjacent cells to the injection surface elements. Besides, it will be possible to define different injection properties for each injection surface element, thus enabling the injection of a varying profile through a given injection surface.

As for particle collisions, well-known approaches like Monte Carlo Collisions (MCC) or Direct Simulation Monte Carlo (DSMC) will be used as well as *ad hoc* hybrid approaches. Those algorithms together with the population control strategies will be focused on improve PIC related statistics reducing numerical noise and getting a better description of the plasma heavy species VDF. The treatment of the heavy species in separated particle lists will help to that purpose, simplifying the collisional processes and the population control of different species.

3.2. Electron fluid module innovative characteristics

3.2.1. Updated Electron fluid model

An updated electron model constitutes the main innovation in the electron-fluid segment of the platform, with regards to previous codes.
The new model results from *dismissing* two of the main assumptions made in the electron-fluid equations posed in HallMA:

- The magnetic field lines are “iso-thermal”
- The electron Velocity Distribution Function (VDF) allows a Maxwellian description with isotropic electron temperature.

The first dismissal implies that the model becomes a two-dimensional description of the plasma when posing the equations in the “Magnetic Reference System”. This system is defined through the local parallel and perpendicular directions to the magnetic field lines, and the azimuthal direction.
A set of “curvilinear magnetic coordinates”, {λ, σ, θ}, is given for the simulation domain under the assumptions that the magnetic field is solenoidal, irrotational and stationary; λ and σ are, respectively, the magnetic stream-line function and magnetic potential function.
The curvilinear coordinates are used to construct the Magnetic Field Aligned Mesh (MFAM), depicted in Fig. 5. Its use is justified on the basis of reducing numerical diffusion errors in the anisotropic media of the magnetized electron fluid [12].

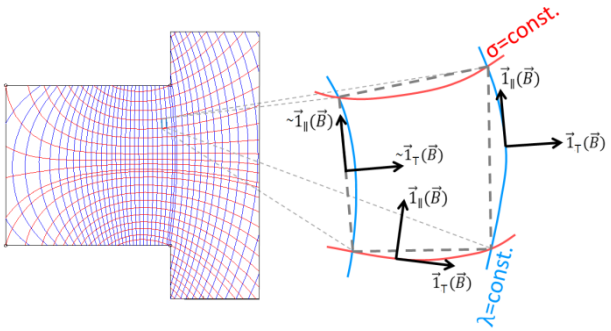

Figure 5. MFAM for a Hall Effect Thruster discharge channel

In regards to the VDF, the new working assumption is that magnetic confinement leads to anisotropicity in the electron distribution function. Two separate electron temperatures, $T_{e\parallel}$ and $T_{e\perp}$, defined, respectively, for the magnetic field parallel direction and perpendicular directions (those contained in the perpendicular plane to the field line), may be used to describe a bi-Maxwellian VDF:

$$f_e = \\ = \frac{1}{(2\pi)^{3/2}}\left(\frac{m_e}{kT_{e\parallel}}\right)^{1/2}\frac{m_e}{kT_{e\perp}}\exp\left[-\frac{1}{2}\frac{m_e}{k}\left(\frac{c_{e\perp}^2}{T_{e\perp}}+\frac{c_{e\parallel}^2}{T_{e\parallel}}\right)\right] \quad (4)$$

The electron fluid equations may then be derived by taking velocity moments of the Boltzmann equation.

The implications for the electron VFD of assuming an anisotropic temperature bi-Maxwellian

distribution versus an isotropic temperature Maxwellian become apparent in Fig. 6.

