A SOFTWARE FOR COMPUTATIONAL PLASMA ENGINEERING FOR EUROPEAN SPACE INDUSTRY

17 – 18 – 19 MARCH 2021

S. Rouwette[(1)], U. Siems[(1)], L. Basov[(1)], S. Chibani[(1)], R. Bouziane[(1)], B. Mockel[(1)], F. Zhu[(1)], R. Schmit[(1)], D. Petkow[(1)], F. Taccogna[(2)], D. Feili[(3)], and G. Deprez[(3)]

(1) Sparc Industries Sárl c/o Technoport 1, 20 Rue du Commerce, L-3895 Foetz, Luxembourg, Email: d.petkow@sparc-industries.com

(2) Institute for Plasma Science and Technologies ISTP, CNR Research Bari Area, via Amendola 122/D, 70126 Bari, Italy, Email: francesco.taccogna@cnr.it

(3) ESA-ESTEC, Keplerlaan 1, 2200AG, Noordwijk, The Netherlands, Email: davar.feili@esa.int

KEYWORDS: electric propulsion, plasma simulation, computational engineering and design

ABSTRACT:

This paper reports about the upcoming release of a professional plasma simulation software targeting specifically plasma engineering and design work in the electric propulsion industry. It describes the plasma engineering issues that are addressed by this software, how these issues are addressed, which market players do support it already, and how users can get involved to maximize the added value provided to them.
The targeted readers are EP developers regardless of their host institution, and decision makers in charge of keeping competitivity at high levels.

1. INTRODUCTION

Plasma codes for EP applications are researched and developed since decades. Performance, physical representation and, in some cases, also usability, have been steadily improved. However, until today there are no easy to use, off-the-shelf solution on the market dedicated to EP engineering (similar to those flow solvers which are well established for example in the automotive industry) while at the same time allowing users to *not* study several years the fundamentals of these solvers in order to interpret their simulation results reliably and with confidence. Consequently, and, in comparison with large engineering industries like automotive, experimental engineering work and computational plasma flow analysis are domains that are largely decoupled from each other and with very little mutual support - while working on the same EP technologies. Thus, cost-, time-, and risk-to-market, remain very high, making today's EP companies struggle under New Space conditions. Those are primarily driven by "being first". This can be observed at EP start-ups, their VC based funds, their short development times, the reluctance to stick to major established R&D protocols in the EP industry developed over decades, and their eagerness to lunch as soon as possible for a successful in-orbit demonstration. In other words: It is not the cost-to-market, it is the time-to-market that decides upon commercialization strategy, R&D risks, and ultimately the market shares.
In order to allow coping with these trends in the EP market competition, SPARC Industries prepares to release a plasma simulation software specifically designed for this purpose.
This paper gives an overview of the software product, the philosophy behind its development, the differentiators, the targeted user group, the availability time frame (product release schedule), and more. Chapter 2 gives an overview of the identified requirements and the baseline categories that define these requirements. The next five chapters consist of a deeper dive of each of these five categories, starting with the credibility efforts for the software in chapter 3. Chapter 4 goes over the applicability of the software in relation to its environment and EP applications. The performance aspects of the software are addressed in chapter 5. Chapter 6 addresses the graphical user interface (GUI) and other aspects of ease-of-use. Reliability of the software and the measures to ensure this is covered in chapter 7. Release timeline and partners are outlined in chapter 8.

2. NEEDS & REQUIREMENTS

Based on interviews and surveys with members of the plasma simulation experts, EP developing companies, satellite manufacturers, space agencies and public research labs, major needs were identified and translated into as high-level user requirements. Based on these, and on own experience with R&D of plasma technologies under very low-pressure conditions we developed a plasma simulation software that addresses these requirements.
Part of the process was breaking down the user feedback into fragments and sorting them into categories. These categories define the baseline for the software product that we are developing, namely

- Credibility,
- Performance,
- Ease of use,
- Reliability, and
- Applicability.

Each of these five categories represent the five pillars which this software is based upon. Their interpretations are subject to debates due to differences across professions, communities, individuals, and levels of the value chain. Our definitions are as follows:

Credibility we define as a mixture of various test suits covering large parts of the code and automated support in result interpretation.

Performance is measured in simulation time but requires a direct link to the user's application and execution on hardware that is accepted by the user in order to become meaningful.

Ease of use we consider everything that removes obstacles which keep a potential user away from experiencing the software and exploring its capabilities to increase productivity and efficiency.

Reliability we achieve by using industrial s/w quality and development standards that minimize the cost per fixed bug and allows for high feature update frequencies.

