FREE JAVA™ CNES FLIGHT DYNAMICS TOOLS

Jean-François Goester (1)
(1) CNES, 18, Av. Edouard Belin, 31401 Toulouse Cedex 9, France, Email: jean-francois.goester@cnes.fr

ABSTRACT

For numerous years, CNES Flight Dynamics teams have made freely available some astrodynamics tools and libraries as MSLIB library. Nevertheless, these tools, essentially coded in Fortran language needed different versions of compilation depending on used platforms (Solaris, Linux, Windows …) which didn't ease its installation and therefore limit their dissemination.

Some years ago, CNES astrodynamics subdirectorate made the decision to switch to Java language in particular to insure portability whatever the target machine was. As a consequence, old generation astrodynamics tools were recoded to Java™ with a restructuration of code and functionalities leading to an overall improvement of the laters. Moreover, and as a consequence of the new language, these new tools (or new versions of tools) became more easily exportable keeping them available as freely available tools and libraries.

This paper will describe these different tools and libraries always linked to Flight Dynamics applications, their interaction and dependency as well as their dissemination mode (open source, freeware). Initially we will describe low-level libraries as PATRIUS uniquely devoted to Flight dynamics aspects and GENIUS for scientific GUI development. Secondly we will also present GENOPUS library which is based on both previous ones and allows providing "intelligent" widgets as the one used for defining orbit parameters. Then, we will present some tools based on these building blocks as PSIMU (for any kind of trajectory extrapolation around Earth) or MIPELEC (optimization of low thrust propulsion). We will also give as example, tools used in operational contexts as ELECTRA. To finish, means to get and use these tools will be described via the CNES Web site, their licenses, Wikis (including tutorials and Javadoc) or even training course.

1. THE OLD SOFTWARE SUIT

From the very beginning, CNES needed to develop and use Flight Dynamics software both for studies and analysis but also for operational applications. Naturally, the need to build these tools on a common reliable basis appeared as necessary since the very beginning. Thus, in the 70's and the 80's, first basic libraries as the famous MSLIB appeared. The common used language was of course FORTRAN (FORmula TRANslator) which was actually a good choice at that time. Later in the 90's some upgrades of these tools in FORTRAN 90/95 lead to the apparition of intermediate layers allowing to take advantage of this more structured language. All these libraries and tools were grouped inside BIBMS.

Moreover, always in the 90's, the need of using such tools via a Graphical User Interface appeared more and more mandatory. In fact, it was already done for operational contexts using the MERCATOR environment allowing to do dozens of Geostationary LEOP. But such operational environment was not really appropriate for studies or expert software. So, some internal initiatives lead to new layers included in PIMS software:

- MADONA: to define a standard for ASCII files structure
- GENESIS/DIAMS: to be able to develop GUI
- XTRACE: to plot data

These three connected layers permit to develop many tools for advanced studies but also to develop new families of operational environment for LEO orbits (FDS G2) and later for the ATV-CC.

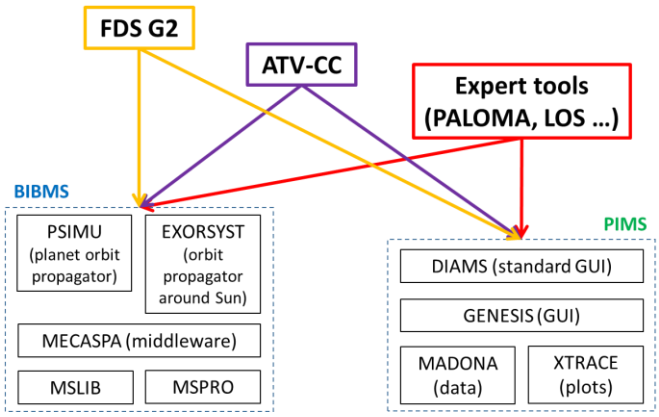

Figure 1: Fortran S/W suit

2. SCILAB DEVELOPMENTS

Before the decision to upgrade our old generation software suit, some other developments began, using Scilab™ language which is more or less equivalent as the Matlab™ one, with the particularity of being freely available. These developments lead to some toolboxes as CELESTLAB. We will not describe them in this paper (cf. [1]) but the reader shall know that these toolboxes are complementary and very powerful for Phase 0 studies.

3. THE CHOICE OF THE JAVA™ LANGUAGE

3.1. Justification

Several needs were identified in order to compare different languages. These technical needs represent different researched qualities of the language which can be summarized as below:

- Technical capacity of the language: CPU time performance, good numerical precision, robustness, portability …
- Existence of tools: compiler, development environment, quality checks …
- Knowledge of the language in CNES or in contractors working for CNES; training course, …

Then, eight criterions were defined to answer to these needs:

1. Numerical performance:
   - computation on doubles with a $1e^{-15}$ precision;
   - respect of the numerical standards (as IEEE_754);
   - comparison functions.
2. CPU performance
3. Portability: even if, for operational purposes, this need is not a strong one (as the kind of platform is fixed relatively soon), for expert tools it is a real need to avoid to take a long time for new compilations then comparisons of different numerical results.
4. Maintenance and development facilities:
   - dynamism of the language;
   - existence of development environments;
   - capacity for developer to well understand it;
   - documentation, wikis, …
   - error management;
   - compiler controls.
5. Interfaces and interoperability:
   - Interface with other languages;
   - Input / Output possibilities.
6. Maturity / obsolescence:
   - Age of the language;
   - Stabilization;
   - Standardization and standard implementations;
   - Variety of the compilers and associated libraries;
   - Maintainers, sellers, distributors, support …
7. Security & reliability
8. Treatments sequencing (for example for parallel computation)

3.2. Selected languages

We focused only on compiled languages which removed automatically other languages as Scala, Python, Perl, Ruby, … Thus, the following languages were selected: Ada, C, C++, Fortran (95, 2003, 2008), Java™.

