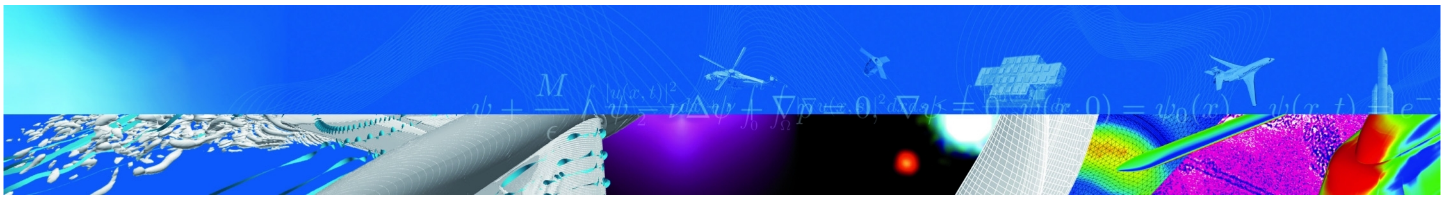


ONERA

THE FRENCH AEROSPACE LAB

P R O G R E S S   R E P O R T

Improved Modelling of Electrical Thruster Induced Plasma Plume Interactions SPIS-EP Detailed Design Document User manual

Authors : S. Hess ; P. Sarrailh ;

J.-C. Mateo Velez ; B. Jeanty-Ruard ; A. Trouche

PHYSICS INSTRUMENTATION ENVIRONMENT SPACE

RA 7/25473 DPHY - September 2018

BP 4025 - 2, avenue Edouard Belin
31055 Toulouse Cedex - FRANCE
Tél. : 05 62 25 25 25 - Fax : 05 62 25 25 50

PHYSICS INSTRUMENTATION ENVIRONMENT SPACE

Progress Report N° RA 7/25473 DPHY

September 2018

Improved Modelling of Electrical Thruster Induced Plasma Plume Interactions SPIS-EP Detailed Design Document User manual

Written by :

S. Hess ; P. Sarrailh ; J.-C. Mateo Velez ; B. Jeanty-Ruard (ARTENUM) ;
A. Trouche (ARTENUM)

Approved by :

Director
Physics Instrumentation Environment space
J.-F. Roussel

[illegible]

IDENTIFICATION CARD of ONERA REPORT N° RA 7/25473 DPHY

Issued by : **PHYSICS INSTRUMENTATION ENVIRONMENT SPACE**	Contracting Agency : **ESA**	Contract Number : **4000116103/15/NL/LF**
		Date : **September 2018**
Title : **Improved Modelling of Electrical Thruster Induced Plasma Plume Interactions SPIS-EP Detailed Design Document User manual**		
Author(s) : **S. Hess ; P. Sarrailh ; J.-C. Mateo Velez ; B. Jeanty-Ruard ; A. Trouche**		
SECURITY CLASSIFICATION : Civil Title : UNPROTECTED ID Card : UNPROTECTED Report : UNPROTECTED	Timing Classification Off Title : Without object ID Card : Without object Report : Without object	

Abstract :

This document aims at guiding users of the Spacecraft-Plasma Interaction Software (SPIS) using the Electric Propulsion version to set-up and run simulations. This software is available on the Linux, Mac OSX 10, Windows 7 and Windows 10. It also describes the modifications of the architecture and the models developed in the frame of the “Improved Modelling of Electrical Thruster Induced Plasma plume Interaction” activity under ESA contract number 4000116103/15/NL/LF, whose objective is to improve the capabilities of SPIS in order to perform more accurate simulations of electric thruster plumes, of their interaction with the spacecraft and of their coupling with the environment.

Key words :

DISTRIBUTION LIST of ONERA REPORT N°RA 7/25473 DPHY

Distribution of report

• Outside ONERA :

ESA	F. Cipriani (diffusion électronique)
Thales Alenia Space	V. Perrin (diffusion électronique)
Airbus Defence and Space	C. Théroude - T. Lagrée (diffusion électronique)
ARTENUM	B. Jeanty-Ruard - A. Trouche (diffusion électronique)

• Inside ONERA :

DPHY/CSE	S. Hess - P. Sarrailh - J.C. Matéo Vélez	3 ex.
DIST	Documentation	1 ex.

Distribution of identification card

Systematic distribution : DSP, DTP, DAI .... 3 ex.

Improved Modelling of Electrical Thruster Induced Plasma plume Interaction
-
SPIS-EP

Detailed Design Document
User manual

September 2018

ESA/ESTEC contract No 4000116103/15/NL/LF

ONERA
ARTENUM

Detailed design document

September 2018

Contributors

Sébastien Hess
Pierre Sarrailh
Jean-Charles Matéo-Vélez

ONERA/DPHY
2 Av. Edouard Belin
31055 Toulouse cedex, France

Benjamin Jeanty-Ruard
Arnaud Trouche

Artenum
20, rue Hermes
31520 Ramonville Saint Agne

Document Status Sheet			
Document Title:			
Issue	**Date**	**Author(s) of document/change**	**Reason of change**
0.1	17/05/2017	SH	Creation
0.2	13/06/2017	SH	Input from Artenum
0.3	28/06/2017	SH	Re-organization
0.4	03/06/2017	PS, SH	Final version
0.5	15/06/2017	SH	Corrected version
0.6	05/05/2018	SH	Updated version
1.0	10/09/2018	PS, SH	Final version
2.0	28/11/2019	SH	Updated version

1 SUMMARY

1. SUMMARY

1.1. Objectives

This document aims at guiding users of the Spacecraft-Plasma Interaction Software (SPIS) using the Electric Propulsion version to set-up and run simulations. This software is available on the Linux, Mac OSX 10, Windows 7 and Windows 10. It also describes the modifications of the architecture and the models developed in the frame of the “Improved Modelling of Electrical Thruster Induced Plasma plume Interaction” activity under ESA contract number 4000116103/15/NL/LF, whose objective is to improve the capabilities of SPIS in order to perform more accurate simulations of electric thruster plumes, of their interaction with the spacecraft and of their coupling with the environment.

1.2. This report structure

This document is divided into three sections:

- User Manual. Section describes the new interfaces of SPIS and the new interactions that can be simulated using SPIS-EP. The set-up of these new interactions is explained.
- Software Design. Section describes in depth the new architecture of the software.
- Models. Section describes in details the new models implemented in SPIS-EP.

1.3. Acronyms

ONERA	Office National d'Etudes et de Recherches Aérospatiales
S/C	Spacecraft
SOW	Statement Of Work
SPIS	Spacecraft Plasma Interaction Software

1.4. Reference Documents

[SPIS]	Spacecraft Plasma Interaction Software, www.spis.org
[SUM]	Spis User Guide, delivered with the SPIS package, www.spis.org
[HV]	PLASMA CURRENT COLLECTION OF HIGH VOLTAGE SOLAR ARRAY: NUMERICAL INVESTIGATION, Hess et al, Proceedings of the 14th Spacecraft Charging Technology Conference, ESA/ESTEC, Noordwijk, NL, 04-08 APRIL 2016 1
[UR]	SPIS-EP user requirement document
[SR]	SPIS-EP software requirement document
[FTR]	SPIS-EP functional test report

1.5 Caution and limit of Warranty

The present document is intended to be a simple User Manual (UM) to help the user to use SPIS in basic mode and follow a step-by-step simple simulation. We remind that SPIS is still an expert tool. This document does not attempt to provide to the future SPIS user the whole expertise and knowledge needed to perform the complete modelling of a spacecraft. The final confidence in the relevance and accuracy of simulation results depends on the user expertise and remains his responsibility.

FOR THIS REASON, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING, THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

SPIS is basically the coupling of an electrical circuit solver and a plasma simulation code. Plasma being by nature a strongly coupled and instable assembly of charged particles and electromagnetic fields, the stability of the simulation is not ensured in the general case. While general guidelines exist for plasma code stability, a strong expertise in plasma physics and plasma simulation is required to setup and interpret the result of SPIS simulations in which the plasma exert a strong influence on the results. Moreover, in spite of the simplification of the user interface, it is still strongly recommended to have good knowledge in space environment, space engineering and mathematical modelling before any simulation of realistic and/or critical systems is performed.

2. VERSION NOTES

IMPORTANT NOTICE: because of the new monitor handling, the simulations built with old versions of SPIS have to be reset (i.e. click on Finalize Simulation in the global parameter view) in order to get the 2D and 3D views.

2.1 Changelog

Bug corrections : all bugs that have been reported were corrected, except for a bug on tabulated distribution that could not be reproduced.

Thin elements: Many improvements concerning thin elements definition and simulation. The Poisson solver underwent a major revision. Some monitors were improved to refine the outputs around the thin elements.

User interface: Group definition reworked to “hide” the parameters that are not relevant for the user. Monitors appear as instruments so they can be added, removed and modified during the simulation.

Electrical circuit: Allow for the use of the wider SPICE netlist syntax. Possibility to use models, time varying elements, multiple grounds, ...

2.2 Bug corrections and major changes

The following table list the major changes between the version 6.0.0 and the version 6.0.4

#	Type[a]	Short description	Origin[b]	Emitter	Status[c]	Long description
1	B	Exact CSat	D	G. Déprez	S	Problem of unit due to non-duplication of Dimscal
2	B	No RIC	D	G. Déprez	S	RIC could not be activated because of a wrong variable type (regression)
3	B	No volume Shading	NRC	S. Hess	S	Bug in the volume shading provoked a crash at initialization that led to wrong results
4	B	Bad Sticking computation	T	M. Villemant	S	Wrong implementation led to wring sticking coefficient evolution wrt. temperature.
5	B	Tabulated distribution	D	A. Waerts	N	Bug could not be reproduced
6	B	File writing	D	Several users	S	File was not written often enough, nor written to the end. Cause: JAVA never implemented the flush command.
7	B	Non Xe Erosion	I	S. Hess	S	Erosion for ions that are not Xenon
8	B	Edit Thin Panel groups	I	S. Hess	S	Impossible to re-edit thin panel groups after their first definition
9	A	Maxwell distribution bins	NRC	S. Hess	S	Problem of accuracy in the Sphere charging backtracking NRC due to bad sampling of the distribution

10	A	Thin wire	C, NRC	F. Cipriani, S. Guillemant	S	Problem in the wire modelling detected by S.Guillemant led to the disconnection of the thin wire handling since v5.2. Poisson solver rewritten.
11	U	Thin element setting	F	M. Diaz-Aguado	S	The settings of thin wire required the change of two properties, which are is obvious and not documented. Group redefinition.
12	U	Multiple 2D element	F	M. Diaz-Aguado	P	Impossible to have several thin panels. Partially treated, by simplified UI settings and NUM modification. But the surface duplication must be made “by hand” in Gmsh
13	U	Monitors as Instruments	I	S. Hess	S	Monitors are defined as instruments, allowing a better control of the monitoring
14	U	SPICE netlist syntax	I	S. Hess	S	Use the SPICE netlist syntax together with the SPIS legacy. Allows for clearer description and comments.
15	U	Node naming	B	B. Thiébault (Bug 341)	S	The node name uses 2 numbers, so they are in good order
16	F	Time varying circuit	C,D	F. Cipriani, P. Henri	S	Request for time varying electrical components. Use of SPICE like models. Required by on-going contractual activities and LPC2E for Rosetta MIP.
17	F	Chamber simulation	F, C	A. Gourdin, P. Sarrailh, O. Jorba	S	Need for simulations with non-floating reference potential (e.g. chamber simulation)
18	F	1D wire alone on node	F, C	M. Diaz-Aguado	P	Wires had to be on nodes with surfaces. Now they can be alone on their node.
19	F	Changing parameters during simulation	B, C	B. Thiébault (Bug 281) F. Cipriani	P	Some parameters that control accuracy vs simulation duration can be changed during simulation
20	O	Singularity visualization	F	Several users	S	Non-linear fields around singularities can now be visualized
21	O	Ghost surface for 3D data view on nodes	F	Several users	S	Non physical triangles appeared sometime between nodes.
22	O	Useless monitoring	D	Several users	S	The number of default quantities monitored has been decreased, better control by global parameters
23	P	Caching data	I	S. Hess	S	Particle data are cached to avoid searching long tables
24	B	View factor for large mesh	D	G. Déprez	P	View factors must be disabled if the mesh has more than 130000 vertices (JAVA array limit)

[a]Type: **B** Bug, **U** Usability improvement, **P** Performance improvement, **A** accuracy improvement, **F** new Functionnality, **O** Output/visualization improvement

[b]Origin: **F** community (Forum), **D** Direct demand to developers, **C** Contractual demand, **I** Internal development, **NRC** problem detected during NRC, **V** Validation activity, **T** Test activity, **B** Bug tracker

[c]Status: **S** Solved, **P**: Partially solved, **N**: Could not be solved.

3. SIMULATION SET-UP

3.1. Plugins

The Electric Propulsion version of SPIS introduces a new architecture using plugins instead of the usual monolithic architecture of SPIS. These plugins are optional: All of the key components of the SPIS numerical core – in particular the solvers – are gathered in the SPIS-NUM-Core package that is mandatory and can run in a stand-alone mode. Plugins are used to introduce new physics in SPIS. For example, all of the interactors, instruments and particle-related components that describe the dust-spacecraft interactions are gathered in the dust package. The electric propulsion physics (i.e. descriptions of thrusters, cathodes and related distributions) is gathered in the “*electric-propulsion*” package, the solar panel interactions are defined in the “*highvoltage*” panel and the erosion and contamination physics is defined in the “*erosion-contamination*” package.

3.1.1. Installation

A plugin appears as a .jar file. To install it, just copy the file in the *lib* directory of your SPIS distribution. It will be automatically loaded next time SPIS is started.

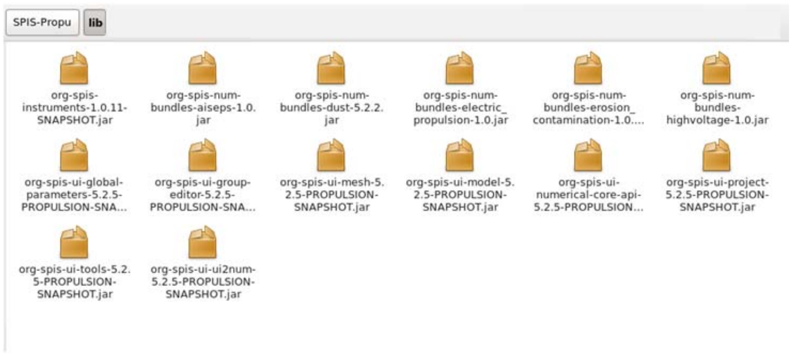

SPIS-Propu
lib
org-spis-instruments-1.0.11-SNAPSHOT.jar
org-spis-num-bundles-aiseps-1.0.jar
org-spis-num-bundles-dust-5.2.2.jar
org-spis-num-bundles-electric_propulsion-1.0.jar
org-spis-num-bundles-erosion_contamination-1.0....
org-spis-num-bundles-highvoltage-1.0.jar
org-spis-ui-global-parameters-5.2.5-PROPULSION-SNA...
org-spis-ui-group-editor-5.2.5-PROPULSION-SNA...
org-spis-ui-mesh-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-model-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-numerical-core-api-5.2.5-PROPULSION...
org-spis-ui-project-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-tools-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-ui2num-5.2.5-PROPULSION-SNAPSHOT.jar

Figure 1: lib folder of the SPIS-EP distribution (snapshot version) with several optional packages: AISEPS, Dust, Electric propulsion, Erosion & Contamination and High Voltages

3.1.2. Using the plugin functionalities

All classes that are to be instantiated explicitly by SPIS-NUM core are classes that are specified in SPIS-UI as SPIS local or global parameters. SPIS first tries to load the class whose name corresponds to the parameter as a SPIS-NUM Core class. If it does not succeed, it tries to load that class from all bundles one by one. It loads the class of the first bundles that contains a class with the correct name. If several bundles share classes with the same name (which should be avoided as much as possible but could happen with plugins from different origins), SPIS-NUM will use the first class it finds, which leads to an unpredictable behaviour. To avoid confusion it is possible (and recommended when using packages that does not belong to an official SPIS distribution) to specify the plugin in the parameters by placing the name of the plugin between brackets {} in front of the class name. Example: putting *{dust}DustSurfSource* as *interactorName* in the global parameters specifies that the surface interactor *DustSurfSource* belongs to the dust package and that the class path is *spis.bundles.dust.Surf.SurfInteract.DustSurfSource*.

The classes that can be instantiated in SPIS are:

Name/use	Belong to package	Derive from class
Volume distribution	spis.bundles.*.Vol.VolDistrib	spis.Vol.VolDistrib.VolDistribWithIO
Surface distribution Source	spis.bundles.*.Surf.SurfDistrib	spis.Surf.SurfDistrib.SurfDistrib
Surface Interaction	spis.bundles.*.Surf. SurfInteract	spis.Surf.SurfInteract.Interactor
Volume Interaction	spis.bundles.*.Vol.VolInteract	spis.VolInteract.VolInteractor
MagneticField	spis.bundles.*.Vol.VolField	spis.Vol.VolField.BField
Transition	spis.bundles.*.Top.Transition	spis.Top.Transition.Transition
Environment	spis.bundles.*.Top.Plasma	spis.Top.Plasma.Environment
Scenario	spis.bundles.*.Top.Top	spis.Top.Top.Scenario
spacecraftType	spis.bundles.*.Top.SC	spis.Top.SC. RLCSC
photoelectronEnvironmentDF **secondaryEnvironmentDF** **pop*X*EnvrinomentDF**	spis.bundles.*.Util.DistribFunc	spis.Util.DistribFunc.DistributionFunction
erosionYield	spis.bundles.*.Surf.SurfInteract	spis.Surf.SurfInteract.ErosionYieldAPI
erosionSampler	spis.bundles.*.Util.Sampler	spis.Util.Sampler.ErosionProductsSampler

3.2. Geometry and Mesh

Geometry and mesh are defined as in previous versions of SPIS. In is recommended to use small cell sizes at the exhaust of the thrusters in order for the plume to be well described and for the charge-exchange reactions to be well computed. Depending on the type of thruster surface distribution used, it is not mandatory to mesh the thruster exhaust. However, it is highly recommended to ensure an accurate simulation. The surface group identified as the thruster does not collect or emit currents (except the currents emitted by the thruster and the cathode that are handle specifically). This allows avoiding emission of secondary emission due to electron, ion and photon impact from the exhaust surface (which would be unphysical as the physics and geometry in the exhaust of the thruster is not described at all). This also prevents to take into account the recollection of the cathode electrons at the exhaust of the thruster (which would also be unphysical for the same reasons).
If the exhaust is unmeshed, it is important to have a thruster surface that is not much larger than the exhaust surface. Otherwise, the currents collected and emitted by the surfaces surrounding the thruster exhaust will not be taken into account which may lead to unphysical results. This would in particular lead to a spacecraft potential heading toward the positive values due to a lack of cathode electron recollection.

It may also be beneficial to use a mesh refinement in the plume region to ensure that the plume divergence and the charge exchanges are accurately modelled.

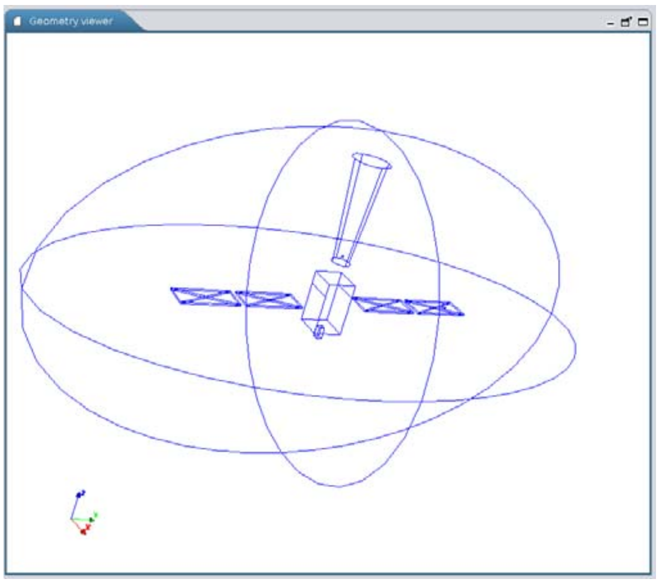


Figure 2: example of CAD with a refined plume region

One of the most numerically challenging parts of the SPIS- Electrical propulsion simulation is the resolution of the non-linear Poisson equation. It is highly recommended to edit the mesh with Gmesh and to use its 3D mesh optimization methods to ensure a correct mesh quality. This quality can be checked by using the “Statistic” option in the tool menu of Gmesh, which gives the cell quality “Gamma” after the “Update” button was clicked on. It is recommended to avoid simulation with average values of Gamma lower than 0.6 and minimal value less than 0.1. The Gamma histogram is displayed at the beginning of the SPIS simulation, in the Summary report window, for control.

3.3. On-Ground geometry definition

The simulation geometry is decomposed in different zones:

- The vacuum tank by itself which is composed by two tanks that are connected (the principal tank and the secondary tank in which the thruster can be retracted)
- The pumping zone which is at the opposite side of the thruster. We consider that the pumping zone is composed by the pumping connection itself plus the cold zones around it.
- The thruster zone which is composed by the surfaces of the ion injection plus the body of the thruster. The cathode is not geometrically modelled.

The geometry obtained is shown in the following figure.

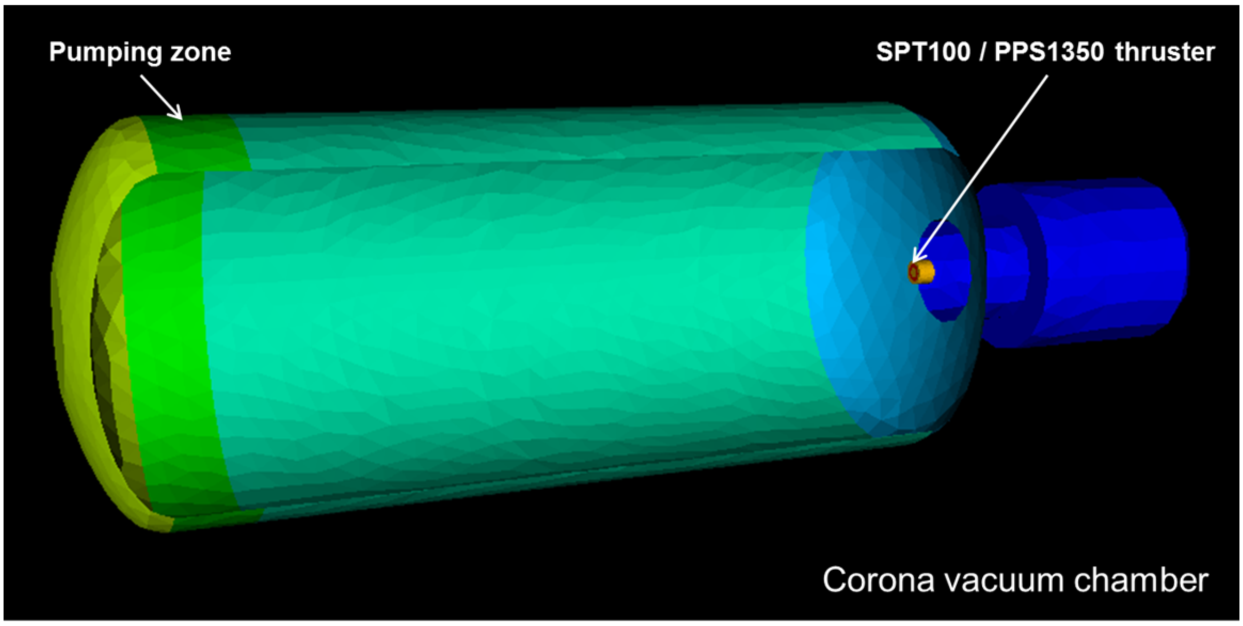
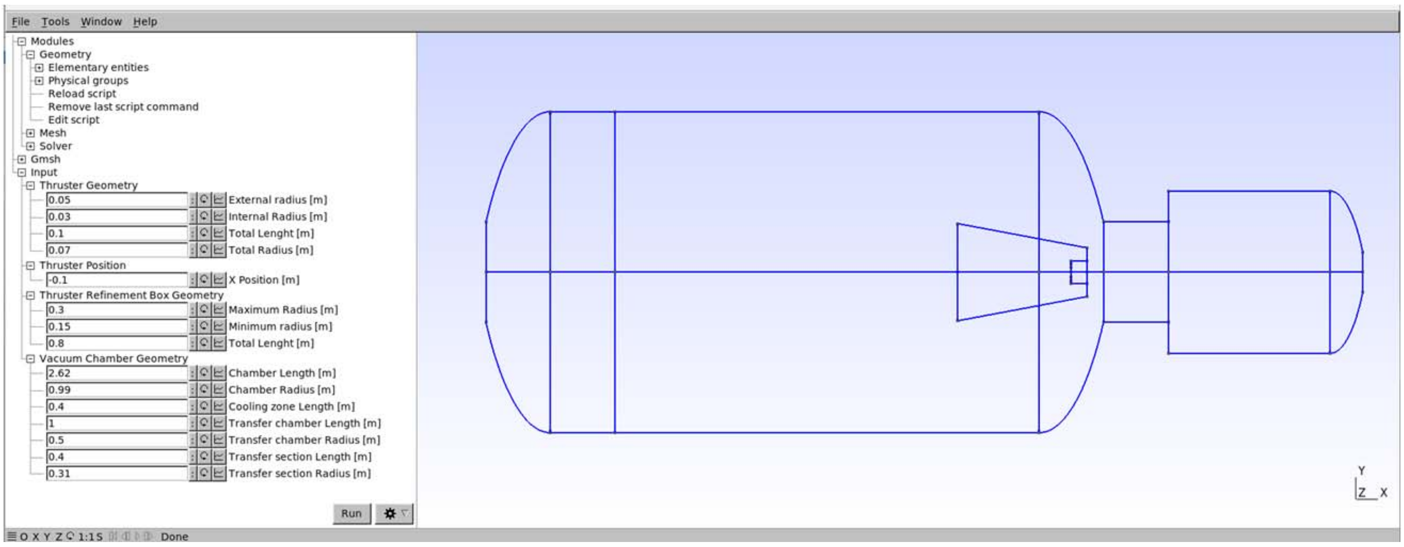


Figure 3: Example of Corona vacuum chamber representation with pumping and cooling zones

In the frame of this project, a geometry with parameter is provided for the user (as can be seen in the picture below). This geometry parameter can be changed in the file itself of using the GMSH user interface (since GMSH 4.0). Different parameters are grouped in 4 sections:

- Thruster geometry
- Thruster position
- Thruster refinement box geometry
- Vacuum chamber geometry

The default parameters are those from the Corona vacuum chamber of ESA as represented in the previous figure.



Thruster Geometry

0.05	External radius [m]
0.03	Internal Radius [m]
0.1	Total Lenght [m]
0.07	Total Radius [m]

Thruster Position

-0.1	X Position [m]

Thruster Refinement Box Geometry

0.3	Maximum Radius [m]
0.15	Minimum radius [m]
0.8	Total Lenght [m]

Vacuum Chamber Geometry

2.62	Chamber Length [m]
0.99	Chamber Radius [m]
0.4	Cooling zone Length [m]
1	Transfer chamber Length [m]
0.5	Transfer chamber Radius [m]
0.4	Transfer section Length [m]
0.31	Transfer section Radius [m]

Figure 4: Example of parameterization of the thruster and the chamber

3.4. Definition of the surface properties and interactors

The group definition offered by SPIS-Electric propulsion is similar to the previous versions of SPIS. It allows defining:

3.4.1. The simulation domain and its characteristics.

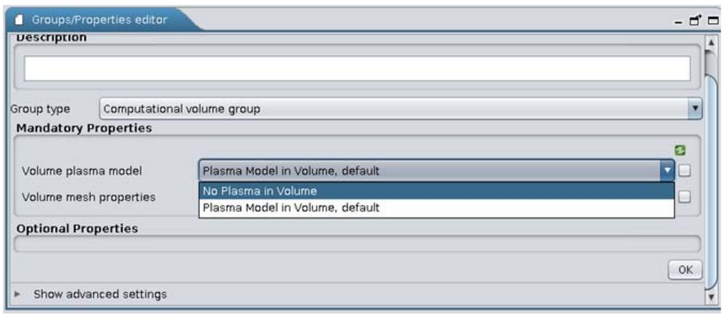
The *volInteractFlag* allows determining how the plasma behaves in the volume

0: no interactions in volume (works only with generic interactions);

1: plasma interacts in volume (default);

-1: no plasma in this volume

The plasma behaviour can be set by changing the *volInteractFlag* characteristics or by choosing one of the predefined plasma models for the volume. When the flag is set to -1, the volume is seen by the plasma as if it was out of the simulation domain: surface emissions occur outward of this volume, fluid distributions assume a 0 density and the PIC particles are not pushed. Nonetheless, the Poisson equation is solved.