Applicability we define as designing the software in such a way that it solves the user's engineering problem under their commercial constraints.

Each pillar contains (and benefits from) a multitude of elements. Describing all of them is far beyond the scope of this paper, but the following chapters give some inside into some of these elements.

3. CREDIBILITY

Credibility enhancing factors, especially in the EP community, are typically linked to the software's physical models and methods. We believe that the user needs to be put into the centre of the attention if credibility is supposed to become meaningful.

3.1. Integration Testing

Integration tests are used to assess the correct behaviour of one or more solvers as a whole as well as its integration in the complete architecture. In the community they are well known as "verification" and "validation" testing. The former tests the models that underline the targeted application. The latter is a reproduction of the targeted application based on experimental input and output data.

Verification

The set of verification tests is built so that the gap between low-level algorithmic tests, the unit tests, and full-fledged validation tests, is covered with a suitable level of confidence. Furthermore, it ensures that each software requirement is appropriately addressed, which also means that it will be extended when the requirements of the software are extended.

With final applications in mind, the verification effort is split into categories where a ladder of simple to complex tests are scheduled such that the fulfilment of a complete set allows approaching validation test with confidence. The references used are mostly cases for which an analytic solution exists, but also comprise code-to-code comparison with reputable literature.

Table 1. Today's snapshot of verification test suits progress.

	Verified/Planned	*Relative*
Ion optics	21/22	95%
HET	7/29	24%
General	4/6	67%
Coupled	5/7	71%
Total	37/64	58%

It can be seen in Table 1 that the *current* verification test goal for ion optics is almost achieved at the time of writing. Some software parts required for simulating HETs are already implemented, others are still under implementation. Hence, the fraction of successfully performed verification tests are lower compared to the ion optics verification test suit. The category "General" represents verification tests of software parts that show commonalities between the two applications. The category "Coupled" represents more complex verification tests, testing the framework as a whole.

Validation

Validation tests assess whether the software is fit for purpose. Here, the purpose is the industrial use. In other terms, validation tests assess the fulfilment of high-level user requirements as they have been identified in user and customer interviews, and specified to obtain acceptance and qualification by ESA. Currently, major constituents are an ion optics test case based on data provided by ArianeGroup Lampoldshausen GmbH [1], and a HET test case with data provided by TU Dresden [2]. The successful validation will demonstrate the ability of the software to tackle state-of-the-art EP applications and thus contribute to generating added value. Furthermore, the validation tests are additionally linked to performance requirements to assess compliance with the market requirements.

3.2. Interpretability

Our concept of interpretability is a composition of two elements: generation of interpretable data, and enabling the user to interpret the data. Together, they represent a hierarchical system since the latter cannot succeed without the former.

To exemplify the generation of interpretable data, we use the evolution of a typical DSMC simulation, shown in Figure 1.

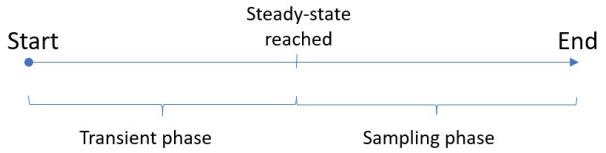

Figure 1. The two phases of a DSMC simulation of a steady-state problem.

In the first phase, the simulated flow evolves towards its steady-state which is considered the solution of the simulation. That phase is characterized by highly dynamic changes in local flow properties, requiring runtime adaptivity of space, and often of time. Sampling of macroscopic data is limited due to limited ensemble sizes.

Once steady-state is reached, sampling phase starts during which quantities of interest are sampled. Given the steady-state character of the flow, sampling over time increases the sample size thus allowing a much better signal to noise ratio.

Interpretability needs valid simulation data, and one can identify several pre-requisites of which we utilize two for the sake of exemplification:

1. Is steady-state reached? If the sampling phase starts too early, the sampled data gets compromised which contributes to simulation result invalidation. If the sampling phase starts too late, the user loses time at extra cost for something that does not bring any added value.
2. Is the correct steady-state reached? For arbitrary flow problems this question is not easy to answer. However, there are indicators that can be utilized, for example: If model assumptions are violated, then it contributes to simulation result invalidation. If the solution changes notably upon relocation of outflow boundaries, then the simulation data might not be entirely invalidated, but at least compromised. When the simulation results show notable dependency on spatial or temporal resolution, one can consider these results to be compromised, and potentially in some cases invalid.

The user should not need to invest large amounts of time to understand how to distinguish between valid simulation results and their counterparts. The software should do this for the user. Even though it is hard to agree on metrics that allow tagging a result valid or invalid, one can use existing methods to assess the numerical “footprint” in the simulation results and translate it into user-understandable terms like the *risk* associated with the obtained simulation results.

