PALOMA

GMV Space
Ground Control Systems

MIPELEC-MU-LOG-0012-GMV
Edition : 01 Date : 15/04/2016
Révision : 01 Date : 28/07/2017
Réf. : intentionnellement vide Code diffusion : E

MANUEL UTILISATEUR

MANUEL UTILISATEUR LOGICIEL MIPELEC

Rédigé par : GMV-Équipe ACIBS	GMV	le : 28/07/2017	
Validé par : EITO Ricardo	GMV	le : 28/07/2017	
Pour application : LLAMAS Iván	GMV	le : 28/07/2017	

BORDEREAU D'INDEXATION

CONFIDENTIALITE : DLP	**MOTS CLES : Mipelec, Exigences, Manuel, Logiciel**		
TITRE DU DOCUMENT : MANUEL UTILISATEUR MANUEL UTILISATEUR LOGICIEL MIPELEC			
AUTEUR(S) : GMV-Équipe ACIBS GMV			
RESUME : Manuel utilisateur MIPELEC			
DOCUMENTS RATTACHES : Ce document vit seul.		**LOCALISATION :**	
VOLUME : 1	**NBRE TOTAL DE PAGES : 19** **DONT PAGES LIMINAIRES : 5** **NBRE DE PAGES SUPPL. : 0**	**DOCUMENT COMPOSITE : N**	**LANGUE : FR**
GESTION DE CONF. : NG	**RESP. GEST. CONF. :**		
CAUSE D'EVOLUTION : Actualisation pour MipElec V2.1 **Réorganisation de MMI**			
CONTRAT : Néant			
SYSTÈME HÔTE : **Microsoft Word 15.0 (15.0.4833)** **C:\DATA\Gdoc\ModeleGDOCIndus.dot** **Version GDOC : v4.2.2.2** **Base projet : Y:\Gdoc\CNESACIBS_PALOMA**			

DIFFUSION INTERNE

Nom	Sigle	Bpi	Observations
GOESTER Jean-François			

DIFFUSION EXTERNE

Nom	Sigle	Observations
CARRO Javier		
EITO Ricardo		
LLAMAS Iván		

MODIFICATION

Ed.	Rév.	Date	Référence, Auteur(s), Causes d'évolution
01	01	28/07/2017	intentionnellement vide GMV-Équipe ACIBS GMV Actualisation pour MipElec V2.1 Réorganisation de MMI
01	00	15/04/2016	intentionnellement vide GMV-Équipe ACIBS GMV Création du document

SOMMAIRE

GLOSSAIRE ET LISTE DES PARAMETRES AC & AD

IHM	Interface Homme-Machine

Liste des paramètres AC :

Liste des paramètres AD :

1. GÉNÉRALITÉS

1.1. DOCUMENTS DE RÉFÉRENCE

1.2. DOCUMENTS APPLICABLES

2. GENERALITES

2.1. PRESENTATION

MIPELEC is a Java implementation of the original CNES program written in F77 to compute the optimal trajectory using low-thrust propulsion. Some modifications have been done (notably allowing for an automatic iteration in the solver problem and initial conditions).

The problem is solved using a multivariate zero solver (converting it into a least-square minimization problem solved by means of Levenberg-Marquadt method), where the values to zero are made of the necessary optimality conditions arising from Pontryagin's Maximum Principle. As such, the problem is numerically sensitive and finding the solution strongly depend on the given initial conditions. See CNES paper "An averaging optimal control tool for low-thrust minimum-time transfers" (by J.Fourcade, S.Geffroy and R.Epenoy) for more information on the mathematical modeling and the original version.

Three types of execution are available:

- MMI: launch a swing-based MMI so as to define the problem and solve it (and optionally plot solution).
- Batch: launch a java headless process loading the problem from a parameter xml file and solving it. Results will be written to parameter xml file, information to stdout and tabulated orbit to "resulP1"/"resulP23D" file.
- Subroutine: the computation can also be requested from another Java program using the public static function Mipelec.compute(…).

Generated tabulated orbit file is called "resolP1" (or "resolP23D" for the P23D problem). It is a flat file (%21.15E formatted values) with N rows of 16 parameters (N depends on MMI parameter and transfer duration), tabulated at constant v steps (the integration variable).