Groups/Properties editor

Description

Group type: Computational volume group

Mandatory Properties

Volume plasma model: Plasma Model in Volume, default
- No Plasma in Volume
- Plasma Model in Volume, default

Volume mesh properties

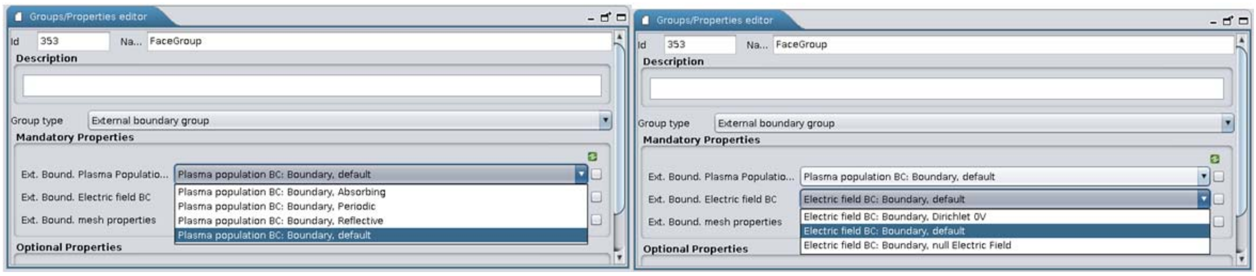
Optional Properties

Figure 5: predefined plasma model in volume

3.4.2. The boundary conditions

New predefined plasma and electric field conditions were added.
Predefined plasma conditions are:

- Open (default): particles can be injected and lost through the surface
- Reflective: No particles injected from the environment, while plasma particles are reflected
- Periodic: No particles injected from the environment, while plasma particles are re-injected on the opposite surface
- Absorbing: No particles injected from the environment, while plasma particles are lost through the surface


Groups/Properties editor

Id: 353 Na...: FaceGroup

Description

Group type: External boundary group

Mandatory Properties

Ext. Bound. Plasma Populatio...: Plasma population BC: Boundary, default
- Plasma population BC: Boundary, Absorbing
- Plasma population BC: Boundary, Periodic
- Plasma population BC: Boundary, Reflective
- Plasma population BC: Boundary, default

Ext. Bound. Electric field BC

Ext. Bound. mesh properties

Optional Properties

Groups/Properties editor

Id: 353 Na...: FaceGroup

Description

Group type: External boundary group

Mandatory Properties

Ext. Bound. Plasma Populatio...: Plasma population BC: Boundary, default

Ext. Bound. Electric field BC: Electric field BC: Boundary, default
- Electric field BC: Boundary, Dirichlet 0V
- Electric field BC: Boundary, default
- Electric field BC: Boundary, null Electric Field

Ext. Bound. mesh properties

Optional Properties

Figure 6: predefined plasma and Poisson boundaries

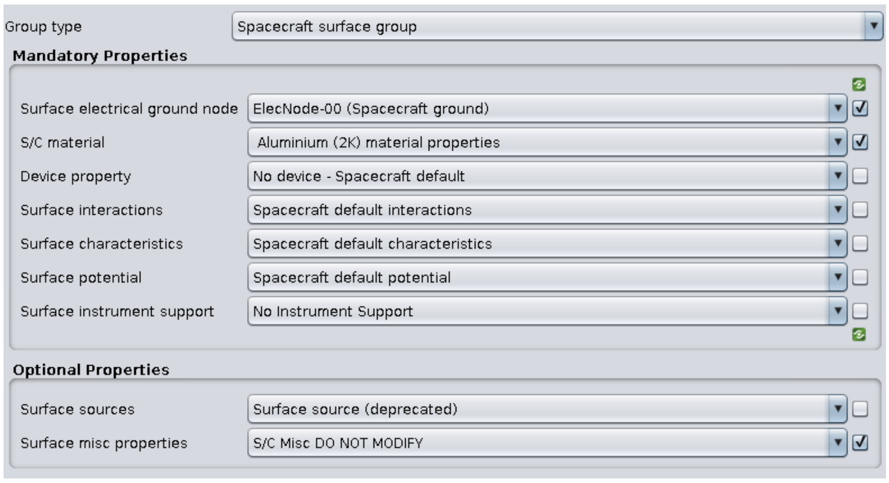
Predefined electric field boundaries are:

- Fourier – Robin conditions (Default): the most general condition on the potential and the electric field $V-\alpha E=\beta$, with $\alpha=1m^{-1}$ and $\beta=0V$ by default.
- Dirichlet condition: the potential is fixed, to 0V by default.
- Fixed electric field: the electric field is fixed to a given value, 0V/m by default.

3.4.3. *The spacecraft surface characteristics*

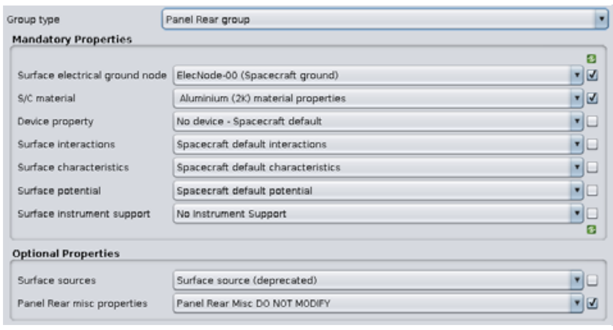
While functionally equivalent to the previous features in SPIS, the definition of the unmeshed surfaces has been simplified in the latest version of SPIS. Note that older definitions are still working (thus ensuring compatibility) but cannot be set under the same layout with the new interface. All the fields that were defined in the older version of SPIS are still present, but the fields which values are not to be modified (because it leads to a nonsensical setup) are now grouped in the misc. category. This simplifies the definition of the groups by highlighting the fields which definitions are of interest for the user. Moreover, it is now possible to define an arbitrary number of any thin elements.

- Regular surface: to define a regular spacecraft surface, assign the "Spacecraft Surface" group type to a surface.
-

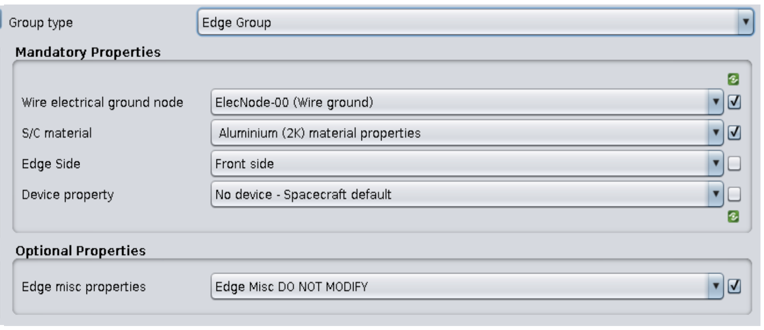


Group type	Spacecraft surface group	
Mandatory Properties		
Surface electrical ground node	ElecNode-00 (Spacecraft ground)	✓
S/C material	Aluminium (2K) material properties	✓
Device property	No device - Spacecraft default	☐
Surface interactions	Spacecraft default interactions	☐
Surface characteristics	Spacecraft default characteristics	☐
Surface potential	Spacecraft default potential	☐
Surface instrument support	No Instrument Support	☐
Optional Properties		
Surface sources	Surface source (deprecated)	☐
Surface misc properties	S/C Misc DO NOT MODIFY	✓

- Panels: to define thin panels, it is needed to define both sides of the panel as independent surfaces, each one facing a volume. Front side must be set as a "Panel Front Group" and rear side as a "Panel Rear Group". Moreover, the panel edges should be defined as "Edge" group, with the "Side" property set to "Front" or "Rear".

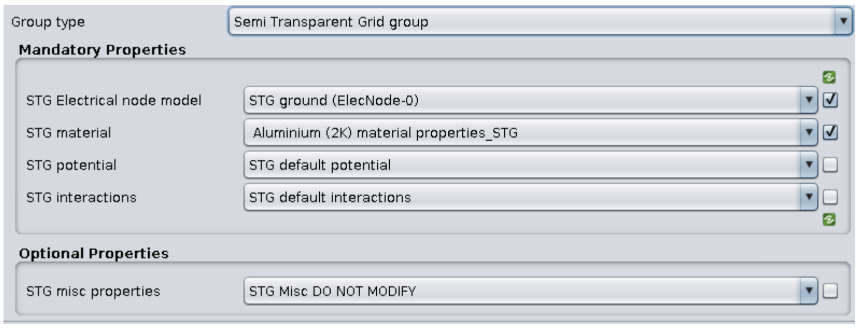


Group type	Panel Rear group	
Mandatory Properties		
Surface electrical ground node	ElecNode-00 (Spacecraft ground)	✓
S/C material	Aluminium (2K) material properties	✓
Device property	No device - Spacecraft default	☐
Surface interactions	Spacecraft default interactions	☐
Surface characteristics	Spacecraft default characteristics	☐
Surface potential	Spacecraft default potential	☐
Surface instrument support	No Instrument Support	☐
Optional Properties		
Surface sources	Surface source (deprecated)	☐
Panel Rear misc properties	Panel Rear Misc DO NOT MODIFY	✓



Group type	Edge Group	
Mandatory Properties		
Wire electrical ground node	ElecNode-00 (Wire ground)	✓
S/C material	Aluminium (2K) material properties	✓
Edge Side	Front side	☐
Device property	No device - Spacecraft default	☐
Optional Properties		
Edge misc properties	Edge Misc DO NOT MODIFY	✓

- Grids: to define a grid, assign the “Semi Transparent Grid” group type to a surface. The surface must be defined between two volumes.
-



Group type: Semi Transparent Grid group

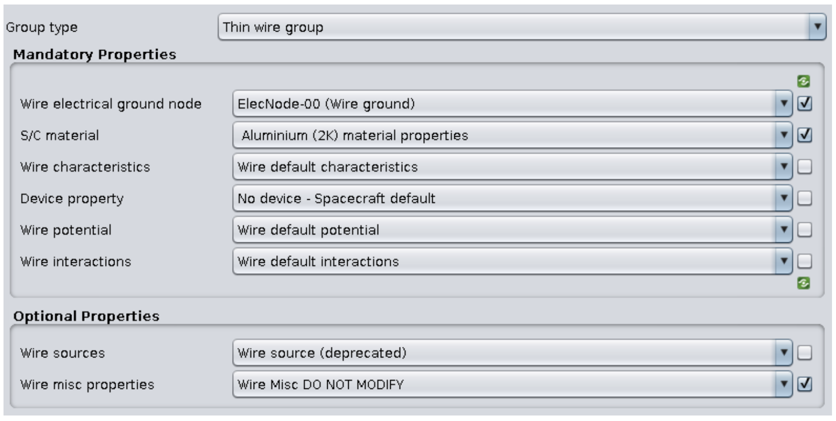
Mandatory Properties

STG Electrical node model	STG ground (ElecNode-0)	✓
STG material	Aluminium (2K) material properties_STG	✓
STG potential	STG default potential	☐
STG interactions	STG default interactions	☐

Optional Properties

STG misc properties	STG Misc DO NOT MODIFY	☐

- Wires and antennas: to define wires and antennas, assign the “Wire” group type to a 1D element.
-



Group type: Thin wire group

Mandatory Properties

Wire electrical ground node	ElecNode-00 (Wire ground)	✓
S/C material	Aluminium (2K) material properties	✓
Wire characteristics	Wire default characteristics	☐
Device property	No device - Spacecraft default	☐
Wire potential	Wire default potential	☐
Wire interactions	Wire default interactions	☐

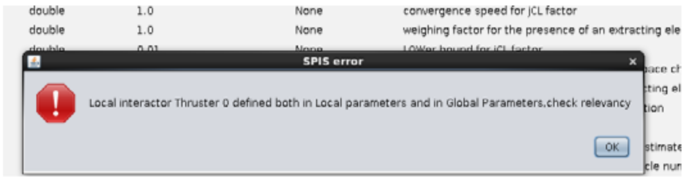
Optional Properties

Wire sources	Wire source (deprecated)	☐
Wire misc properties	Wire Misc DO NOT MODIFY	✓

Most of the surface characteristics were not modified in the version of SPIS Nonetheless, a novelty appears in the definition of the surface interactions.

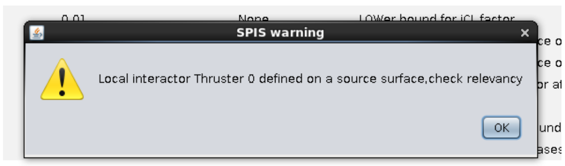
Surface interactions are of two types. The definition of the global interactions that applies to all spacecraft surfaces and that are entirely defined in the global parameter window remains unchanged. Local interactions that are related to a particular surface group are usually defined in the global parameter window, while their domain of application is defined by a value of the *SourceId* in the Local parameter corresponding to 1000 plus the interactor number.

In the electric propulsion version of SPIS, these local interactors now can be directly affected to the surface in the group editor as a device. Only one device can be defined per surface group. It still remains possible to define supplementary interactors using the previous method describe above. In this later case, a message will appear in a popup window with a warning message if the interactors are of the same type. This message has been added to avoid confusion and dual definition of the same interactor, nonetheless this message is only informative and is not blocking. A message also appears if a local interactor is defined on a surface to which a source is associated. If SPIS runs in batch mode, these messages only appear in the log file.


double 1.0 None convergence speed for jCL factor
double 1.0 None weighing factor for the presence of an extracting ele

SPIS error

Local interactor Thruster 0 defined both in Local parameters and in Global Parameters.check relevancy


SPIS warning

Local interactor Thruster 0 defined on a source surface,check relevancy

Figure 7: Warning messages appearing when two interactors of the same type are defined in the local and global parameters (top) and when a local interactor is defined on a surface associated to a source (bottom).

The spacecraft surface definition categories have been reworked to simplify the setup. Fields that are not to be modified by the user are gathered in a “miscellaneous properties” category. Thus, only the properties of interest for the user appear (in a much more limited number) in the category of interest. As a general rule, the categories which are frequently modified appear by decreasing order in the “Mandatory properties” panel, whereas those which are rarely modified (or not to be modified) appear in the “Optional properties” panel. The spacecraft characteristic categories of interest for the user allow defining:

- The surface electrical node: this electric node corresponds to the metallic surface underneath the coating. For conductive coatings, this electric node also corresponds to the material surface, but for dielectrics an equivalent RC circuit is set between this electrical node and the surface elements.

- The surface coating material: this material can be defined by selecting a material from the material databank. These materials correspond to that coating the spacecraft surface. An exception is for the materials to assign to the exhaust of droplet emitting thrusters. In this case the material to assign to the exhaust is that corresponding to the droplets.

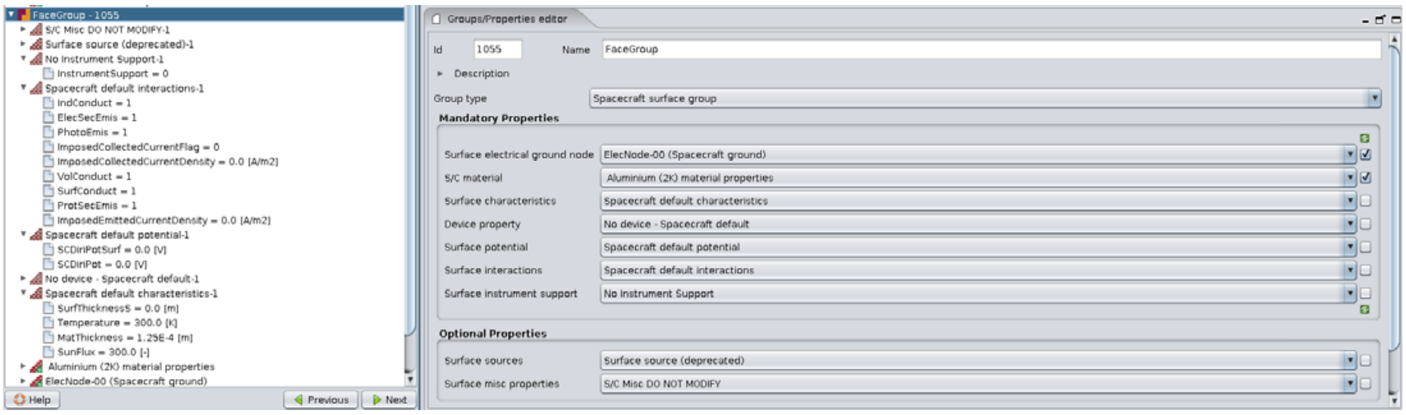
- The macroscopic characteristics: depending on the type of element (regular surface, grid, wire or panel), this allows to define the characteristic thickness of the elements. For surfaces, it allows to define the structure thickness (*SurfThicknessS or EdgeRadiusS*) and for group allowing it the coating thickness (*MatThickness*). Moreover, this also permits -to define the coating material temperature and exposure to the sun flux (used if set so in the global parameter), which is of importance for contamination studies.

- The device characteristics: this allows defining the behaviour of the surface by assigning to it an “interactor” (i.e. a physical model of the interaction of this surface with the plasma). This is how to define a thruster. SPIS provides a databank of predefined thruster models, amongst which the SPT100 Hall thruster.

- The potential and currents across the surface: used to fix the initial potential and currents at the surface.

- Global interaction characteristics: used to turn on/off the global interactions, i.e. the photo- and secondary emissions, and to force the currents on surfaces. The source property is deprecated and should not be modified by the user. Warning messages appear if a source or and interactor is associated to the surface in addition to a device.

- Miscellaneous properties are specific to the type of surface element and should not be modified by the user.



```
▼ FaceGroup - 1055
  ▸ S/C Misc DO NOT MODIFY-1
  ▸ Surface source (deprecated)-1
  ▾ No Instrument Support-1
      InstrumentSupport = 0
  ▾ Spacecraft default interactions-1
      IndConduct = 1
      ElecSecEmis = 1
      PhotoEmis = 1
      ImposedCollectedCurrentFlag = 0
      ImposedCollectedCurrentDensity = 0.0 [A/m2]
      VolConduct = 1
      SurfConduct = 1
      ProtSecEmis = 1
      ImposedEmittedCurrentDensity = 0.0 [A/m2]
  ▾ Spacecraft default potential-1
      SCDiriPotSurf = 0.0 [V]
      SCDiriPot = 0.0 [V]
  ▸ No device - Spacecraft default-1
  ▾ Spacecraft default characteristics-1
      SurfThicknessS = 0.0 [m]
      Temperature = 300.0 [K]
      MatThickness = 1.25E-4 [m]
      SunFlux = 300.0 [-]
  ▸ Aluminium (2K) material properties
  ▸ ElecNode-00 (Spacecraft ground)
```

Groups/Properties editor

Id: 1055 Name: FaceGroup

Description

Group type: Spacecraft surface group

Mandatory Properties

Surface electrical ground node	ElecNode-00 (Spacecraft ground)	✓
S/C material	Aluminium (2K) material properties	✓
Surface characteristics	Spacecraft default characteristics	☐
Device property	No device - Spacecraft default	☐
Surface potential	Spacecraft default potential	☐
Surface interactions	Spacecraft default interactions	☐
Surface instrument support	No Instrument Support	☐

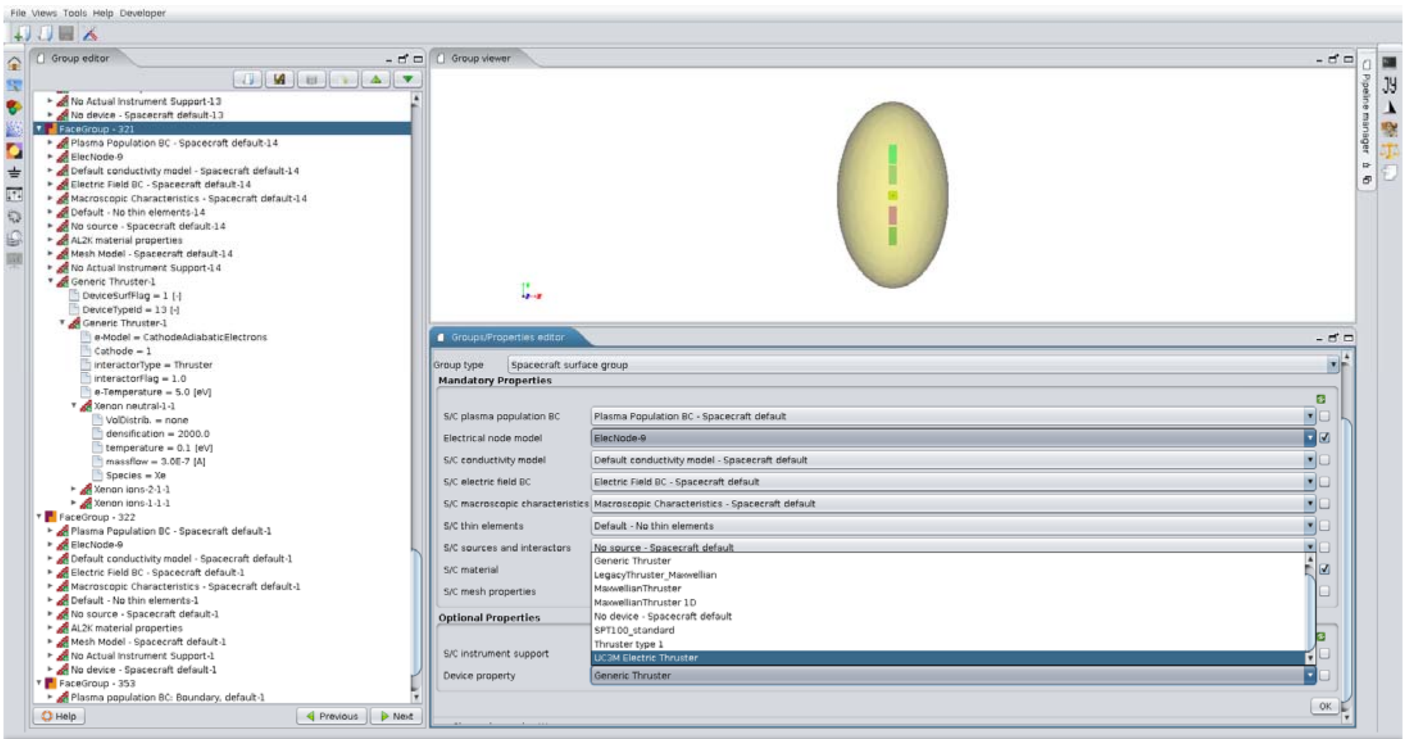
Optional Properties

Surface sources	Surface source (deprecated)	☐
Surface misc properties	S/C Misc DO NOT MODIFY	☐

Figure 8: New layout for the definition of a regular spacecraft surface. The number of categories and of characteristics per categories has been reduced to those of interest for the user. Other characteristics are gathered in the misc. properties category.

3.4.4. Device characteristics

The devices are created by defining the optional device property of a spacecraft surface. Some local interactors have predefined templates. It is noticeably the case for the Thruster and Solar Panel interactors. Once selected, the device property appears in the group editor tree in a manner similar to the surface materials.



```
▸ No Actual Instrument Support-13
▸ No device - Spacecraft default-13
▾ FaceGroup - 321
  ▸ Plasma Population BC - Spacecraft default-14
  ▸ ElecNode-9
  ▸ Default conductivity model - Spacecraft default-14
  ▸ Electric Field BC - Spacecraft default-14
  ▸ Macroscopic Characteristics - Spacecraft default-14
  ▸ Default - No thin elements-14
  ▸ No source - Spacecraft default-14
  ▸ AL2K material properties
  ▸ Mesh Model - Spacecraft default-14
  ▸ No Actual Instrument Support-14
  ▾ Generic Thruster-1
      DeviceSurfFlag = 1 [-]
      DeviceTypeId = 13 [-]
    ▾ Generic Thruster-1
        e-Model = CathodeAdiabaticElectrons
        Cathode = 1
        interactorType = Thruster
        interactorFlag = 1.0
        e-Temperature = 5.0 [eV]
      ▾ Xenon neutral-1-1
          VolDistrib. = none
          densification = 2000.0
          temperature = 0.1 [eV]
          massflow = 3.0E-7 [A]
          Species = Xe
      ▸ Xenon ions-2-1-1
      ▸ Xenon ions-1-1-1
▾ FaceGroup - 322
  ▸ Plasma Population BC - Spacecraft default-1
  ▸ ElecNode-9
  ▸ Default conductivity model - Spacecraft default-1
  ▸ Electric Field BC - Spacecraft default-1
  ▸ Macroscopic Characteristics - Spacecraft default-1
  ▸ Default - No thin elements-1
  ▸ No source - Spacecraft default-1
  ▸ AL2K material properties
  ▸ Mesh Model - Spacecraft default-1
  ▸ No Actual Instrument Support-1
  ▸ No device - Spacecraft default-1
▾ FaceGroup - 353
  ▸ Plasma population BC: Boundary, default-1
```

Group type: Spacecraft surface group

Mandatory Properties

S/C plasma population BC	Plasma Population BC - Spacecraft default	☐
Electrical node model	ElecNode-9	☑
S/C conductivity model	Default conductivity model - Spacecraft default	☐
S/C electric field BC	Electric Field BC - Spacecraft default	☐
S/C macroscopic characteristics	Macroscopic Characteristics - Spacecraft default	☐
S/C thin elements	Default - No thin elements	☐
S/C sources and interactors	No source - Spacecraft default	☐
S/C material		☑
S/C mesh properties		☐

Optional Properties

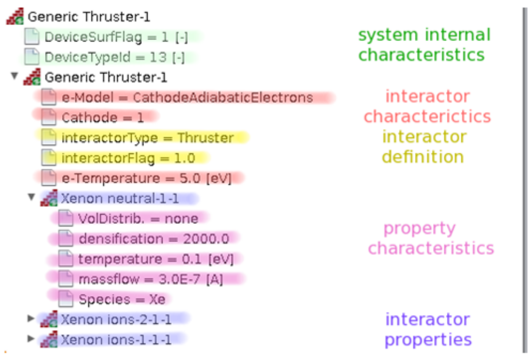
S/C instrument support		☐
Device property	Generic Thruster	☐

```
Generic Thruster
LegacyThruster_Maxwellian
MaxwellianThruster
MaxwellianThruster 1D
No device - Spacecraft default
SPT100_standard
Thruster type 1
UC3M Electric Thruster
```

Figure 9: Screen capture of the group editor while selecting a predefined device property (Local interactor). A device was already attributed to this surface and its characteristics appear in the group editor tree on the left panel.

The device characteristics are complex and appear as a sub-tree. An example of such device characteristics is displayed on Figure 10. It is composed of several sub-characteristics. First of all, internal system characteristics appear at the top level in the same manner that for the material characteristics, they serve as references for the numerical core and should not be modified by the user. All user defined characteristics are gathered in the top level property named the same way than the device property.

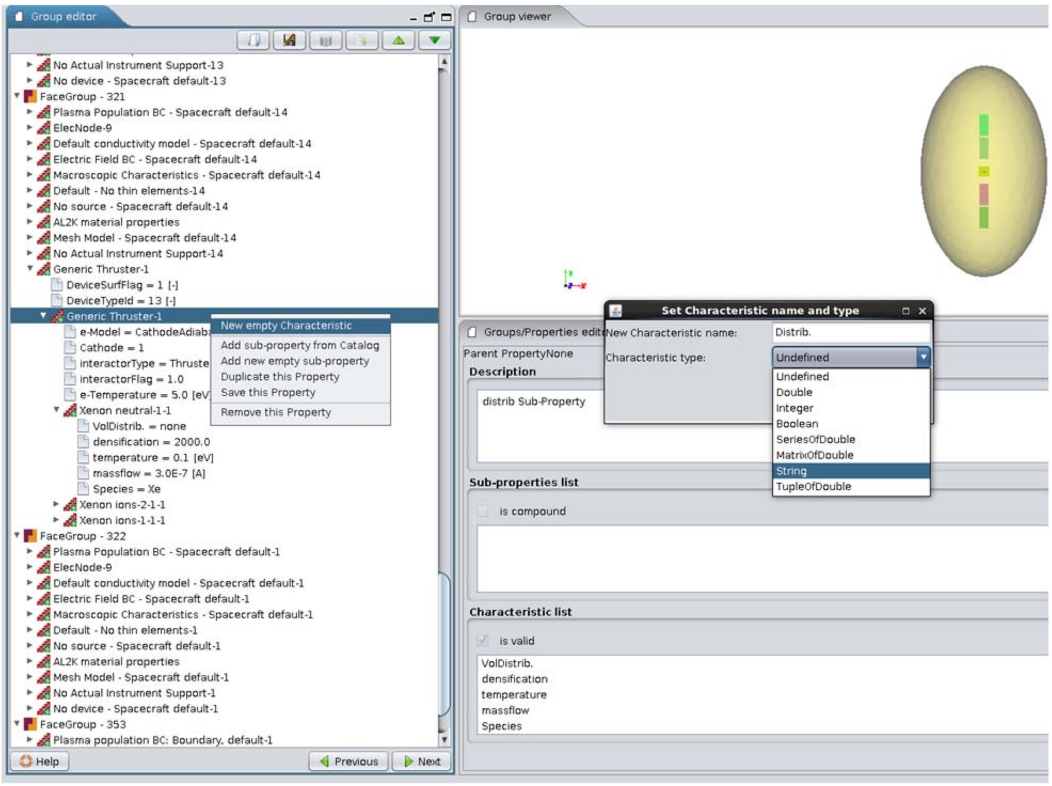
It is composed of characteristics (meaning a name-value-unit triplet) and properties (meaning a sub-tree element). Two characteristics are mandatory and independent of the device as they are use to define the interaction: the *interactorType* is a String naming the interactor type to use (here a Thruster) and the *interactorFlag* is an integer set to one to instantiate the interaction and to zero to ignore it.



```
Generic Thruster-1
  DeviceSurfFlag = 1 [-]
  DeviceTypeId = 13 [-]
  Generic Thruster-1
    e-Model = CathodeAdiabaticElectrons
    Cathode = 1
    interactorType = Thruster
    interactorFlag = 1.0
    e-Temperature = 5.0 [eV]
    Xenon neutral-1-1
      VolDistrib. = none
      densification = 2000.0
      temperature = 0.1 [eV]
      massflow = 3.0E-7 [A]
      Species = Xe
    Xenon ions-2-1-1
    Xenon ions-1-1-1
```

Figure 10: details of a device (here a thruster) characteristics

Other characteristics appear at the same level and serve to set up the interaction (in our example, they define the cathode electron characteristics). Templates may not define all of the possible characteristics. To define a new characteristics, right-click on the property to which you want to add the characteristic and select "*New empty characteristic*". A popup widow appears in which you can define the characteristic name and type (Figure 11).



```
No Actual Instrument Support-13
No device - Spacecraft default-13
FaceGroup - 321
  Plasma Population BC - Spacecraft default-14
  ElecNode-9
  Default conductivity model - Spacecraft default-14
  Electric Field BC - Spacecraft default-14
  Macroscopic Characteristics - Spacecraft default-14
  Default - No thin elements-14
  No source - Spacecraft default-14
  AL2K material properties
  Mesh Model - Spacecraft default-14
  No Actual Instrument Support-14
  Generic Thruster-1
    DeviceSurfFlag = 1 [-]
    DeviceTypeId = 13 [-]
    Generic Thruster-1
      e-Model = CathodeAdiaba
      Cathode = 1
      interactorType = Thruste
      interactorFlag = 1.0
      e-Temperature = 5.0 [eV
      Xenon neutral-1-1
        VolDistrib. = none
        densification = 2000.0
        temperature = 0.1 [eV]
        massflow = 3.0E-7 [A]
        Species = Xe
      Xenon ions-2-1-1
      Xenon ions-1-1-1
FaceGroup - 322
  Plasma Population BC - Spacecraft default-1
  ElecNode-9
  Default conductivity model - Spacecraft default-1
  Electric Field BC - Spacecraft default-1
  Macroscopic Characteristics - Spacecraft default-1
  Default - No thin elements-1
  No source - Spacecraft default-1
  AL2K material properties
  Mesh Model - Spacecraft default-1
  No Actual Instrument Support-1
  No device - Spacecraft default-1
FaceGroup - 353
  Plasma population BC: Boundary, default-1
```

New Characteristic name: Distrib.
Characteristic type: Undefined

Undefined
Double
Integer
Boolean
SeriesOfDouble
MatrixOfDouble
String
TupleOfDouble

Parent PropertyNone

Description

distrib Sub-Property

Sub-properties list

is compound

Characteristic list

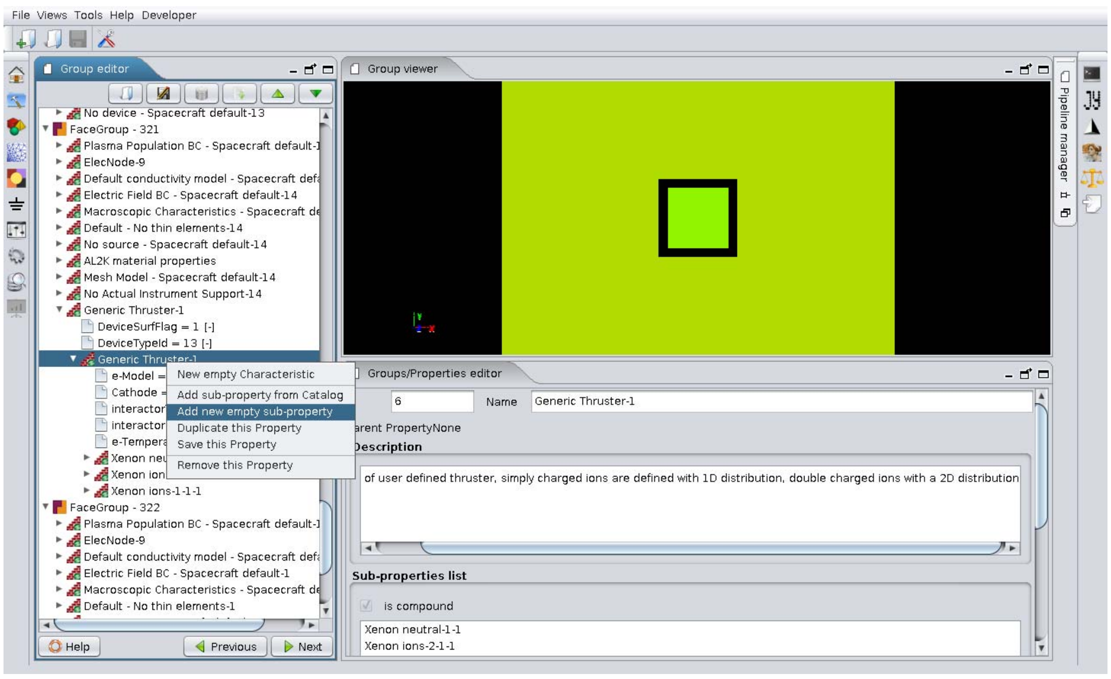
✓ is valid

VolDistrib.
densification
temperature
massflow
Species

Figure 11: Addition of a new characteristic to the "Generic Thruster-1" device property.

