SPACECRAFT PLASMA INTERACTION SOFTWARE (SPIS): NUMERICAL SOLVERS – METHODS AND ARCHITECTURE.

Jean-François Roussel, François Rogier, Dominique Volpert,
ONERA, 2Av E Belin, 31055 Toulouse cedex 4, France,
Jean-Francois.Roussel@onecert.fr, tel +33 5 62 25 27 35, fax +33 5 62 25 25 69

Julien Forest
Artenum, Julien.Forest @artenum.com

Guillaume Rousseau
University Paris 7, Guillaume.Rousseau@obspm.fr

Alain Hilgers
ESA, Alain.Hilgers@esa.int

Abstract. The development of a new software for spacecraft plasma interactions modelling was started in Europe at the end of 2002. This Spacecraft Plasma Interaction Software (SPIS) is developed for and by SPINE community (Spacecraft Plasma Interaction Network in Europe) on an open source basis. The ESA contractors, ONERA and Artenum, are in charge of the development of SPIS framework and main numerical modules. In the framework of this collaborative development, SPINE community has several roles. It first expressed its requirements. It is now testing and validating the code on challenging test cases, and is starting to develop extra modules. The software framework is based on the integration or interfacing with available open source tools for CAD, 2D meshing, 3D meshing, GUI, post-processing and graphical display. The numerical routines allow the modelling of the plasma dynamics (kinetic or fluid, electrostatic with possible extension to electromagnetic) and its coupling with the spacecraft (equivalent circuit approach). The modelling of all types of environments and devices is supported (LEO/GEO/PEO..., EP/solar arrays...). The emphasis was put on the modularity of the code to allow the interoperability of the modules, through an object-oriented (OO) approach throughout the code (Java language, Jython script language). Requirements and design of SPIS code were defined during the first semester of 2003. The first version of the code was released in spring 2004, and extended versions later in 2004 and 2005. This paper mostly focuses on the numerical core of the solvers and their OO architecture.

Introduction

Charging is becoming more and more of a concern to spacecraft designers, as recent mission failures, partial or total, can attest. As the computing power and our understanding of charging physics were improving, computer simulation has also become more and more popular for assessing spacecraft charging. In the field of technology, important concerns are the triggering of electrostatic discharges (ESD) by high level charging, and in particular the integration of relatively new active devices such as electric thrusters and high voltage solar arrays. In the field of scientific missions, plasma or field measurements can be spoiled by even small potentials reached on spacecraft. Computer simulation may help avoid such low charging, or interpret the data in its presence (reverse engineering to get the undisturbed charged particle distribution function for example).

In spite of this need, no satisfactory code was available to Europeans to model spacecraft plasma interactions. Some proprietary codes did exist (see e.g. [Roussel 1998]) but were not available to all and were costly to maintain by a single company. Commercial codes were mostly American and not available to export. This led to the building up of a community of specialists, SPINE (Spacecraft Plasma Interaction Network in Europe), in 2000. Its members were from various origin, space technology; space science, plasma or computer science. Since they shared some common needs in spacecraft plasma interaction modelling, they tried to join their efforts to build a simulation code.

A first effort led to PicUp3D code, mostly written in the framework of J. Forest's PhD [Forest 2005], which also had a few other contributors and a very significant number of users around the world. Aware of the need to go beyond this student work, and encouraged by its success, ESA (A. Hilgers) then initiated a new project, called SPIS, Spacecraft Plasma Interaction Software. The aim of this action was to give the opportunity to SPINE community to build the code it needed. So, from start, the code objectives were to take into account all the needs of the community, or at least make the code versatile enough so that they can easily be taken care of later, through code evolutions. The code

This document is provided by JAXA.

was first to be a research code, with the possibility to later derive more packaged versions for engineering, as e.g. a specialised version for telecom GEO, with simplified user controls. Emphasis was also set on the modularity of the code, so as to allow a collaborative development within the community, an easy maintenance, medium or long term evolutions and enhancements. An open source strategy was thus defined, aiming at efficient co-development (better than black box juxtaposition), mutualisation of pre and post processing resources among the community, and extensive validation within the community (comparison between different solvers and with experimental data) [Roussel 2003].

This new development was thus structured as depicted on Fig. 1. The ESA contractors (ONERA as prime, Artenum, and Universiy Paris 7) propose code requirements and design, develop the core modules of the software (framework and basic solvers), and support the community work. A Software Development Advisory Board (SDAB) was set up, which supervises and orients the development. Within SPINE community, Working Groups (WG) were defined, which express requirements, participate to the development, and test the software.