Java™ language appeared as the « favorite » one for several reasons as the development environment, the user community or its portability. Other advantages "appeared" later as the potential of evolutivity thanks to heritage and interface mechanisms. Of course, "favorite" did not mean the best for each criterion but considering a global perimeter.

4. JAVA™ SOFTWARE SUIT

Once decided to use the Java™ language (2010), it remained to re-develop our software, basically on the same architecture as the ancient one as presented on Figure 1.

First, we had to redefine the BIBMS equivalent. We could do it relatively "easily" thanks to the existing OREKIT (ORbits Extrapolation KIT) open-source library ([3]) developed by CS and based itself on another open-source library: Apache Commons Math ([1]). This new library, called PATRIUS, whose development started in 2011 is today considered as a very powerful library with many features, fully tested and validated, ready to be used in next generation FDS development as well as in mission analysis tools and internal studies.

Then, based on the feedback of our previous flight dynamics suit, a dedicated library for GUI and plots has also been developed which is called GENIUS. This graphical library is used for expert tools developments. For operational tools, specific graphical libraries have been developed.

Note also that the PSIMU propagator has been extracted from PATRIUS perimeter (as it was part of BIBMS) considered finally as an expert tool itself depending on PATRIUS.

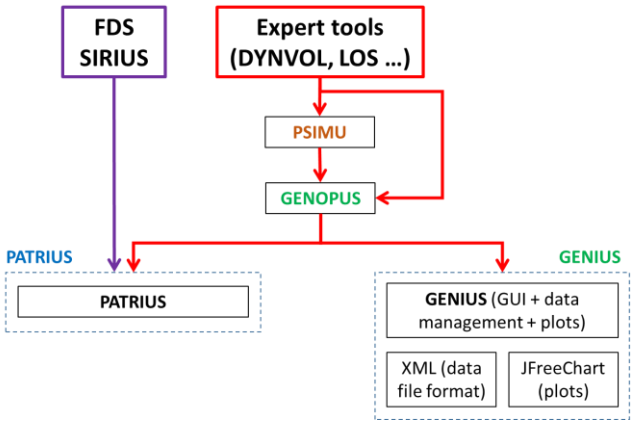

Figure 2: Java S/W suit

5. PATRIUS

Originally PATRIUS was based on a Mathematics package corresponding to the Apache Commons Math library ([2]) and other packages issued from OREKIT ([3]). But after several years of evolution, it was more and more difficult to keep such an organization as these both libraries have been considerably changed and in parallel, more and more add-ons were present. So, since the V4.0 version, it has been decided to shift on a clearer organization

The main packages of PATRIUS are described on figure 3.

Mathematics: Dispersions, Matrices, Rotations, Interpolations, Geometry, Numerical Integration	**Flight dynamics**: Time, Orbits, Frames, Celestial Bodies, SpacecraftState
Attitude: Attitude Laws, Kinematics, Transformations, Guidance Commands, Slew	**Orbit Determination:** Propagation, Physical Models, Measure and Filtering
Spacecraft: Assembly, Sensors, Mass and Forces	**Mission:** Events Detection, Maneuvers, Postprocessing

Figure 3: PATRIUS library features overview

It is not possible to describe all the content of such a library, for more detail confer to ([4]), even if it not present the last state of the library, as the latter has evolved since thanks to user feedbacks (for vehicle characteristics or attitude laws for example), keeping a high level of validation (comparison with ZOOM, the CNES POD tool) that allows to be used for the next generation of FDS.

6. GENIUS

6.1. Why GENIUS?

GUI for flight dynamics tools (or, more generally, scientific tools) need most of the time:

- To enter numerical input data from the screen or the keyboard;
- To read / write these data into files;
- To execute computation thanks to these data;
- To visualize results.

That is the reason why GENESIS had been developed as no dedicated software was available (especially in the beginning of 90's!).

Now, and specifically in Java™ world, several tools are available: basic ones, as Swing, or more elaborated ones. But, they do not include such functionalities or, if they have, it will be partially.

So, GENIUS, is a CNES higher level layer based on Swing (as GENESIS was based on TCL-TK), fully written in Java™ (no need of code generation) but allowing to create more easily such scientific GUI with data and result visualization widgets.

6.2. Main advantages

Besides the fact that coded in Java™, this product is available for any kind of platforms owning Java™, GENIUS includes some interesting advantages:

- It includes numerical widgets!
- It uses some simplified approaches, in particular about events management (BEFORE, AFTER via the GListener interface) and the fact that you will not have to manage how to refresh the display as it will be done automatically each time it is necessary (for example, if a data was changed);
- It performs very easily conditional display;
- There are units' management:

Figure 4: GENIUS units management

- Always, linked to numerical data inputs, it is possible to manage interval of validity:

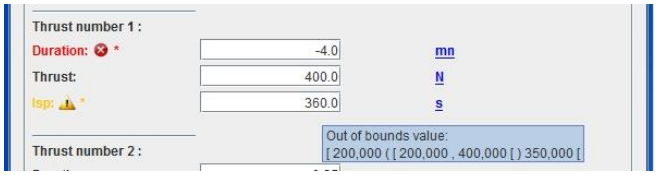

Figure 5: GENIUS validity controls

- Possibility of internationalization of the labels (several languages for a single application)
- Its process management is compatible of all OS (thanks to Java™).
- Another important point, not really linked to graphical aspects, is the fact that data files read/write mechanism is directly integrated (as it was already the case for GENESIS): in fact, the basic idea is the following one:

1. First, you have to write some code to display your data
2. Then, as you need to save your data (if possible in an easy readable format), you will need to write another part of code.
3. Finally, as you need to read these saved data, you will have to write a third part of code.

… and if we think about it, we will have to write three times the same logic in three different parts of code! Thus, GENIUS allows to write it once. And the format of the generated files is XML (not necessary to redefine a format as MADONA).