It is also possible to define at the same level sub-properties (sub-trees), themselves containing characteristics. Their (optional) use depends on the device. As examples, they are used to define thruster populations for thruster devices and define solar panels elements for high voltage solar panels.

Templates of these devices usually provide predefined properties. However, it may be needed to add new ones (to add populations for thrusters or sub panel for solar panels, for example). To possibilities are offered to add new properties. The most general way is to create a new empty property by right-clicking on the device property and selecting "*add new empty sub-property*" (Figure 12). There are no restrictions or recommendations on the naming of the sub-properties, as this name is not accessible to the numerical kernel. Then, it is possible to fill the sub-property with new characteristics as discussed previously.


File Views Tools Help Developer
Group editor
Group viewer
Pipeline manager
No device - Spacecraft default-13
FaceGroup - 321
Plasma Population BC - Spacecraft default-1
ElecNode-9
Default conductivity model - Spacecraft defa
Electric Field BC - Spacecraft default-14
Macroscopic Characteristics - Spacecraft de
Default - No thin elements-14
No source - Spacecraft default-14
AL2K material properties
Mesh Model - Spacecraft default-14
No Actual Instrument Support-14
Generic Thruster-1
DeviceSurfFlag = 1 [-]
DeviceTypeId = 13 [-]
Generic Thruster-1
e-Model =
Cathode =
interactor
interactor
e-Tempera
Xenon neu
Xenon ion
Xenon ions-1-1-1
FaceGroup - 322
Plasma Population BC - Spacecraft default-1
ElecNode-9
Default conductivity model - Spacecraft defa
Electric Field BC - Spacecraft default-1
Macroscopic Characteristics - Spacecraft de
Default - No thin elements-1
New empty Characteristic
Add sub-property from Catalog
Add new empty sub-property
Duplicate this Property
Save this Property
Remove this Property
Help
Previous
Next
Groups/Properties editor

6 Name Generic Thruster-1

arent PropertyNone

Description

of user defined thruster, simply charged ions are defined with 1D distribution, double charged ions with a 2D distribution

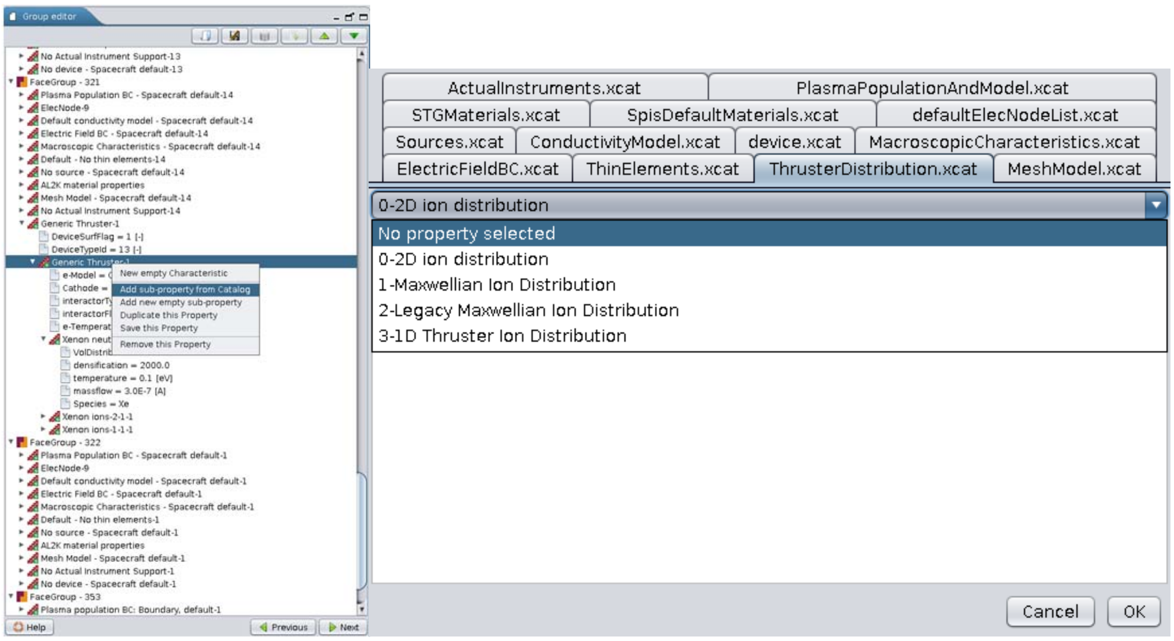
Sub-properties list

is compound

Xenon neutral-1-1
Xenon ions-2-1-1

Figure 12: Addition of a new empty property to the "Generic Thruster-1" device property.

The other possibility is to insert a predefined property. A few templates of population descriptions have been defined for thruster populations. To use these template, right-click on the device property and select "*add sub property from catalog*". A popup window appears that provides the list of existing templates. The thruster distribution templates can be found in the "*ThrusterDistribution*" tab (Figure 13).



```
No Actual Instrument Support-13
No device - Spacecraft default-13
FaceGroup - 321
  Plasma Population BC - Spacecraft default-14
  ElecNode-9
  Default conductivity model - Spacecraft default-14
  Electric Field BC - Spacecraft default-14
  Macroscopic Characteristics - Spacecraft default-14
  Default - No thin elements-14
  No source - Spacecraft default-14
  AL2K material properties
  Mesh Model - Spacecraft default-14
  No Actual Instrument Support-14
  Generic Thruster-1
    DeviceSurfFlag = 1 [-]
    DeviceTypeId = 13 [-]
    Generic Thruster-1
      e-Model =
      Cathode =
      interactorTy
      interactorFl
      e-Temperat
      Xenon neut
        VolDistrib
        densification = 2000.0
        temperature = 0.1 [eV]
        massflow = 3.0E-7 [A]
        Species = Xe
      Xenon ions-2-1-1
      Xenon ions-1-1-1
FaceGroup - 322
  Plasma Population BC - Spacecraft default-1
  ElecNode-9
  Default conductivity model - Spacecraft default-1
  Electric Field BC - Spacecraft default-1
  Macroscopic Characteristics - Spacecraft default-1
  Default - No thin elements-1
  No source - Spacecraft default-1
  AL2K material properties
  Mesh Model - Spacecraft default-1
  No Actual Instrument Support-1
  No device - Spacecraft default-1
FaceGroup - 353
  Plasma population BC: Boundary, default-1
```

Figure 13: Addition of a new property to the "Generic Thruster-1" device property using templates.

3.5. Thruster settings

3.5.1. General considerations

Thrusters can be defined in SPIS electric propulsion by setting the Device property of a group in the group editor. The *InteractorType* characteristic must be set to "*Thruster*" and the *InteractorFlag* characteristic set to 1. Other characteristics may be added at this level to determine the cathode associated to the thruster and the cathode electron properties. The thruster definition itself is done through the definition of population emitted by the thruster. For each population (or group or population) and new sub property must be defined (right click on the thruster property and select "add new characteristic").

Each population must contain a species name (specified by the "*Species*" string characteristic), which corresponds to the chemical formula of the species. A special case concerning droplets is detailed at the end of this chapter. For Maxwellian population distributions or legacy distributions (see hereafter) it is possible to define several populations at the same time is these populations are different levels of ionization of the same neutral species. In this case, the *ion1Fraction* and *ion2Fraction* characteristics must be defined. An example of multiple populations is given in the "*LegacyMaxwellianThruster*" template.

For each population, the thruster source model is defined by the "*Distrib.*" property. This property default value is "*ThrusterSurfDistrib*" which corresponds to the generic thruster source model.

It is also possible to specify how each thruster population is model in volume by specifying the "*VolDistrib*." characteristic. This characteristic default value is "*PICVolDistrib*" which corresponds to population modelled by numerical particles.

3.5.2. Thruster electrical properties

The capacitance of the thruster is automatically recomputed is there are other surfaces than that of the thruster. Because the thruster emits very dense plasma compared to the natural plasma in which the spacecraft evolves, the sheath size above the thruster is much smaller than around the other parts of the spacecraft. Thus, the thruster capacitance is multiplied by the environment Debye length divided by the thruster plasma Debye length.

The currents emitted and collected by the thruster are automatically set to 0. A net thruster current is computed by the thruster interactor and used for the spacecraft circuit calculations. This avoids in particular having unphysical secondary electrons currents emitted from the thruster.

Because of the particular treatment imposed to the surfaces corresponding to the thruster, it is recommended to attribute to each thruster an independent electrical node.

3.5.3. Thruster axis computation

In the group editor, for a spacecraft group, the device property is used to indicate devices types such as thrusters. If at least one group has a thruster type, an automatic computation of the thruster axis is performed. It is necessary for several reasons:

- It is used to define the thruster position and orientation in SPIS-NUM, which are used in particular to define the sources of thruster species.

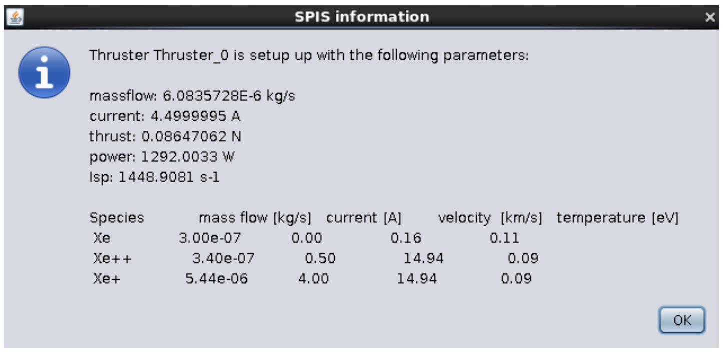
- The thruster axis is used for a tailored data mining operations: the probing of data along the thruster axis. This option is presented in section 6.2.

The thruster axis is automatically computed for each spacecraft surface group type with a device property defined in the group editor. Several thruster axes may be associated with one spacecraft group type with a device property if there are several non-continuous surfaces. More information on the thruster axis computation can be found in section 4.6.

3.5.4. Thruster parameter verifications

After validation of the global simulation parameters, before the start of the simulation, the thruster parameters are computed from the numerical distribution associated to the thruster. The thruster thrust, mass

flow, current power and specific impulse are computed and some moments of each population distribution are displayed.
The information appearing in this popup window is not a reminder of the user settings, it is actually computed by running the thruster population emission and computing the moment of the emitted particles.



```
Thruster Thruster_0 is setup up with the following parameters:

massflow: 6.0835728E-6 kg/s
current: 4.4999995 A
thrust: 0.08647062 N
power: 1292.0033 W
Isp: 1448.9081 s-1

Species         mass flow [kg/s]   current [A]       velocity  [km/s]   temperature [eV]
 Xe             3.00e-07         0.00             0.16             0.11
 Xe++             3.40e-07         0.50             14.94             0.09
 Xe+             5.44e-06         4.00             14.94             0.09
```

Figure 14: panel that list the thruster parameters appearing at the end of the simulation definition phase. For batch simulation, this information only appears in the log file.

3.5.5. Legacy thruster models

3.5.5.1. Maxwellian Thruster

To use the legacy Maxwellian thruster model, the "*Distrib.*" Property must be set to "*MaxwellianThruster*". The *current*, *temperature* and *mach* number properties must be defined or it should be possible to compute them from the specified scalar properties specified (amongst *density*, *massflow* and *velocity*).

This legacy thruster model uses an old particle sampler which badly samples the drifting Maxwellian distributions and underestimates the population kinetic energy by a factor of two. As this function is used for the fitted AISEPS distributions, it cannot be changed without new fits of the AISEPS thrusters.

3.5.5.2. Axisymetric model

To use the legacy Axisymetric thruster model, the "Distrib." Property must be set to "*AxisymTabulatedVelocitySurfDistrib*". The *current* properties must be defined or it should be possible to compute it from the specified scalar properties specified (amongst *density*, *massflow*, *temperature*, *mach* number, *velocity*). The tabulated distribution shall be in a file named "*SourceXAxisymData.txt*", where X is the number of the thruster.

3.5.5.3. AISEPS Thruster

To use the legacy AISEPS thruster model, the "*Distrib.*" Property must be set to "*AisepsThrusterSurfDistrib*". Moreover, the AISEPS plugin must be installed. The *current* properties must be defined or it should be possible to compute it from the specified scalar properties specified (amongst *density*, *massflow*, *temperature*, *mach* number, *velocity*). The thruster parameters must be gathered in an AISEPS file whose name is specified by the "*SourceFile*" property of the thruster.

3.5.6. Generic thruster model

The generic thruster model that is provided with the electric propulsion package provides the user with more flexibility for the definition of the thruster population distribution. The generic thruster model is selected by default, but it can be explicitly defined by setting the "*Distrib.*" property to "*ThrusterSurfDistrib*".

3.5.6.1. How to define distribution

User can define the distribution and the thruster geometry in many ways. The definition used by SPIS is determined by a sequential order presented hereafter. For each parameter to determine, SPIS uses the first method for which all the mandatory parameter are provided in the distribution characteristics (example: SPIS will define the energy distribution from a 1D tabulated distribution if the *f(energy)* parameter is defined). Details and examples of 1D and 2D distribution settings are given in sections 3.5.6.3 and 3.5.6.4.

SPIS determine four main characteristics of the thruster, in this order:

(1) The thruster geometry definition

The thruster geometry can simply correspond to the geometry of the group on which it is defined. This is the default case, but it is also possible to specify the geometry of the thruster in the population distribution definition in which case the thruster exhaust is not necessarily meshed. The radial dependency of the distribution is only possible if the geometry is specified. The geometry is selected by the "*geometry*" property of the population distribution and can be either "*annular*", "*circular*" or "*rectangular*". For the annular geometry, it is needed to specify the "*radius*" and "*innerradius*" properties. For the circular geometry, only the "*radius*" property is needed. For the rectangular geometry, the "*sizex*" and "*sizey*" properties must be defined.

(2) The energy distribution

a. 1D distribution: SPIS will define the energy distribution from a 1D tabulated distribution if the *f(energy)* parameter is defined as two *seriesOfDouble*, the first one giving the energy in eV, the second the distribution. An example of such a distribution is shown in section 3.5.6.3.

b. 2D energy-radius distribution: SPIS will define the energy distribution from a 2D tabulated energy versus radius distribution if the *f(energy,radius)* parameter is defined as a *matrixOfDouble*, with the abscissa (first column) being the energy in eV and the ordinate (first row) the radius in meters. The rest of the components stand for the distribution. An example of such a distribution is shown in section 3.5.6.4. In this case, the radial distribution is defined at the same time than the energy distribution.

c. 2D energy-angle distribution: SPIS will define the energy distribution from a 2D tabulated energy versus radius distribution if the *f(energy,angle)* parameter is defined as a *matrixOfDouble*, with the abscissa (first column) being the energy in eV and the ordinate (first row) the angle in degrees. The rest of the components stand for the distribution. An example of such a distribution is shown in section 3.5.6.4. In this case, the angular distribution is defined at the same time than the energy distribution.

d. Maxwellian distribution: SPIS defines the energy distribution from the temperature and the Mach number.

(3) The radial distribution
If the radial distribution was not defined at the same time than the energy, then:

a. 2D radius-angle distribution: SPIS will define the energy distribution from a 2D tabulated energy versus radius distribution if the *f(radius,angle)* parameter is defined as a *matrixOfDouble*, with the abscissa (first column) being the radius in meters and the ordinate (first row) the angle in degrees. The rest of the components stand for the distribution. If the angle distribution was defined at the same time than the energy, this does not change the angular distribution. Then, it only gives the radius distribution for a given angle; otherwise the radial distribution is defined at the same time than the energy distribution.

b. 1D distribution: SPIS will define the radius distribution from a 1D tabulated distribution if the *f(radius)* parameter is defined as two *seriesOfDouble*, the first one giving the energy in eV, the second the distribution.

c. Predefined functions: if the *f(radius)* parameter is defined as a String or not defined, then the radial distribution is either "*uniform*" (default), "*cosine*" or "*cosine2*" (cosine squared).

To be defined, a radial distribution needs that the thruster exhaust geometry be explicitly defined.

(4) The angular distribution
If the angular distribution was not defined at the same time than the energy or the radius, then:

a. 2D energy-angle distribution: SPIS will define the energy distribution from a 2D tabulated energy versus radius distribution if the *f(energy,angle)* parameter is defined as a *matrixOfDouble*, with the abscissa (first column) being the energy in eV and the ordinate (first row) the angle in degrees. The rest of the components stand for the distribution. As the energy distribution was previously determined in (1), it only gives the angle distribution for a given energy;

b. 2D radius-angle distribution: SPIS will define the energy distribution from a 2D tabulated energy versus radius distribution if the *f(radius,angle)* parameter is defined as a *matrixOfDouble*, with the abscissa (first column) being the radius in meters and the ordinate (first row) the angle in degrees. The rest of the components stand for the distribution. If the angle distribution was defined at the same time than the energy, this does not change the angular distribution. As the radius distribution was previously determined in (1), it only gives the angle distribution for a given radius;

c. 1D distribution: SPIS will define the angular distribution from a 1D tabulated distribution if the *f(angle)* parameter is defined as two *seriesOfDouble*, the first one giving the angle in degrees, the second the distribution.

d. Divergence: SPIS will define the angular distribution as a cosine function between two angles defined by the "divergence" and "*innerdivergence*" parameters. The emission axis is the difference of these angles. "*innerdivergence*" fixes the maximum angle in the direction pointing toward the thruster axis, while the "divergence" fixes the maximum angle in the outward direction. If "*innerdivergence*" is not defined, it has the same value than "*divergence*". If "*divergence*" is not defined it default value is 90°.

3.5.6.2. Maxwellian thruster

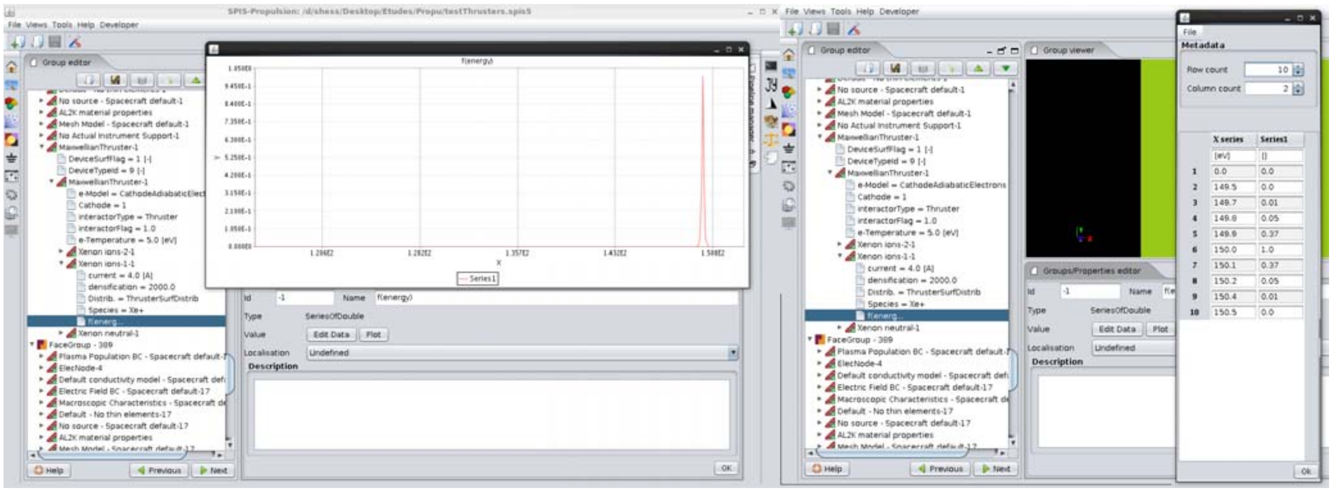
To define a Maxwellian thruster, one should proceed in the exact same way than for the legacy Maxwellian thruster, except for that the "*Distrib*." Property should be "*ThrusterSurfDistrib*" (or be absent). The *current*, *temperature* and *mach* number properties must be defined or it should be possible to compute them from the specified scalar properties specified (amongst *density*, *massflow* and *velocity*).

3.5.6.3. Setting up 1D energy, radial and angular distributions

To set up 1D energy, radial or angular distributions, you must specify in the species distribution parameters the name of the distribution you intend to define by right clicking on the distribution name and selecting "*add a new property*". This property name must be *f(energy)* for the energy distribution, *f(radius)* for the radial distribution and *f(angle)* for the angular distribution. In any cases, the type of the property must be *seriesOfDoubles*. Then edit the distribution value by clicking on the "Edit Data" button in the lower left window. A new window appears with a two column table. The first column stands for the energy, radius or angle values, and the second for the distribution values. The first row is used to define the units. Energies must be in [eV] while radii are in [m].

Once the distribution is validated, it is possible to display it by clicking on the "Plot" button.

An example of a 1D energy distribution is provided in the "*MaxwellThruster1D*" template provided with the SPIS electric propulsion distribution. To use it, select "*MaxwellThruster1D*" as the Device Property of the group that stands for the thruster. The singly charged Xenon ions are defined as a shifted Maxwellian distribution similar to that defined in section 3.5.6.2, but they are defined by a 1D energy distribution and a 1° angular divergence.



MaxwellianThruster-1
DeviceSurfFlag = 1 [-]
DeviceTypeId = 9 [-]
MaxwellianThruster-1
e-Model = CathodeAdiabaticElectrons
Cathode = 1
interactorType = Thruster
interactorFlag = 1.0
e-Temperature = 5.0 [eV]
Xenon ions-2-1
Xenon ions-1-1
current = 4.0 [A]
densification = 2000.0
Distrib. = ThrusterSurfDistrib
Species = Xe+
f(energ...
Xenon neutral-1

Id: -1 Name: f(energy)
Type: SeriesOfDouble
Localisation: Undefined

Metadata
Row count: 10
Column count: 2

	X series	Series1
	[eV]	[]
1	0.0	0.0
2	149.5	0.0
3	149.7	0.01
4	149.8	0.05
5	149.9	0.37
6	150.0	1.0
7	150.1	0.37
8	150.2	0.05
9	150.4	0.01
10	150.5	0.0

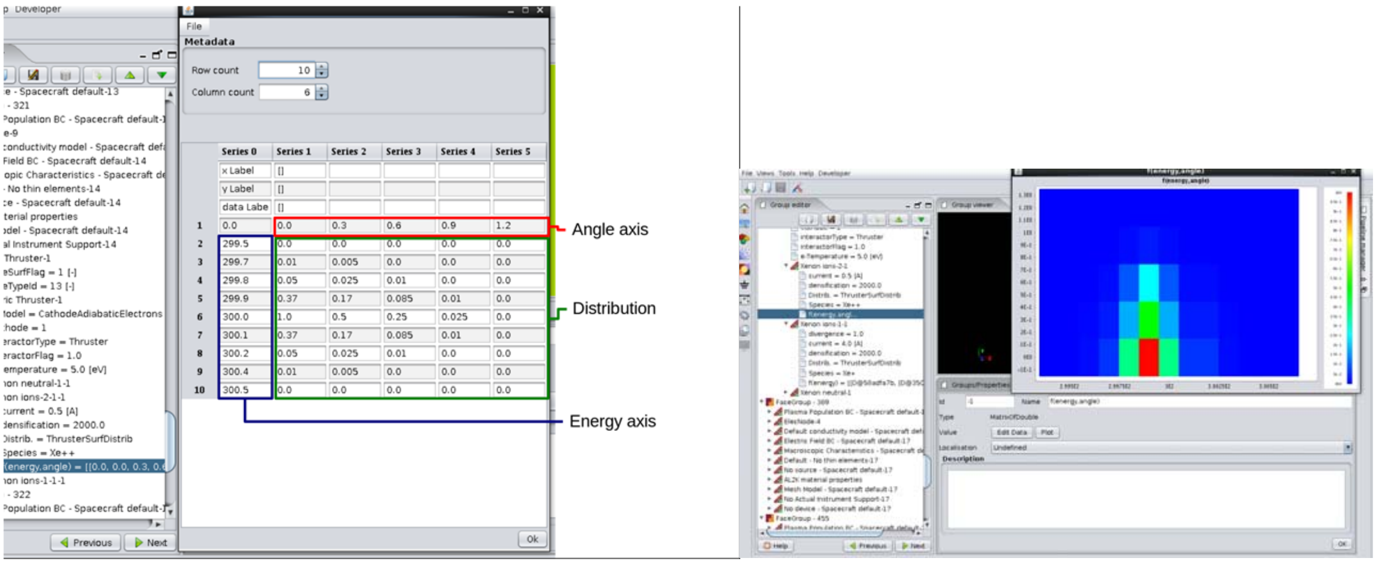
Figure 15: Example of 1D distribution in Energy. (Right) table of data, (left) distribution plot.

3.5.6.4. Setting up 2D energy, radial and angular distributions

To set up 2D energy, radial or angular distributions, you must specify in the species distribution parameters the name of the distribution you intend to define by right clicking on the distribution name and selecting "*add a new property*". This property name must be *f(energy,radius)* for the energy vs radius distribution, *f(energy,angle)* for the energy vs angle distribution and *f(radius,angle)* for the radial vs angular distribution. In any cases, the type of the property must be matrix*OfDoubles*. Then edit the distribution value by clicking on the “Edit Data” button in the lower left window. A new window appears with a matrix table. The first column stands for the energy or radius values, and the first numerated row for the radius or angle values. The first rows without numbers are used to define the units. Energies must be in [eV] while radii are in [m].

Once the distribution is validated, it is possible to display it by clicking on the “Plot” button.

An example of a 2D energy distribution is provided in the “*GenericThruster*” template provided with the SPIS electric propulsion distribution. To use it, select “*GenericThruster*” as the Device Property of the group that stands for the thruster. The doubly charged Xenon ions are defined as a shifted Maxwellian distribution similar to that defined in section 3.5.6.2, but they are defined by a 2D energy vs angle distribution.



	Series 0	Series 1	Series 2	Series 3	Series 4	Series 5
	x Label	[]				
	y Label	[]				
	data Labe	[]				
1	0.0	0.0	0.3	0.6	0.9	1.2
2	299.5	0.0	0.0	0.0	0.0	0.0
3	299.7	0.01	0.005	0.0	0.0	0.0
4	299.8	0.05	0.025	0.01	0.0	0.0
5	299.9	0.37	0.17	0.085	0.01	0.0
6	300.0	1.0	0.5	0.25	0.025	0.0
7	300.1	0.37	0.17	0.085	0.01	0.0
8	300.2	0.05	0.025	0.01	0.0	0.0
9	300.4	0.01	0.005	0.0	0.0	0.0
10	300.5	0.0	0.0	0.0	0.0	0.0

Figure 16: Energy –angle distribution of the doubly charged Xenon ion in the GenericThruster template.

3.5.6.5. Setting up droplets

It is possible to define droplets emitted by the thruster by setting the species type to “*dust*”. Then the droplet radius, mass and charge are defines in by the “*dustDefaultRadius*”, “*dustDefaultMass*” and “*dustInitialCharge*” global parameters. Note: the dust package must be installed (it is by default in the Electric propulsion distribution).

3.5.7. *Thruster magnetic field*

It is possible to specify magnetic fields associated to the thruster in the group editor. To do so, the name of the magnetic field model must be assigned to the thruster properties named “*MagneticField*”, “*MagneticField2*”… The parameters of the magnetic field models must be defined as for the environment magnetic fields.