This combination of transparency and translation represents the second element of interpretability where the user is enabled to interpret the simulation results, provided that the data is generated in line with the first element of interpretability. To ensure this, respective control algorithms and checks are implemented in VSTRAP. They are used to assess the risk in the simulation results based on the concept of interpretability. Note, interpretability is such a broad and deep topic that we only scratch at its surface. Given the importance of it, we will provide additional aspects of interpretability in the presentation.

4. APPLICABILITY

This chapter exemplifies some elements that contribute to the applicability of the simulation software. These sections cover elements of software architecture, selected EP technologies and some of the implemented models and physics.

4.1. Enabling thruster technologies

Different thruster technologies require different model sets, covering different physical phenomena.

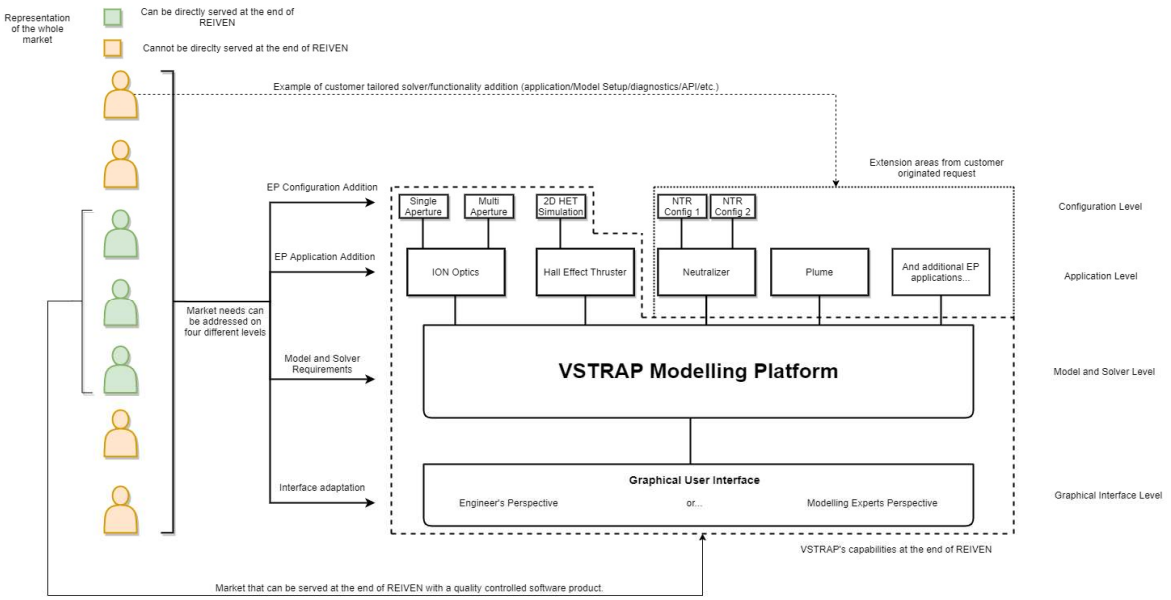

Figure 2. Representation of the modular and extensible modelling framework.