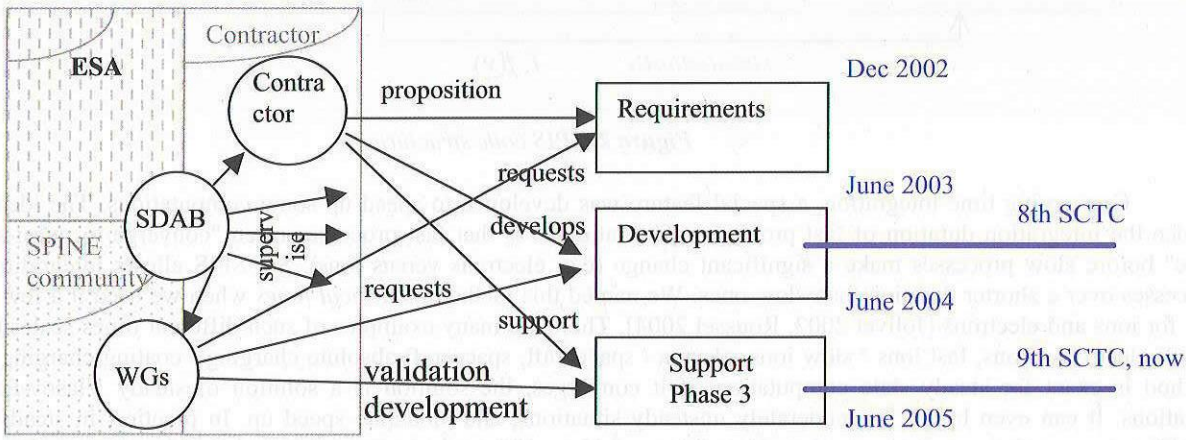


***Figure 1.** SPIS project organization*

This figure also gives the schedule of the project during the present phase funded by ESA contract, which is now close to come to an end. The requirements phase (Dec. 2002 - June 03) led to the major requirements of the code (mostly flexibility and capability to model virtually any plasma-SC situation), and major software choices as the selection of Java and Jython languages (Java-base Python interpreter), and of unstructured mesh. The multi-scale modelling capability was the main driver for the choice of the unstructured mesh. During the (short) development phase (June 03 - June 04), the code framework (SPIS-UI) [Forest 2005b] and the major routines of the numerical core (SPIS-Num) were developed. In the present support phase (June 04 – June 05), main contractors' activities are the support to SPINE community users, through debugging and some specific developments for Working Groups. Three WGs were defined on specific topics and test cases, with a special focus to support their needs in successive period of time:

- WG1, (supported from 2004/06 to 2004/12): sheaths, test case = Cluster
- WG2, (supported from 2004/12 – 2004/05): artificial plasma (EP), test case = SMART 1
- WG3, (supported from 2005/05 – 2005/07): material interactions, test cases = Freja, SCATHA...

Code Structure

The structure of a typical spacecraft plasma simulation and consequently of SPIS software is depicted on Fig. 2. It simply describes how the plasma dynamics is obtained from field and matter coupling, while plasma spacecraft coupling is performed at top simulation level.

Time integration naturally follows this nested architecture. Each level has its own constraints for the largest admissible integration time. Various matter models may have their own special constraints. Examples of such constraints limiting the integration time steps are cell crossing time (Monte Carlo populations), plasma period (plasma), eigenvalues of $\dot{\underline{U}} = \underline{\underline{C}}^{-1}\underline{I}$ system to be solved for spacecraft, etc. These constraints are applied recursively at each level, with possible sub-cycling. Users can define explicitly each of these times steps, or let the code determine them automatically.

This document is provided by JAXA.

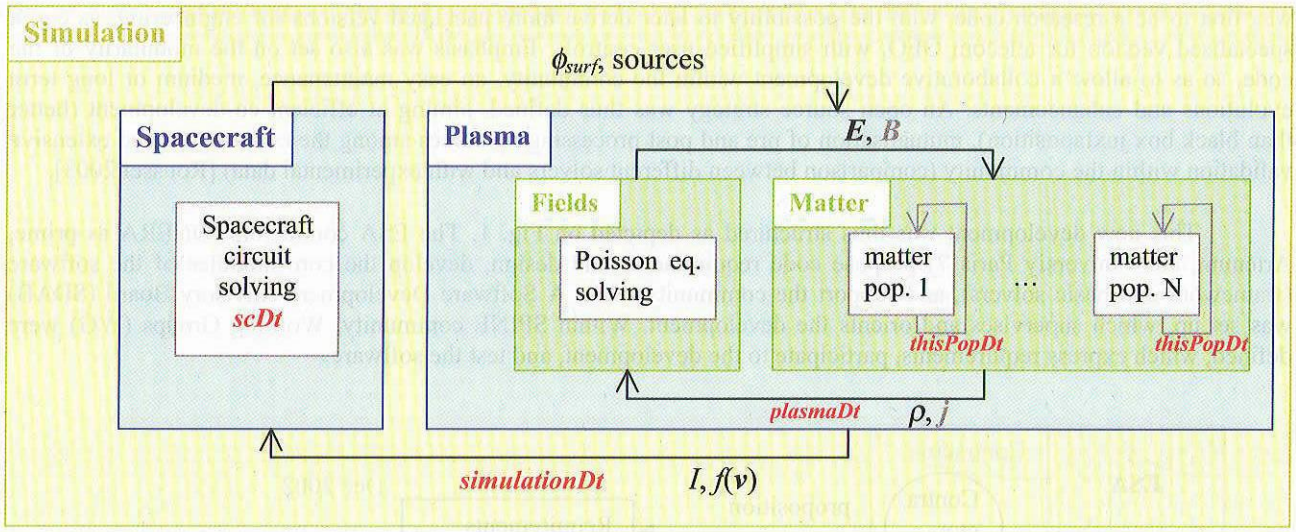


Figure 2. *SPIS code structure*

Concerning time integration, a special feature was developed to speed up some computations. The idea is to reduce the integration duration of fast processes. The rationale is that fast processes often "converge to quasi-steady state" before slow processes make a significant change (e.g. electrons versus ions). So, SPIS allows integrating fast processes over a shorter duration than slow ones. We named this method *numerical times* when we used it a few years ago for ions and electrons [Jolivet 2002, Roussel 2004]. There are many examples of such different times scales (fast / slow): electrons / ions, fast ions / slow ions, plasma / spacecraft, spacecraft absolute charging / coating charging. This method is exact for steady state computation: if it converges, the solution is a solution of steady Vlasov-Poisson equations. It can even be ok for moderately unsteady situations, and moderate speed up. In practice, in steady state conditions, computations can be sped up by a factor of 10 to 100, typical of electron to ion velocity ratio. For large ratios of capacitance values, this kind of method can be implemented by artificially increasing the small capacitance (C_{sat}), which is equivalent to decreasing the integration duration for this degree of freedom. It leads to a non-physical time evolution, but to correct steady state solution, if well controlled. It is an interesting alternative to implicit spacecraft circuit solvers, which rely on assumptions on dI/dV matrix.