3.5.8. *Setting a thruster from a ion current fit measured on a shell*

The definition of a thruster flux angular distribution is possible by only defining the current measured on a shell at a certain distance of the thruster. For example to define an ion distribution in angle from measurement on a shell at 0.7 m from the thruster (see below example from AISEPS project).

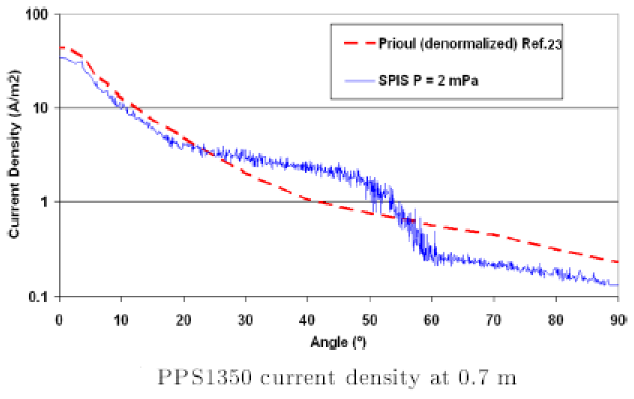
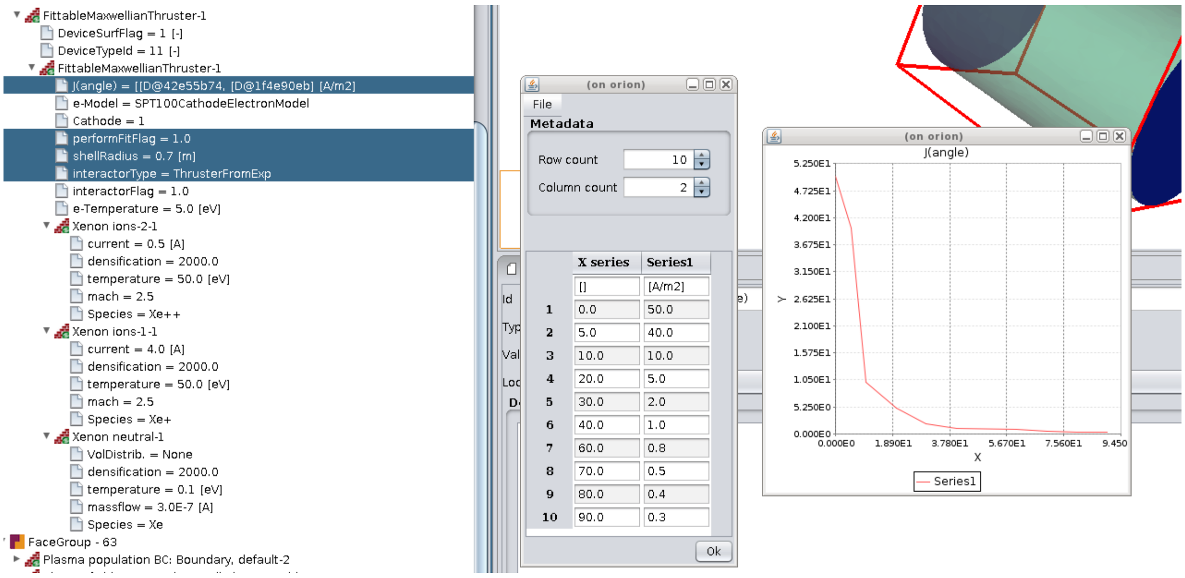


Figure 17: Current density of ions measured as a function of angle at a distance of 0.7 m (data extracted from AISEPS documentation)

In order to define such a thruster, the user has to define in the local parameters a new type of thruster: *ThrusterFromExp* instead of *Thruster*.



```
▼ FittableMaxwellianThruster-1
    DeviceSurfFlag = 1 [-]
    DeviceTypeId = 11 [-]
  ▼ FittableMaxwellianThruster-1
      J(angle) = [[D@42e55b74, [D@1f4e90eb] [A/m2]
      e-Model = SPT100CathodeElectronModel
      Cathode = 1
      performFitFlag = 1.0
      shellRadius = 0.7 [m]
      interactorType = ThrusterFromExp
      interactorFlag = 1.0
      e-Temperature = 5.0 [eV]
    ▼ Xenon ions-2-1
        current = 0.5 [A]
        densification = 2000.0
        temperature = 50.0 [eV]
        mach = 2.5
        Species = Xe++
    ▼ Xenon ions-1-1
        current = 4.0 [A]
        densification = 2000.0
        temperature = 50.0 [eV]
        mach = 2.5
        Species = Xe+
    ▼ Xenon neutral-1
        VolDistrib. = None
        densification = 2000.0
        temperature = 0.1 [eV]
        massflow = 3.0E-7 [A]
        Species = Xe
FaceGroup - 63
▶ Plasma population BC: Boundary, default-2
```

(on orion)

File

Metadata

Row count 10

Column count 2

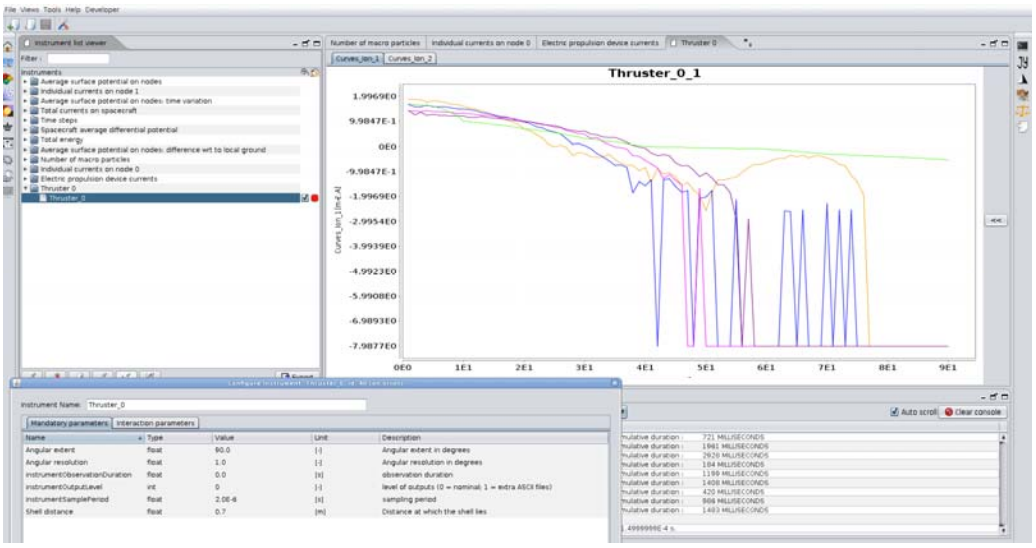
	X series	Series1
	[]	[A/m2]
1	0.0	50.0
2	5.0	40.0
3	10.0	10.0
4	20.0	5.0
5	30.0	2.0
6	40.0	1.0
7	60.0	0.8
8	70.0	0.5
9	80.0	0.4
10	90.0	0.3

Ok

Figure 18: Local parameters of a fittable thruster

In addition to the standard parameters of a thruster, the user has to define a shell radius (i.e. *shellRadius* and *performFitFlag* parameters). The tabulated current measured has to be define in the parameter *j(angle)*.

This thruster type creates automatically an instrument that measures the flux on a shell during the simulation and that performs an optimisation of the divergence angle of the thruster (if the *performFitFlag* = 1).



Instrument Name: Thruster_0

Mandatory parameters | Interaction parameters

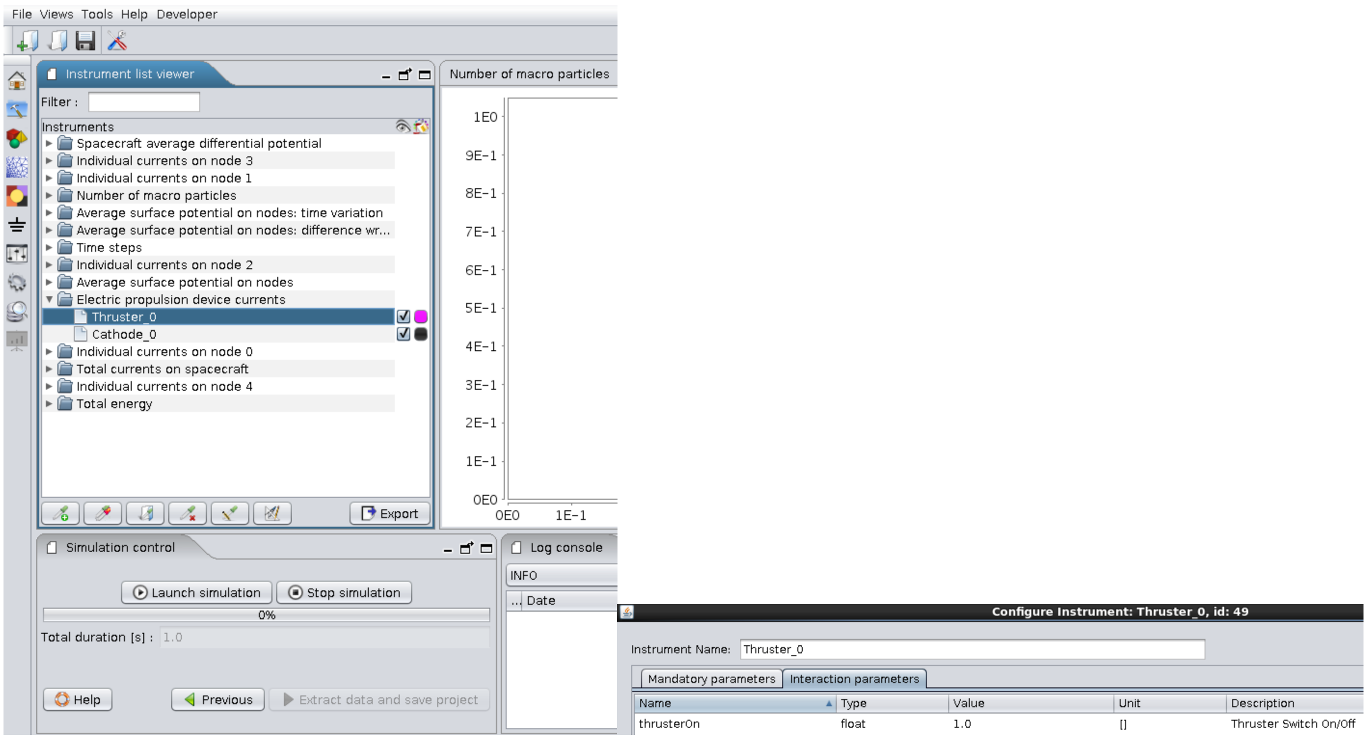
Name	Type	Value	Unit	Description
Angular extent	float	90.0	[-]	Angular extent in degrees
Angular resolution	float	1.0	[-]	Angular resolution in degrees
instrumentObservationDuration	float	0.0	[s]	observation duration
instrumentOutputLevel	int	0	[-]	level of outputs (0 = nominal; 1 = extra ASCII files)
instrumentSamplePeriod	float	2.0E-6	[s]	sampling period
Shell distance	float	0.7	[m]	Distance at which the shell lies

Figure 19: View of the current density in log scale as a function the angle on the shell at 0.7m at different time. At t=0, a large divergence is defined (curve in purple) and the divergence is automatically decreased (last time step = orange) to reach the measured current (green curve)

At the end of the simulation, the parameters of the fit are automatically saved in a form of a new thruster inside the device catalogue (i.e. in a xcat file format). This new thruster can be used in new simulation without doing a parameter fit each time.

3.5.9. *Switch On/Off of the thruster*

It is possible to switch On/Off the thruster by selecting and editing in the instrument panel the *Thruster* instrument that belongs to the "*Electric propulsion device currents*" category of instruments. The thruster switch is the *thrusterOn* parameter in the "*Interaction parameters*" tab. A value of 1.0 means that the thruster is on. Put it to 0.0 to switch the thruster off. It is also possible to schedule the start and stop times of the thruster by setting the start time and stop time parameters (in seconds).



Filter :

Instruments
- Spacecraft average differential potential
- Individual currents on node 3
- Individual currents on node 1
- Number of macro particles
- Average surface potential on nodes: time variation
- Average surface potential on nodes: difference wr...
- Time steps
- Individual currents on node 2
- Average surface potential on nodes
- Electric propulsion device currents
  - Thruster_0
  - Cathode_0
- Individual currents on node 0
- Total currents on spacecraft
- Individual currents on node 4
- Total energy

Total duration [s] : 1.0

Configure Instrument: Thruster_0, id: 49

Instrument Name: Thruster_0

Name	Type	Value	Unit	Description
thrusterOn	float	1.0	[]	Thruster Switch On/Off

Figure 20: in order to switch On/Off the thruster, select the thruster monitoring instrument and change the value of the thrusterOn parameter.

The thruster can only be fully turned off if the population distributions correspond to the "*ThrusterSurfDistrib*" generic type. Otherwise, the current emitted is divided by 10^{20} but not set to 0 (which allows switching on the thruster later by multiplying the current by 10^{20}).

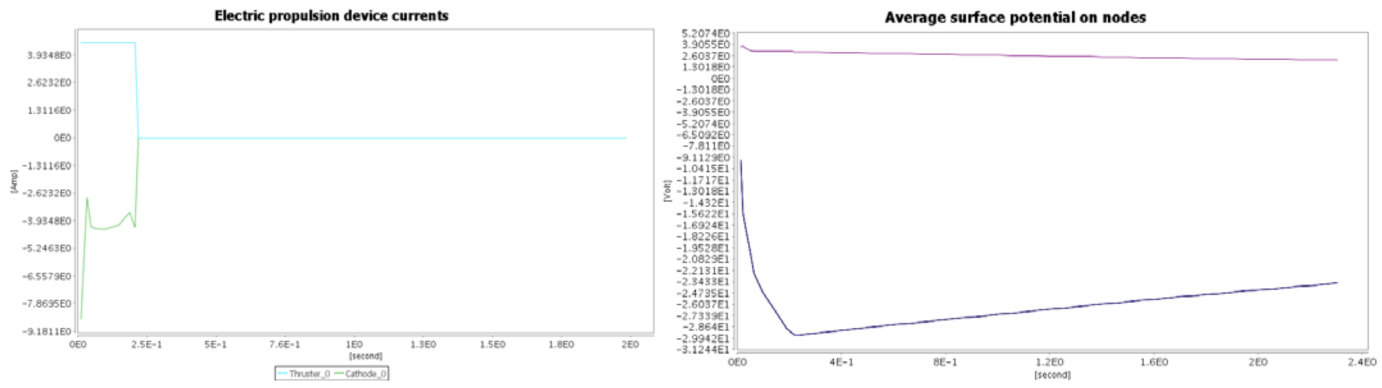


Figure 21: Simulation of a thruster and cathode switch off in a 2.8eV plasma. The thruster and cathode were switched on until 0.2s and then switched off. (Left) Thruster (blue) and Cathode (green) emitted currents. (Right) Thruster assembly (purple) and spacecraft ground (blue) potentials. The spacecraft ground potential first gets very negative because of the cathode electron recollection and return gradually to a few Volts negative by coupling with the environment.

3.6. Cathode settings

3.6.1. Setup of a cathode linked to a thruster

In order to associate a cathode to the thruster the Cathode characteristic of the thruster must be set to 1 or the name of the cathode model to use. SPIS-EP only delivers a single cathode model named "*Cathode*" which is used by default.
Depending on the cathode model, several characteristic may be specified. For the default model, one can specify:

- The cathode electron distribution: The volume distribution to be used to simulate the electrons must be defined by the parameter. The list and description of the distribution of cathode electrons is available in sections 5.1.1 and 5.1.2.
- The cathode electron temperature: The electron temperature must be defined in eV as a double characteristic named "*e- temperature*";
- The cathode position: The 3 component vector defining the position of the cathode in the simulation reference frame can be defined as a *seriesOfDouble* characteristic named "*cathodePos.*".
- The cathode direction: The 3 component vector defining the direction of the cathode in the simulation reference frame can be defined as a *seriesOfDouble* characteristic named "*cathodeDir.*".
- It is also possible to define a cathode voltage bias relative to the thruster using a Double characteristic named *"cathodeBias"*.

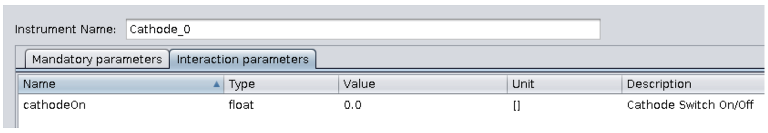
3.6.2. *Setup of a stand-alone cathode*

It is possible to define a cathode without associating it to a thruster. Then, it as to be defined as a new surface interactor in the global parameters by setting "*InteractorTypeX=Cathode*" (with X the interactor number).
The cathode parameters must also be defined in the global parameters:

- The cathode electrical node is set with the integer parameter *CathodeXElecNode*.
- The cathode electron distribution: The volume distribution to be used to simulate the electrons must be defined by the global parameter *CathodeDistribution*. The list and description of the distribution of cathode electrons is available in sections 5.1.1 and 5.1.2.
- The cathode position: The 3 component vector defining the position of the cathode in the simulation reference frame can be defined as a *series* characteristic named "C*athodeXposition*"
- The cathode direction: The 3 component vector defining the direction of the cathode in the simulation reference frame can be defined as a *series* characteristic named "*CathodeXdirection*"
- It is also possible to define a cathode voltage bias relative to the thruster with the "*CathodeXBias*" parameter.

3.6.3. *Switch On/Off of the cathode*

It is possible to switch On/Off the thruster by selecting and editing in the instrument panel the *Cathode* instrument that belongs to the "*Electric propulsion device currents*" category of instruments. The cathode switch is the *cathodeOn* parameter in the "*Interaction parameters*" tab. A value of 1.0 means that the cathode is on. Put it to 0.0 to switch the cathode off.



Name	Type	Value	Unit	Description
cathodeOn	float	0.0	[]	Cathode Switch On/Off

Figure 22: Setting of the cathode switch On/Off.

3.6.4. *Biasing the cathode*

It is possible to impose a voltage bias to the cathode. This bias corresponds to the voltage difference between the cathode and the thruster exhaust and is in principle negative (cathode more negative than the exhaust). It effectively acts by modifying the reference density for the cathode electrons (see section 5.1.1), this density is divided by $\sqrt{1-\frac{eV}{kT}}$ to model the fact that the electrons at the cathode exhaust are accelerated to neutralized the high density at the thruster exhaust. For a model as generic as the adiabatic model, this bias should be approximately equal to the CRP, but model specific models such as those from UC3M already take into account the standard CRP of the cathode, so it should be set to 0V (or not set at all).

3.6.5. Multi-Cathodes

3.6.5.1. General considerations

The multi cathode functionality aims to equilibrate the current balance between two different cathodes in case of different thrusters are implemented on the spacecraft. Due to difference in the electric circuit, on the plasma behaviour or to voltage bias, even if the thrusters and cathodes are the same, the cathode current can be not identic from the cathode. In another sense, basically, a cathode is made to neutralize the ions coming from its thruster but due to external component it could be not the case.

The multiple cathode interaction is designed as a surface interaction in SPIS that can be activated or not in the global parameters of the IME. This interactor computes the correlation coefficients between an ion plume and the electrons population coming from all cathodes:

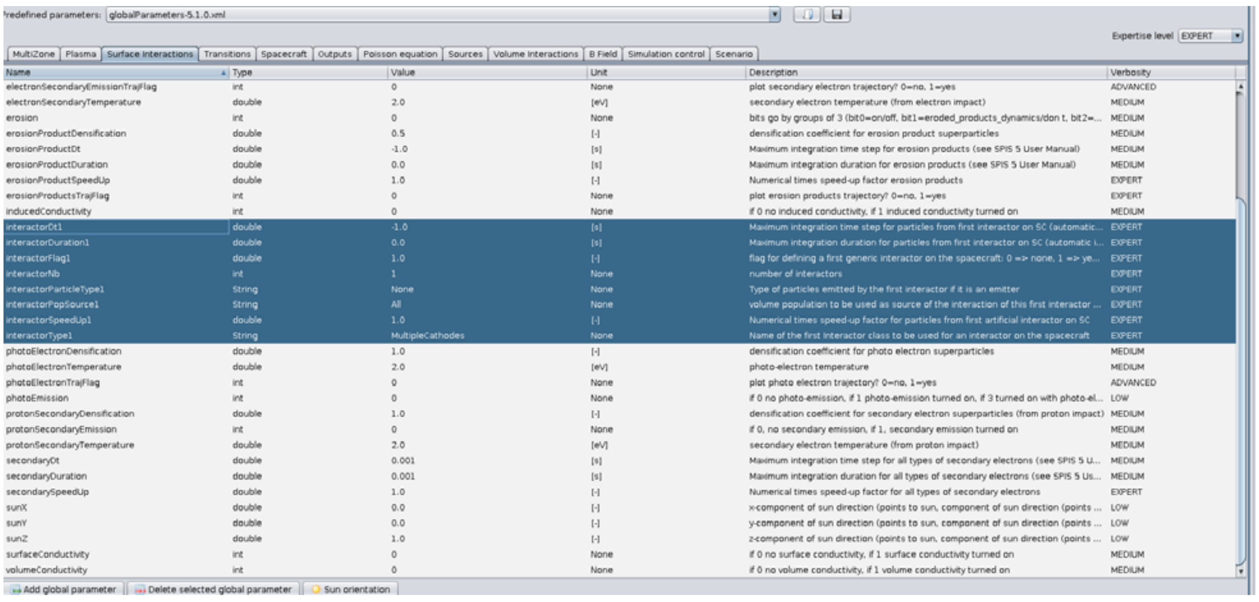
$$corr = \frac{\sigma_{\times ne}}{\sigma \quad \times \sigma_{ne}}$$

With $\sigma_{\times ne}$ the cross variance of the density of electron of a cathode with the density of ions of a thruster plume, σ the variance of the thruster plume density and σ_{ne} the variance of electron cathode density.

When a correlation is found between the densities of the ion plume of a thruster and the electron population of another cathode, a fraction of the ion density from the plume is exchanged with the other plume. This fraction of ion will be neutralized in the next iteration by the new cathode.

3.6.5.2. Multi-Cathode interactor settings

The multiple cathodes interactor should be created in SPIS as a standard surface interaction that is defined in the Global Parameter Editor as follow:



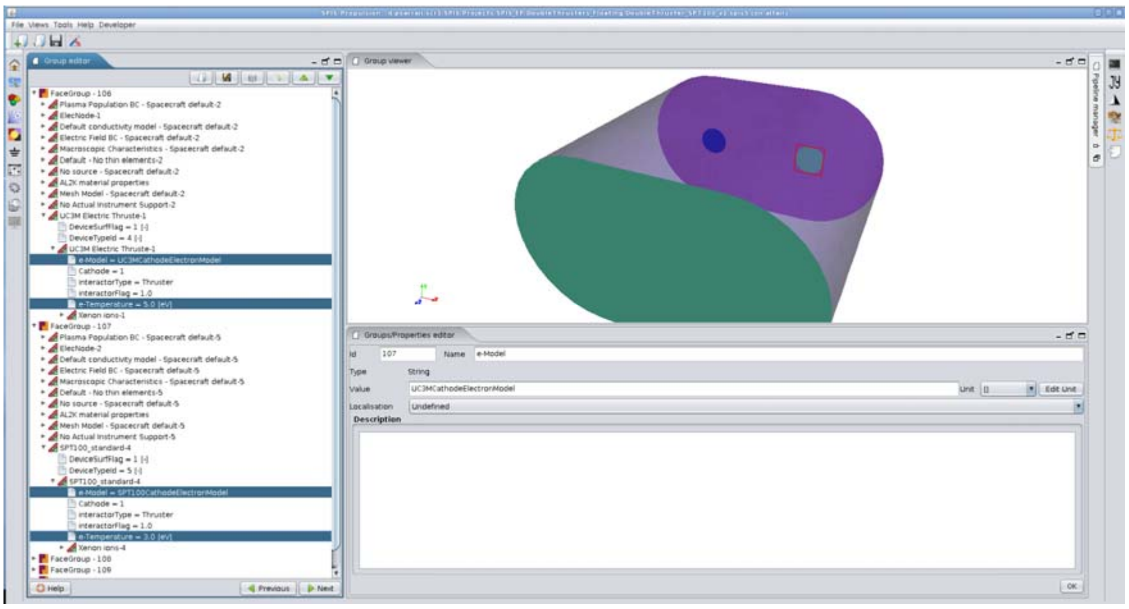
Name	Type	Value	Unit	Description	Verbosity
electronSecondaryEmissionTrajFlag	int	0	None	plot secondary electron trajectory? 0=no, 1=yes	ADVANCED
electronSecondaryTemperature	double	2.0	[eV]	secondary electron temperature (from electron impact)	MEDIUM
erosion	int	0	None	bits go by groups of 3 (bit0=on/off, bit1=eroded_products_dynamics/don t, bit2=...	MEDIUM
erosionProductDensification	double	0.5	[-]	densification coefficient for erosion product superparticles	MEDIUM
erosionProductDt	double	-1.0	[s]	Maximum integration time step for erosion products (see SPIS 5 User Manual)	MEDIUM
erosionProductDuration	double	0.0	[s]	Maximum integration duration for erosion products (see SPIS 5 User Manual)	MEDIUM
erosionProductSpeedUp	double	1.0	[-]	Numerical times speed-up factor erosion products	EXPERT
erosionProductsTrajFlag	int	0	None	plot erosion products trajectory? 0=no, 1=yes	EXPERT
inducedConductivity	int	0	None	if 0 no induced conductivity, if 1 induced conductivity turned on	MEDIUM
interactorDt1	double	-1.0	[s]	Maximum integration time step for particles from first interactor on SC (automatic...	EXPERT
interactorDuration1	double	0.0	[s]	Maximum integration duration for particles from first interactor on SC (automatic i...	EXPERT
interactorFlag1	double	1.0	[-]	flag for defining a first generic interactor on the spacecraft: 0 => none, 1 => ye...	EXPERT
interactorNb	int	1	None	number of interactors	EXPERT
interactorParticleType1	String	None	None	Type of particles emitted by the first interactor if it is an emitter	EXPERT
interactorPopSource1	String	All	None	volume population to be used as source of the interaction of this first interactor ...	EXPERT
interactorSpeedUp1	double	1.0	[-]	Numerical times speed-up factor for particles from first artificial interactor on SC	EXPERT
interactorType1	String	MultipleCathodes	None	Name of the first Interactor class to be used for an interactor on the spacecraft	EXPERT
photoElectronDensification	double	1.0	[-]	densification coefficient for photo electron superparticles	MEDIUM
photoElectronTemperature	double	2.0	[eV]	photo-electron temperature	MEDIUM
photoElectronTrajFlag	int	0	None	plot photo electron trajectory? 0=no, 1=yes	ADVANCED
photoEmission	int	0	None	if 0 no photo-emission, if 1 photo-emission turned on, if 3 turned on with photo-el...	LOW
protonSecondaryDensification	double	1.0	[-]	densification coefficient for secondary electron superparticles (from proton impact)	MEDIUM
protonSecondaryEmission	int	0	None	if 0, no secondary emission, if 1, secondary emission turned on	MEDIUM
protonSecondaryTemperature	double	2.0	[eV]	secondary electron temperature (from proton impact)	MEDIUM
secondaryDt	double	0.001	[s]	Maximum integration time step for all types of secondary electrons (see SPIS 5 U...	MEDIUM
secondaryDuration	double	0.001	[s]	Maximum integration duration for all types of secondary electrons (see SPIS 5 Us...	MEDIUM
secondarySpeedUp	double	1.0	[-]	Numerical times speed-up factor for all types of secondary electrons	EXPERT
sunX	double	0.0	[-]	x-component of sun direction (points to sun, component of sun direction (points ...	LOW
sunY	double	0.0	[-]	y-component of sun direction (points to sun, component of sun direction (points ...	LOW
sunZ	double	1.0	[-]	z-component of sun direction (points to sun, component of sun direction (points ...	LOW
surfaceConductivity	int	0	None	if 0 no surface conductivity, if 1 surface conductivity turned on	MEDIUM
volumeConductivity	int	0	None	if 0 no volume conductivity, if 1 volume conductivity turned on	MEDIUM

Figure 23: Settings of the multi-cathode parameters.

3.6.5.3. *Example*

In the following example, two SPT100 thrusters have been selected with the same distribution of ions and the same current of ions (i.e. 4.5A each) but the cathode have been considered differently with:

- Two different electron models
- Two different electron temperatures



FaceGroup - 106
- Plasma Population BC - Spacecraft default-2
- ElecNode-1
- Default conductivity model - Spacecraft default-2
- Electric Field BC - Spacecraft default-2
- Macroscopic Characteristics - Spacecraft default-2
- Default - No thin elements-2
- No source - Spacecraft default-2
- AL2K material properties
- Mesh Model - Spacecraft default-2
- No Actual Instrument Support-2
- UC3M Electric Thruste-1
  - DeviceSurfFlag = 1 [-]
  - DeviceTypeId = 4 [-]
  - UC3M Electric Thruste-1
    - e-Model = UC3MCathodeElectronModel
    - Cathode = 1
    - interactorType = Thruster
    - interactorFlag = 1.0
    - e-Temperature = 5.0 [eV]
    - Xenon ions-1

FaceGroup - 107
- Plasma Population BC - Spacecraft default-5
- ElecNode-2
- Default conductivity model - Spacecraft default-5
- Electric Field BC - Spacecraft default-5
- Macroscopic Characteristics - Spacecraft default-5
- Default - No thin elements-5
- No source - Spacecraft default-5
- AL2K material properties
- Mesh Model - Spacecraft default-5
- No Actual Instrument Support-5
- SPT100_standard-4
  - DeviceSurfFlag = 1 [-]
  - DeviceTypeId = 5 [-]
  - SPT100_standard-4
    - e-Model = SPT100CathodeElectronModel
    - Cathode = 1
    - interactorType = Thruster
    - interactorFlag = 1.0
    - e-Temperature = 3.0 [eV]
    - Xenon ions-4

FaceGroup - 108

FaceGroup - 109

Id: 107 Name: e-Model
Type: String
Value: UC3MCathodeElectronModel Unit: []
Localisation: Undefined
Description

Figure 24: Geometry and settings of the thrusters and associated cathodes.

In the next figures, the current emitted by the cathode and the surface potential can be observed during the course of the simulation.