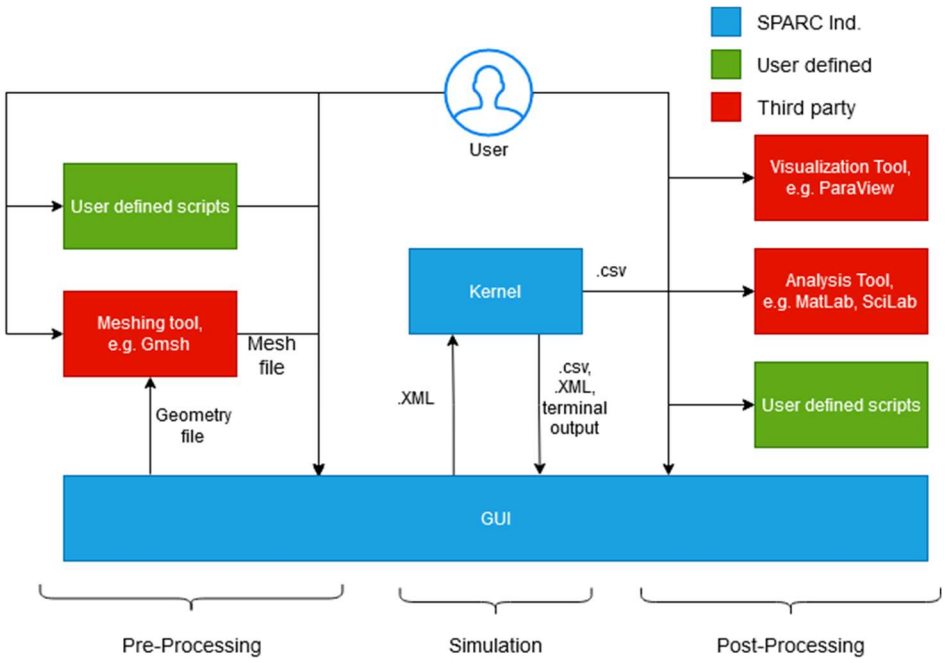

Figure 3. High-level representation of the software interfaces.

To be commercially attractive, aforementioned requirements need to be fulfilled. Today, available plasma codes have either a very general scope of applications (within the frame of the implemented models), or have been developed for a specific type of EP thrusters. Neither can match these requirements without drastic re-working and associated investment risks. To cope with that, the software architecture as foundation of the overall software needs to be designed to be fit for purpose. This is typically not addressed appropriately within the scientifically oriented funding frameworks.

VSTRAP allows subsequent extension of models and EP applications - based on customer requirements. That phasing is enabled by VSTRAP's modular design. The first two EP technologies being available upon release are Hall Effect Thrusters (HETs), and dual grid ion optics which are typically used Gridded Ion Thrusters (GITs/RITs) regardless of the discharge method.

Currently, HETs are simulated with rotational symmetricity in r, z space, while the ion optics are simulated in full 3D. Due to a modular design of the software on various levels, new modelling capabilities can easily be added, or dimensions. Thus, new/additional EP technologies' capabilities can be integrated very efficiently upon customer demand.

4.2. Modularity

Consideration was put in the design of the software architecture behind VSTRAP. The software is set up in a modular fashion providing a strict separation between individual modules. With this separation it is possible to implement components in parallel to one another which highly reduces the development time of new solvers. Additionally, this architecture makes new implementations safe in the sense that each developer can focus on the assigned component without the fear of breaking another part of the software. This reduces communication and synchronization overhead during development further reducing the overall time to develop a new software component. As such, new models can easily be implemented to extend the range of applicability of the software and to access the possibility to simulate more EP technologies over the lifetime of the software.

4.3. Workflow integration

The software has been designed with a clear separation between the GUI and the kernel. Since the GUI and the kernel have different requirements, the development of these components have different foci and need to be developed independently. As such, the GUI and kernel can also be executed independently from each other. As an example, the GUI can be executed on a regular workstation, while the kernel can be executed on a high-performance machine to make use of available computational power to allow timely delivery of the simulation results.

To link the simulation as configured by a user from the GUI to and its execution in the kernel, a system of XML files acts as the interface between the GUI and the kernel. These sets of files are generated by the GUI and are interpreted by the kernel to execute the simulation as configured. To interpret the results of an executed simulation, a set of CSV files are generated by the kernel during runtime, allowing data visualization and post-processing. Additional output formats can be added easily to allow smooth integration into the user's work environment.

Additionally, these output files act as the interface

between the kernel and the GUI for the simulation analysis. The GUI interprets and analyses the files to extract information about typical quantities of interests extracted from the simulation. The interface between the user and the software is detailed in Figure 3.

4.4. Hall Effect Thrusters

One target application of VSTRAP is the simulation of axisymmetric thruster configurations such as Hall Effect Thrusters. The inherent axis-symmetry is exploited by using a 2D simulation domain representing the radial and axial directions and neglecting gradients in the azimuthal direction, thereby drastically reducing the computational effort. All particles are simulated kinetically. This allows modelling of secondary electron emission and anomalous electron transport, which both play a crucial role in the physics underlying the Hall effect thruster [3] [4]. In detail, the contribution of the near wall conductivity to the anomalous electron transport induced by SEE can be properly resolved in a kinetic simulation [4]. The different time scales on which the dynamics of the neutrals and charged particles take place, are addressed by a staggered approach, i.e., separate model execution of neutral and plasma phases in an alternating fashion. The model employs a particle-in-cell and Monte-Carlo-collision (PIC-MCC) method in order to solve the Poisson and Boltzmann equation and obtain self-consistent plasma dynamics. An external static magnetic field is used as induced ones can be neglected [5] [6].
Coulomb interactions between charged particles on length scales smaller than the cell size, i.e., the Debye length, are resolved using the Nanbu method [7] [8]. The MCC method is also used to perform the interactions between charged particles and neutrals and to simulate the anomalous electron transport in the cross B-field direction. For the latter, a Bohm-like elastic scattering model is used [3]. Secondary Electron Emission (SEE) due to charge impact is implemented via a state-of-the-art model [9]. SEE is considered as an important interaction, contributing to anomalous electron transport and with an impact on the produced thrust [4].
Spatially resolved charge currents and erosion rates on solid walls exposed to the plasma are examples for possible diagnostics, which are computed by the software.
The neutralizer sustaining the plasma is not simulated directly. Instead, an electron current at low thermal energy is injected into the near-field plume region. That current is determined by a current balance [5].
Due to its high level of flexibility, with VSTRAP a tool is offered to easily perform variations in thruster configurations to optimize the thruster performance. Meaningful quantities, such as the erosion yield, important for lifetime assessment, or the produced thrust, are computed and outputted during the simulation.
Generally, features are in place that allow the user to easily make modifications to the simulation parameters and geometrical aspects to individual requirements.