The Object Oriented (OO) structure of the code follows this structure. It was already presented at last Charging Conference [Roussel 2003], we will thus only comment on the "plug in" classes implemented in SPIS. In a basic OO approach, new versions of a class have to follow the class model they derive from. For example in SPIS, a volume distribution, such as a PIC distribution (Particle-In-Cell) or a Boltzmann distribution have to implement the abstract method defined in the parent class, typically the move(*dt*) method (time integration) and the getMoment(*order*) method (provide volume density or current). Following that, the code can call them generically in the matter part of Fig. 2 chart (perform time integration by calling move(*dt*) methods, and get densities for Poison equation by calling the getMoment(0) methods). This is already very powerful. In some cases as e.g. for these volume distributions, we went one step further by also defining a standard constructor so that it can be invoked automatically (basically with user-defined parameters). Using the introspection capabilities of Java, this permits to find and invoke such a constructor only from the class name. As a result, if the user defines a new type of volume distribution (e.g. fluid with a Euler solver), there is no need to modify the least piece of the old code to integrate his new class. The steps are simply to:

- Write the class
- Make it accessible (including it in spis.jar in the right package)
- Type its name in the user interface at execution time

The code then automatically instantiates an object of such a class and uses it in the simulation. More details can be found on these plug in classes on SpisNum.ppt and SpisNumWithRelatedPages.zip presentations that can be downloaded from http://dev.spis.org/projects/spine/home/meeting/mviii/data/.

This document is provided by JAXA.

Plasma Model

As already explained the plasma is presently modelled as the coupling of matter and field models, even though a single model of both, as e.g. MHD is (MagnetoHydroDynamics), could be implemented in SPIS. As of today, two versions of matter models are implemented in SPIS, PIC model (Particle-In-Cell, with linear charge deposit) and global Maxwell-Boltzman distribution. They are integrated on the same footing (the OO approach described above) which makes it very easy to add e.g. a fluid distribution as soon as an Euler equation solver is implemented.

Volume interactions are another part of the matter dynamics model. Charge Exchange reaction (CEX) is the only implemented model, following a MCC (Monte Carlo Collision) scheme. Basically, each incoming ion takes an electron from a neutral (described by a background density) with a probability given by the neutral density multiplied by cross section, relative velocity and path length. If this event happens, an ion is generated with the velocity of the neutral distribution. Specific extra functionalities have been added for electric thruster plume modelling, as e.g. the capability to describe the neutral density from the far field distribution resulting from neutral emission over the same mesh elements as the fast ions are emitted from, with a user-defined fraction of non-ionised propellant.

The only field considered as dynamical is the electric field and the field solver is thus a Poison equation solver. It is a finite elements model (P_1: linear potential) on unstructured mesh. The linear system obtained is solved by conjugate gradient method. Boundary conditions can be Dirichlet or Fourier (i.e. mixed Dirichlet-Neumann: $\alpha\,\phi + E_n = g$) with specific capability to define α and g locally so as to a mimic $1/r$ or $1/r^2$ behaviour at external boundaries (typical of sheath or pre-sheath). We also implemented an explicit handling of singularities, as thin wires (1D) and plate edges (2D), which is described below (only wires, plate edges is still under finalisation).

This solver also solves non-linear Poisson equation, which includes the Boltzman form of electron density (indeed possible with two electron distributions in SPIS):

$$-\Delta\phi = e(n_i - n_0\, e^{e\phi/kT}) / \varepsilon_0$$

The major advantage of using Poisson non-linear equation is that the field-matter loop becomes stable even for cells larger than Debye length (since electrons are assumed to have reached thermodynamic equilibrium, contrarily to models of electron dynamics as e.g. a PIC model). The drawback is that the iterations on non linear Poisson equations become unstable for this regime of cells larger than Debye length. We thus had to implement an implicit method to stabilise the solver (Newton-type method).

We give a few examples showing applications of these models. Most of them are more detailed in other papers of that conference on SPIS applications, and are thus simply mentioned here for illustration. Figure 3 depicts the potentials around a generic spacecraft equipped with an electric propulsion (Hall Effect thrusters) system. Due to the very small Debye length at thruster exit (well below 1 mm), while cell size is above the centimetre, a non linear Poisson equation with implicit solver had to be used. In high density regions non linear Poisson equation is equivalent to a quasi-neutrality assumption (sometimes used in such situations), but it is much better in depleted regions, where quasi-neutrality gives very negative non physical potentials. Charge exchange is computed by SPIS in the plume. The plume model is still very simple here, a Maxwellian (MaxwellianThruster class) with $T_i = 5$ eV, Mach number = 7, $I_{tot} = 4.5$ A. Concerning particles, electrons are a global Boltzman distribution, but numerical times speed up was used for fast Xe+ versus slow CEX ions, and ambient ions. Computation time is a matter of few tens of minutes.

This document is provided by JAXA.

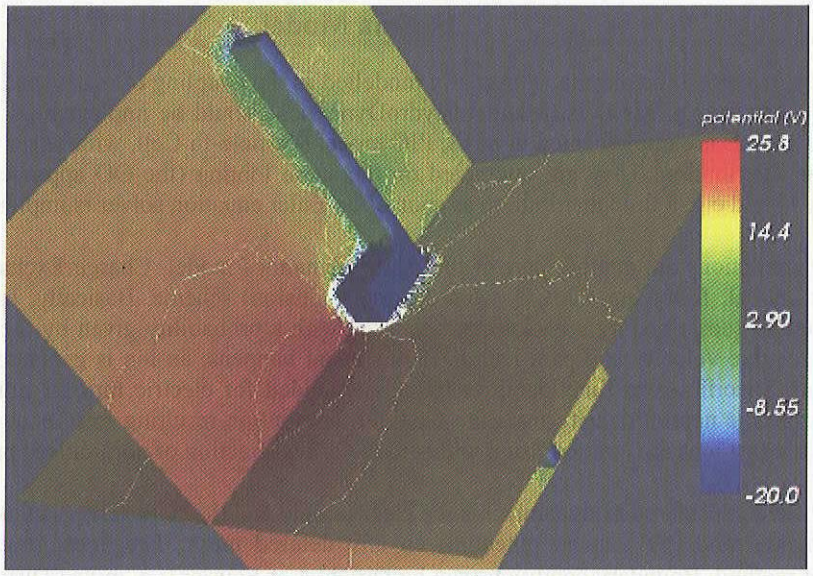


Figure 3. *potentials around a generic spacecraft with an SPT engine.*

Figure 4 deals with a spherical Langmuir probe in the large Debye length regime (λ_D = 2.3 m, r_{sphere} = 0.25 m, $T_e = T_i$ = 1 eV, n = 10 cm^{-3} of H^+): where OML (Orbital Motion Limitation) theory is valid. The comparison to the theory is satisfactory on a 0-10 V sweep (left hand side). Here both ions and electrons were modelled by a PIC method, and numerical times (electrons speed up factor = 40) were used to reduce computation time (to the hour range).