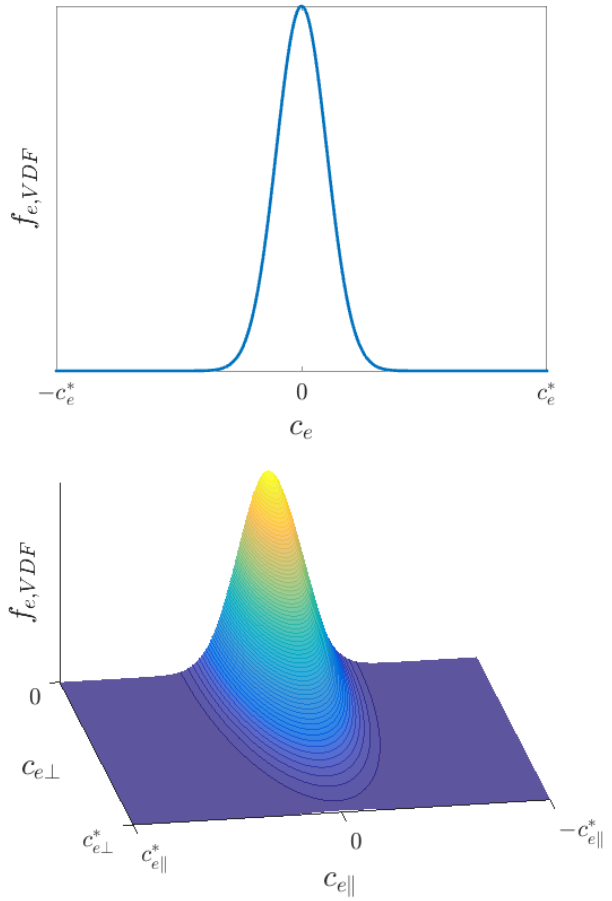

Figure 6. (top) *Isotropic Maxwellian VDF;* (bottom) *Anisotropic bi-Maxwellian VDF*

The higher-order moment closure for the electron fluid equations, along with the formal definitions of the various terms, are based on Barakat & Shunk's *16 moment approximation* [13], which includes density, momentum, energy, heat-flow and viscosity moments.
The equations to be integrated in NOMADS are a reduced version of this approximation, obtained through the following assumptions:

- Plasma is quasi-neutral.
- Electron drift kinetic energy may be neglected with respect to electron thermal energy.
- Viscous terms are neglected: a 12 moment approximation will be used.
- Convective and non-stationary terms in the momentum and heat-flow equations will be neglected with respect to electron pressure terms.
- The collisional terms for the momentum equation will be modelled using Maxwell-type molecule interactions. The ones for the heat-flow equations will be modelled using Krook's relaxation model.
- Azimuthal terms in the equations will be time-and-spatially-averaged to obtain "turbulent forcing" terms.

Other notable upgrades on the electron model will include: plasma production terms being obtained using updated experimental cross-section models for various gases (Xe, Ar, I_2, etc.) and a generalized plasma sheath model for a bi-maxwellian electron-fluid with arbitrary-incidence-angle of the magnetic field lines to the channel walls.

3.2.2. Numerical treatment

The use of the MFAM is best supported by a numerical approach based on a Cell-Centered (CC) Finite Volume Method (FVM); this method provides a formulation in the "strong" conservative, or integral, form (versus the weak forms of the Finite Element Method). Spatial discretization of gradients in the FVM method is handled through the Weighted Least Squares method (WLSQR) as defined by Sozer [14].
The temporal evolution of the problem, given solely by the energy equations when applying the aforementioned assumptions, will be discretized using an explicit 2nd order Adam-Bashforth scheme. This explicit-type scheme is a trade-off between stability of the method, complexity in its implementation and the toll on numerical resources; this last issue takes into account the implicit terms in the equation and the interdependencies with the momentum and continuity equations.
Finally, the momentum and continuity equations may be solved together as a generalized Ohm's law in the problem, which requires the use of solvers for large linear system of equations; the solvers proposed for this task are PETSc and PARDISO.

3.2.3. New modelling opportunities

The updated electron model will allow for simulating magnetic field topologies which were not possible in HPHall and subsequent versions: magnetic fields with singular points, cusps, "magnetically shielded" regions [15]. In addition, the two-temperature approach allows for non-uniform magnetic field effects to be modelled, such as the magnetic mirror or the exchange of parallel and perpendicular energies occurring in magnetic nozzles [16].