4.5. Dual Grid Ion Optics

Ion optics are subject to constant erosion which defines the lifetime, dictated by the maximum allowed performance drop during operation. VSTRAP is designed as tool for performance optimization of which the wall erosion and respective lifetime are one of several metrics.

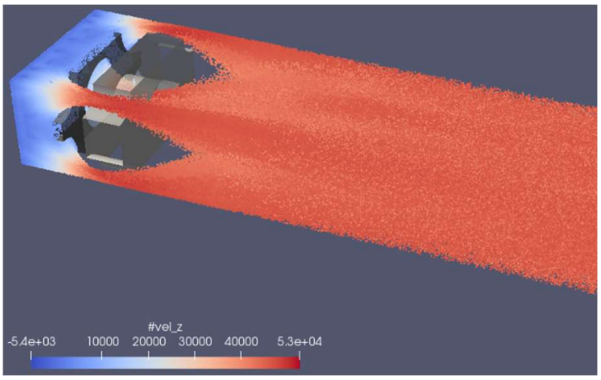

Figure 4. Ion optics single beamlet simulation at 499.5 ns showing ion extraction and acceleration. CEX processes ignored for visualization purposes.

In case of gridded ion optics, 3D plasma simulations are crucial to correctly resolve geometrical influences. VSTRAP allows simulating a single beamlet in an infinite pattern for hexagonal and rectangular grid patterns. We also provide the possibility to simulate a wedge-shaped sector of the full thruster utilizing 90°, 60° or 30° rotational symmetries. This is especially valuable for small ion optics configuration where the complex geometry plays a major role, like in the micro ion thruster [1] [10] [11], also used for one of our validations. To resolve the relevant physics, we implemented a combination of the Fast Multipole Method (FMM) and the Boundary Element Method (BEM) to solve the Poisson equation in a gridles manner. While the fully kinetic simulation is always available, the simulation can be accelerated by a hybrid method utilizing the Boltzmann relation for the electrons and kinetic representation only for the ions. The example in Figure 4 shows a dual grid configuration with screening and acceleration grids. Grid potentials, grid materials, propellant, and the ion inflow condition from the ionization stage are the main input parameters to be specified by the user. The grid spacing, the hole diameters and grid lengths are subject to parameter variations for design improvements, evaluated against user defined metrics like grid currents and erosion rates, to name a few.

5. PERFORMANCE

Generally, kinetic plasma simulations are computationally intensive and require a long time to

run, which conflicts with the user defined timeline requirements. To comply with them, reduction of simulation time to attractive time scales has a high priority. Methods include, among others, CPU parallelization, GPU acceleration, and algorithm optimization and different abstraction levels. In this chapter we will outline some of them.

5.1. GPU Acceleration

In High performance systems, Graphical Processor Units (GPUs) are used as accelerators thanks to their ability to accelerate computing using a large number of computational cores. Following this philosophy, VSTRAP is now including parts (solvers and utility code) of the software are ported to GPU using Compute Unified Device Architecture (CUDA) [12], developed by NVIDIA [13]. Using CUDA, the required simulation data is transferred to the GPU from the CPU and a large number of CUDA threads are launched simultaneously for parallel execution. Once all threads have finished, the computational results are transferred to the CPU. The cost of communication between CPU and GPU in the process of porting to GPU is a necessary and essential factor. We analyse and control the overhead that could limit the performance gain, effectively reducing the cost of data transfer between host and device as much as possible. Furthermore, we perform continuous performance evaluation of the kernel within a regular interval to prevent negative performance changes after feature updates. These tests are among our verification and validations tests for which the performance constraints must be satisfied.