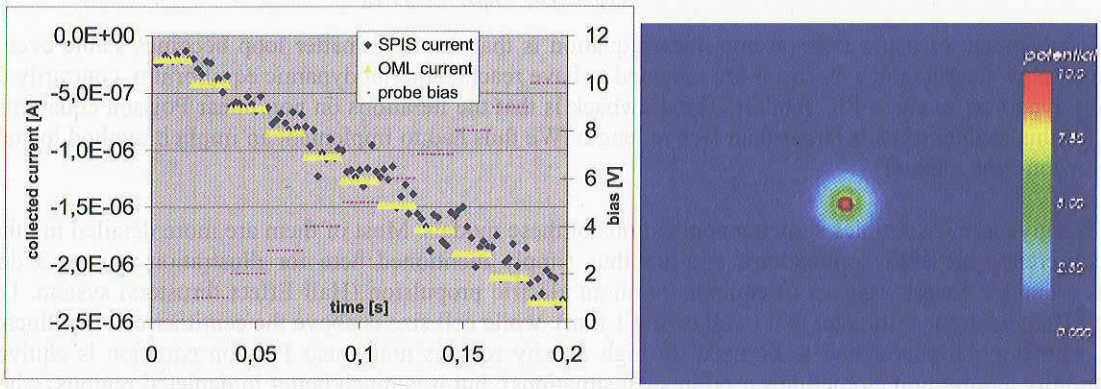


Figure 4. *spherical Langmuir probe: I-V characteristics and comparison to OML theory, potential plot*

Finally Fig. 5 illustrates multi scale issues. Unstructured mesh is well suited for multi-scale modelling (ex of the sphere above: box = 10 λ_D = 100 r_{sphere}), but when going to very large ratios (for an CNES-funded ESD triggering study in progress, we went to a 1 mm resolution and a centimetre scale box, hence a ratio of 10,000), GMSH, the open source mesh generator integrated in SPIS, needs to be helped. It can seen in Fig 5 that when meshing a cylinder at 10 cm scale in a 10 m box, without any help GMSH builds a lot of degenerated tetrahedra (graphically on the left, and on the mesh quality histogram on the right). Figure 6 shows that it improves greatly when helping GMSH, for example by forcing it to mesh at intermediate cell sizes on intermediate nested boxes (visible in the left-most chart). When defining such boxes, each step (e.g. from D to 10D) has the same cost (in cell number), which keeps the overall cost to be simply proportional to log(overall ratio). This is what allowed us to treat a ratio as big as 10,000 as mentioned above. In case of more reasonable ratios (e.g. the 25 cm radius sphere above in a 20 m box), it was not necessary to use such a technique. Multi scale meshing may also raise another issue, the one of the super-particle density, which may be insufficient in small cells. We did not really meet this issue yet since particles are often emitted or focussed in the small cell zone (true in the two examples above). If the problem is encountered, a particle splitting/merging scheme would have to be implemented.

This document is provided by JAXA.

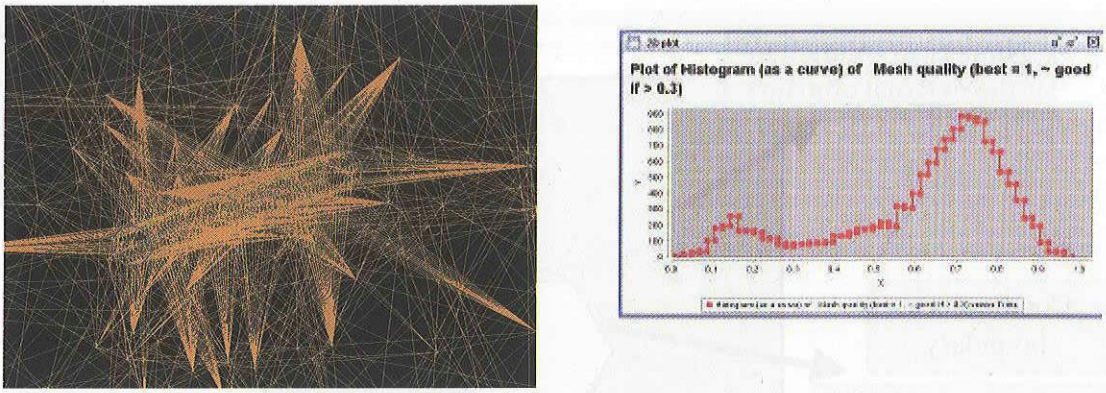


Figure 5. *Mesh and mesh quality histogram (raw usage of GMSH in a difficult multi-scale situation)*