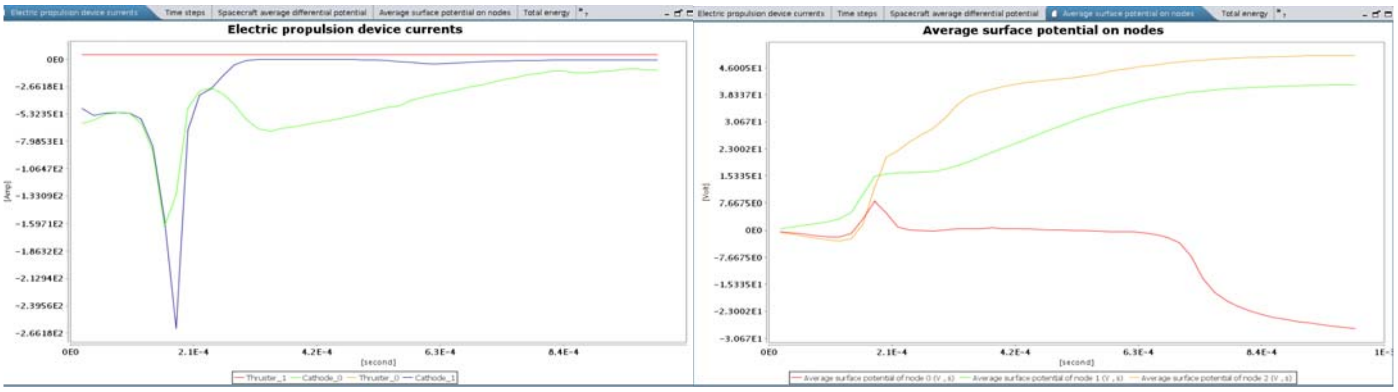
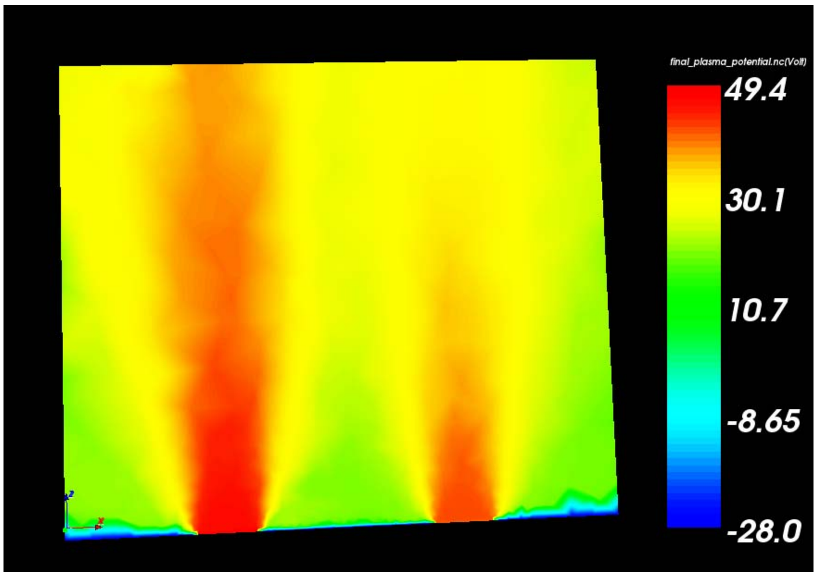


Figure 25: evolution of the surface potentials and currents emitted by the cathodes and thrusters.

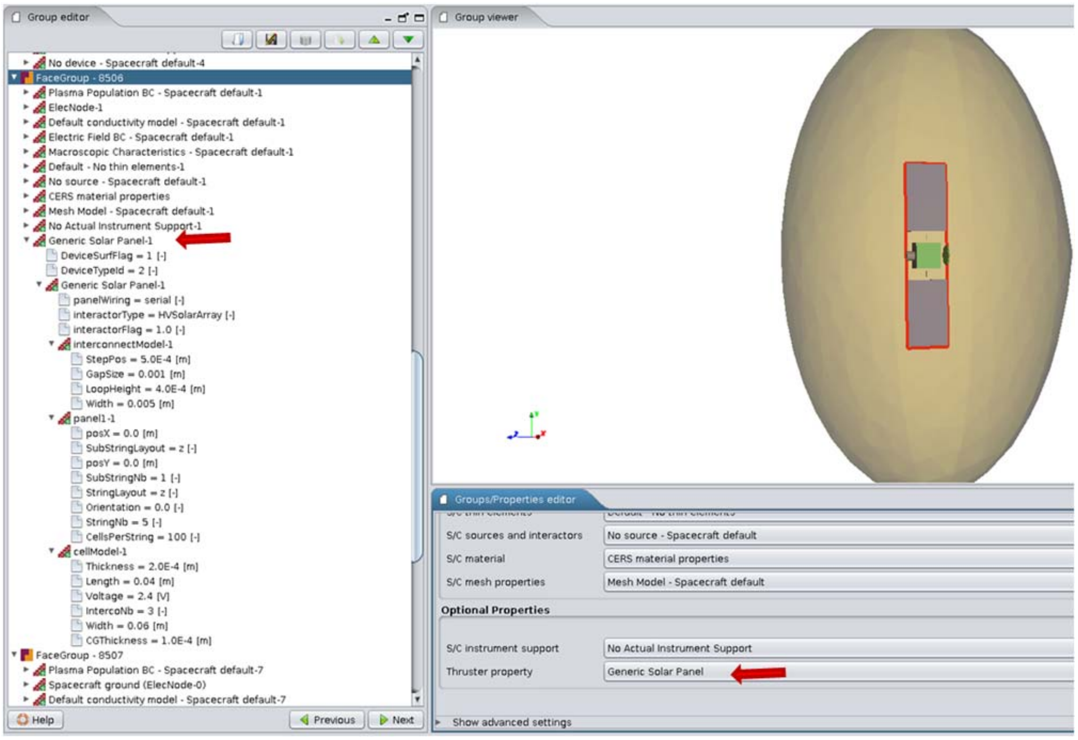
After 0.8 ms, the current and the potential stabilize to different values due to the fact the electron temperature is not the same in the both thrusters. The thruster with the electron temperature higher tens to neutralize a larger part of the ions plumes. Instead to read a current of 4.5A, the first cathode stabilizes to a current of 0.6 A and the second cathode to a current of about 9 A.

The potential of the cathode is also not the same at the end of the simulation. The body of the thrusters tends to about -30V, the reference potential of the first thruster tends to +48 V and the thruster 2 to +40 V.



3.7. **Solar panels and exposed conductors**

3.7.1. Setting



```
▸ No device - Spacecraft default-4
▾ FaceGroup - 8506
  ▸ Plasma Population BC - Spacecraft default-1
  ▸ ElecNode-1
  ▸ Default conductivity model - Spacecraft default-1
  ▸ Electric Field BC - Spacecraft default-1
  ▸ Macroscopic Characteristics - Spacecraft default-1
  ▸ Default - No thin elements-1
  ▸ No source - Spacecraft default-1
  ▸ CERS material properties
  ▸ Mesh Model - Spacecraft default-1
  ▸ No Actual Instrument Support-1
  ▾ Generic Solar Panel-1
      DeviceSurfFlag = 1 [-]
      DeviceTypeId = 2 [-]
    ▾ Generic Solar Panel-1
        panelWiring = serial [-]
        interactorType = HVSolarArray [-]
        interactorFlag = 1.0 [-]
      ▾ interconnectModel-1
          StepPos = 5.0E-4 [m]
          GapSize = 0.001 [m]
          LoopHeight = 4.0E-4 [m]
          Width = 0.005 [m]
      ▾ panel1-1
          posX = 0.0 [m]
          SubStringLayout = z [-]
          posY = 0.0 [m]
          SubStringNb = 1 [-]
          StringLayout = z [-]
          Orientation = 0.0 [-]
          StringNb = 5 [-]
          CellsPerString = 100 [-]
      ▾ cellModel-1
          Thickness = 2.0E-4 [m]
          Length = 0.04 [m]
          Voltage = 2.4 [V]
          IntercoNb = 3 [-]
          Width = 0.06 [m]
          CGThickness = 1.0E-4 [m]
▾ FaceGroup - 8507
  ▸ Plasma Population BC - Spacecraft default-7
  ▸ Spacecraft ground (ElecNode-0)
  ▸ Default conductivity model - Spacecraft default-7
```

S/C sources and interactors	No source - Spacecraft default
S/C material	CERS material properties
S/C mesh properties	Mesh Model - Spacecraft default
Optional Properties	
S/C instrument support	No Actual Instrument Support
Thruster property	Generic Solar Panel

Figure 26: Settings of a solar panel using the template

The high voltage elements can be set in the simulation by defining a device property with the *interactorType HVSolarArray* or *HVConductor*. It is recommended to use the templates as a starting point to define the interactions.

3.7.2. High Voltage Solar Array

In order to simulate the current collection by unmeshed interconnects on the solar arrays it is needed to set the *interactorType* as *HVSolarArray*. A solar array definition is done through the definition of sub-elements:

- Solar Cell
  - *Thickness*: thickness of the solar cell (excluding the cover glass)
  - *CGThickness*: thickness of the cover glass
  - *Length*: dimension along the string direction
  - *Width*: dimension perpendicular to the string direction
  - *IntercoNb*: number of interconnects per cell
  - *Voltage*: voltage across the solar cell

- Interconnect
  - *GapSize*: size of the gap between cells
  - *LoopHeight*: largest distance between the interconnect and the insulating substrate.
  - *Width*: size of the interconnect along the gap
- Panels
  - The panel can be divided into subpanels. The template proposes only one subpanel, but other can be added following the same pattern. These panels can be wired in series (*panelWiring: serial*) or in parallel (*panelWiring: parallel*)
  - The subpanels have a position defined by the position of the (x0,y0) corner in the group surface reference plane (*posX* and *posY*). This reference plane is computed by SPIS as follows: z is the normal to the solar panel surface oriented toward the plasma, x is along the shortest dimension of the panel and y is along the longest dimension of the panel. It is recommended to start a very short simulation and to have a look at the interconnect potential map in the data mining section to ensure that the panel position and orientation is correct.

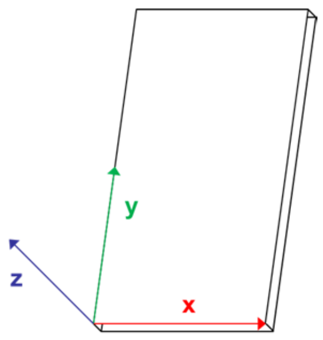


Figure 27: orientation of the panel reference frame

  - Subpanel may be composed of several string (*StringNb*), themselves composed of substrings (*SubStringNb*). Strings are wired in parallel while substrings are in series.
  - From a layout point of view, substrings of the same strings are always contiguous.
  - The successive strings and substrings can be disposed in parallel (potential gradient always in the same direction) or in zig-zag (potential gradient direction alternates). This is set by setting *[Sub]StringLayout* to “=” or “z”.

3.7.3. Exposed conductor

It is possible to obtain the current collection by an unmeshed exposed polarized conductor by setting the *interactorType* to *HVConductor*. The conductor is defined by:

- *posX* and *posY* define its position in the group reference frame (see Figure).
- *width* and *length* define its size.
- *Orientation* defines its orientation (in degrees) relative to the x direction.
- *BusPotential* defined its potential relative to the group electrical node potential.

3.8. Erosion

Up to SPIS version 5.2, the erosion modelling in SPIS was hard coded and the user could only turn it on or off without any control on the physical model. This has been modified in SPIS-EP, allowing the user to select the physical models determining (1) the yield as a function of the impinging particle energy and incidence and (2) the distribution of the eroded product(s). The yield model can be selected thanks to the *erosionYield* global parameter, whereas the product distribution can be selected with the *erosionSampler* global parameter.

3.8.1. Garcia-Rosales-Bodhansky-Oeshner yield model (default)

This model is the default model. In order to preserve backward compatibility of the SPIS simulation files, it will be used if the *erosionYield* parameter is not defined. To explicitly select this model, *erosionYield* must be set to “*GRBO_Oeschner_ErosionYield*”. The model is based on the model published in C. Garcia-Rosales et al, J. of Nuclear Materials 218 (1994) 8-17 for the energy dependence and on the Oeshner model for the incidence dependence.

3.8.2. Garcia-Rosales-Bodhansky-Yamamura yield model

This model is part of the *erosion_contamination* bundle. It is included in the SPIS-EP distribution. In case an error message appears indicating that the class was not found, please check that the plugin file is present in SPIS lib directory. To select this model, *erosionYield* must be set to “*GRBO_Yamamura_ErosionYield*”. The model is based on the model published in C. Garcia-Rosales et al, J. of Nuclear Materials 218 (1994) 8-17 for the energy dependence and on the Yamamura model for the incidence dependence.

3.8.3. Tabulated erosion yield

This model is part of the *erosion_contamination* bundle. It is included in the SPIS-EP distribution. In case an error message appears indicating that the class was not found, please check that the plugin file is present in SPIS lib directory. To select this model, *erosionYield* must be set to “*Tabulated_Erosion_Yield*”. In this model, the yield is defined by the user for each couple of impinging species and material. For each material, a property named “*XXXErosionYield*” (with *XXX* the species name) must be created. This parameter is of type matrix, and gives the yield as a function of the energy (in eV) and of the pitch angle (in degrees).

3.8.4. Tondu eroded product model (default)

This model is the default model of eroded product emission. In order to preserve backward compatibility of the SPIS simulation files, it will be used if the *erosionSampler* parameter is not defined. It is based on the Tondu model.

3.8.5. Tabulated eroded product model

This model is part of the erosion*_contamination* bundle. It is included in the SPIS-EP distribution. In case an error message appears indicating that the class was not found, please check that the plugin file is present in SPIS lib directory. To select this model, *erosionSampler* must be set to "*TabulatedProductSampler*". In this model, the emission pattern is defined by the user for each triplet of impinging species, material and erosion product. For each material, a property named "*XXXErosionProducts*" (with *XXX* the species name) can be created. It is not, a single generic erosion product will be created. This parameter is a string listing all products by their chemical formula, separated by a comma. The branch ratio of all product can be specified by writing it as a float separated from the chemical formula by an underscore. Example: "Al_0.1,C_0.9" indicates that the eroded products are for 90% carbon and for 10% aluminium.

If the eroded products are specified, the emission pattern of each must be specified in two material properties named "*XXX_YYY_ErosionDir*" and "*XXX_YYY_ErosionWidth*", where XXX is the impinging species and YYY the eroded product. If the eroded products are not specified, the emission pattern is specified by material properties named "*XXX_ErosionDir*" and "*XXX_ErosionWidth*".

These parameters are of type matrix, and gives the direction of emission (*ErosionDir*) and the angular dispersion (*ErosionWidth*) of the eroded product(s) as a function of the energy (in eV) and of the pitch angle (in degrees) of the impinging species. Both the erosion direction and the angular dispersion are given in degrees.

3.9. **Contamination**

It is possible to estimate the level of contamination of the spacecraft due to heavy particle collection by setting up the *Contamination* interactor in the global parameters as a surface interactor. All particles heavier than helium are taken into account. The flux of particles impinging the spacecraft is collected a fraction of which sticks to the spacecraft, the rest being reemitted as a Lambertian whose temperature is the surface temperature. The sticking coefficient depends on the spacecraft surface temperature as:

$$S = \begin{cases} 1 & T < T_0 - 0.5\Delta T \\ 0.5 - (T - T_0) * \Delta T^{-1} & \\ 0 & T > T_0 + 0.5/\Delta T \end{cases}$$

The parameters T_0 and ΔT^{-1} are material parameter in eV named "*STKE*" and "*STKW*", respectively.By default, T_0 is equal to 10^{12} eV and ΔT value is equal to 10^{10} eV, i.e. the sticking coefficient equals 1.
Along with the particles that do not stick to the surface, a part of the particles that is deposited on the surface evaporates. The fraction of the contaminant that evaporates is:

$$E=1-\exp(t.\tau^{-1})$$

where the contamination evaporation time scale τ^{-1} is a material parameter in s^{-1} named "*CETS*". By default τ^{-1} equals $0s^{-1}$ (no evaporation).

The contamination integration can be accelerated by setting the parameter *contaminationSpeedUp* to the desired acceleration factor.

3.10. Volume interaction

3.10.1. General considerations

Several general actions have been taken to improve the accuracy of the volume interactions:

(1) The interactions may now be computed along the particle trajectory, i.e. once in each cell that is crossed by each interacting particles.
This is done by default when using the *GenericVolInteraction* which is available in the *Erosion&Contamination* package. It is also possible to use this functionality with the older *CEXInteractor* available in SPIS-Core by setting the *VolInteractXTrajectory* flag to 1.
Although this leads to more accurate results, it also results in slower simulations (because more computations are needed).

(2) The sampling of the products is improved to favour the particles that imping the satellite surfaces. This allows getting a better statistics for erosion computations.

(3) The generation of fast neutral is taken into account. For accuracy and simulation speed reasons, fast neutrals are not simulated are particles but there flux is computed using a factor of view approach between the 3D source distribution and the satellite surfaces.

A generic volume interactor has been defined that allows computing several physico-chemical interactions in the plasma. To use it, the *Erosion&Contamination* package shall be present in the *lib* directory of SPIS (see section 3.1.1).

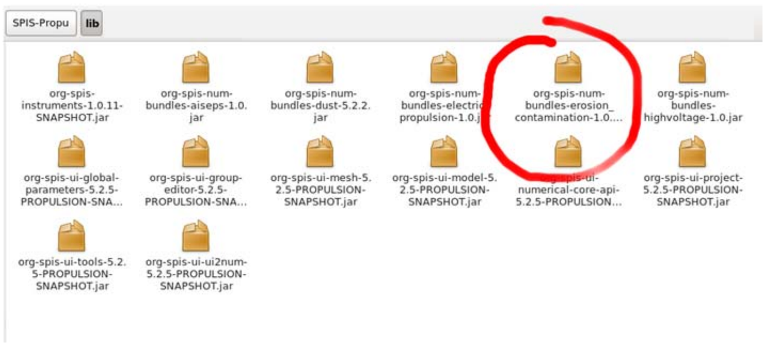
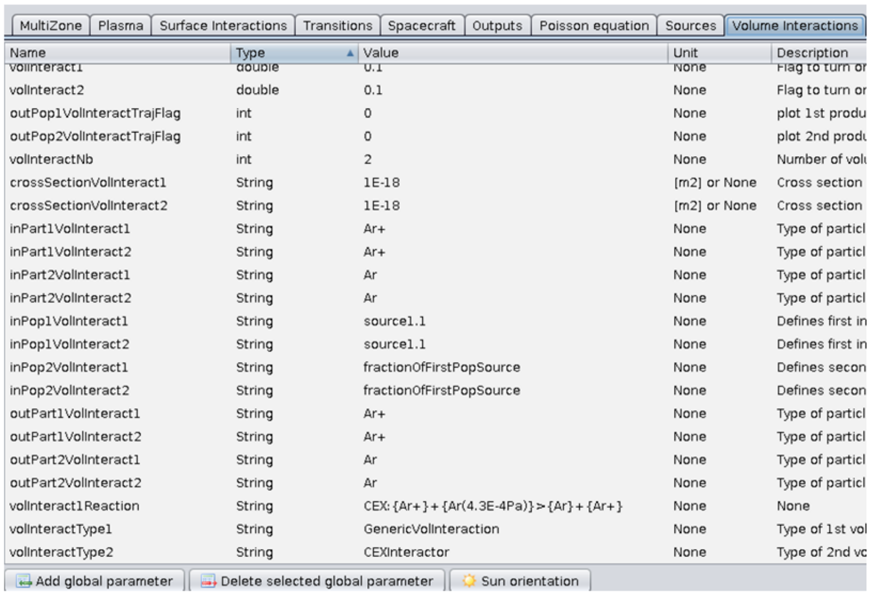

SPIS-Propu
lib
org-spis-instruments-1.0.11-SNAPSHOT.jar
org-spis-num-bundles-aiseps-1.0.jar
org-spis-num-bundles-dust-5.2.2.jar
org-spis-num-bundles-electric propulsion-1.0.j
org-spis-num-bundles-erosion_contamination-1.0....
org-spis-num-bundles-highvoltage-1.0.jar
org-spis-ui-global-parameters-5.2.5-PROPULSION-SNA...
org-spis-ui-group-editor-5.2.5-PROPULSION-SNA...
org-spis-ui-mesh-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-model-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-numerical-core-api-5.2.5-PROPULSION...
org-spis-ui-project-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-tools-5.2.5-PROPULSION-SNAPSHOT.jar
org-spis-ui-ui2num-5.2.5-PROPULSION-SNAPSHOT.jar

Figure 28: Package Erosion&Contamination in the lib directory

Then, the *GenericVolInteraction* shall be assigned to a *VolInteractTypeX* global parameter. The *volInteractX* flag shall then set to a positive value standing for the densification coefficient as for previous volume interactions. Do not forget to set the correct number of volume interactors with the *volInteractNb* parameter. The volume interaction modelled by the *GenericVolInteraction* is then defined by the *volInteractXReaction*.



Name	Type	Value	Unit	Description
volInteract1	double	0.1	None	Flag to turn or
volInteract2	double	0.1	None	Flag to turn or
outPop1VolInteractTrajFlag	int	0	None	plot 1st produ
outPop2VolInteractTrajFlag	int	0	None	plot 2nd produ
volInteractNb	int	2	None	Number of volu
crossSectionVolInteract1	String	1E-18	[m2] or None	Cross section
crossSectionVolInteract2	String	1E-18	[m2] or None	Cross section
inPart1VolInteract1	String	Ar+	None	Type of particl
inPart1VolInteract2	String	Ar+	None	Type of particl
inPart2VolInteract1	String	Ar	None	Type of particl
inPart2VolInteract2	String	Ar	None	Type of particl
inPop1VolInteract1	String	source1.1	None	Defines first in
inPop1VolInteract2	String	source1.1	None	Defines first in
inPop2VolInteract1	String	fractionOfFirstPopSource	None	Defines secon
inPop2VolInteract2	String	fractionOfFirstPopSource	None	Defines secon
outPart1VolInteract1	String	Ar+	None	Type of particl
outPart1VolInteract2	String	Ar+	None	Type of particl
outPart2VolInteract1	String	Ar	None	Type of particl
outPart2VolInteract2	String	Ar	None	Type of particl
volInteract1Reaction	String	CEX: {Ar+} + {Ar(4.3E-4Pa)} > {Ar} + {Ar+}	None	None
volInteractType1	String	GenericVolInteraction	None	Type of 1st vol
volInteractType2	String	CEXInteractor	None	Type of 2nd vc

Figure 29: example of generic volume interaction (interaction 1). It defines a charge exchange reaction between Argon ions and neutrals, with the reactant neutrals defined as a population with a constant pressure.

The reaction rate is defined by the *crossSectionVolInteractX* parameter. This parameter's unit shall be [m$^{(3r-4)}$] where *r* is the number of reactants (e.g. in m^2 for two reacting species). This parameter can either be a double (constant cross-section), a series (table of cross-sections versus the impinging particle energy in eV) or the name of a file containing the table of cross-sections (same format as CEXInteractor, for backward compatibility).

The number of product particles generated is sampled such as the statistics on a target gets improved. The space and velocity domain is divided in 10 bins in each dimension, and the number of particles in each bins is computed to send more particles on the target than in other directions The target on which the statistics is improved is define by attributing to it a *sourceId* in the group editor. Then, this id is reported in the *TargetIdVolInteractX* global parameter. The increase of particles toward the target is controlled with the *TargetDensVolInteractX* global parameter (1= no particle number increase, default=3).

3.10.2. Reaction description

The general manner of defining a reaction is:

Name : N_1\{reactant$_1$\}+ N_2\{reactant$_2$\}+…+N_N*\{reactant$_N$\}*> M_1\{product$_1$\}+…+*\{$M_M$product$_M$\}*

Italic font stands for optional elements.

The Name is a simple name for the reaction that is used in the outputs naming. Several interactions can have the same name as the interaction number is always added to the name. If omitted the name is "VolInteract".

N_x and M_x stand for the stoichiometric coefficients. Those must be integer or decimal numbers. If omitted the stoichiometric coefficient is one.

reactants and *products* can be either :

- Chemical formula of species. Example: Xe, Xe+,… For reactants, and except for particular cases detailed below, these species must correspond to existing populations in the simulation.
- Name of existing populations. Example: ions1,…
- Photons. In which case it should appear either as "photon" or "hv"

<u>For reactants:</u>
It is possible to use as reactant a species that was not yet defined. It can be done by defining the species chemical formula and either a constant pressure or a influx that is a fraction of the first population influx (these possibilities correspond to that offered by the *CEXInteractor* and where developed to ease the validation of the interactor).
To define a population with a constant pressure the reactant name should be: {reactant(*xx*Pa)} with *xx* the pressure in Pascals (example: {Ar(10Pa)}).
To define a population with an influx being a fraction of that of the first population, the reactant name should be {reactant(*xx*%OfFirstPopulation)}, with *xx* the fraction of the first population in flux. Obviously, such a population cannot be defined as the first population itself.
In these two special cases, the neutral temperature must be defined in eV by the *parameter2VolInteractX* parameter.

<u>For products</u>
If the product is a photon, an electron or a neutral defined by a chemical formula, it is ignored (meaning no particle created). If the product is defined by a population name (ex: ions1,…) it will not be ignored.
If the product corresponds to fast neutral its name must be: {fastproduct} (example: fastXe for fast xenon neutrals).

3.10.3. Examples of reactions

CEX:{Xe+}+{Xe}>{fastXe}+{Xe+}: charge exchange reaction between Xenon ions and neutral

{Thruster0_Xe+}+{Xe}>{Thruster0_Xe+}+{Xe}: elastic collision between Xenon ions

{Xe}+{hv}>{Xe+}+{electron}: photoionization, note that the{electron} is not mandatory.

{Xe}+{electron}>{Xe+}+2{electron}: ionization by electron impact.

{Xe+}+{electron}>{Xe}: recombination

3.11. **Magnetic fields**

Previous versions of SPIS only allowed defining a single uniform magnetic field associated with the environment. SPIS-DUST also allowed defining a single magnetic dipole in the simulation. SPIS-EP now introduces the possibility of having an unlimited number of magnetic fields. To define a magnetic field in the global parameters, it is needed to determine the magnetic field model by defining the "*MagneticModel*" parameter (for the first model) or the "*MagneticModelX*" parameter (with X the model number, starting with 2). SPIS-EP provides three models of magnetic field: "*UniformBField*" (uniform field), "*DipolarBField*" (magnetic dipole) and "*SolenoidBField*" (solenoid magnetic field). SPIS-EP also provides a new particle pusher to compute the particle motion in the magnetic field (see section 4.5).

3.11.1. Uniform Field

To define a uniform magnetic field, one must provide the components of the magnetic field by specifying the global parameters "*Bx*", "*By*" and "*Bz*" (or *BxN*, *ByN*, *BzN* if the field number is N). The magnetic field unit is Tesla.

3.11.2. Dipolar Field

To define a uniform magnetic field, one must provide the components of the magnetic moment by specifying the global parameters "*MagMomentx*", "*MagMomenty*" and "*MagMomentz*" (or *MagMomentxN*, *MagMomentyN*, *MagMomentzN* if the field number is N). The magnetic moment unit is T.m^3. Then, it is needed to define the dipole centre by specifying the global parameters "*MagDipoleCenterx*", "*MagDipoleCentery*" and "*MagDipoleCenterz*" (with the model number if needed). The dipole centre coordinates are in meters.

3.11.3. Solenoid Field

To define a uniform magnetic field, one must provide the components of the magnetic field at the center of the solenoid by specifying the global parameters "*B0x*", "*B0y*" and "*B0z*" (or *B0xN*, *B0yN*, *B0zN* if the field number is N). The magnetic field is in Tesla. Then, it is needed to define the solenoid centre by specifying the global parameters "*SolenoidCenterx*", "*SolenoidCentery*" and "*SolenoidCenterz*" (with the model number if needed). The dipole centre coordinates are in meters. Finally, the solenoid length and radius are given by the global parameters "*SolenoidRadius*" and "*SolenoidLength*" (with the model number if needed) both given in meters.

3.12. **Settings of the spacecraft and simulation parameters**

2.8.1 SPIS summary report

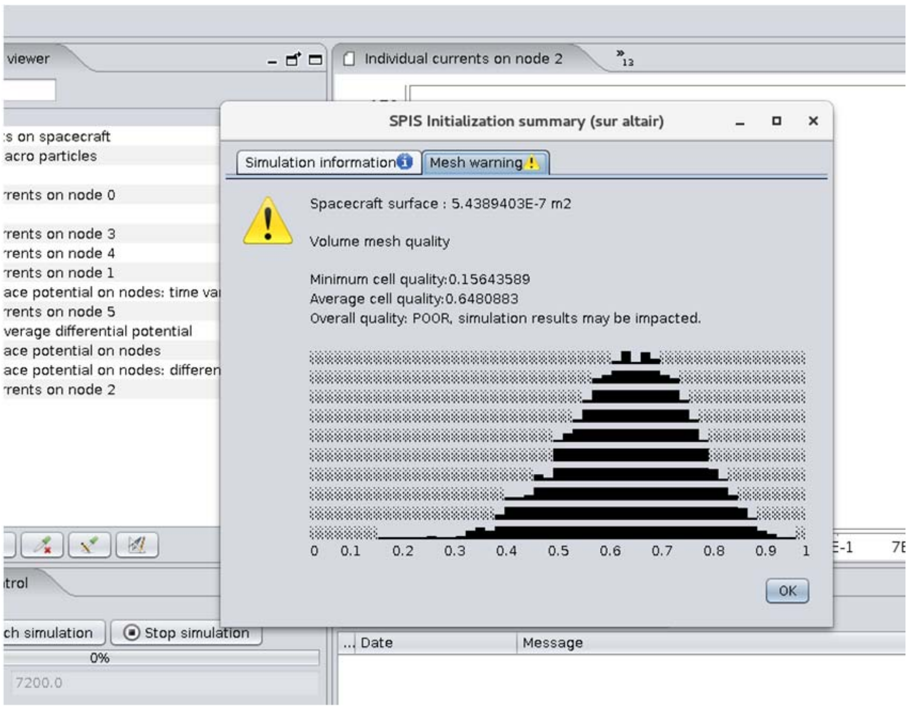
In order to ease the control of the simulation setting and initialisation, a summary report has been added to SPIS, which is available both when SPIS is used with a Graphical User Interface, from command lines or in batch mode.