5.2. Performance Customization

It is possible to choose between a full CPU execution and a hybrid execution, involving both CPU and GPU. The latter supports full exploitation of available features, specifically targeting multi-parameter design variations for the sake of matching user-specified performance metrics. Currently, the implementation is tested on a single NVIDIA TESLA V100 GPU [14]. Pre-production runs have demonstrated significant speedup in comparison to full CPU executions. The released product will allow utilizing multi-GPU configurations of the latest GPU generation (A100).
As the total simulation time is steadily reduced, it allows the user to not just reduce the time for a complete EP design variation. The targeted time frames are competitive in the sense that it allows to replace multiple design variations tested by experiment by simulations, thereby omitting various risks, e.g., associated with the supply chain (component prices, availabilities, delivery time) and personnel availability (for assembling the next EP design variant, testing it, analysing it, and generating the next design variant).

6. EASE OF USE

Ease-of-use is maximized by several elements and features. Some of these are outlined in this chapter.

6.1. Graphical User Interface

Dealing with a command line software does not provide the best user experience. The software input data is not really human readable and has a complex format. The user can easily make mistakes when working with such complex inputs. A gateway between the user and the simulation kernel to manage the interactions between them is highly efficient and generally desired.
The graphical user interface (GUI) of our simulation software is the interactive component between the user and the kernel. The user can only interact with the GUI and the GUI translates the user defined configuration into a format readable by the kernel.
The GUI allows setting up different kinds of simulations. Today, one can select between 2D plasma, 3D plasma, and 3D gas. Several features that are not known yet in the EP community make identification of valid and optimized thruster designs very convenient and efficient.
The GUI facilitates the use of the software kernel by providing the user all the visual controls to configure a simulation and pass it easily to the kernel. It provides a clean display of the different configuration sections inside a constructive model builder. The user can go through it and start setting up the simulation: defining the geometry, meshing it, defining the boundaries and their conditions for fields and particles, configure the species, the chemistry, the diagnostics, the simulation controls, and so on.
In the simulation setup, the GUI provides different kinds of automated processes to make the setup easier, for example by automated parameter variations, automated mesh generation, and automated job scheduling.
Another great feature of the GUI is the modern 3D visualization of the surface and volume meshes. Here we follow concepts known from other industries in order to allow the user to familiarize with the visualization controls much faster.
After finishing the simulation setup, the user executes the specified simulation job(s) via the GUI.

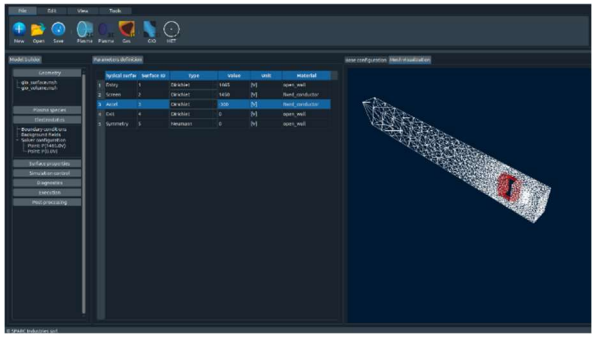

Figure 5. GUI exemplifying electrostatic boundary definition (dark-blue theme).

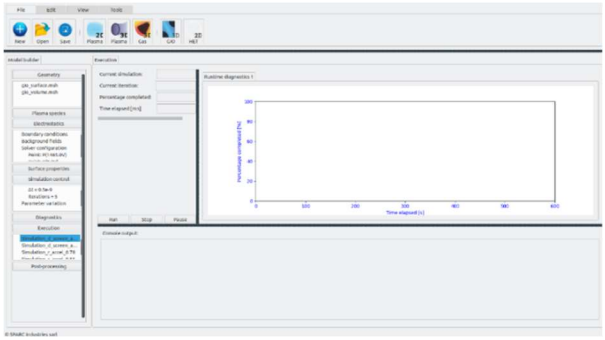

Figure 6. Representation of the GUI at the start of a simulation execution (light-coloured theme).

During the simulation, the GUI displays runtime diagnostics of various standard and user-defined outputs.
To match individual user preferences, the GUI comes with several themes, see Figure 5 and Figure 6.

6.2. Failsafe Operation

While setting up the simulation, the GUI supports the user in different ways, for example by checking the consistency and the correlation between the user input parameters and informing the user.

6.3. Low barrier to entry

During the different phases, from setting up the simulation to analyzing the output results, the GUI assists the user and explains the different parameters.
In general, the GUI is user friendly, it ensures an easy navigation between the different sections and it provides a kind of clarity that allow its users to rapidly begin engaging and interacting with it in a meaningful way.