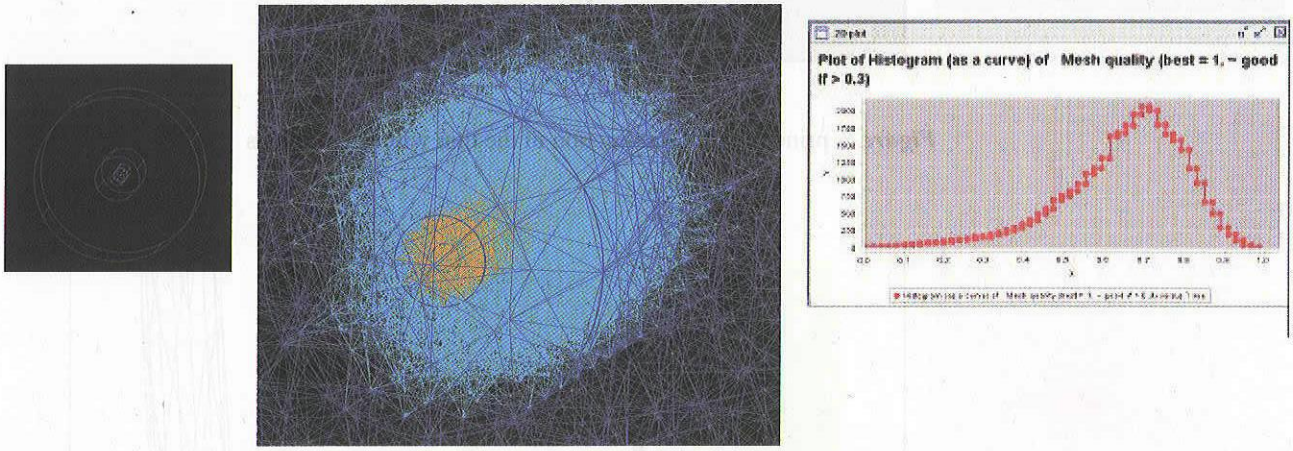


Figure 6. *Mesh and mesh quality histogram (better usage of GMSH in a difficult multi-scale situation: introduction of intermediate boxes)*

When the small structure extends in one dimension, as in a wire, the mesh refinement becomes intractable. In the direction of the wire, the surface mesh can either involve a huge number of small triangles or a smaller amount of degenerated triangles (much longer than wide). None of these was acceptable, at least in extreme cases, as e.g. CLUSTER booms of radius around 1 mm for tens of meters of length. This is why we concluded that the only correct solution was to geometrically model the wire as a 1D object (zero thickness), while of course its radius was used in the solvers. The method consists in extracting the field singularity around such wires and treating it analytically. The computation domain is decomposed into a small sub domain (close to be cylindrical) of a radius a containing the wire, and the complementary domain (external sub domain). The solution of the Poisson equation is expanded in Fourier series in the small sub domain, while at the boundary the external solution is matched with the analytical expansion in the inner domain, as depicted in Fig. 7.

In the next figures, the two possibilities, actually meshing a cylinder or using our wire approximation, are compared. Figs 8 and 9 show how the meshes look like in both cases, while Fig 10 compares the errors of both methods. Our method is of course much more accurate with fewer cells. Fig. 11 is a simple illustration.

This document is provided by JAXA.

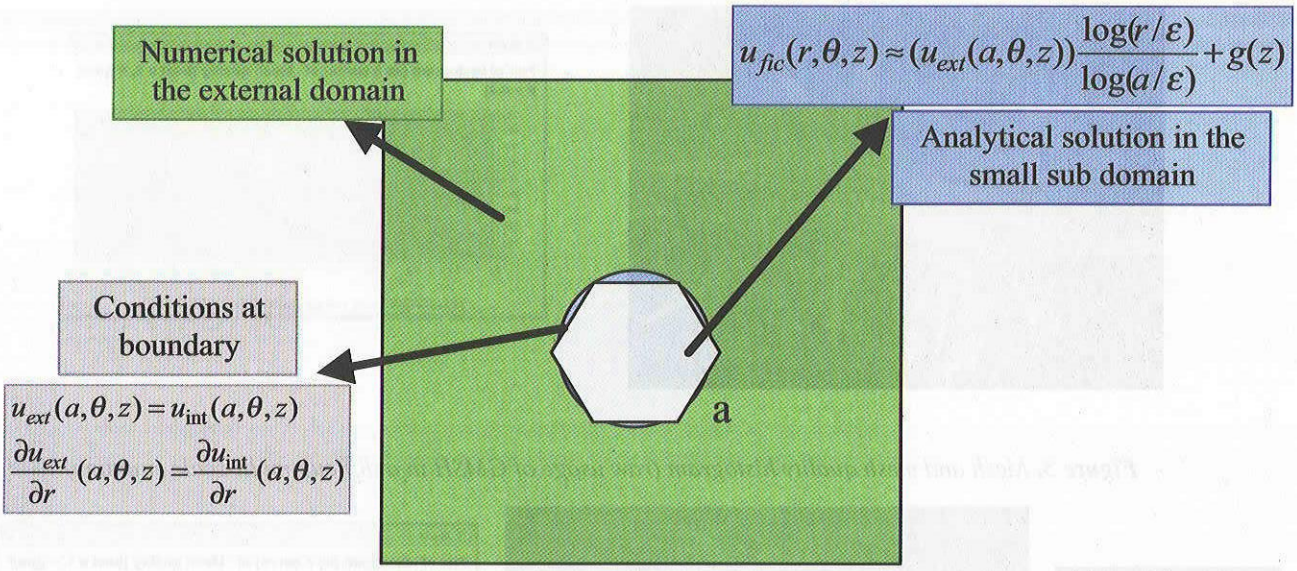
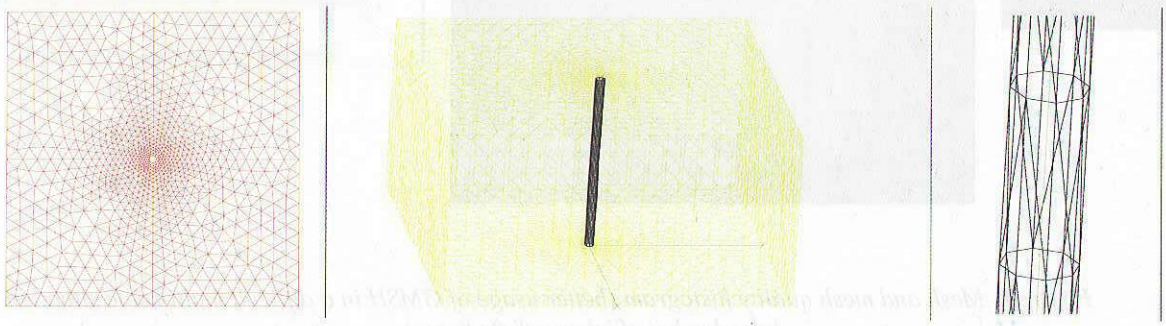
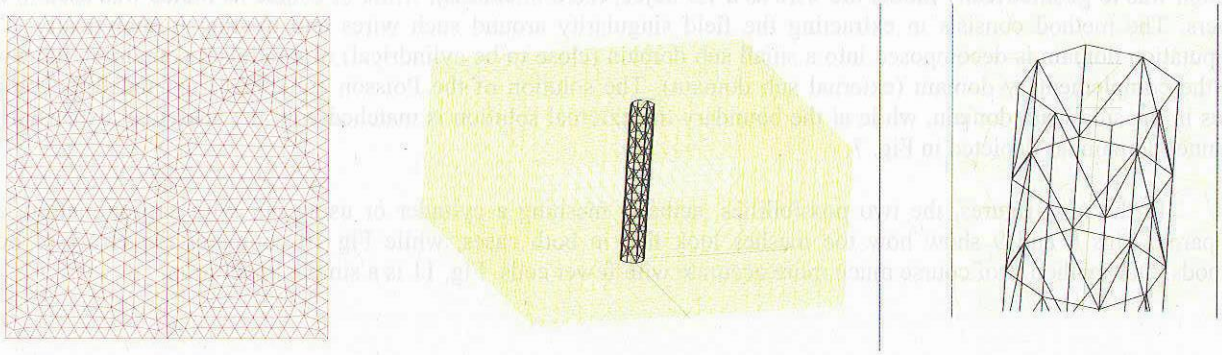