Several modules may deliver information in this summary report; each of them appearing in a different tab. Up to now, two tabs at least will appear. The first one lists all the populations simulated with their main characteristics. The second one presents a histogram of the mesh quality. Depending on the quality of the mesh, the tab icon may represent information, a warning or a problem.

Some devices, namely the thrusters and the high voltage solar arrays, also present their main characteristics.

3.12.2. Mesh quality

As for any SPIS simulation, it is of primary importance that the simulation volume mesh be of the best possible quality. By "mesh quality" one should understand that the mesh tetrahedral are as close as possible as regular tetrahedral (i.e. all surfaces, angles, edges,... being equals). This is particularly the case for electric propulsion simulation in which the mesh refinement close to the thruster exhausts must be refined to catch all the gradients. One should always test the mesh with the quality tools embedded with the Gmsh software delivered with SPIS. In order to ease this process, a quality check tool has been added to the SPIS simulation summary report appearing at the simulation initialization (as a popup if the User interface is used, in plain text otherwise).



Spacecraft surface : 5.4389403E-7 m2

Volume mesh quality

Minimum cell quality:0.15643589
Average cell quality:0.6480883
Overall quality: POOR, simulation results may be impacted.

Figure 30: Summary report of the mesh quality. It is recommended to reach the GOOD overall quality marker (minimum cell quality >0.2, mean quality >0.67).

3.12.3. Current generator

Until the electric propulsion version, spacecraft circuit in SPIS could only be composed of RLC components and voltage generators. SPIS circuit solver aims at nullifying the net current on an open circuit (i.e. closed by the plasma). This formulation of the circuit does not allow for explicit current generators. In a closed circuit case, current generator can be included because the current can circulate on closed loops, preserving the consistency of the system. To include current generators in an open circuit, it is thus needed to translate the net current generated in the circuit into a net current between the circuit and the plasma.

This is why previous versions of SPIS allowed simulating current generators by defining sources in the group editor: in order to introduce a current I between to nodes, one should select the surfaces corresponding to each nodes and define a source delivering a current +I or –I depending on the node. Since SPIS offers to possibility to have sources that only delivers a current without actually generating particles, this is effectively equivalent to defining a current source. This option is still possible with SPIS-EP.

The previous option poses two difficulties: It is not obvious to do, in particular if several groups belong to the same node and it is questionable if it is applied to dielectric surfaces. Thus, SPIS-EP now offers the possibility to define a current generator directly in the circuit editor. It follows the usual SPIS netlist syntax:

writing “I 0 1 2” in the circuit editor creates a current generator between the nodes 0 and 1 that delivers a current of 2 Amperes. This current is directly injected in the electrical nodes, rather than on the surface like with the previous method. This solves the problem of having to deal with multiple surfaces per nodes and better handles dielectric surfaces.

The current generators defined one or the other ways can be defined in addition to any other electrical component, since they are treated as supplementary currents exchanged with the plasma rather than actual electrical components. The downside of it is that the current generator adds a current between the two nodes rather than forcing it to a given value. To simulate an ideal generator it is thus possible to define a large resistance between the two nodes.

3.12.4 SPICE netlist description of the circuit

SPIS v6.0.4 extends the way the spacecraft circuit can be described, based on the standard SPICE netlist syntax. Since the traditional SPIS netlist syntax is a subset of the larger SPICE syntax, the compatibility with older description is fully ensured.

The new syntax accepts comments both as full lines, using the * symbol, and at the end of a line using either the ; or $ symbols.

The general syntax of a SPIS netlist is:

T<*name*> N1 N2 [value|model<(param<=value>)>] <parameter<=value>...>

where T is the component symbol (currently R,C,L,I,V), name an optional component name, N1 and N2 are the nodes connected by the component. The component is defined by either a numerical value (following SPICE syntax, see below) or by a model identified by its name. If a model is defined, extra parameters may be added, either explicitly using the “parameter=value” syntax, where parameter is the parameter name, or by specifying the model parameters in the model specific order.

As an example, the following circuit is simulated as a functional test case each time SPIS is built:

```
*This is a test circuit for non-regression purposes
I 0 1 1
R 1 2 10
C 1 2 10
L 1 3 10
R 0 2 10
V 2 3 0; node 2 and 3 are short-circuited
Vsin 4 5 SIN(10 5 0.01)
```

NOTE FOR DEVELOPPERS: Both components and models can be defined in classes that follow a specific API, so that the list of components and models can be easily extended in plugins. The naming of the components must follow the NGSPICE nomenclature [RD4], except for “P” which is reserved for Photovoltaic cells in SPIS (although not yet implemented). The API of the components and model can be obtained by looking at the org.spis.Circ.Circ.ElecComponent class source or by contacting ONERA’s SPIS development team.

Models that are currently implemented in SPIS are the standard models for potential and current sources [RD4]. A complete description of the parameters is given in the NGSPICE manual. Note that the default values for the SPIS models may slightly differ from the NGSPICE ones.

3.12.4.1 PULSE

General form:

```
PULSE(V1 V2 TD TR TF PW PER)
```

Examples:

```
VIN 3 0 PULSE(-1 1 2NS 2NS 2NS 50NS 100NS)
```

Name	Parameter	Unit	Default value
V1	Base value	Component unit	Mandatory
V2	Pulsed value	Component unit	Mandatory
TD	Delay Time	Seconds	0
TR	Rise Time	Seconds	0
TF	Fall Time	Seconds	0
PW	Pulse Width	Seconds	1E20
PER	Period	Seconds	1E20

Intermediate points are determined by linear interpolation.

3.12.4.2 SINusoidal

General form (the PHASE parameter is only possible when XSPICE is enabled):

```
SIN(VO VA FREQ TD THETA PHASE)
```

Examples:

```
C1 3 0 SIN(0 1 100MEG 1NS 1E10)
```

Name	Parameter	Unit	Default value
V0	Offset value	Component unit	Mandatory
VA	Amplitude value	Component unit	Mandatory
FREQ	Frequence	Hertz	Mandatory
TD	Initial time shift	Seconds	0
THETA	Damping factor	1/Seconds	0
PHASE	Pulse Width	Degrees	0

3.12.4.3 EXPonential

General Form:

```
EXP(V1 V2 TD1 TAU1 TD2 TAU2)
```

Examples:

```
R2 3 0 EXP(-4 -1 2NS 30NS 60NS 40NS)
```

Name	Parameter	Unit	Default value
V1	Base value	Component unit	Mandatory
V2	Final Value	Component unit	Mandatory
TD1	Rise delay	Second	0
TAU1	Rise time	Seconds	1E-6
TD2	Fall delay	Seconds	1E20
TAU2	Fall time	Seconds	1E-6

3.12.4.4 Piece-Wise Linear

The definition of the piece-wise linear model differs slightly from that of NGSPICE :
General form:

```
PWL(T1 V1 <T2 V2 T3 V3 T4 V4 ...>) <STEPS=[0|1]>
```

Name	Parameter	Unit	Default value
Tn	Time of the nth value	second	First mandatory
Vn	nth value	Component unit	First mandatory
STEPS	Step flag	-	0

Nominal behaviour (NGSPICE) is to interpolate values between the specified times, is STEPS is set to 1, then no interpolation is performed and the last value is used.

3.12.5. Poisson solver parameters

Because of the strong and abrupt variation of the thruster ion densities in the plume, the electrons density and temperature are also varying abruptly over small distances, which put strong constraints on the Poisson solver. It may thus be needed to increase the maximum number of iterations allowed for solving the Poisson equation (by increasing the values of the *iterGradientNI* and *iterNewton* global parameters) and to decrease the value the convergence threshold (by decreasing the values of the *tolGradientNI* and *tolNewton* global parameters). The simulation stability is very sensitive to the quality of the Poisson solution. If the solution is not satisfying at some step, the simulation may diverge.

3.12.6. Spacecraft capacitance

Because of the strong currents emitted by the thrusters and cathodes, it may be needed to adjust the spacecraft capacitance in order to avoid brutal surface potential variation that could lead to a divergence of

the simulation. The spacecraft capacitance is given by the *CSat* global parameter. The capacitance of dielectric materials is given by their thickness defined in the group parameters. A good practice for all of SPIS simulation is to adjust the *CSat* to filter effect faster than those of interest (i.e. a large *CSat* should be set when only the quasi-static charge state is of interest).
In order to obtain more stable simulations and to better fit the physics, the equivalent electrical circuit used to model the dielectric coatings has been modified in SPIS-EP. In previous versions, the sheath capacitance *"CSat"* was applied to the metallic conductor which underlies the dielectric coating. This made little sense and was modelled this way only to avoid zero capacitance electrical nodes which could have been numerically unstable. In the new version of SPIS, the sheath capacitance is applied to both the dielectric coating and the underlying metallic conductor. This is more physical and better makes the distinction between the overall charging due to the environment and the differential charging due to the dielectric capacitance.

3.12.7. Using view factors for neutrals

In order to accelerate computation related to neutral particles, it is possible to use view factors. These view factors are computed in real time, but the visibility of each pairs of nodes is computed once and for all at the start of the simulation. Use of view factors is impossible when using semi transparent grids and is currently not accurate for unmeshed wires. Moreover the initial computation of the visibility between each pairs of nodes may be long for some geometry. Thus, it is possible to select the use of the view factors in the global parameters by changing the value of the *useViewFactors* parameter. A value of 0 deactivate the view factors, 1 (default) activates them and a value of 2 forces their use for the computation of the shadowing.
View factor are not used for shadowing computation by default because the photon angular distribution function is a Dirac. Thus, using the visibility between the nodes to compute the transfer function between surfaces may be inaccurate. Still, it may be efficient for dusty plasma simulation with changing sun orientation, as the re-computation of the shadowing in volume is much faster using view factors.

4.12.7 Electrical setup of the thruster assembly

The thruster assembly is exposed to a very strong plume particle collection that is not correctly modelled, in particular concerning the collection of cathode electrons which is largely overestimated. Thus, this assembly electric node should either be set to the same electrical node than the thruster exhaust or the current collection and emission from it should be set to 0 and the node be connected to the exhaust node through a voltage bias corresponding to that expected in space.
Nonetheless, the flux of ions on the thruster assembly cannot be accurately computed so that the sputtering due to erosion by ion may be overestimated. Thus, a particular care should be taken while interpreting the erosion and contamination data from the thruster assembly.

4. SPIS-EP ARCHITECTURE DESCRIPTION

4.1. Overview of SPIS architecture

The numerical kernel block diagram of the next figure presents the SPIS-NUM top level structures and their main interactions. The top level classes that encode the core functionalities and object definition of the simulation (initialization from UI, Simulation main loop, plasma and spacecraft definition) are gathered in the top (TOP) java package. The plasma object is composed of volume distributions associated to a Poisson solver, located in the solver (SOLVER) package. This package also provides Matter classes to move particles. The particle pusher of the Matter solver moves particles on the volume mesh and mark particles arriving on spacecraft and grid surface meshes. The volume (VOL) package is composed of classes relative to volume meshes, data fields living on this mesh, population distributions and interactions. The surface (SURF) package handles particle collection and emission, surface interactions and surface fields. The spacecraft is composed of a circuit solver located in the circuit (CIRC) package. The circuit solver also takes account of semi-transparent grid objects. Finally, the UTIL package collects a large number of methods (sampling, mathematics, etc…) used in all packages. Several of the SPIS-NUM elements are dynamically loaded from the user specification (Global parameters) in UI.

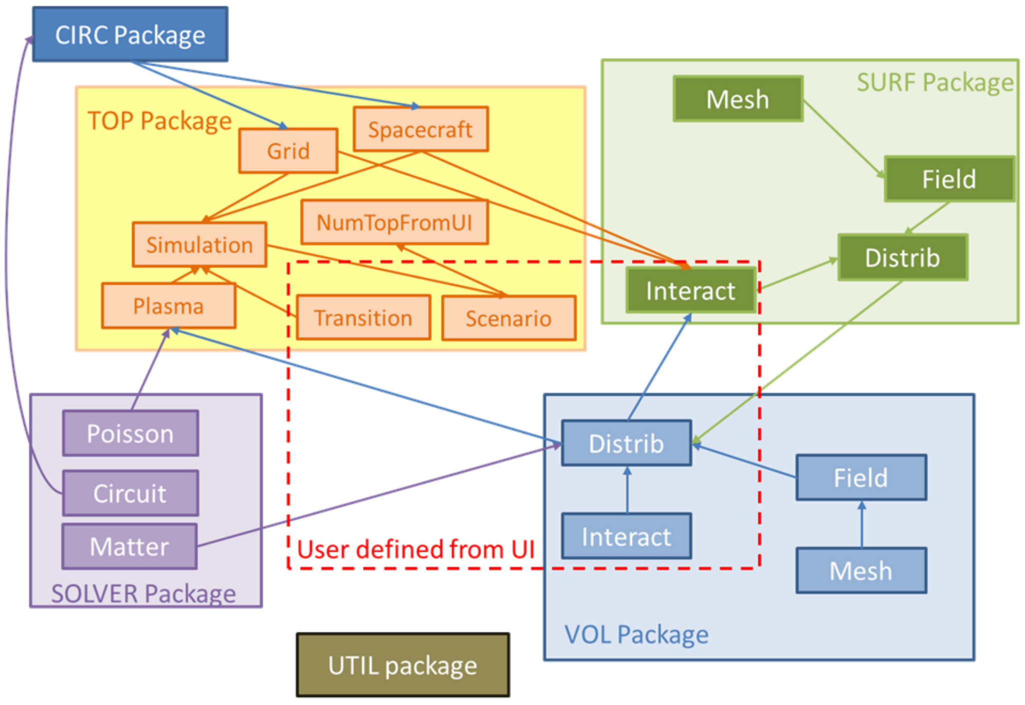


Figure 31 - Block diagram of SPIS-NUM packages, with main interactions between them.

For a complete understanding of SPIS-NUM classes' hierarchy and interactions, a proper documentation can be generated using the Javadoc command of java.

4.2. Packaging modification

SPIS-EP introduces a particularly large number of new physical modules, i.e. classes describing a certain physics that is dynamically loaded by SPIS during its execution on the request of the User in the UI through Global parameters. In SPIS 5.1 physical modules cover scenarios, transition, surface and volume interactions and distributions. The fact that these modules are loaded dynamically leads them to be de-correlated from the core of SPIS-NUM and allows to have them packaged in separated packages, which then permits the code maintainers to better handle the code evolutions (in particular for parallel projects), the code long term maintenance (since the core is shared, bug fixes are more easily performed for all projects) and a simple and fast addition of new physics (in particular for new developers in the SPINE community that are not familiar with the deepest layers of the numerical core).

This also permits SPIS to load only the needed packages. This is obtained through the use of OSGi bundles, in the same manner that each major functionality of SPIS-UI are packaged in a separate bundle (i.e. in a separate java file). The details of the installation, setup and use of these packages is given in the next section.

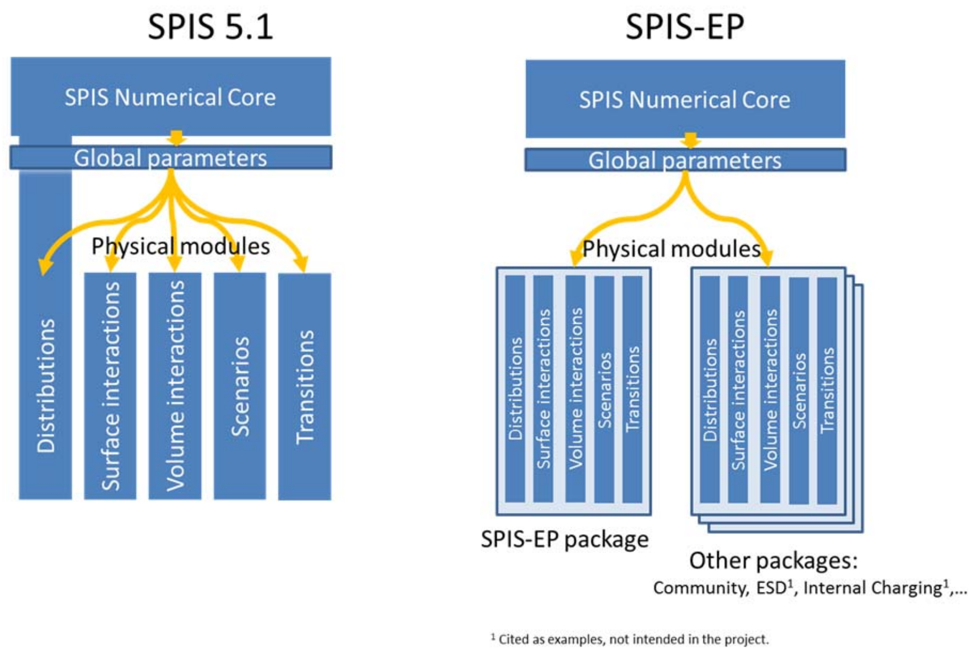


Figure 32: Block diagram of SPIS-NUM, separating the numerical core and the physical modules that are dynamically loaded on User's request. In SPIS 5.1, some parts of the definition of the distributions are hard-coded in the core of SPIS (Poisson solver,…).

4.3. Plugins architecture

A plugin is a group of java classes that is independent from the core in the sense that SPIS-NUM core can run and perform a simple simulation without the presence of the plugin. It means that SPIS-NUM core do not import or make any reference to any of the plugin classes. SPIS-NUM Core knowledge of the plugins only occurs at runtime when the package manager Felix loads the different all the SPIS components [SCIENCE]. In order to be recognized as a plugin, the SPIS-NUM plugin, the plugin must follow a precise naming and architectural convention. In particular, all the packages and the classes in the bundles must be contained in a package name: spis.bundles.*bundleName*, where *bundleName* is the name of the bundle. In order to avoid confusions, it is required that bundles that are not part of the community distribution be named as spis.bundles.*compagnyName-bundleName*, where *companyName* is the name of the company (or the person) that developed and/or maintain the bundle. Moreover, Maven's Symbolic name must be org-spis-num-bundles-*compagnyName-bundleName*. To do so, the bundle's pom.xml must contain:

```
<plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
        <extensions>true</extensions>
        <configuration>
                <instructions>
                        <Bundle-SymbolicName>org-num-bundles-compagnyName-bundleName <Bundle-SymbolicName>
                        <Export-Package>
                                spis.bundles.compagnyName-bundleName.*
                        </Export-Package>
                </instructions>
        </configuration>
</plugin>
```

SPIS-NUM core, as well as other plugins can be declared as dependencies if necessary.

Once Felix loaded all bundles, SPIS-NUM Core search across the loaded bundles those whose name match the convention and store them in a list. At the same time, SPIS-NUM Core performs an invocation of the optional spis.bundles.*compagnyName-bundleName*.osgi.BundleInit.init() method. This method is executed before at SPIS start, before any simulation or data be loaded. It is thus exclusively used to define static data (exemple: it is used by the dust plugin to add the dust particles type to SPIS-NUM default particle list).

The architecture and the naming conventions in the plugin must follow that of SPIS-NUM. As an exemple, a volume distribution must be stored in the spis.bundles.*compagnyName-bundleName*.Vol.VolDistrib package and derive from the spis.Vol.VolDistrib.VolDistribWithIO class. This allows SPIS-NUM core to retrieve the classes in the bundles. All classes that can be loaded from a class name defined by the user as a SPIS local or global parameter can be instantiated by SPIS-NUM core, given that they respect the naming convention and that the derive from the correct SPIS-NUM Core classes.

In addition, there are special classes and methods that can be specified at that will be called by SPIS-NUM at specific time if they exist (all are optional):

We already presented the *osgi.BundleInit.init()* method earlier in this section. This method must only be used to add some static properties to SPIS-NUM.

The *Top.Simulation.BundleSimulationInit* class is instantiated during the simulation initialisation process, just before the loading of the surface interactors. It derives from *BundleAPI_SimulationInit* and contains a single public method *initSource(boolean fixedDt).*

The *Util.MonitorBundleFieldMonitor* class derives from *BundleAPI_FieldMonitor* and is used to define a field monitor specific to the plugin.

The *Util.Instrument.BundleInstrumentsCatalogue* class is used to define the instruments that are implemented in the plugin and derives from *BundleAPI_InstrumentsCatalogue*.

The *Util.Instrument.BundleDefaultInstruments* class is used to define the instruments that must be set up by default at the start of any simulation when the plugin is loaded (even is no classes from the plugins are specifically declared to be used by the user). It derives from *BundleAPI_DefaultInstruments*.

4.4. Workflow modifications

SPIS 5.1 is based on a main (time) loop that executes sequentially three elements: The plasma evolution, the surface interactions and the spacecraft circuit evolution. This assumes that these three elements can evolve following the same clocking. This is usually true: the plasma evolution is itself composed of a sub-loop that allows providing a rather slowly varying plasma condition that matches the spacecraft circuit evolution without introducing abrupt variations that would lead to numerical instabilities (this may require the user to be experienced enough to parameterize correctly the different loop steps). The surface interaction itself is rather passive from this point of view and only serves as an interface between plasma and spacecraft (although secondary emission may lead to a strong coupling between plasma and surface interactions).

A new problem arises with electric propulsion: the recollected current may directly change the huge current emitted from the cathode which itself directly impact the recollected current. This retroaction is faster and much larger than the plasma response and thus a sub-loop should be created between the current collection/emission and the spacecraft circuit evolution. This can be done using the current SPIS architecture by making sub-loops on the circuit solver in which the current scalers (function that predicts the current evolution as a function of the spacecraft potential evolution) are recomputed.

Another modification that is needed by the developments of SPIS-EP is the coupling between the volume interactions and the particle motion. As for SPIS 5.1, the plasma loop is composed of three elements: The population evolution (particle motion for PIC distributions), the volume interactions and the potential evolution. In order to have an accurate computation of the processes related to particle collisions (in particular CEX), it is necessary to compute the volume interaction in each mesh cell crossed by the particles (volume interaction and population evolution must be computed at the same time). This leads to a modification of the general SPIS workflow.

Figure 33 shows a sketch of the existing (left) and SPIS-EP (right) workflows.

The proposed modification stays consistent with the current simulation scheme and will not introduce problems of backward compatibility with the existing SPIS simulations.

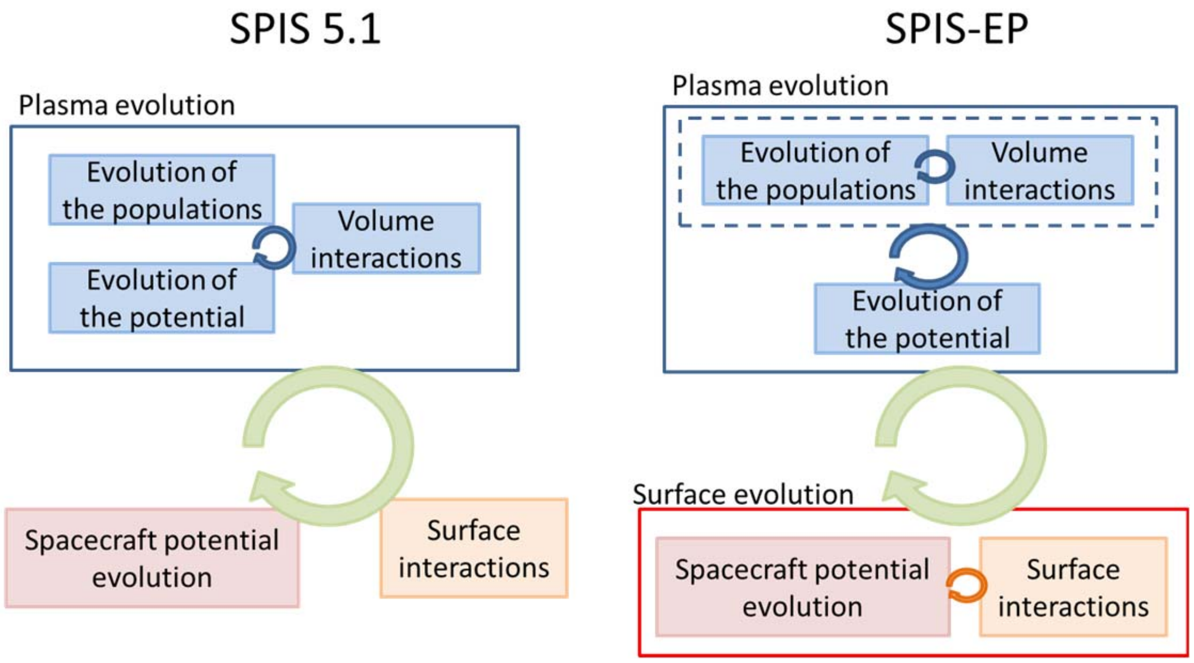

SPIS 5.1
Plasma evolution
Evolution of the populations
Volume interactions
Evolution of the potential
Spacecraft potential evolution
Surface interactions
SPIS-EP
Plasma evolution
Evolution of the populations
Volume interactions
Evolution of the potential
Surface evolution
Spacecraft potential evolution
Surface interactions

Figure 33: Schematic flow chart of the SPIS software. As for SPIS version 5.1, there is a main loop on three elements: Spacecraft circuit evolution, Surface interaction and plasma evolution. The plasma evolution is itself composed of a loop on three elements: population evolution, volume interactions and plasma potential evolution. The better performances required by SPIS-EP implies to deeper connection between the spacecraft evolution and the surface interaction on one hand and between the population evolution and the volume interactions on another hand.

4.4.1. *Modification of the circuit solver*

The current balance is particularly difficult to compute for an electric propulsion simulation. Thus, the Poisson and circuit solvers were refactored.
A wider and more accurate use of the current scalers is made in the circuit solver. The current scalers predict the current variation that arises from a potential variation. Because there is a closed electrical circuit between the cathode and the spacecraft surface (closed on one side by the spacecraft wiring and on the other by the plasma), there is a strong retroaction of the cathode and surface potential variations on the current circulating in this circuit. New current scalers were thus designed to accurately model this current circuit. This also includes the computation of the cathode net current as a function of the cathode potential.

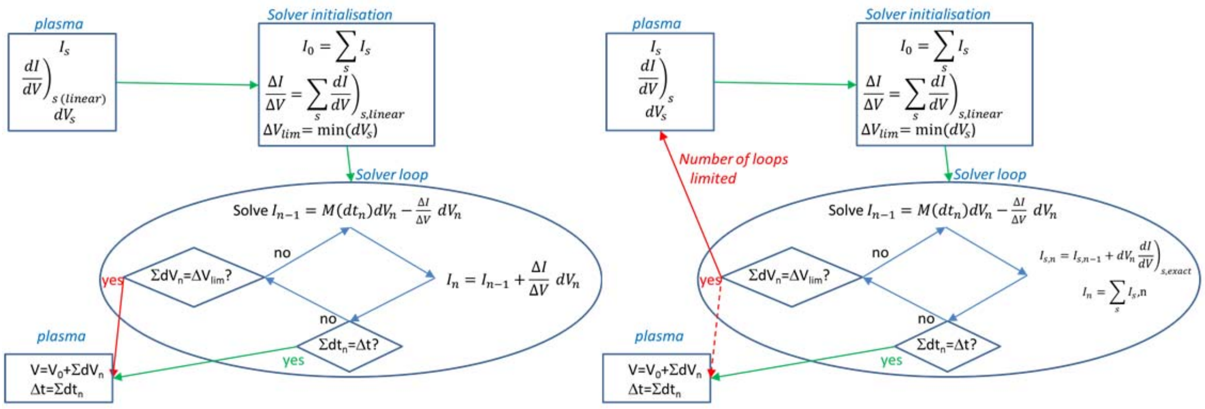

plasma
I_s
$\left.\frac{dI}{dV}\right)_{s\,(linear)}$
dV_s
Solver initialisation
$I_0 = \sum_s I_s$
$\frac{\Delta I}{\Delta V} = \sum_s \left.\frac{dI}{dV}\right)_{s,linear}$
$\Delta V_{lim} = \min(dV_s)$
Solver loop
Solve $I_{n-1} = M(dt_n)dV_n - \frac{\Delta I}{\Delta V}\,dV_n$
no
yes
ΣdV$_n$=ΔV$_{lim}$?
$I_n = I_{n-1} + \frac{\Delta I}{\Delta V}\,dV_n$
no
Σdt$_n$=Δt?
yes
plasma
V=V$_0$+ΣdV$_n$
Δt=Σdt$_n$
plasma
I_s
$\left.\frac{dI}{dV}\right)_s$
dV_s
Number of loops limited
Solver initialisation
$I_0 = \sum_s I_s$
$\frac{\Delta I}{\Delta V} = \sum_s \left.\frac{dI}{dV}\right)_{s,linear}$
$\Delta V_{lim} = \min(dV_s)$
Solver loop
Solve $I_{n-1} = M(dt_n)dV_n - \frac{\Delta I}{\Delta V}\,dV_n$
no
yes
ΣdV$_n$=ΔV$_{lim}$?
$I_{s,n} = I_{s,n-1} + dV_n \left.\frac{dI}{dV}\right)_{s,exact}$
$I_n = \sum_s I_{s,n}$
no
Σdt$_n$=Δt?
yes
plasma
V=V$_0$+ΣdV$_n$
Δt=Σdt$_n$

Figure 34: (right) circuit solver algorithm in SPIS 5.1, (left) circuit solver algorithm in SPIS-EP. A better computation of the current evolution vs potential improves the accuracy of the solution, while external loops in which the dI/dV and maximum potential variation are re-evaluated allows for larger time steps.

The current circuit solver had always sub-iterated to solve the circuit (at least twice), until it reached a threshold of potential variation at which the (linear) current scaler prediction loose it accuracy. If this threshold is reached over duration shorter than the current time step, the time steps decreases and the simulation will run for a longer time. To improve this, the current scalers are recomputed each time the threshold is reached, thus allowing for longer time steps. The number of time the current scaler can be recomputed is determined automatically by SPIS based on a timing process (if the circuit solution is longer than the other processes there are no re-computation, if it is fast it can be recomputed up to three times).

To have faster circuit solver computation, the task of computing the dI/dV matrix has been parallelized. Also, the estimate of the current variation between two iterations was improved. Formerly, a global (i.e. summed over all populations) dI/dV matrix was computed and the current variation was computed in a linear way as dI = (dI/dV).dV. This is acceptable as long as dI << I for all species independently. This condition was in fact rarely verified and led to strong divergences in the case of an electric propulsion simulation. Thus, the current is now updated in by each current scaler for the current they model in an analytical way (not necessarily linear). In particular this prohibits having |dI| > |I| in which case the current for a given species could change sign (which is unphysical).