P1	t	ν	a	e_x	e_y	h_x	h_y	p_a	p_{ex}	p_{ey}	p_{hx}	p_{hy}	m	p_m	ξ	ψ
P23D	t	ν	a	e_x	e_y	h_x	h_y	p_a	p_{ex}	p_{ey}	p_{hx}	p_{hy}	J	u/u_{max}	ξ	ψ
[units]	days	rev	m										kg		deg	deg

Where **p** values are the problem adjoint variables;. Thrust direction in TNW frame is given by ξ the in-plane component atan(N/T) and ψ the out-of-plane elevation. J is the cost function [m^2/s^3] of problem P23D.

2.2. COMMAND LINE ARGUMENTS

Mipelec is compiled to contain all required dependencies, so as no extra jar is required for executing it. To the normal execution as a java program (eg "java -jar mipelec.jar") several arguments and options can be given in the command line (though mostly they are used for validation and debugging purposes, notably those greyed):

Argument	**Description**	**Comment**
--input <FILE.xml>	Give input parameters file and execute MIPELEC in **batch mode**	Alternative options
--read <FILE.xml>	Give input parameters to initialize MMI	
--output <FILE.xml>	Define name of output file (by default MIPELEC.xml)	

Option switches do not have arguments and can be grouped in a single one (eg –ordf)

Switch	**Description**	**Comment**
-o	Use same input name as output	Incompatible with --output
-r	Do not create "resulP1" file	
-f	"Fixed" mode. Do not optimize (just use given input to compute result solution and generate result file)	Alternative options. Only for batch mode; can be achieved from MMI
-p	"Iter" mode. Force automatic iteration	
-P	"IterBest" mode. Force automatic iteration (return partial solution as waypoint)	
-s	No screen output (debugPrint-1)	Alternative options. Only for batch mode; can be achieved from MMI

2.3. MIPELEC IHM

2.3.1. Main Window

The following main actions are available:

- **Load**: Read a mipelec parameters file (eventually can also read an old F77 format file).
- **Save Context**: Store MMI values as a mipelec parameters file [XML format].
- **Save Results:** Store computation results, context file and console data.
- **Compute/Stop compute**: Toggle button to execute computation (or abort current one). Launching an execution will automatically toggle to the Console tab and execute a separate JVM running this process in batch mode to compute solution to given problem.

  If the MMI fields contain errors (value outside “valid range”; it is shown in red in the MMI) user will be requested confirmation, but execution can proceed (program may later fail if they are really bad, or take a very long time if too much accuracy was requested).
- **Clear Results:** button to clear the text in the console tab.
- **Clear Messages:** button to clear the error console.
- **Parameters**: access detailed algorithm parameters in a dedicated modal window.
- **Quit**: button to leave MMI (and eventually kill process if running). Note the window “X” has not been disabled and will also kill the process (without asking for confirmation if still running).