Figure 7. principle of the extraction of the thin wire singularities



2D cut of the mesh | General view: 3D mesh including the wire | Mesh on the wire skin

Figure 8. *Test case number 1:refined mesh around the wire, cell number: 170,000.*



2D cut of the mesh | General view: tetrahedra connex to the wire are singled out | Faces close to the wire

Figure 9. *Test case number 2: wire approximation, celle number: 50,000.*

This document is provided by JAXA.

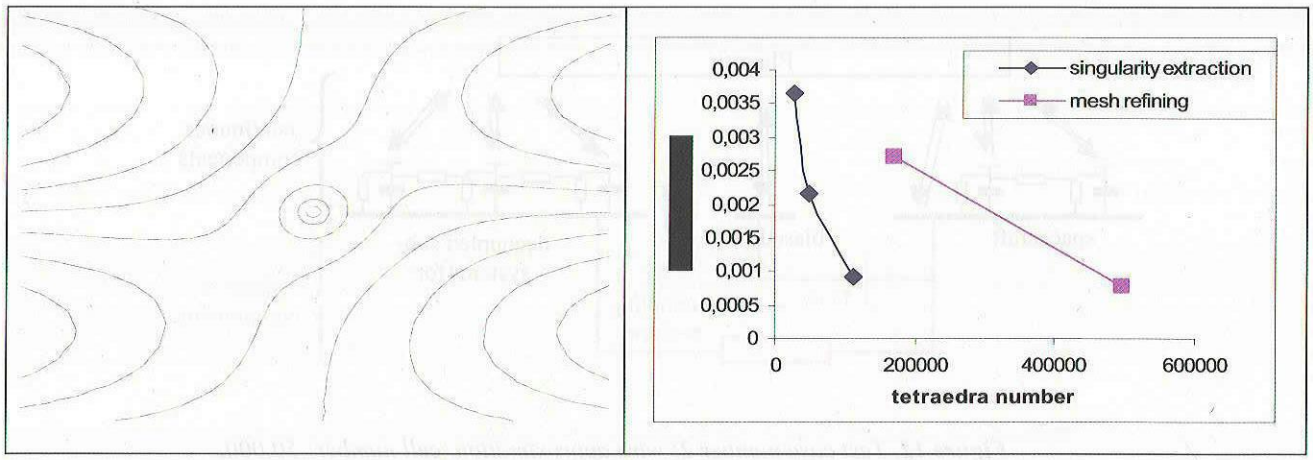


Figure 10. Comparison of both methods: isopotential contours (differences are only at small scale in the centre), comparison of relative errors.

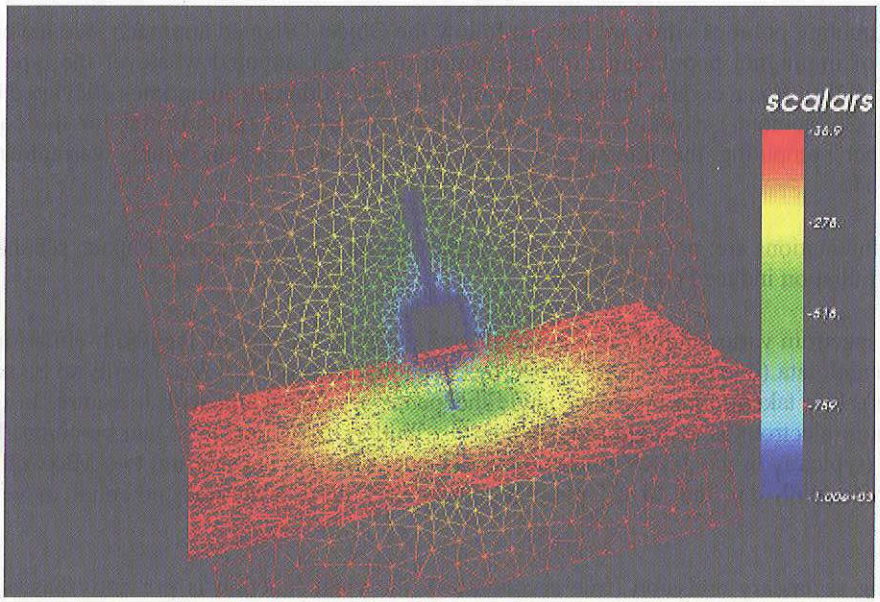


Figure 11. Plasma potential plot around a cubic spacecraft with two wire-like booms (the SC itself, at -1000V, is not represented, only two cutting planes are depicted, with potential on edges to show the mesh)

Spacecraft Model

The movements of electrical charges on a spacecraft can be modelled through an equivalent circuit. In the absence of very fast transients, as e.g. an ESD triggering, it is mostly capacitors and resistors. Figure 12 gives a simplified view of such a circuit. We first have coatings which can be considered as "continuous components", i.e. a capacitor spread all over their surfaces, with resistors in parallel for leakages through them. We also offer the capability to have "discrete components", corresponding to real electronic components (a decoupling resistor, an active bias, etc.), which can be added between electric super nodes (i.e. sub-systems). Continuous components are generated automatically while discrete components are user-defined in an ASCII file. The present version of the circuit solver is explicit, which forces to have an integration time step smaller than the smallest eigenvalue of the linear system describing time evolution. It may be interesting to write an implicit solver to overcome this limitation. However, for the search of steady state solutions, using non-physical capacitances, as e.g. a larger spacecraft absolute capacitance, can provide a larger eigenvalue, hence allow integrating with a suitable larger time step.