6.3. Some high level widgets

Some interesting high level widgets have also been added thanks to several user feedbacks. We may identify for example:

- GComponentList: a widget allowing to manage a list of widgets

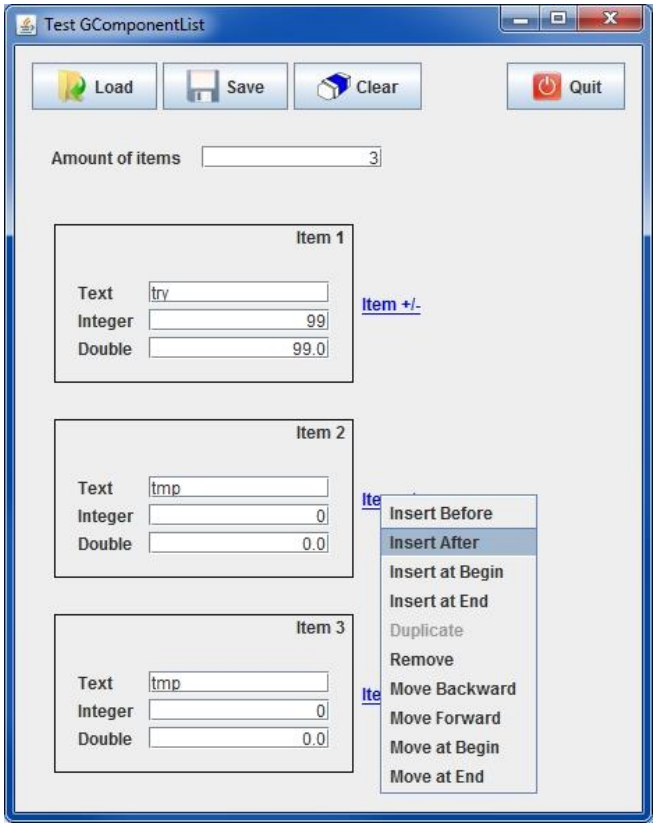

Figure 6: GENIUS list of widgets

- GPlotPanel: a widget allowing to get 2D plots (thanks to the JFreeChart library) after reading data in ASCII files or SQLite ones. It replaces XTRACE in the old Fortran suit.

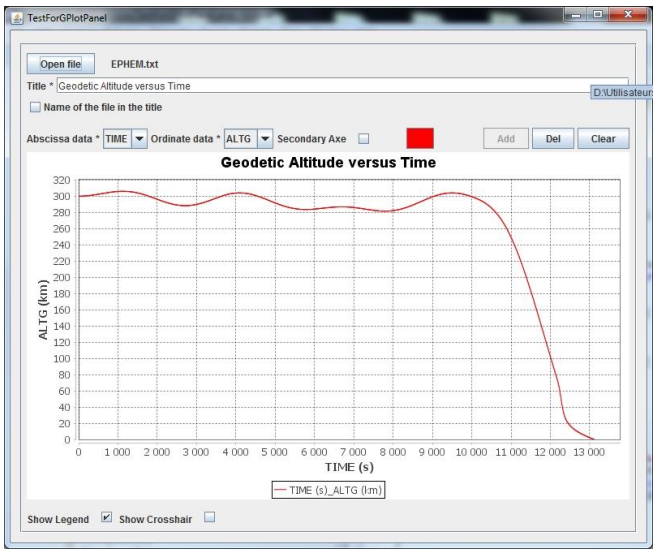

Figure 7: GENIUS plots

- Same kind of widget also exists but allowing to plot directly ground tracks:

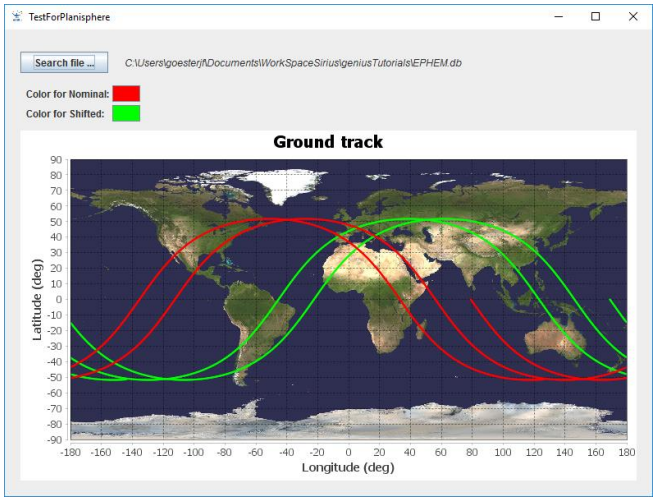

Figure 8: GENIUS ground tracks

- Possibility to build a "standard application GUI" managing:
  - context file loading and saving;
  - computation launching;
  - result files saving;
  - displaying input data as well as output ones.

Most of our Java™ expert tools uses this standard frame as PSIMU as depicted in Figure 9. It allows to the future users not to be lost from a tool to another one.

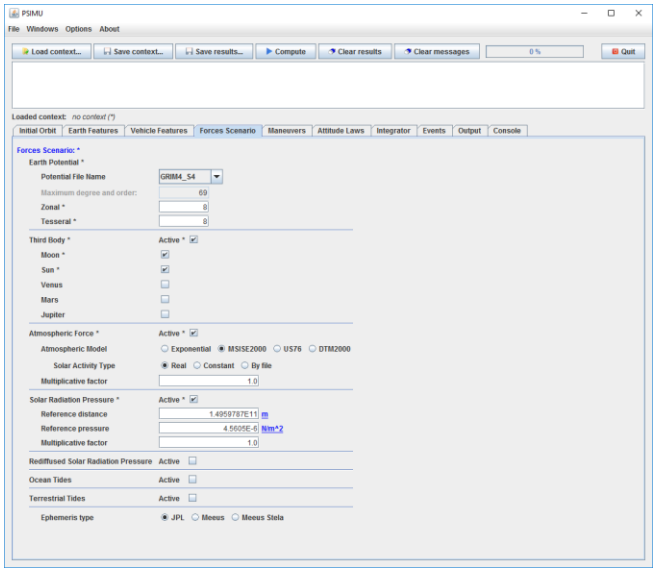

Figure 9: example of GENIUS "standard application" with PSIMU

7. GENOPUS

On one side we have a library of flight dynamics algorithms and data structures (i.e. objects) with PATRIUS and on the other side a GUI library specialized for scientific applications with GENIUS. Both are useful for developing expert tools but it seemed obvious that each of these tools would not have to redeveloped common widgets as classically the one used to define orbit parameters!