4.4.2. *Smoothing of the current taking into account the statistical noise*

SPIS v5 already provided the mean to smooth the currents on surfaces before solving the equivalent electrical circuit of the spacecraft. With SPIS-EP, this smoothing also uses the information of the statistical noise (for PIC populations) to perform the smoothing. The result of the noise-dependent method is not necessarily smoother that the old smoothing method, as it favours the surface elements with the lowest noise, which may be in minority for very noisy simulations.

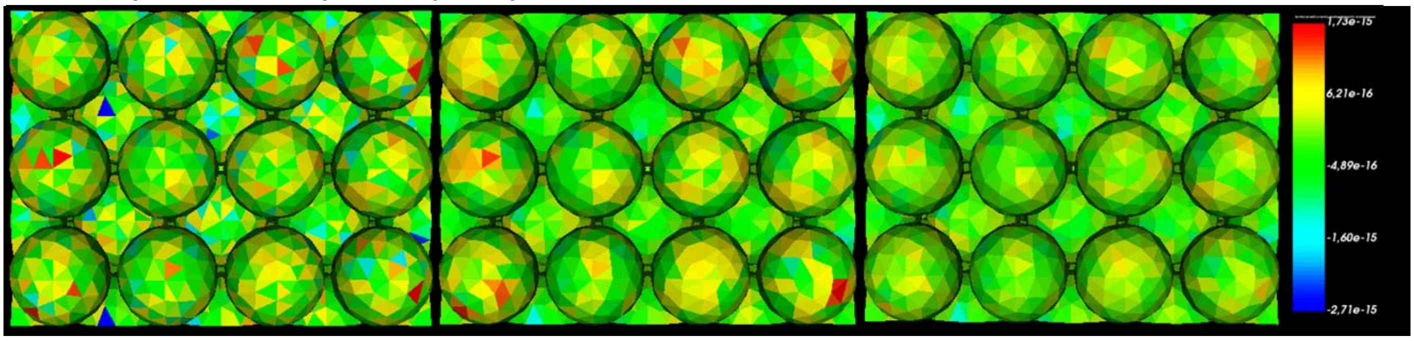


Figure 35: From left to right, current collected on spheres before the smoothing, with the noise dependent smoothing, with the old smoothing method. The noise-dependent method better preserve the current difference due to photo-emission than the old smoothing method

4.4.3. *Modification of the Poisson solver*

The electron source term in the Poisson equation is implicit, meaning that the electric potential and the electron densities are computed at the same time. In order to solve this non-linear equation, SPIS 5.1 Poisson solver included expressions of the density variation against potential variation for all existing distributions. This was not satisfactory for several reasons: (1) It put particle distribution dependent data in the Poisson solver, meaning that modification in the distribution must be manually reported in the Poisson solver, (2) it prohibits to easily add new distributions, as new switches have to be implemented in the Poisson solver to define these distributions. Thus, the implementation of the computation of the distribution dependent terms is now deported to the distribution classes. The methods to implement are now gathered in the *ImplicitableVolDistrib* interface. All distributions that can be computed implicitly must implement this interface. The code cleaning related to this change also allowed to suppress the limitation of the number of implicit distribution. Up to SPIS 5.1, only the two first electron distributions could be implicit, whereas the number is implicit distributions is now unlimited.
The cathode electron population is particular in the sense that the reference densities and temperatures are not uniform and vary in time. Moreover, both can vary over several orders of magnitude across the simulation domain. In order to handle these constraints, the first guess of the Poisson solution has been improved by implementing in the implicit distributions the computation of the potential as a function of the density population and of its derivative. This allows having exact zero order approximations for each distribution that can be averaged and corrected using the derivative terms. Without this modification, the Poisson solver does not converge.

Because the cathode electron population reference density, temperature and potential are time dependant, the moments of the implicit distributions are now updated at the beginning of the Poisson solver to have a well time-centred equation to solve and to have consistent reference (surface) potentials between the Poisson boundary conditions and the cathode electron density.

4.4.4. Modification of the population evolutions and interactions

In order to better handle the particle evolution along their trajectory, the particle pusher and the volume interaction interface were reworked.

A new API was defined in the *VolInteractionAlongTrajectory* abstract class for volume interactions occurring along the particle trajectory. It permits to define for each interaction the acceleration applied to the particles in each cells (*computeAccelerationInCell* method). This method is called when the particles enter a cell (or when its motion starts). It also permits to define the interaction that occurred while crossing the cell (*computeInteractionInCell* method). This method is called when the particles exit the cell (or when the motion stops). In order to accelerate the above method computations, the interaction is initialized once at the beginning of the particle pushing phase (*initInteractionForTimeStep* method).

In SPIS 5.1 parallelisation of the particle motion was performed by invoking N instances of the *ComplexPusher* class. Each of these instances was responsible for moving one particle over N. This architecture of full duplication of the pusher was not adapted because the new interaction interface needs to be initialized only once at each time step. Instead, multiple instances of the *ComplexPusher* were replaced by a single instance of the new *PICPusher*. This pusher performs a single initialisation step common to all particles, and then invokes multiple instances of the *TaskOfPushingParticleBatch* object. Contrary to the old *ComplexPusher*, this object divides the particle list in several batches (usually much more than the number of threads) and starts several pushing threads. Each thread asks for and obtains a batch of particles. Once they finished their batch, they ask for the next one until all particles are pushed. Thus, the threads do not necessarily push the same number of particles, but rather share a more balanced computational load (if a thread computes the motion of particles that cross more tetrahedrons, it will computes less particle motions). This improves the efficiency of the particle pusher.

Several methods exist for computing the particle motion across a tetrahedron, depending on the field uniformity and on the magnetic field. Each of this method is now implemented in a separate class which inherits from the *CrossTetrahedron* abstract class which defines a common API. This improves the readability of the code and opens the possibility for a future modularity of the particle pusher.

4.5. Improved particle pusher

Many electric thruster technologies rely on particular magnetic topologies. Previous versions of SPIS implemented two algorithms that push the particles in presence of a magnetic field: One of them approximates the motion by a dichotomy algorithm, while the second one uses a Runge-Kutta algorithm. The latest is precise, but numerically demanding; while the first is fastest but lacks precision. A new algorithm that is both faster and more precise than the previous algorithms is implemented in SPIS-EP.

It is a two stage algorithm: if the cell size is significantly larger than the Larmor radius, the guiding centre motion is computed using the same algorithm than the pusher without magnetic field (by solving in barycentric coordinates the 2nd order equation of motion $x+v\ \delta t + dv/dt\ \delta t^2/2 = 0$). Because amplitude of the particle motion around the guiding centre trajectory is the Larmor radius, the guiding centre motion is computed up to a distance of one Larmor radius from the cell surface.
Then, the particle motion is computed using an algorithm similar to that used to compute the guiding centre motion but using a fourth order equation of motion in barycentric coordinates.

$$\left(\frac{E}{|B|}+v\wedge b\right)\wedge b\wedge b\frac{\omega_c^3}{24}\delta t^4+\left(\frac{E}{|B|}+v\wedge b\right)\wedge b\frac{\omega_c^2}{6}\delta t^3+\left(\frac{E}{|B|}+v\wedge b\right)\frac{\omega_c}{2}\delta t^2+v\delta t+x=0$$

The trajectory computed this way is accurate enough over time steps such that ($\omega_c\delta t < 0.2$). If the motion duration obtained from the equation of motion is larger, the guiding centre (and the particle position around it) is pushed over this time step (up to $\omega_c\ \delta t = 1$). If the particle ends up out of the cell, the motion is limited to two-third of the time step. The algorithm is then re-iterated by solving a new fourth order equation of motion.

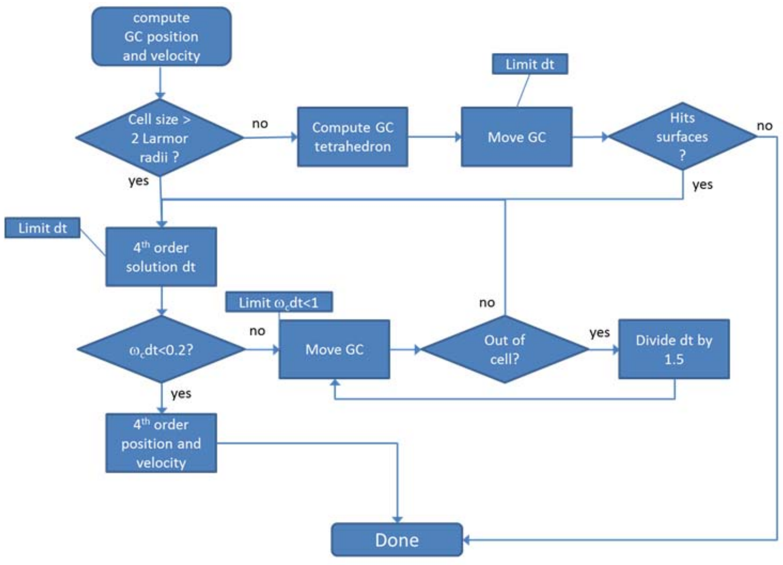


Figure 36: Algorithm of the magnetic field pusher.

4.6. Thruster axis computation

To identify several thrusters surfaces defined in one spacecraft group, the following algorithm is used:

Input: the mesh where is defined the design of the geometry.

Output: the list of list of triangle mesh elements. Each list of triangle mesh elements is a continuous surface.

```
• While all the face mesh elements of the thrusters have not been considered
    o Consider one random mesh element (triangle) from the thrusters mesh mask and put it in the
      list of neighbours faces to treat
        ▪ While there are untreated neighbours faces
    o End While
• End while
```

For each continuous surface, thanks to the following algorithm, an axis, defined by a position and a direction, is computed:

Input: list of triangle mesh elements defining a plane continuous surface.
Output: the position and the direction of the thruster axis.

- Compute the normal of the first triangle to know the direction of the thrusters axis and orient it with the tetrahedral connected to this mesh element
- Compute the barycentre C of the continuous surface using the following formula to weight the barycentre of each cell of the surface by its area

$$C = \frac{\sum_{i=0}^{N-1} A_i R_i}{\sum_{i=0}^{N-1} A_i} \text{ with } \begin{array}{l} R_i = \frac{(a_i+b_i+c_i)}{3} \\ A_i = \|(b_i - a_i) \otimes (c_i - a_i)\| \end{array}$$

Where ai, bi and ci are the three vertices of the current triangle.

These calculations have been validated using several approaches. First a unit test case has been implemented to verify the validity of the algorithm on a simple case. A hexagon mesh is built in the test case following the coordinates shown on Figure 37. Then the thruster axis for this surface is computed. We obtain the correct centre (3, 3, 0) and the correct direction (0, 0, 1).

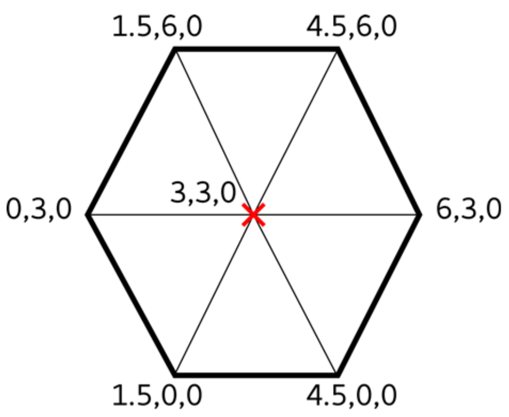
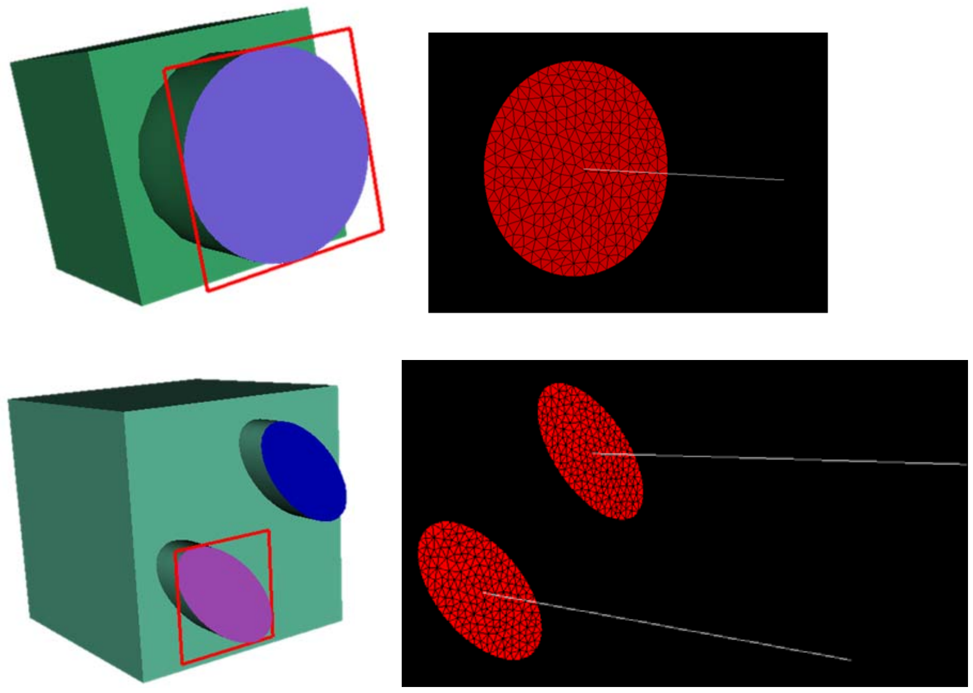


Figure 37 Schema of the "Hand-made" mesh for the unit test case

Several tests have been performed with different shapes and with uniform and non-uniform meshing. The results are shown on the Figure 38. At the left the geometry is displayed in the group editor (spacecraft body in green, device surface in purple) and on the right we display the computed axis by displaying it in Cassandra with the device surface (in red).



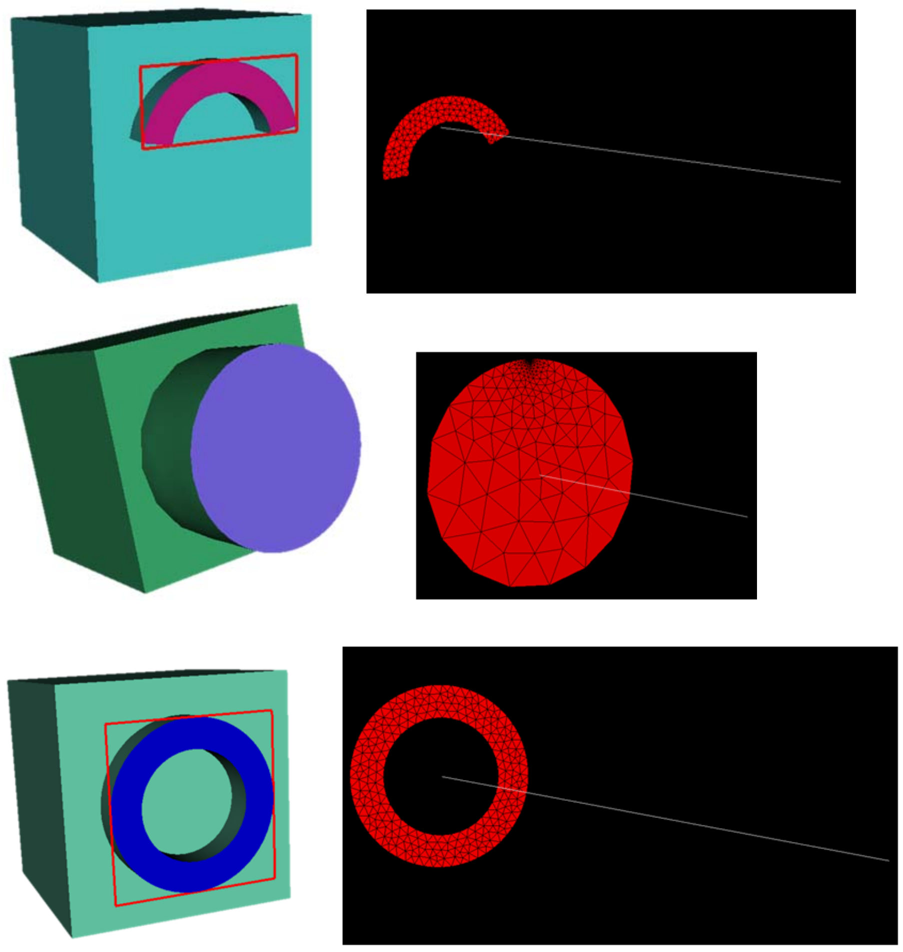


Figure 38 Examples of thruster surfaces (left) and the resulting thruster axis (in white on the right)

The computed device axis is then transmitted to the numerical kernel using a Global Parameter with an array of *FloatScalTables*. Each *FloatScalTable* represents a device axis with the following values:

- *id*: id of the device surface in the *DeviceTypeId* data field.
- *x*: x coordinate of the start point of the device axis.
- *y*: y coordinate of the start point of the device axis.
- *z*: z coordinate of the start point of the device axis.
- *dx*: x value of the direction of the device axis.
- *dy*: y value of the direction of the device axis.
- *dz*: z value of the direction of the device axis.
-

4.7. View factors

In order to accelerate computation related to neutral particles, it is possible to use view factors. The view factors are implemented as a component of the 3D unstructured volume mesh in the *ThreeDUnstructVolMeshWithViewFactor* class which is part of the core of SPIS-NUM. The first step in the view factor calculation is the computation of the visibility of each pairs of nodes in the computational volume. It is done once and for all the first time a distribution projection between to meshes using view factors is performed. The visibilities are computed by searching for each pairs of nodes if a surface sits in between. The visibility computation is multi-threaded and based on a tree algorithm. The result is binary and thus stored in a binary symmetric matrix for memory usage efficiency. This forbids taking into account semi-transparent. Thus, the view factor usage is deactivated if the volume mesh contains grid meshes.

The view factors themselves are re-computed for each projection of a surface or volume distribution on a surface or volume mesh. Because the view factors are distribution dependant and because these distributions are neither unique nor constant, the view factors cannot be stored. Their storage would also have posed memory usage difficulties.

In order to compute the view factors, the surface and volume distributions must implement the getMomentsInDirection(**float**[] direction, **float**[] surface, **int** index, **int** node, **float**[] results) method that compute the density fraction, the flux, the three velocity components and the temperature of the distribution in the direction of the *dir* vector over the oriented *surface* issued from the *node* node of the surface or volume element *index*.

These distributions must also implement the buildDistribFromMoments(**ScalSurfField** density, **ScalSurfField** flux, **VectSurfField** velocity, **ScalSurfField** temperature) or buildDistribFromMoments(**ScalVolField** density, **ScalVolField** flux, **VectVolField** velocity, **ScalVolField** temperature). These methods create the projected surface or volume distribution from the projected moments.

5. MODELS

5.1. Cathodes

5.1.1. Cathode electron modelling

The cathode electron models in SPIS all derivate from a model based on the plume quasi-neutrality with local Poisson-Boltzmann perturbations, implemented in the *GenericCathodeElectronDistribution* abstract class.
This model is based on the quasi neutral approximation in which the quasi-neutral electron density equals the ion density:

(1) $n_{e,qn}=n_{i.}$

Since, there are no net charges, the electric potential balances the thermal pressure forces:

(2) $q\, dV_{qn} = k/n\, dn_{e,qn}T_{e,qn}$

Note that the electron temperature $T_{e,qn}$ can depend on the quasi-neutral electron density. Typically, such dependence is given by a polytropic index. We note hereafter f the function giving the temperature from the quasi-neutral electron density.

(3) $T_{e,qn} =f(n_{e,qn})$

Such a quasi-neutral model is already implemented in SPIS in the AISEPS package. But this model suffers from the fact that the potential is given by the pressure gradient rather than by the Poisson equation. This leads to some difficulties as there is a minimum potential allowed by the model, depending on the temperature law. This prohibits having consistent volume and surface potentials. Physically, this translates the fact that the quasi-neutral approximation prohibits the existence of non-neutral sheath near the surfaces.
In the electric propulsion model, we compute the temperature from the ions, using a smoothing function S:

(4) $T_{e,qn} =S\, f(n_i)$

The electron quasi-neutral density is obtained from equations (3) and (4):

(5) $n_{e,qn}= f^{-1}\, S\, f(n_i) \sim n_i$

We then allow the electron density to deviate locally from the quasi neutral approximation:

(6) $n_e = n_{e,qn} + \delta n_e$

Then, solving Poisson equation one obtains:

(7) $e(n_e - n_{e,qn})/e_0 = \Delta V \sim (V-V_{qn}-V_{cathode})/\lambda_D^2$

(8) $n_e/n_{e,qn} = 1+ (n_e/n_{e,qn})e\, (V-V_{qn}-V_{cathode})/kT_{e,qn}$

(9) $n_e = n_{e,qn} \exp(e(V-V_{qn}-V_{cathode})/kT_{e,qn})$

In this model, $n_{e,qn}$, V_{qn} and $T_{e,qn}$ are function of the ion density exclusively, and not on the electron density. Their exact expression depends on the selected model. Some of these models are implemented in SPIS and are describe in the next sections. $V_{cathode}$ is the potential of the electrical node to which the cathode is connected. This later potential is computed dynamically by the circuit solver.

The mathematical form of the above equation is exactly the same as that of the usual Poisson-Boltzmann distribution, except that the reference density temperature and potential of the distribution are local and not global and that they may vary in time. Except for the additional constraints it imposes to the Poisson solver in term of preconditioning to converge (see section 4.4.1), this equation can be solve using the exact same solver than the usual Poisson equation.

Contrary to the pure quasi-neutral case, it is possible to simulate non-neutral sheaths close to polarized surfaces with this model. Another interest of this model is that the electron density equals the ions density tends naturally towards 0, without requiring infinite potentials. Thus, it becomes possible to couple the cathode electrons with the environment.

The cathode current is computed from two sources:

(1) The current of cathode electrons collected by the spacecraft.

(2) The current of electrons needed to neutralize the plume

The latest is computed as the larger of two currents: the current corresponding to the net thruster ion charge creation in the volume and the net electron current at infinity. The first one dominates at the start of the thruster, before the plume reaches the simulation boundaries, while the second dominates during the rest of the simulation. The current at infinity is computed from the current on the boundary corrected from the cooling and the boundary potential effects:

(10) $$J_\infty = J_{boundary}\sqrt{\frac{T_\infty - m\in(q\cancel{V},0)}{T_{boundary}}}\,exp\left(\frac{-max(q\cancel{V},0)}{T_{boundary}}\right) \text{ with} \cancel{V} = V_{boundary} - V_{qn;boundary} + V_{qn;\infty}$$

ONERA THE FRENCH AEROSPACE LAB

ARTENUM, PARIS Science & Groupware

5.1.2. *Cathode electron models*

5.1.2.1. Adiabatic model

The adiabatic model is – after the isothermal model – the simplest polytropic model that is used to close the fluid equation. However, the usual model predicts (1) a constant polytropic index that does not correspond to the measured polytropic index variation in the plume and (2) a zero temperature for zero density, i.e. a zero temperature at infinity. However, this adiabatic model is computed for plasma at rest. For expending plasma, the model must be computed using a pressure term taking into account the plasma (ions) rest frame velocity:

(11)
$$P = \frac{n_e m_e}{\sqrt{\pi} v_{Te}} \int_0^\infty v(v - v_i)\, exp\left(\frac{-v^2}{v_{Te}^2}\right) dv$$

(12)
$$P = n_e \left(kT_e - \frac{m_e v_i v_{Te}}{\sqrt{\pi}}\right)$$

In this case, the adiabatic relation between the density and temperature is:

(13)
$$\frac{P}{V} dV = \frac{3}{2} n_e k dT_e$$
$$\left(1 - \frac{2v_i}{\sqrt{\pi} v_{Te}}\right) \frac{dn_e}{n_e} = \frac{3}{2} \frac{dT_e}{T_e}$$
$$\frac{dn_e}{n_e} = \frac{3}{2} \frac{dT_e}{T_e\left(1 - \frac{2v_i}{\sqrt{\pi} v_{Te}}\right)}$$

This allows determining an equivalent, temperature dependent polytropic index:

(14)
$$\gamma_{eq} = 1 + \frac{2}{3}\left(1 - \frac{2v_i}{\sqrt{\pi} v_{Te}}\right)$$
$$\gamma_{eq} = \begin{cases} \frac{5}{3} v_i \ll v_{Te} (adiabatic) \\ 1 v_{Te} \to v_i (isothermal) \end{cases}$$

Contrary to the usual adiabatic model, this model predicts a finite temperature at infinity, allowing computing the current balance using Eq. 10.

To select this model, the thruster property "*e-Model*" must be set to "*CathodeAdiabaticElectrons*".

5.1.2.2. *Polytropic model*

The polytropic law is the simplest law that the user is able to define for the electron distribution from a cathode. The equation solved is the same as in the previous case but the user is able to define a polytropic coefficient γ_{eq}that is different from the adiabatic one.The user can define:

1) a unique parameter whatever the plasma density
This coefficient is defined in the global parameters using the global parameter *variableTeGamma* that defines the gamma value.

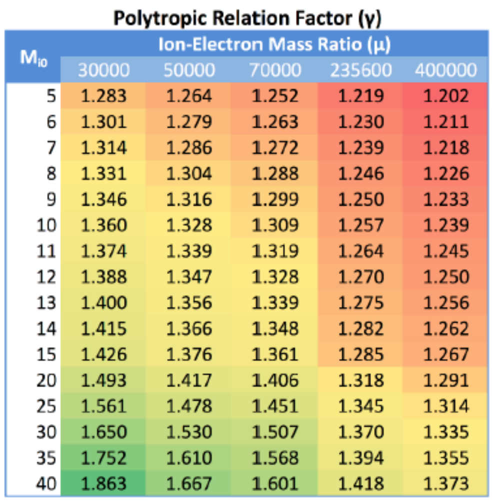
2) a step wise definition of gamma as a function of plasma density. In this case the user may define the global parameter *variableTeGamma* as a table. The user has to enter a two column table with:

- in the first column the value of gamma
- in the second column the minimum value of density for which we have this value of gamma (with decreasing densities)

To select this model, the thruster property "e-Model" must be set to "CathodePolytropicElectrons".

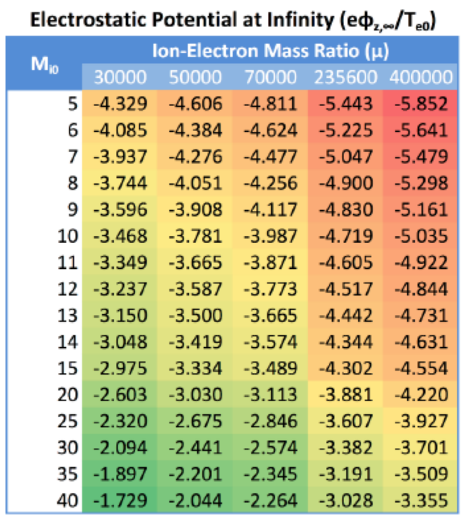
6.1.2.3 *University Carlos III Madrid model (default)*

The *UC3MCathodeElectronModel* cathode model derives from the polytropic model. A single value of gamma is used but its value is computed from the TN2 of the MODEX project:



Polytropic Relation Factor (γ)

M_{i0}	Ion-Electron Mass Ratio (μ)				
	30000	50000	70000	235600	400000
5	1.283	1.264	1.252	1.219	1.202
6	1.301	1.279	1.263	1.230	1.211
7	1.314	1.286	1.272	1.239	1.218
8	1.331	1.304	1.288	1.246	1.226
9	1.346	1.316	1.299	1.250	1.233
10	1.360	1.328	1.309	1.257	1.239
11	1.374	1.339	1.319	1.264	1.245
12	1.388	1.347	1.328	1.270	1.250
13	1.400	1.356	1.339	1.275	1.256
14	1.415	1.366	1.348	1.282	1.262
15	1.426	1.376	1.361	1.285	1.267
20	1.493	1.417	1.406	1.318	1.291
25	1.561	1.478	1.451	1.345	1.314
30	1.650	1.530	1.507	1.370	1.335
35	1.752	1.610	1.568	1.394	1.355
40	1.863	1.667	1.601	1.418	1.373



Electrostatic Potential at Infinity ($e\phi_{z,\infty}/T_{e0}$)

M_{i0}	Ion-Electron Mass Ratio (μ)				
	30000	50000	70000	235600	400000
5	-4.329	-4.606	-4.811	-5.443	-5.852
6	-4.085	-4.384	-4.624	-5.225	-5.641
7	-3.937	-4.276	-4.477	-5.047	-5.479
8	-3.744	-4.051	-4.256	-4.900	-5.298
9	-3.596	-3.908	-4.117	-4.830	-5.161
10	-3.468	-3.781	-3.987	-4.719	-5.035
11	-3.349	-3.665	-3.871	-4.605	-4.922
12	-3.237	-3.587	-3.773	-4.517	-4.844
13	-3.150	-3.500	-3.665	-4.442	-4.731
14	-3.048	-3.419	-3.574	-4.344	-4.631
15	-2.975	-3.334	-3.489	-4.302	-4.554
20	-2.603	-3.030	-3.113	-3.881	-4.220
25	-2.320	-2.675	-2.846	-3.607	-3.927
30	-2.094	-2.441	-2.574	-3.382	-3.701
35	-1.897	-2.201	-2.345	-3.191	-3.509
40	-1.729	-2.044	-2.264	-3.028	-3.355

Figure 39: Tables summarizing the polytropic factor and the electrostatic potential difference between the environment and the cathode for the University Carlos III Madrid model.

The gamma value is automatically computed for each thruster as a function of the Mach number of the ions coming from the thruster and the ion mass emitted by each thruster. The previous tables are imbedded in the model implementation and cannot be modified.

To select this model, the thruster property "*e-Model*" must be set to "*UC3MCathodeElectronModel*".