This document is provided by JAXA.

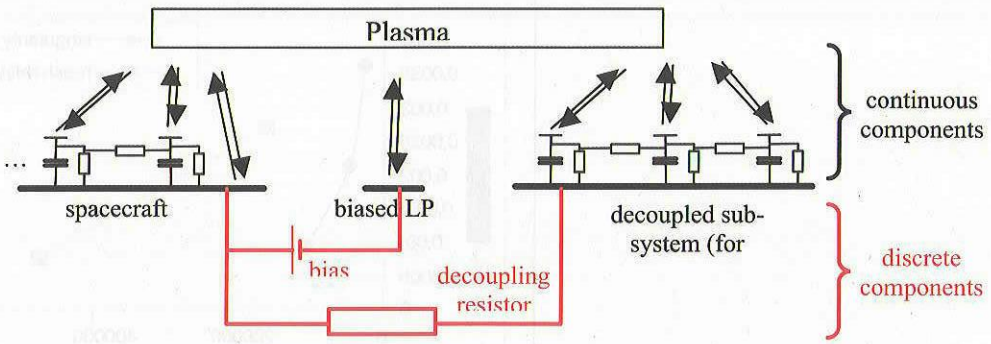


Figure 12. *Test case number 2: wire approximation, cell number: 50,000.*

Spacecraft model can also support particle sources to model thrusters, electron or ion guns, etc. But the major ingredients for a spacecraft model are the material plasma interaction models. They control spacecraft charging in many situations (GEO, PEO, solar wind…).

On the computing point of view, we have to follow the Object Oriented approach. We have to be respectful of the polymorphism of impinging populations: the interaction must be computed whatever the type of impinging flux (typically kinetic or fluid). It is a certain burden on the model writer (although sometimes alleviated by the existence of generic routines as GetMoment(), working on any type of flux), but it is fundamental for the code flexibility. The interactor (the object computing the interaction) itself must follow a model, which warranties it will be called generically in the code.

Modelled interactions are: photo-emission, secondary emission from electron impact, secondary emission from proton impact, and radiation induced conductivity.

Since sun spectrum varies slightly in the wavelength under consideration, the photo-emission production rate is simply an experimental data (one of NASCAP properties is photo emission at 1 AU, simply to be scaled with sun flux for non earth orbits). It is taken to be incidence angle independent, consistently with literature. In the present version, the spectrum of photo-electrons is a single Maxwellian distribution. Its temperature can be controlled through a user-defined parameter: (typically in the 2-3 eV range). Possible extensions are considering two Maxwellian distributions or an arbitrary spectrum (with e.g. the 40 eV population that may drive spacecraft potential in very tenuous plasma conditions).

Considering secondary emission (true secondaries), the yield function is presently based on the position of yield function maximum at normal incidence, so as to be able to use NASCAP properties databases. The yield at other energies and angles is derived from a supplied range function. As usual, this interaction model was coded taking care of modularity. The range function can easily be changed (using a different RangeFunction object) in the same approach, or, changing approach, a different yield function of angle and energy can easily be used. Depending on the type of incident particle model, the microscopic yield can be applied to each particle in case of PIC model, or the isotropic yield is used in case of fluid model (Boltzmann). The spectrum of (true) secondary electron is considered a Maxwellian, of which temperature can be controlled through a user-defined parameter (typically 2-3 eV).

Backscattered electrons are also considered in the secondary emission model. The yield follows NASCAP model, with a small modification (probably still to be improved), since the backscattering yield was assumed to go to 0.5 for dielectrics at low energy (some experimental evidence showing that). Again, depending on the type of incident particle model, the backscattering is computed particle per particle (PIC) or as a global fluid backscatterred flux (if impinging fluid Boltzmann). Concerning the spectrum of backscattered electrons, it diffused in angle (lambertian distribution) and slightly accommodated in term of energy (tuneable accommodation parameter of 0.05).

For the time being, the radiation induced conductivity (RIC) follows the traditional approach:

$$RIC = k\ \dot{D}^{\Delta}$$

This document is provided by JAXA.

where $\dot{D}$ is the dose rate. Again it allows using NASCAP database values. Following a parallel experimental activity (L. Levy other contract, same ESA mini project), an improved RIC model may be implemented, in particular taking into account the effect of past irradiation (delayed RIC). These tests are performed with homogeneous dose rate throughout sample depth (high energy irradiation), which allows extracting model parameters. In flight, the real electron spectrum results in a depth-depending dose profile throughout the material. Yet, most models simply use the RIC formula above with a single dose rate $\dot{D}$, not a whole 1D profile. SPIS model for RIC is currently of that kind. We compute the minimum dose rate, i.e. at the deepest position within the material, since it will be the most limiting place for current conduction through the coating. However we plan to improve that very soon and consider the dose rate profile (leading to conductances in series).

Next two figures illustrate some aspects of material interactions. Figure 13 shows how basic material interaction data can be plotted in SPIS user interface. Figure 14 is a simple example of photo-electron sheath. The large amount of photo-electrons generated on the sun side leads to a negative space charge and potential on top of the sunlit side. A more precise simulation of a photo-sheath with a spherical symmetry (homogeneous emission over the sphere) is reported in [Hilgers 2005], where it is very successfully compared to semi-analytical results (turning point method).