That is the reason why GENOPUS has been developed based both on PATRIUS and GENIUS (in fact an equivalent library existed in the old Fortran suit: GSLIB). GENOPUS is a software library including widgets, fully written in Java™ , by using GENIUS product and corresponding to flight dynamics objects available via PATRIUS library. So, for example, we could find widgets allowing to:

- Entry of a date with timescale and conversions;
- Entry for inertial and rotating frames configurations;
- Entry of orbit definition (date, frame, parameters) and conversions;
- Entry of impulsive maneuver, continuous thrust maneuver or a sequence of maneuvers combining both types;
- Entry of attitude laws individually or via a sequence of laws;
- Entry of orbital events (eclipse …);
- Entry of vehicle characteristics;
- Entry of force models (potential, atmosphere, solar pressure radiation …);
- Many other basic widgets to define for example an ellipsoid, a rotation or more recently ground stations coordinates.

Thus, this library allows getting very quickly complex flight dynamics widgets fully consistent with PATRIUS objects.

Moreover, an important point is that, for some specific widgets as the orbit one, conversions (always based on PATRIUS algorithms) are available.

Here, we have an example of how the user may choose between different date formats and time scales:

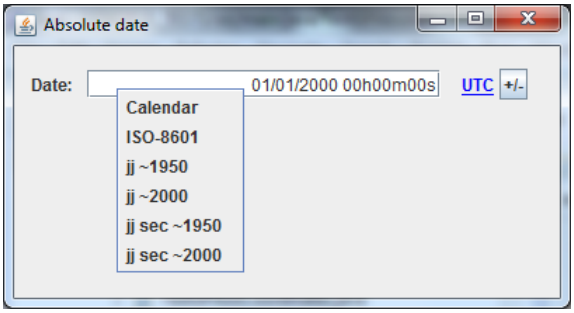

Figure 10: GENOPUS date widget (format)

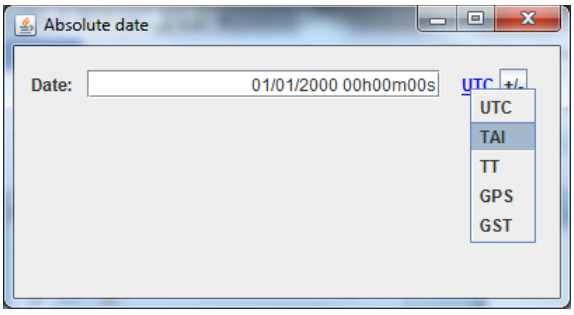

Figure 11: GENOPUS date widget (time scales)

On the next example, we can see the conversion from Keplerian parameters (defined with perigee/apogee altitudes) to Cartesian ones. Of course, it is also possible to go on with such conversions by changing the frame.

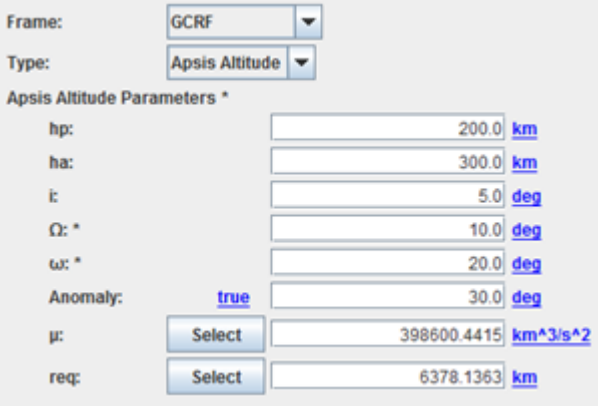

Figure 12: GENOPUS orbit widget

Figure 13: GENOPUS orbit widget after conversion

Thus, GENOPUS includes very basic widgets (even if there is some "intelligence" inside them) as for entering a date or a simple rotation up to more complex widgets as for entering orbit characteristics (as presented above), the definition of an impulsive, a continuous thrust maneuver or a sequence of such maneuvers, the definition of attitude laws or a sequence of attitude laws …

At last, each widget may be initialized with a predefined PATRIUS object as it owns a getPatrius() method allowing to get the equivalent PATRIUS object. Thus, it is very easy to switch between both libraries.

8. PSIMU

Originally, PSIMU in the old Fortran suit was very useful as there were no sufficiently high level layers to propose something to propagate trajectories. Only very basic functions as frames or parameters conversions, numerical integrators or forces computation were available. So, PSIMU was there to deal with a lack of such level.

This need has been replaced in PATRIUS by the very powerful capacity of the NumericalPropagator class. Nevertheless, another need was to propose to users not a solution for coding but a "on the shelf" tool with its own GUI: it is not the problem to code in Fortran, Java™ or Scilab if you just want to propagate an orbit.

So, it has been decided to build a PSIMU Java™ version, at least with its GUI and batch version. As most of our tools are using this kind of architecture, it was very easy to access to the PSIMU main class and therefore use the core of PSIMU internally in some expert tools as OSCAR/DRAGON ([5]).

So, this Java™ version of PSIMU is a tool allowing to propagate trajectories around the Earth (the Fortran version was also able to propagate around Mars or Venus). These trajectories may be:

- Elliptical ones (all kind of orbits, from LEO to GEO passing through MEO or HEO),
- Hyperbolic ones,
- Atmospheric ones (in particular in case of debris reentries).