6.4. Flexible simulation setup

The GUI ensures a flexible simulation setup where the user can configure the sections as preferred while dependencies between the sections and choices made therein are respected.
Right after starting a simulation, the user may want to update some parameters. In that case it is not required to re-setup the simulation from scratch as the GUI provides dedicated data records. Thus, the user can stop a simulation and update parameters and re-run the simulation while keeping the other configurations intact.

7. RELIABILITY

Similar to the other development pillars, multiple elements contribute to the reliability of VSTRAP.

7.1. Automated Testing

One of the major challenges in software development is the release of updates. Releases can unintentionally break parts of the code not touched by the update or turn functionalities not retro compatible. To combat this problem automated executions of the unit and regression tests are routinely performed.
This process is controlled by the continuous integration (CI) environment and procedures regarding the branching system associated with the used version control system. This allows for a testing framework which executes a test suit consisting of unit and regression tests when code is pulled through the development branches.

7.2. Testing

This section describes two of the testing efforts being conducted both to optimize the efficiency of the development and to meet industrial standards of quality assurance. The effort is lead on two fronts:

- Unit testing of individual, separable parts of the code (typically individual functions).
- Regression testing of physical functionalities of the code making sure that changes to the code do not alter the simulation results compared to previous simulations.

Unit tests
Unit tests are series of tests operating at algorithm level with little to no interactions with other parts of the software. These allow the developers to first isolate a section of the code and to assess its conformity in both functionality as well as input parameter ranges. All modifications of the code trigger a unit test event in our CI environment to ensure that the low-level integrity of the software is unaffected. Although common practice in industrial s/w development, it is very uncommon in the academic world of plasma solver development. During the development phase, unit code coverage ranges from 80-99% of lines of code.

Regression tests
Regression tests are carried out on a scheduled basis (time triggered) and whenever a substantial modification is applied to the reference version of the code (event triggered). Those tests mostly comprise integration tests and verification tests tailored for automation. The execution is triggered by the CI environment which then reports on any potentially occurring, metricated simulation result changes.
At the time of writing, the regression testing procedure includes 18 complex regression tests selected to maximize the functionality coverage.

7.3. Code quality

The software is developed based on test driven development principles. All new features are defined by a set of test cases which run automatically with every push to the version control system server. This creates a consistent test environment that makes sure that all additions to the code are tested. Additionally, such a test suit allows for safe changes of the code in the future as the

developer can be sure that their changes do not break the code when the test environment passes. Additionally, committing code to the CI environment triggers a code review process, where a peer developer reviews the changes for mistakes that might have slipped through testing, compliance to QA guidelines, and overall readability and understandability of the code. This improves the quality of the code by strengthening the confidence of the code, by keeping a high level of maintainability, and by being well maintained.

7.4. Continuous Deployment & Support

The outlined framework of code development and testing methods allows performing continuous deployment, delivering frequent updates. As the code is continuously extended in a commercial environment towards additional features to cover a growing number of different EP applications and features, customer support is in place by a standing team of developers, testers, and inhouse users.

8. SUMMARY

SPARC Industries develops professional plasma simulation software for the European space industry. Targeted users and customers are EP developers but also satellite manufacturers which are in need of additional exhaust data that allows to reduce their uncertainty-driven design margins for satellite contamination and charging analyses. The reduction of cost-, time-, and risk-to-market for novel EP developments, as well as the enhanced data sets EP developers can offer to their customers will increase their competitivity as well as their resilience against the rapidly changing conditions of the New Space market.

Given the challenge, but also the opportunity to massively speed up the innovation cycles of EP technologies, great support is provided already today by EP companies, space agencies, universities, and plasma simulation experts.

To harvest the added value brought to the market at each level of the value chain, the software is designed to be credible, applicable, fast, reliable, and, last but not least, easy to learn and use. There are different ways to implement these characteristics. Given the strong experience in the plasma simulation community across the world in the field of electric propulsion, and the experience in the industrial s/w development communities, we see the solution in combining the best of both worlds and write plasma simulation software that does what it is supposed to do, but at industrial standards.

A development team composed of physicists, mathematicians, space propulsion engineers, software developers, and computer scientists is preparing to release a beta version in the next months. The beta testing period is until the summer of 2022 when the full software will be released with features that can be nutshell-ed as follows: plasma flow solver for kinetic and hybrid flow problems; covers electrostatics and magnetostatics; noble gas chemistry; various plasma-wall interactions like erosion, secondary electron emission, and surface charging; GPU accelerated; user-friendly GUI, data base, and documentation; initially targeting HETs and ion optics; made for easy extension towards more EP applications. Equipped with features allowing to focus on business-driven simulation objectives.