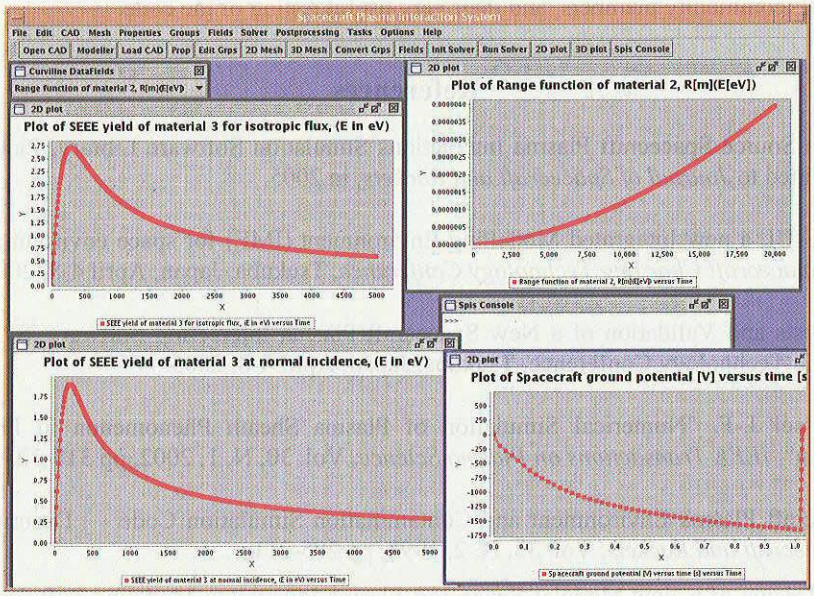


Figure 13. Basic material interaction data can be plotted in SPIS user interface: secondary electron emission yield (left) and range function (top right)

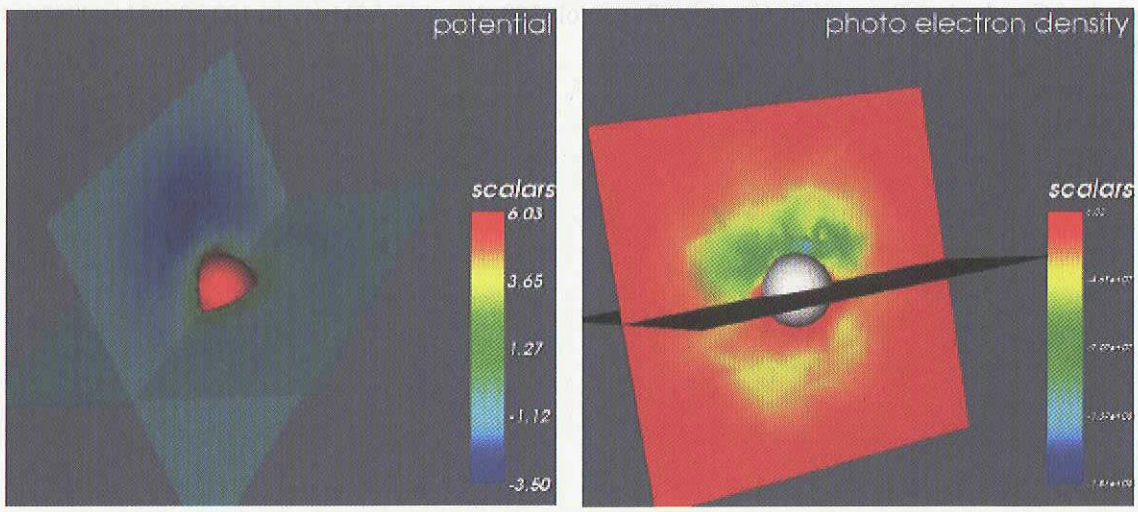


***Figure 14.** Example of photo-electron barrier build up*

This document is provided by JAXA.

Conclusions

In conclusion, it can now be said that SPIS has reached enough maturity to be used by most people. Many domains of application have already been tested (sheaths, EP...), while some still need to be done (material interactions / GEO). An already significant range of situations led to validations [Hilgers 2005]. Some work is still needed for a few extra specific developments, and validations.

It can also be reminded that the first release version of the code is freely available and open source (http://www.spis.org) under a GPL licence (General Public Licence). The logics is to have users anywhere in the world making use of the code for their own needs, also testing it and improving it (adding modules or proposing modifications of existing ones) and returning their extensions to the community.

Acknowledgments

This work has been funded by ESA under contract No. 16806/02/NL/JA. We also acknowledge interesting discussions with SPINE community members, and their very useful testing of the code.

References

Forest, J., "New Open Source Spacecraft Plasma Interactions Simulation Software Library, PicUp3D: First Tests and validations", submitted to *Journal of Spacecraft and Rockets*, in 2005.

Forest, J. et al, " SPIS-UI, a new Integrated Modelling Environment (IME) for space environment simulation codes", *proceeding of 9th Spacecraft Charging Technology Conference*, Tsukuba, Japan, April 4-8, 2005b.

Hilgers, A. et al, " Tests and Validation of a New Spacecraft Plasma Interaction Software, SPIS", proceeding of 9th Spacecraft Charging Technology Conference, Tsukuba, Japan, April 4-8, 2005.

Jolivet, L., and Roussel J.-F. "Numerical Simulation of Plasma Sheath Phenomenon in Presence of Secondary Electronic Emission", *IEEE Transactions on Plasma Science*, Vol. 30, N. 1, 2002, pp 318-325.

Roussel, J.-F. "Spacecraft Plasma Environment and Contamination Simulation Code — Description and first Tests", *Journal of Spacecraft and Rockets*, Vol. 35, N. 2, 1998, pp 205-211.

Roussel, J.-F., et al, "Design of a New Modular Spacecraft Plasma Interaction Modeling Software (SPIS)", *proceeding of 8th Spacecraft Charging Technology Conference*, Huntsville, AL, USA, Oct. 20-24, 2003.

Roussel, J.-F., and Berthelier, J.-J., "A study of the electrical charging of the Rosetta Orbiter: I- Numerical Model", *Journal of Geophysical Research* J. Geophys. Res., Vol. 109, No. A1, A01104, 10.1029/2003JA009836, Jan. 13, 2004

This document is provided by JAXA.