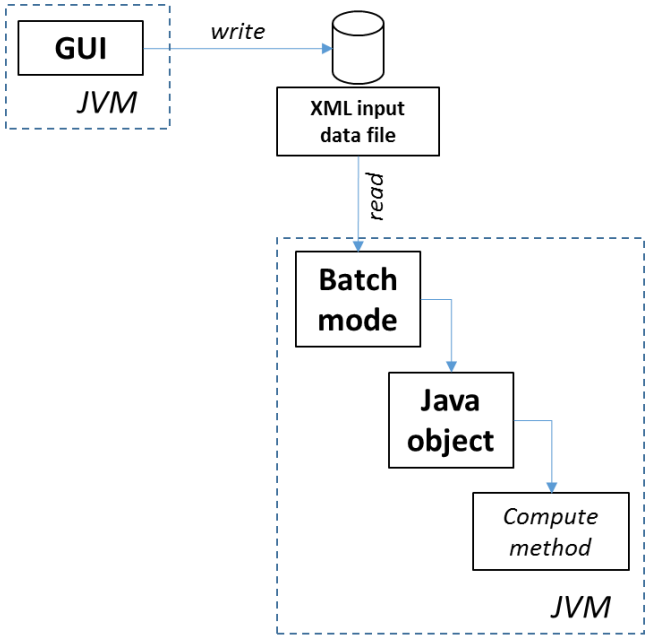

Figure 14: S/W architecture between GUI/batch modes

Its initialization is made by:

- The initial orbital parameters definition (epoch, frame, coordinates) with a great number of available options.

- The vehicle modelling:
  - Vehicle shapes (sphere, cylinder and parallelepiped) with or without solar panels
  - Dimensions or surfaces,
  - Aerodynamic characteristics and/or radiative pressure ones,
  - Propulsive characteristics (engines, tanks).
- A maneuver sequence (optional) including impulsive maneuvers and/or continuous ones.
- A sequence of attitude laws (optional); indeed, if PSIMU does not manage 6 DDL motion, it owns as input data, attitude laws depending on orbital events, allowing to know at every moment the vehicle attitude and thus, to deduce from it the forces applied to the vehicle.
- Numerical integrator parametrization (Runge Kutta or Dormand Price) with, for usual applications, by default settings.
- Choice of force models within:
  - Potential with several models available as the possibility to manage degree and order of zonal and tesseral terms
  - Other bodies attraction: Moon and Sun via analytical or numerical ephemeris,
  - Atmospheric forces using different atmospheric models and their associated settings,
  - Solar radiative pressure (direct or rediffused one)
  - Oceanic and terrestrial tides

Most of the widgets used for entering such data via the GUI are of course issued from GENOPUS!

PSIMU also allows to set its output data within several tens of variables, the output frame and, of course, the output step. A graphical interface for plots is also integrated.

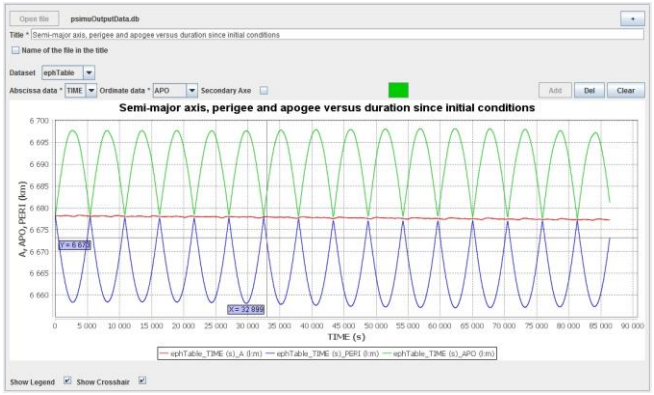

Figure 15: PSIMU plots

9. OTHER APPLICATIONS

9.1. MIPELEC

Thanks to PATRIUS, GENIUS and GENOPUS but also sometimes PSIMU, several other expert tools have been (re)developed in Java™. The first one as it did not require too much flight dynamics properties (only Keplerian mode, no attitude, …) was MIPELEC. This tool is may be one of the most ancient tool to be freely delivered by CNES but only via its source code to be recompiled sometimes with some difficulties. So, it has been decided to propose, always freely, a new Java™ version with its own GUI.

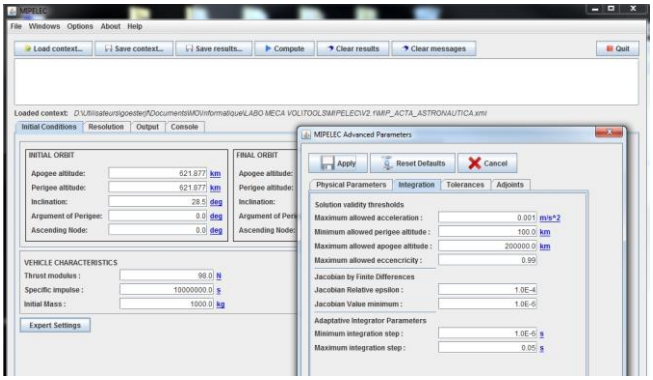

Figure 16: MIPELEC input data

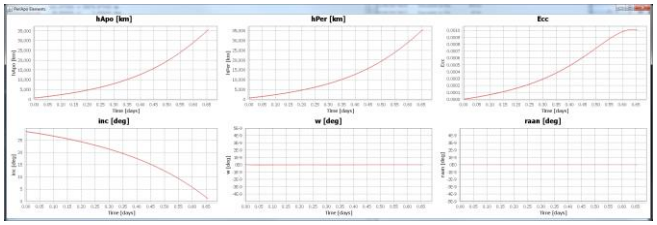

Figure 17: MIPELEC plots

9.2. Other maneuvers tools

Several other expert tools have been recoded in Java™ using these basic libraries (but not yet available outside CNES). We can list for example:

- CRASH: a tool allowing to compute guided reentry trajectories.
- DOORS: a tool computing deorbit scenario as well as an estimation of the debris fallout areas. The previous Fortran version of this tool had been used for ATV operations.
- OSCAR / DRAGON: a set of tools to compute phasing / rendezvous scenarii. The previous Fortran version has been used for ATV operations and is still used for the current GALILEO station acquisition operations.