Acknowledgements

This work is supported by the European Space Agency via the LuxIMPULSE programme of the Luxembourg Space Agency, Prof. Francesco Taccogna, the ArianeGroup Lampoldshausen GmbH, and TU Dresden.

9. REFERENCES

[1] T. Binder, M. Pfeiffer and S. Fasoulas, “Validation of grid current simulations using the particle-in-cell method for a miniaturized ion thruster,” in *AIP Conference Proceedings 2132*, 2019.

[2] N. Gondol, C. Drobny, Neunzig and M. Tajmar, “Development and Characterization of a Miniature Hall-Effect Thruster using Permanent Magnets,” in *36th International Electric Propulsion Conference (IEPC)*, 2019.

[3] I. D. Kaganovic, A. Smolyakov, Y. Raitses, E. Ahedo, I. Mikellides, B. Jorns, F. Taccogna, R. Gueroult, S. Tsikata, A. Bourdon, J. Boeuf, M. Keidar, A. T. Poweis, M. Merino, M. Cappeli, K. Hara, J. Carlsson, N. J. Fisch, P. Chabert, I. Schweigert, T. Lafleur, K. Matyash, A. V. Khrabrov, R. W. Boswell and A. Fruchtman, “Physics of E×B discharges relevant to plasma propulsion and similar technologies,” *Physics of Plasmas,* vol. 27, no. 12, p. 120601, 2020.

[4] E. Bultinck, S. Mahieu, D. Depla and A. Bogaerts, “The origin of Bohm diffusion, investigated by a comparison of different modelling methods,” *Journal of Physics D: Applied Physics,* vol. 43, no. 29, p. 292001, Jul 2010.

[5] F. Taccogna, S. Longo, M. Capitelli and R. Schneider, “Stationary plasma thruster simulation,” *Computer Physics Communications,* vol. 164, no. 1, pp. 160-170, 2004.

[6] F. Taccogna and L. Garrigues, “Latest progress in Hall thrusters plasma modelling,” *Reviews of Modern Plasma Physics,* vol. 3, no. 1, 2019.

[7] K. Nanbu, “Theory of cumulative small-angle collisions in plasmas,” *Phys. Rev.,* vol. E, no. 55, p. 4642–52, 1997.

[8] A. V. Bobylev and K. Nanbu, “Theory of collision algorithms for gases and plasmas based on the Boltzmann equation and the Landau-Fokker-Plack equation,” *Phys Rev E,* vol. 61, p. 4576–86, 2000.

[9] M. A. Furman and M. T. F. Pivi, “Probabilistic model for the simulation of secondary electron

emission," *Phys. Rev. ST Accel. Beams,* vol. 5, no. 12, p. 124404, 2002.

[10] D. Feili, B. Lotz, S. Bonnet, B. Meyer, H. Loeb and N. Puetmann, "μNRIT-2.5 -A New Optimized Microthruster Of Giessen University," in *31 International Electric Propulsion Conference*, 2009.

[11] D. Feili, H. Loeb, K. Schartner, D. Kirmse, S. Weis, B. Meyer, R. Kilinger and H. Mueller, "Testing of new μN-RITs at Giessen," in *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference Exhibit*, 2005.

[12] "CUDA Zone," NVIDIA, [Online]. Available: https://developer.nvidia.com/cuda-zone. [Accessed 2021].

[13] "NVIDIA main page," NVIDIA, [Online]. Available: https://www.nvidia.com/. [Accessed 2021].

[14] NVIDIA, "Volta architecture whitepaper," 2017. [Online]. Available: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf. [Accessed 2021].

[15] K. Nanbu and Y. Kitatani, "An ion-neutral species collision model for particle simulation of glow discharge," *Journal of Physics D: Applied Physics,* vol. 28, no. 2, pp. 324--330, 1995.

[16] J. Szabo, N. Warner, M. Martinez-Sanchez and O. Batishchev, "Full Particle-In-Cell Simulation Methodology for Axisymmetric Hall Effect Thrusters," *Journal of Propulsion and Power,* vol. 30, no. 1, pp. 197-208, 2014.

[17] S. E. Olson and A. J. Christlieb, "Gridless DSMC," *J. Comput. Phys.,* vol. 227, no. 17, p. 8035–8064, 2008.

[18] R. Jambunathan and D. A. Levin, "Grid-Free Octree Approach for Modeling Heat Transfer to Complex Geometries," *Journal of Thermophysics and Heat Transfer,* vol. 30, no. 2, pp. 379-393, 2016.