This will expand not only the group's capabilities in modelling of next generation HETs but also of other EP thruster types such as the ones mentioned in Section 1: HPT, ECRA, and HEMPT.

4. CONCLUSIONS AND FUTURE WORK

The main intended capabilities of NOMADS platform have been presented in this manuscript by giving an update on the current development status. Current efforts are focused on the model development and discretization, coding and integration of the different modules and

subroutines.

5. AKNOWLEDGEMENTS

This work has been supported by the Spanish R&D National Plan, Grant No. ESP2013-41052-P.

6. REFERENCES

1. Snyder, J., Brophy, J., Hofer, R., Goebel, D., Katz, I. (2012) Experimental Investigation of a Direct-Drive Hall Thruster and Solar Array System at Power Levels up to 10 kW. *Presented at the 2012 Joint Propulsion Conference*

2. Hofer, R., Jorns, B., Polk, J., Mikellides, I., Snyder, J. (2013). Wear test of a magnetically shielded Hall thruster at 3000 seconds specific impulse. *Presented at the 33rd International Electric Propulsion Conference*

3. Ahedo, E. & Navarro-Cavallé, J. (2013). Helicon thruster plasma modeling: Two-dimensional fluid-dynamics and propulsive performances. *Physics of Plasmas (1994-present)*, *20*(4), 043512

4. Cosmo, M. (2014). Horizon 2020 Strategic Research Cluster COMPET-03-2014: IN-SPACE ELECTRICAL PROPULSION AND STATION KEEPING. *Presentation material*

5. Escobar, D. & Ahedo, E. (2008). Two-Dimensional Electron Model for a Hybrid Code of a Two-Stage Hall Thruster. In IEEE Transaction on Plasma Science, Vol. 36, No. 5.

6. Parra, F., Fife, J. M. & Martinez-Sanchez, M. (2006) A two dimensional hybrid model of the Hall thruster. In J. Appl. Phys., vol. 100, no. 2, pp. 023 304-1--023 304-11

7. Fife, J. M. (1998). Hybrid-PIC modeling and electrostatic probe survey of Hall thrusters (Ph.D. dissertation), MIT, Cambridge, MA

8. Cichocki, F., Domínguez, A., Merino, M. & Ahedo, E. (2016). A 3D hybrid code to study electric thurster plume. *Space Propulsion Conference*, Rome, Italy.

9. Hockney, R.W. & Eastwood, J.W. (1988). Computer simulation using particles. *Adam Hilger*, Bristol and New York.

10. Birdsall, C.K. & Langdon, A.B. (1991). Plasma physics via computer simulation. *Adam Hilger*, Bristol, Philadelphia and New York.

11. Verboncoeur, J.P. (2001). Symmetric spline weighting for charge and current density in particle simulation. *Journal of Computational Physics,* 174, 421-427.

12. Pérez-Grande, D., Gonzalez-Martinez, O., Fajardo, P. & Ahedo, E. (2015). Benchmarks for Magnetic Field Aligned Meshes in Electromagnetic Plasma Thruster Simulations. *Presented at IEPC-2015* (Conference paper 203)

13. Barakat, A. R. & Schunk R. W. (1982). Transport Equations for Multicomponent Anisotropic Space Plasmas: A review. *Plasma Physics* Vol. 24, no. 4

14. Sozer, E., Brehm, C. & Kiris, C. (2014). Gradient Calculation Methods on Arbitrary Polyhedral Unstructured Meshes for Cell-Centered CFD Solvers. *52nd Aerospace Sciences Meeting*

15. Mikellides, I. G., Katz, I., Hofer, R. R., and Goebel, D. M., de Grys, K., and Mathers, A. (2011). Magnetic Shielding of the Channel Walls in a Hall Plasma Accelerator. *Physics of Plasmas*, Vol. 18, No. 3

16. Correyero, S., Navarro-Cavallé, J. & Ahedo, E. (2016). Kinetic modelling of collisionless electron cooling on magnetized plasma expansions. *Presented at Space Propulsion Conference 2016*