9.3. French Space Operation Act (FSOA) Tools

As French National Space Agency, CNES has the responsibility to validate technically that launchers and satellites operated by French operators respects the law. To this purpose, and to help the operators to proof that their mission is in agreement with the FSOA, CNES developed efficient state-of-the-art tools for such evaluation: STELA, DEBRISK and ELECTRA.

A specific paper is dedicated to the last two tools ([6]) and for the first, we may refer to [7].

9.3.1. STELA

The Semi-analytic Tool for End of Life Analysis (STELA) reflects the standard concerning the protection of LEO and GEO regions (lifetime and protected regions crossing of disposal orbits) and provides the user with tools to assess compliance with the requirements. The software allows efficient long-term propagation of LEO, GEO, and GTO based on a semi-analytical models and assessment of protected regions criteria. Thus, STELA produces a report file that summarizes the computation (spacecraft characteristics, initial and final orbits, computation parameters, criteria status) and optionally an ephemeris file.

STELA is probably one of the first significant tool developed in Java™. That is the reason why most of its code does not use PATRIUS and, a fortiori, GENIUS/GENOPUS libraries (GUI is directly coded in Swing). Nevertheless, the main algorithms, linked to its semi-analytical models for orbit propagation have been included in PATRIUS, which permits to use them directly for other tools.

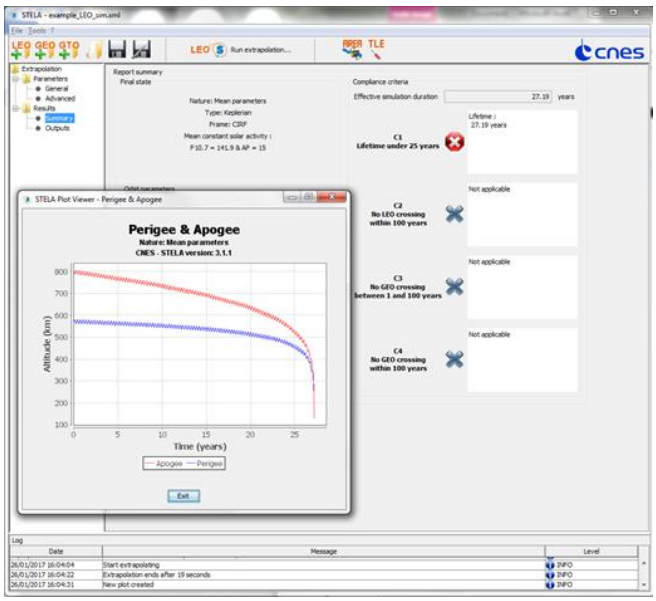

Figure 18: STELA GUI

9.3.2. DEBRISK

DEBRISK evaluates the survivability of fragments from a satellite entering the Earth's atmosphere. This software is available for space operators to check the compliance of their vehicles with this technical regulation. It computes trajectories and ablation of fragments from a space vehicle during re-entry.

This software uses an object oriented approach: it assumes the satellite to be a multiple interdependent objects set, modelled by simple forms. Each object is characterized by its geometrical shape, its size, its mass and its material. DEBRISK provides a list of the surviving objects and their characteristics upon ground arrival.

DEBRISK is more recent than STELA, so its Java™ development used very soon PATRIUS library. However, its GUI does not use GENIUS/GENOPUS (also directly in Swing) as these products did not exist at its creation. The possibility to upgrade its GUI using GENIUS/GENOPUS is under study.

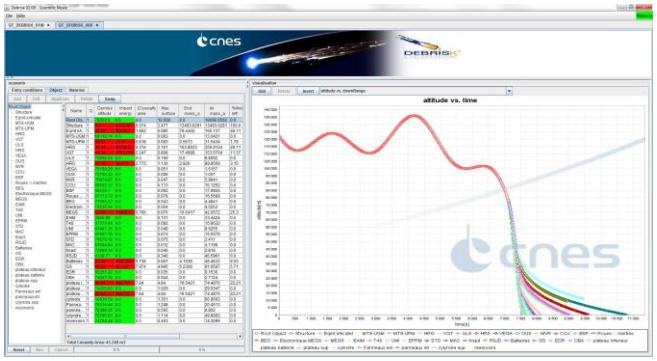

Figure 19: DEBRISK GUI

9.3.3. electra

ELECTRA tool meets the requirement for precise quantification of the risks involved in the launch and the re-entry of a spacecraft. It computes the risk of making a victim due to atmospheric reentries, with or without taking into account protection coefficients.

Using a lot of input data as the debris characteristics eventually provided by DEBRISK, ELECTRA can compute the risk of making a victim in several contexts:

- RA mode: in this case the space object (satellite of launcher part) is not controlled and it is extremely difficult to predict the impacts location. The method considers only the latitudes the object flies over, meaning the risk depends on the inclination of the orbit.

- RL mode: this context starts from the trajectory of a launching (which is guided by definition) eventually dispersed. The method computes the risk due to failures occurring during this launching phase.
- RC mode: this context deals with controlled trajectories following deorbit maneuvers and evaluates the risk associated with maneuver failures.
- RF mode: in this case, we consider uncontrolled re-entries but only some days before the final fallout. It is then possible to compute the risk more precisely than in RA mode. This mode is only available since the V4.1 version.

Except for RA mode, numerical propagations are used with Monte Carlo method leading to relatively important CPU time (depending of the amount of required simulations).