6.1.2.4 SPT100 model

The SPT100 cathode model derives from the polytropic model but using the possibility to define a tabulated function of gamma as a function of the plasma density. This table has been defined for a standard SPT100 thruster during the MODEX project and the function implemented fit the following plot:

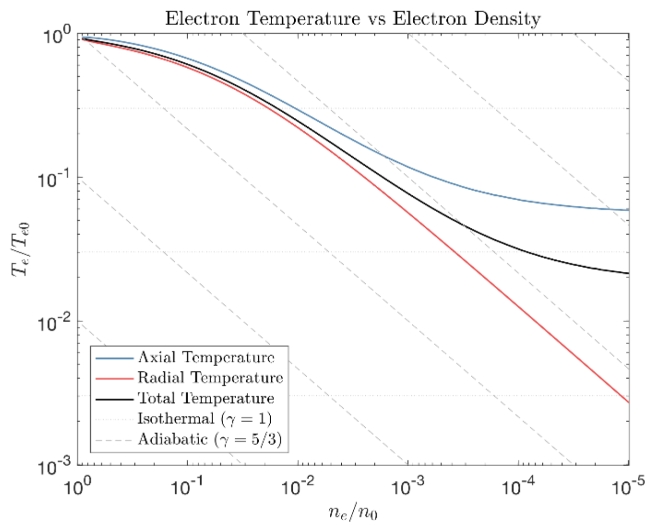


Figure 40: relation between the electron temperature and density in the SPT100 model.

To select this model, the thruster property "*e-Model*" must be set to "*SPT100CathodeElectronModel*".

5.2. **High voltage elements**

5.2.1. Model

A more detailed version of the model can be found in [HV], which also gives results of validation cases. The current collected by a charged surface is usually determined using an Orbital Motion Limited (OML) law. This kind of law allows computing an effective collection surface that corresponds to the limit after which all particle trajectories must necessarily cross the charged surface. The following sketch shows the trajectory for a particle with a trajectory tangent to the effective collection surface, which happens to be tangent to the charge surface too. This approximation is simple to compute for cylinder or spheres in free space, because of the symmetries: particles can arrive from all directions and the problem has a central symmetry in 2D (cylinder) or 3D (sphere).

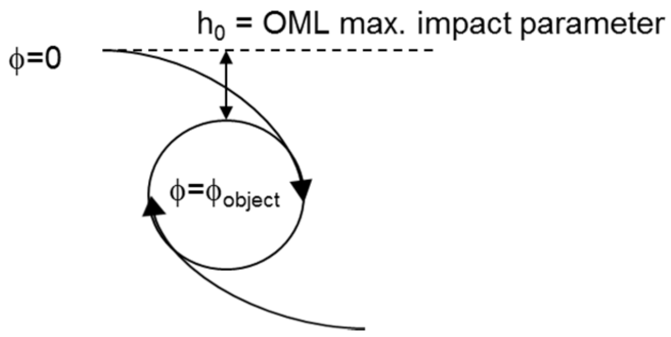


Figure 41: OML limit impact parameter in a cylindrical geometry.

But for an interconnect on a solar panel, only some directions of arrival are possible. We build a model based on OML, but instead of collecting particles arriving from all directions, we define an acceptance angle. This angle depends on the impact factor: for interconnect at the surface of the panel the acceptance angle is larger for smaller impact factors (it can be different for an interconnect buried in the gap between the cells.

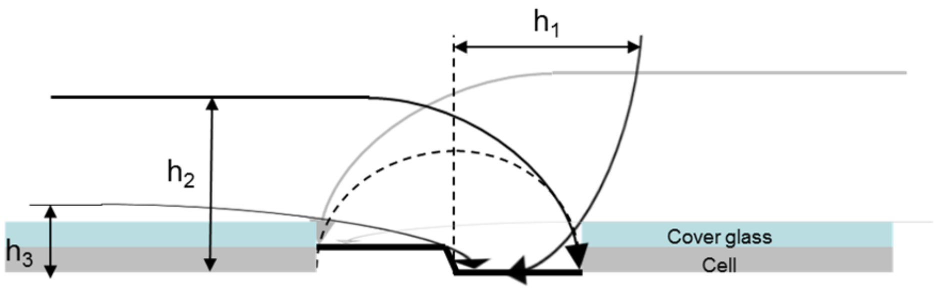


Figure 42: Sketch of the impact parameters associated with an interconnect geometry. Three impact parameter must be defined for each half of the interconnect.

The acceptance angle can be expressed as the following expression: $\Delta i = \Delta i_1 + \Delta i_2 + \Delta i_3 - \pi$

$$\Delta i_{n=1-2} = \begin{cases} \operatorname{acos}\left(\frac{h - h_n}{h_0 - h_n}\right); h_n < h < h_0 \\ \frac{\pi}{2}; 0 < h < h_n \end{cases} \qquad \Delta i_3 = \begin{cases} \frac{\pi}{2} - \operatorname{atan}\left(\frac{h_3 - h}{h_0}\right); 0 < h < h_3 \\ \frac{\pi}{2}; h_3 < h < h_0 \end{cases}$$

h_n / h_0 functions are sigmoid functions of the logarithm of $q\phi/m\,v^2$ with a vertex around $q\phi = mv^2$.

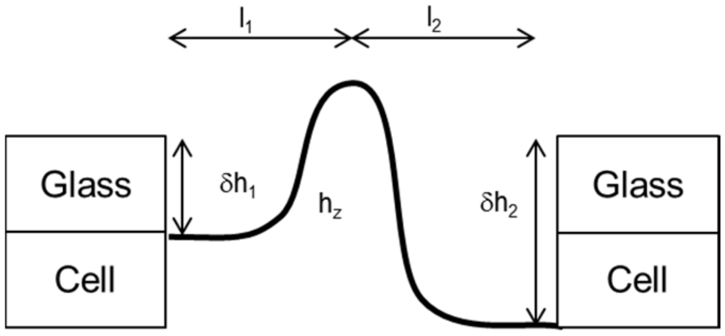


Figure 43: Sketch of the parameters defining the interconnect and cell gap geometry

Because of dissymmetry, both halves of the interconnect are treated separately.

$$h_1/h_0 = \begin{cases} 1; q\phi \ll mv^2 \\ \frac{(1 - sin\Phi)}{2cos\Phi}; q\phi \gg mv^2 \end{cases}$$
$$h_2/h_0 = \begin{cases} 0; q\phi \ll mv^2 \\ 1; q\phi \gg mv^2 \end{cases}$$
$$h_3/h_0 = tan\Phi$$

When the cover glass surface charges in a way that repels the species attracted by the interconnects, a Poisson-Boltzmann factor must be applied to the collected current.

6. OUTPUTS

6.1. Numerical parameters

It is possible to check the quality of the simulation by adding the “Numerics monitor” instrument to the simulation. This instrument displays in live the largest ratio between the Debye length and the cell size. If this ratio is too large and the electrons are computed as PIC particles, the simulation may diverge.

It also provides 2D map of the ratio between the Debye length and the cell size, as well as the map of the noise on the net current on the surfaces. These maps are created with a frequency N time smaller than the live monitoring, with N being the “2D measure periodicity” instrument parameter.

Finally the instrument is also providing 3D maps of the ratio between the Debye length and the cell size. These maps are created with a frequency N time smaller than the live monitoring, with N being the “3D measure periodicity” instrument parameter.

7.2 Default instruments

The list of default instrument was deemed to be too long for usual simulations. A shortened list has been selected for SPIS v6.0.4. A new instrument was defined which monitors the currents of each population and interactor over the whole spacecraft, replacing the individual current monitoring per node. The old list of default instruments can be selected by setting the “detailedIVOuput” Global Parameters to 1.

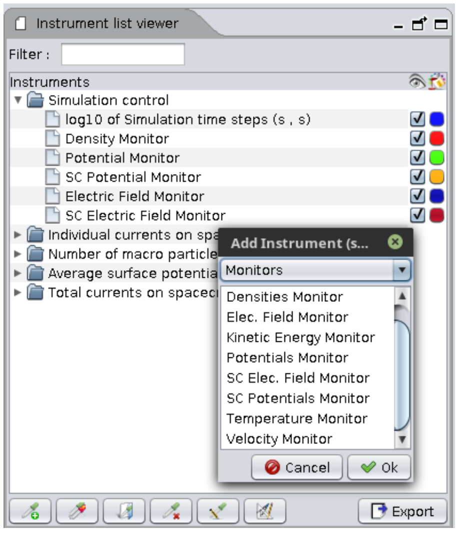


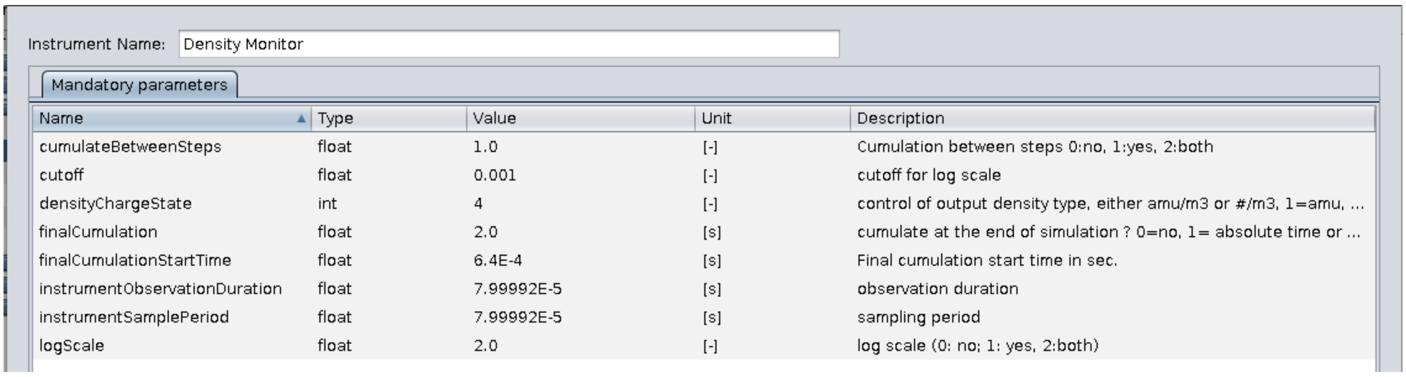
Figure 44: View of the standard default instrument list, with the list of monitors present by default. New monitors can be added as instruments under the “Monitors” category.

7.3 Simulation control

The parameters controlling the simulation convergence (*CSat*, *validityRenormalization*, *smootingI* and *smoothingPot*) can be modified by editing the “Simulation Control – time step” instrument, which replaces the “Time Step” monitor.

7.4 Monitors

In order to simplify the monitor settings and to provide a better control on the output, most of the quantity monitors now appear as instruments in the UI. The monitors appear in the “Simulation control” category. Although no live monitoring is associated to them, they can be edited to change to monitoring settings.


Instrument Name: Density Monitor

Mandatory parameters

Name	Type	Value	Unit	Description
cumulateBetweenSteps	float	1.0	[-]	Cumulation between steps 0:no, 1:yes, 2:both
cutoff	float	0.001	[-]	cutoff for log scale
densityChargeState	int	4	[-]	control of output density type, either amu/m3 or #/m3, 1=amu, ...
finalCumulation	float	2.0	[s]	cumulate at the end of simulation ? 0=no, 1= absolute time or ...
finalCumulationStartTime	float	6.4E-4	[s]	Final cumulation start time in sec.
instrumentObservationDuration	float	7.99992E-5	[s]	observation duration
instrumentSamplePeriod	float	7.99992E-5	[s]	sampling period
logScale	float	2.0	[-]	log scale (0: no; 1: yes, 2:both)

Figure 45: Instrument parameters controlling a typical monitor. It is possible to tune the cumulation and, sampling period for each monitor individually.

The monitors appearing by default are the same than is previous versions. The list of default monitor is controlled by the following Global Parameters:

- **detailedIVOutput**. If set to 1, the following instruments are set by defaults: “Spacecraft average differential potential”, “Average surface potential on nodes: time variation”, “Average surface potential on nodes: difference wrt to local ground” and “Individual current on nodes”. (default 0)

- **momentMonitoringFlag**. If set to 1, the following monitors are set by default: “Temperature monitor”, “Velocity monitor”, “Kinetic energy monitor”

Moreover additional monitors can be added in the course of the simulation the same way than instruments. The available monitors appear in the “Monitor” category.

It is possible to use the instrument “Perform measurement” feature to monitor quantities at any time.

6.5. Automated sampling along the thruster axis

It is possible to display results along the thruster axis. After selecting a data in the data mining tool, click on "*Show advanced operations*" and click on the "*Visualize selected data along device axis*" button as shown on Figure 46.

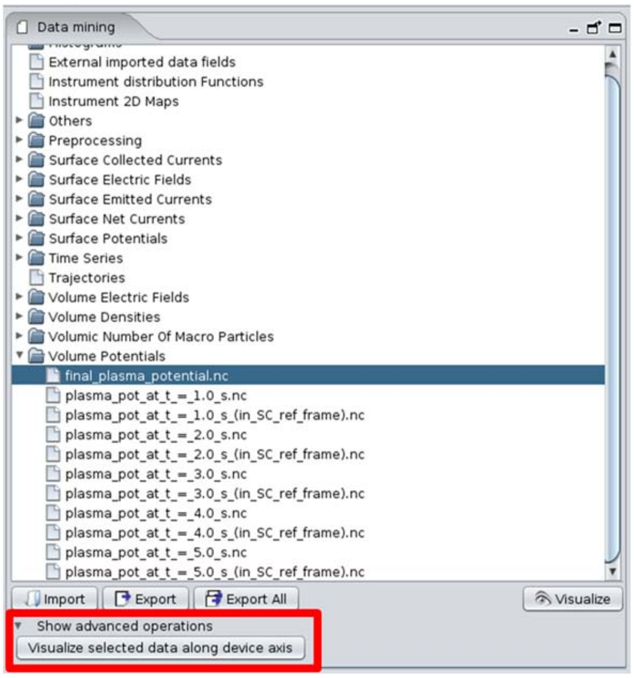

Data mining
External imported data fields
Instrument distribution Functions
Instrument 2D Maps
Others
Preprocessing
Surface Collected Currents
Surface Electric Fields
Surface Emitted Currents
Surface Net Currents
Surface Potentials
Time Series
Trajectories
Volume Electric Fields
Volume Densities
Volumic Number Of Macro Particles
Volume Potentials
final_plasma_potential.nc
plasma_pot_at_t_=_1.0_s.nc
plasma_pot_at_t_=_1.0_s_(in_SC_ref_frame).nc
plasma_pot_at_t_=_2.0_s.nc
plasma_pot_at_t_=_2.0_s_(in_SC_ref_frame).nc
plasma_pot_at_t_=_3.0_s.nc
plasma_pot_at_t_=_3.0_s_(in_SC_ref_frame).nc
plasma_pot_at_t_=_4.0_s.nc
plasma_pot_at_t_=_4.0_s_(in_SC_ref_frame).nc
plasma_pot_at_t_=_5.0_s.nc
plasma_pot_at_t_=_5.0_s_(in_SC_ref_frame).nc
Import
Export
Export All
Visualize
Show advanced operations
Visualize selected data along device axis

Figure 46 Advanced operations panel in Data Mining view

This tool uses the computed thruster axis as described in section 3.5.3. This axis is made of point (barycenter of the thruster surface) and a direction. In order to sample the selected data, an end point is needed. Thus the intersection of the direction and the limit of the computational volume give the final point for the probing line. Thus, the computed graph represents the data along the device axis from the thruster surface to the limit of the computational volume.

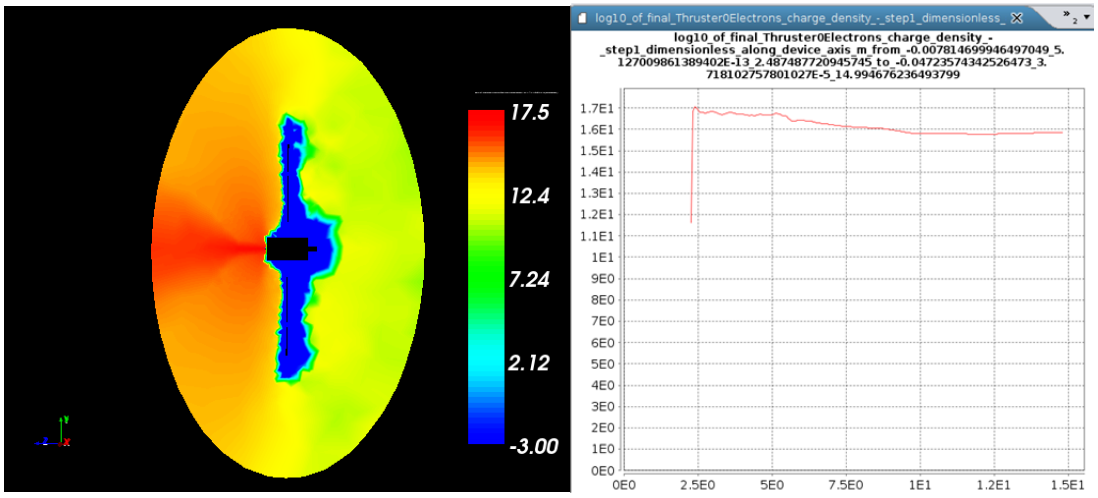


Figure 47: Example of automated monitoring along the thruster axis

6.6. Thruster diagnosis on a shell

It is possible to visualize the ion current through a shell located at a given distance from the thruster.
To do so, create a Thruster shell instrument in the instrument panel. It is then possible to define as interactor parameters the radius and the angular size of the shell, as well as the angular step.

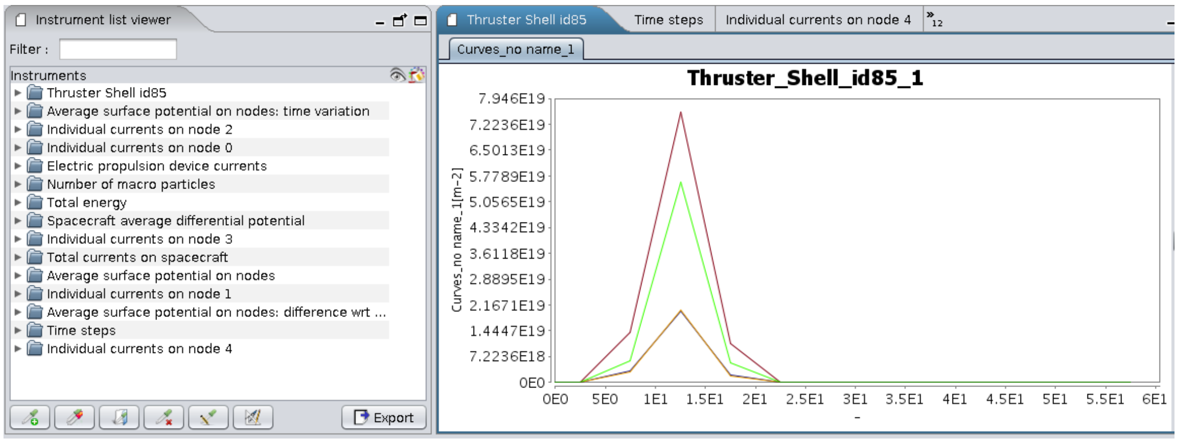

Instrument list viewer
Filter :
Instruments
- Thruster Shell id85
- Average surface potential on nodes: time variation
- Individual currents on node 2
- Individual currents on node 0
- Electric propulsion device currents
- Number of macro particles
- Total energy
- Spacecraft average differential potential
- Individual currents on node 3
- Total currents on spacecraft
- Average surface potential on nodes
- Individual currents on node 1
- Average surface potential on nodes: difference wrt ...
- Time steps
- Individual currents on node 4

Figure 48: Angular distribution of ion current at 30 cm of the exhaust of an annular thruster (before the convergence of the plume).

6.7. Improved trajectory monitor

SPIS v5 offered the possibility of following the trajectories of a given number of PIC particles from a given distribution [SUM]. However, the selection of the particle is arbitrary (actually the user has to provide a "period" and SPIS tracks one particle over this "period") and cannot be controlled by the user. Finally, SPIS v5 trajectory monitor only allows the tracking of PIC distributions.

In SPIS-EP, the tracking is extended to fluid distribution. In this case, the tracking occurs through test particles whose trajectory is computed using the electromagnetic field at the time of the monitoring (static field, contrary to the case of the PIC particles). Moreover, the trajectories of fluid distribution particles may be chosen by specifying a target surface: only trajectories hitting a surface whose "*sourceID*" (defined in the group editor) matches the "*particleTrajectoriesTargetID*" global parameter value are saved.

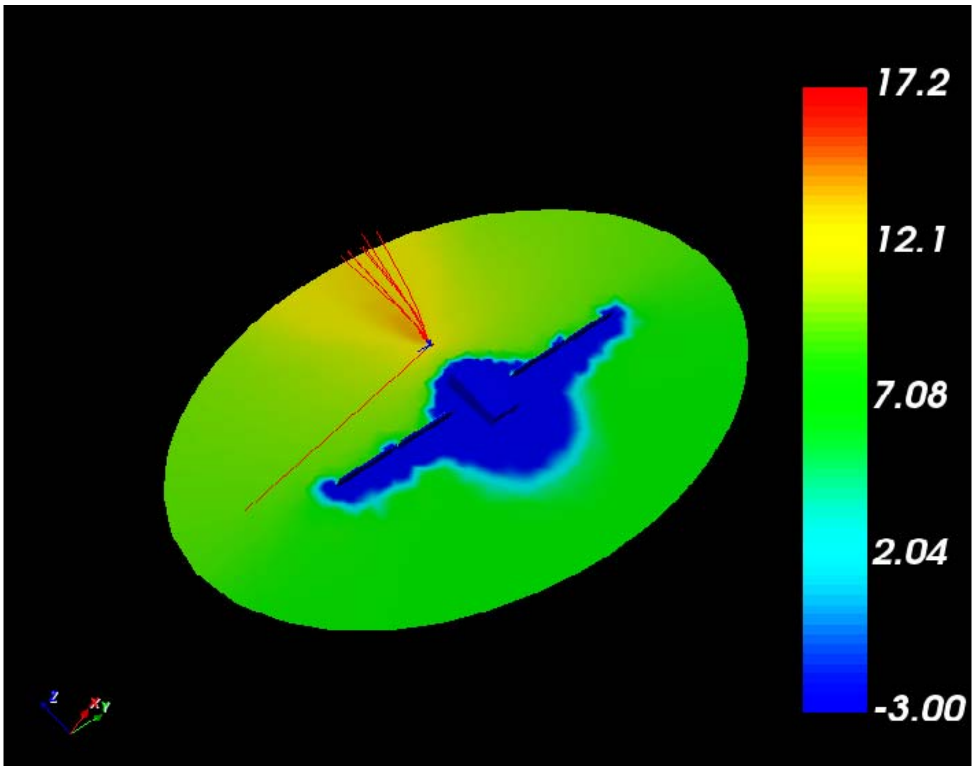


Figure 49: cathode electron trajectories superimposed to the cathode electron density (log scale)

This tracking also applies to cathode electrons. The tracking starts from the cathode position which is defined either by:

- The used defined position (see section 3.6)
- from the centre of the associated thruster if the cathode is associated to a thruster
- from the barycentre of the electrical node of the cathode is the cathode is stand-alone.

The cathode electrons are emitted as a Lambertian distribution (Maxwellian distribution emitted through a surface) following the cathode orientation define either by:

- The used defined orientation (see section 3.6)
- from the orientation of the associated thruster if the cathode is associated to a thruster
- from the average normal to the surfaces of the electrical node of the cathode is the cathode is stand-alone.

In order to start the cathode electron monitoring, one should proceed as for any other distribution [SUM]:
In the global parameter list, the flag "*nameOfThePopulation*TrajFlag" must be set to 1. In the cathode case, "*nameOfThePopulation*" is either "Thruster*X*Electrons" if the cathode is associated to the thruster X, or "cathodeElectrons" if the cathode is stand alone.

6.8. Improved Langmuir probe display

When creating a Langmuir probe, data now appear in the live monitoring. If the instrumentOutputLevel parameter is set to 0, the IV curves are displayed, if it is set to 1, the particle distribution function are displayed (see the description of this parameter in SPIS-Science documentation).

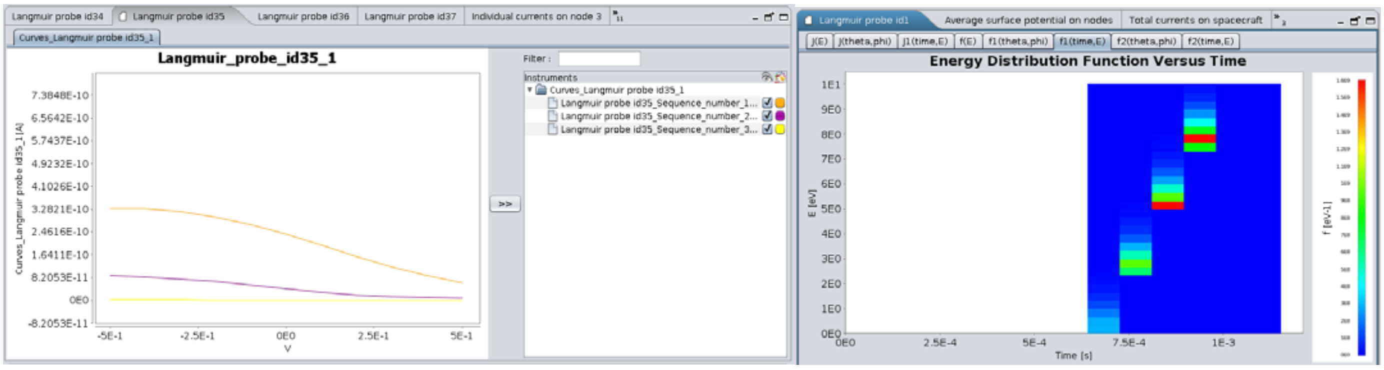


Figure 50: Live monitoring of Langmuir probes: mode 0 (left) showing the IV curve and mode 1 (right) showing the energy distribution.

7 COMPLIANCE MATRICES

7.1 User Requirements

7.1.1.1 Definition level

UR #	Title	Corresponding sections
UR-DL-01	Multi-Thruster definition	3.5
UR-DL-02	Thruster geometry and global characteristics definition	3.5
UR-DL-03	Distributions of the ions and neutral populations from the thruster	3.53.5.6.5
UR-DL-04	Distribution of droplets from the thruster	3.5.6.5
UR-DL-05	Population definition from measurements away from the thruster	3.5.8
UR-DL-06	Thruster characteristics definition from UI	3.4.4
UR-DL-07	Improved distribution function definition in the UI	3.5.6
UR-DL-08	Thruster magnetic fields	3.5.7
UR-DL-09	External magnetic fields	3.11
UR-DL-10	Switch of the active device from UI	3.5.9, 3.6.3
UR-DL-11	Control of the active device from UI	3.5.9, 3.6.3
UR-DL-12	Verification of the thruster parameters	3.5.1
UR-DL-13	Ground based environment	3.3

7.1.1.2 Simulation level

UR #	Title	Corresponding sections
UR-SL-01	Distribution of the electrons emitted by the cathode	3.6
UR-SL-02	Coupling with the environment	5.1
UR-SL-03	Electron thermalisation in electric thruster plume	5.1
UR-SL-04	Current emission from the cathode	5.1
UR-SL-05	Multi-cathode modelling	3.6.5
UR-SL-06	Plasma interaction on the power distribution unit	5.1
UR-SL-07	Current collection by unmeshed conductors	3.7, 5.2
UR-SL-08	Ion-neutral charge exchange in the plume	3.10
UR-SL-09	Ion-neutral elastic collisions in the plume	3.10
UR-SL-10	Electron impact ionization in the plume	3.10
UR-SL-11	Recombination in the plume	3.10
UR-SL-12	Fast neutral modelling	3.10
UR-SL-13	Erosion by electric thruster plumes	3.8
UR-SL-14	Yamamura erosion model	3.8.2
UR-SL-15	Sputtering by electric thruster plumes and associated back-flow	3.8.3, 3.8.4, 3.8.5
UR-SL-16	Neutral reflection and adsorption by surfaces	3.9
UR-SL-17	Neutral evaporation from surfaces	3.9
UR-SL-18	Erosion of contaminant from surfaces	3.9
UR-SL-19	Optimized time integration for contamination	3.9
UR-SL-20	Particle motion in the thruster magnetic field	3.11, 4.5
UR-SL-21	Switch-on/Switch-off of the plasma plume injection	3.5.9, 3.6.3

7.1.1.3 Output level

UR #	Title	Corresponding sections
UR-OL-01	Accuracy verification functionality	6.1
UR-OL-02	Probing lines in the thruster(s) axis	6.2
UR-OL-03	Plasma monitor on a shell	6.6
UR-OL-04	Smart particle trajectory monitor	6.7

7.2 **Software Requirements**

7.2.1.1 Definition Level

SR #	Title	Corresponding sections
SR-DL-01	Thruster source definition	3.5
SR-DL-02	Cathode definition	3.6
SR-DL-03	Thruster parameter control	3.5.9
SR-DL-04	Environment Definition	3.3

7.2.1.2 Simulation Level

SR #	Title	Corresponding sections
SR-SL-01	Thruster Source	3.5
SR-SL-02	Cathode electron distribution	5.1
SR-SL-03	Cathode modelling	5.1
SR-SL-04	Spacecraft circuit	3.6, 4.4.1, 4.4.2
SR-SL-05	Current collection by unmeshed elements	3.7.3, 5.2
SR-SL-06	Volume interaction along particle trajectory	3.10
SR-SL-07	Physico-chemical processes in the plume	3.10
SR-SL-08	Fast neutral modelling	3.10, 4.7
SR-SL-09	Thruster magnetic field	3.5.7, 3.11, 4.5
SR-SL-10	Erosion modelling	3.8
SR-SL-11	Sputtering from erosion	3.8.3, 3.8.4, 3.8.5
SR-SL-12	Contamination	3.9
SR-SL-13	Optimized contamination integration	3.9
SR-SL-14	Simulation progress and interactivity	3.5.9, 3.6.3
SR-SL-15	Noise Computation	4.4.2, 6.1

7.2.1.3 Output Level

SR #	Title	Corresponding sections
SR-OL-01	Accuracy verification functionality	6.1
SR-OL-02	Probing lines in the thruster(s) axis	6.2
SR-OL-03	Plasma monitor on a shell	6.6
SR-OL-04	Smart particle trajectory monitor	6.7

7.2.1.4 Package level

SR #	Title	Corresponding sections
SR-PL-01	User Interface modules for SPIS-EP	3.3, 3.5.3
SR-PL-02	Numerical kernel modules for SPIS-EP	3.1, 4.4
SR-PL-03	Targeted platforms	1
SP-PL-04	Testing	[FTR]