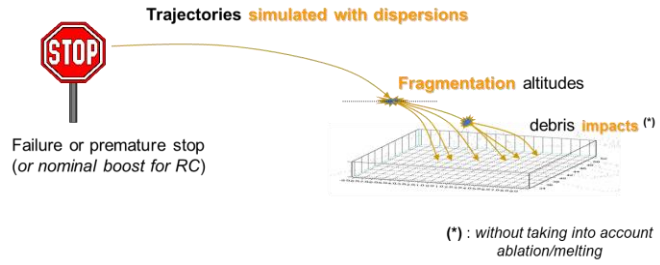

Figure 20: ELECTRA Monte Carlo

It is the more ancient tool developed for FSOA as its development undertook in 2007. At this date, it was naturally coded in Fortran using the previous suit using BIBMS and PIMS. But since 2015, after the internal CNES decision to use Java™ language, a new development started for ELECTRA leading to new V4 versions. The next version (V4.2) will be available at the beginning of 2019.

ELECTRA fully uses PATRIUS functionalities as well as for the orbital or the atmospheric phase. GENIUS is also used a lot as, may be, the ELECTRA GUI is the most complex ever done with such a product.

Figure 21: ELECTRA main frame

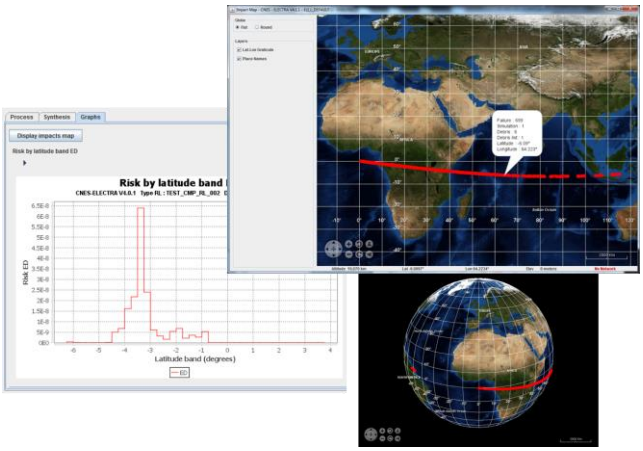

Figure 22: ELECTRA display

10. DISTRIBUTION

10.1. Why?

What is the interest for the CNES for a software distribution? Of course such a distribution deals with the CNES and especially CNES flight dynamics outreach but some other advantages exist:

- To be more easily adopted by our contractors and then to get a better efficiency;
- A good mean for cooperation;
- Support to education (universities but even high schools);
- To become a reference (for example in the FSOA context);
- Making our tools more and more robust by increasing the amount of users.

Anyway, it is not foreseen to distribute all our flight dynamics tools. Particularly the operational branch may have some distribution opportunities but not in the same context as basic libraries or expert tools.

10.2. Which kind of distribution?

Bad license terms as the content of the distribution could occur a brake to it:

- Free of charge or not?
- Binary or source code?
- Duration of the license?
- Possibility of further commercialization?

Some internal discussion leads towards the conclusion that, for software candidates to an external distribution, a free of charge position was the best solution as the choice between binary versus source code will depend of the software considered.

10.2.1. Basic libraries

Such libraries are only useful for developers. That is the reason why it has been decided to deliver source code and to have an « Open Source » distribution to give the following benefits:

- Possibility, for the developers to debug their problems or to do some evolutions without the need to immediately contact CNES;
- Best confidence in the durability of the product.

Moreover, to be consistent with other similar products and for an easy use, it has been decided to associate an Apache 2.0 license.

Note: CELESTLAB (written in SCILAB) is already with such kind of distribution but directly managed in the frame of SCILAB toolboxes as an associated product.

10.2.2. Expert Tools

For expert tools, the philosophy is a bit different as these tools applied not to developers but to "simple" users who, most of the time, need to get results of a computation without need of knowing with which language the tool has been written. That is the reason why the distribution mode will be as executables including their own GUI.

The kind of license is based on the one being created when STELA was firstly distributed.

10.2.3. Specific case of FSOA tools

STELA has not a specific kind of distribution as it can be used of course in the frame of FSOA studies but also as an orbital propagator, all kind of data useful for orbit determination and plenty other needs.

On the contrary, DEBRISK and ELECTRA has been considered as more sensitive tools and their distributions are, up to now, restricted to such a FSOA use. Internal CNES discussions could enlarge the distribution.

10.3. Web site

Rather than to create a specific site to be able to download these products, it has been decided do use a preexisting site where other CNES tools were already distributed as MSLIB (https://logiciels.cnes.fr/en). As this site was not a model of modernity and usability, some evolutions have been set up as for example the possibility to categorize the tools available (not only a single alphabetical list as before) or the fact that the description spreads on different tabs rather than on a single page.

Thus, the flight dynamics tools are listed in a single category except for FSOA tools that are placed in a specific one.

Up to now, the available Flight Dynamics tools, including old Fortran MSLIB library and tools associated with SCILAB developments are the following ones:

- PATRIUS
- PATRIUS_DATASET (a set of data necessary for using some PATRIUS functionalities (for example UTC-TAI gaps)
- GENIUS
- GENOPUS
- PSIMU
- MIPELEC
- CELESTLAB
- CELESTLABX
- MSLIB
- VTS (a graphical 2D/3D visualization tool)

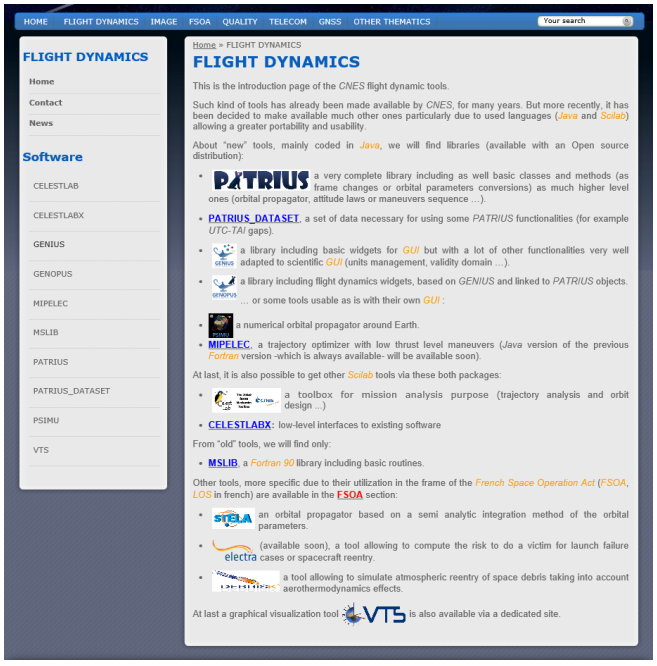

Figure 23: Flight Dynamics tools list

To this list, we can add the three FSOA tools:

- ELECTRA
- STELA
- DEBRISK

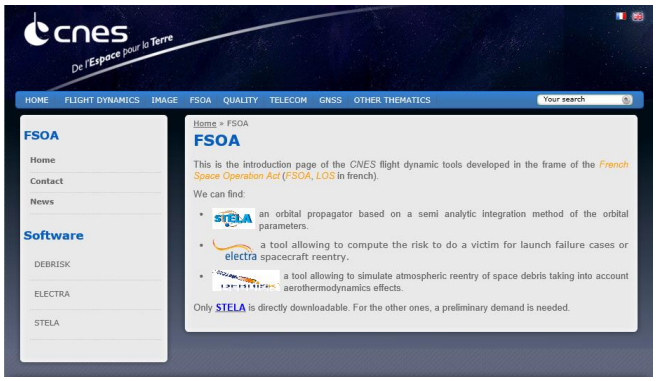

Figure 24: FSOA tools list

10.4. Wikis

In parallel to this web site, some Wikis, based on MediaWiki (https://www.mediawiki.org) format, have been created to help for using some of these tools. Of course, these Wikis are accessible by everybody. Up to now, the available Wikis are dedicated to:

- GENIUS (http://genius.cnes.fr)
- GENOPUS (http://genopus.cnes.fr)
- PATRIUS (http://patrius.cnes.fr)
- PSIMU (http://psimu.cnes.fr)

10.5. Training courses

Via the Wikis, a lot of tutorials are available but CNES provides some training courses for using PATRIUS and GENIUS libraries … as well as for using the Java™ language for scientific tools.

11. CONCLUSION

Since 2010, most of all the developments of CNES flight dynamics software use the Java™ language based on basic libraries as PATRIUS for algorithms and GENIUS for Graphical User Interface (others use the Scilab™ language with links between both).

Thanks to the Java™ (and Scilab™) portability on about all the existing platforms and Operating System, it becomes easier to distribute these tools outside CNES. So, it has been decided to propose an Open Source approach using Apache 2.0 license (or equivalent for Scilab™ toolboxes) for basic libraries and freeware conditions for higher level expert tools as PSIMU in their binary versions including their own GUI.

These tools are downloadable via the CNES dedicated Website (https://logiciels.cnes.fr) and some information is available through several Wikis.

12. REFERENCES

[1] Thierry Martin, Alain Lamy, Guillaume Azema, Maria-Luz Hernandez, "*CELESTLAB: A FREE AND OPEN SOURCE SCILAB LIBRARY FOR FLIGHT DYNAMICS*", 4th International Conference on Astrodynamics Tools and Techniques, Madrid, Spain, 3 – 6 May 2010

[2] Apache Commons Math: The Apache Commons Mathematics Library. Online at http://commons.apache.org/math/ (as of 9 May 2012).

[3] Maisonobe, L. (Feb. 2011). About OREKIT. Online at https://www.orekit.org/blog/index.php?pages/About (as of 9 May 2012).

[4] Houdroge, R., Claude, D., Anton, J., Sabatini, T., Cardoso, P., Mercadier, G., Trapier, T., Tanguy, Y. "*THE SIRIUS FLIGHT DYNAMICS LIBRARY FOR THE NEXT 25 YEARS*", 5th International Conference on Astrodynamics Tools and Techniques, The Netherlands, 29 may – 1 June 2012.

[5] Ivan Sumelzo Martinez, Pierre Labourdette "*JOSCAR/JDRAGON: TOOLS FOR MANEUVER STRATEGY COMPUTATION DEVELOPED IN JAVA AND USING PATRIUS*" 6th International Conference on Astrodynamics Tools and Techniques, Darmstadt, Germany, 14-17 March 2016

[6] A. Bellucci, J.F. Goester, P.Omaly, S. Christy, F. Delmas, "*RISK ANALYSIS BETWEEN AIRCRAFT AND SPACE DEBRIS DURING ATMOSPHERIC RE-ENTRY*" 7th International Conference on Astrodynamics Tools and Techniques, Oberpfaffenhofen, Germany, 6-9 November 2018

[7] Fraysse, H., Morand, V., Le Fevre, C., Cauhert, A., Lamy, A., Mercier, P., Dental, C., Deleflie, F., STELA a Tool for Long-Term Orbit Propagation, Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques, 29 May-1 June 2012, ESA/ESTEC, Netherlands.

13. ACRONYMS

ATV-CC: Automated Transfer Vehicle Control Center

CNES: Centre National d'Etudes Spatiales

FDS: Flight Dynamics System

FSOA: French Space Operation Act (LOS in French)

GENIUS: GENeration of Interface for Users of Scientific S/W.

GEO: Geostationary Earth Orbit

GUI: Graphical User Interface

LEO: Low Earth Orbit

LEOP: Launch Early Orbit Phase

POD: Precise orbit Determination