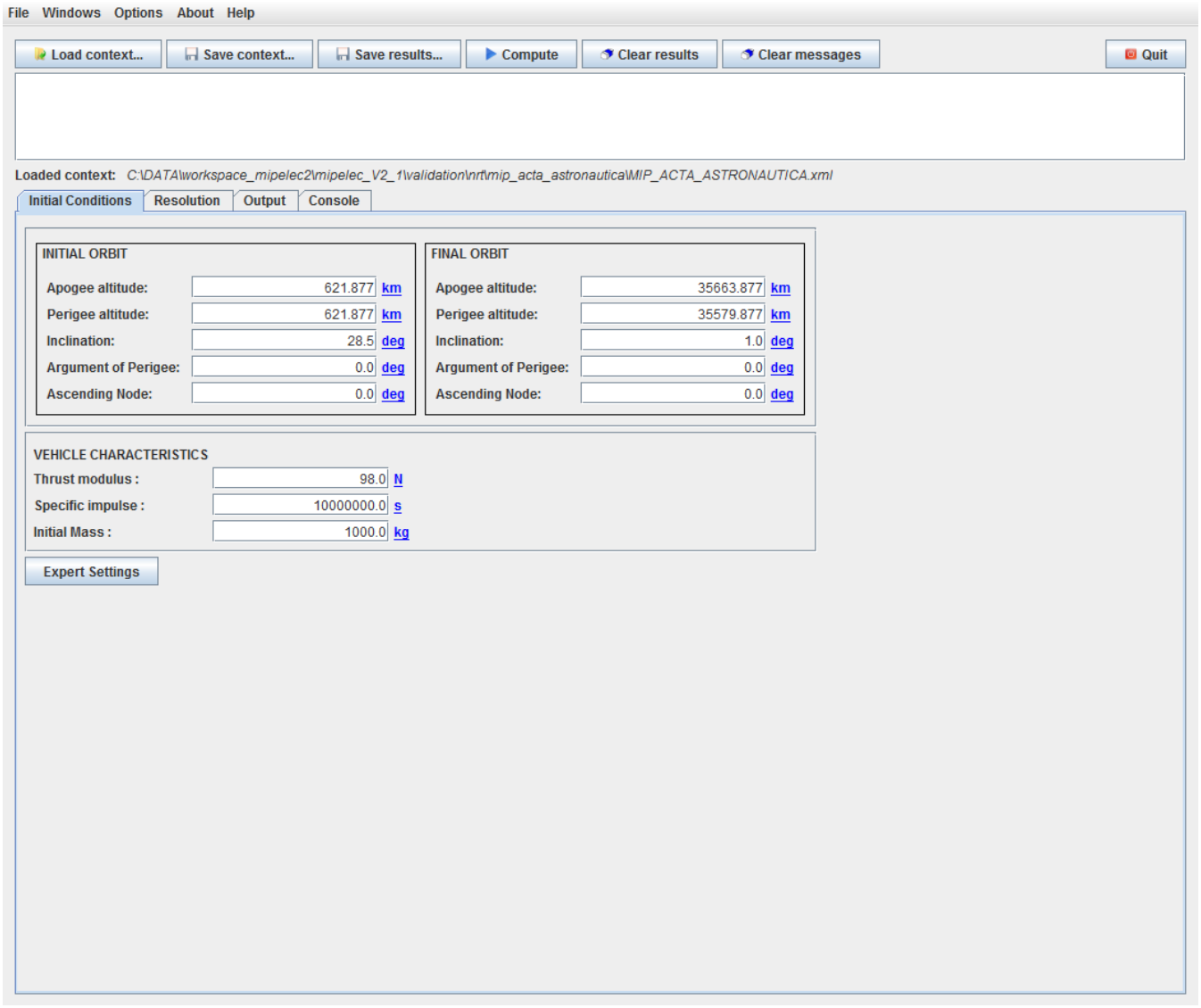

Figure 1: Initial Conditions tab

The **Initial Conditions** tab contains the definition of the problem to be solved. Main data are initial and final orbit (as a 5 element state vector, since true anomaly/longitude is not present in the equations). Apogee and Perigee are defined as an altitude from a spherical body radius (that can be found/modified in the Parameters panel). Other problem required data are initial mass and thrust specific impulse and maximum value.

On the **Resolution** tab two Different problems can be handled:

- **P23D**: find trajectory with minimum energy (WITHOUT respecting thruster constraint). User must also define for this case the desired transfer longitude “L1”.

- **P1**: find trajectory with minimum time (at constant thruster).

Solver state vector initial values can be either:

- **Manual** : use given adjoint values to initialize solver.

- **Auto**: do an automatic initialization of Solver, initially converging on P23D problem (starting from ~zero values) and using that solution to initialize P1 problem (if computing it; user can also give a manual correction factor “LD” to P23D estimated value for L1)

Four Different modes are allowed for execution:

- **Fixed**: use given adjoint values as final values (do NOT optimize). Used whenever we already have a computed solution parameters file and we just want to recreate the resulP1 file. This mode is not compatible with Auto parameter initialization!

- **Simple**: do a solver execution starting from desired initial values (either given or automatic)

- **Iter**: same as *Simple*, but do several iterations (each executing the solver), sampling L1 and the target end orbit so as to iteratively approach the solver initial conditions to a solvable problem.

- **IterBest**: same as *IterAuto*, but allow, in case of failure to converge, to return with the solution to a target orbit that is the closest one for which a solution was found (in that case, the solution will be returned as a WayPoint).

Iter is the default mode (as should be best for most usages). In iterative mode, the user is allowed (but not required at all and most likely NOT needed) to enter waypoints, or intermediate target orbits allowing to “direct” the iterative process (eg converge first in inclination, then change orbit shape). The ClearWP button (hidden if none present) will remove them all if no longer wanted.

> If the transfer goes to very highly retrograde orbits (and no waypoints are present) the program may automatically insert some with lower inclination (as it is known that convergence to that is very slow).

In the Manual and Fixed mode, the Solver parameters are given (5 for the equinoctial elements rates and, for the P1 problem, 2 more m and l). Note these values are normalized (with DU=a1, MU=m0 and TU so as to make μ==1).

On exit, if the problem was correctly solved a file will be generated (**resulP1** for P1 problem and **resulP23D** for P23D problem) containing a tabulation of integration for the solution found. This tabulation will contain the number of points per orbit revolution specified in the MMI (use 0 to impose the fixed number of points defined in the Configuration window).

The **Console** tab contains the printout of the problem execution, mostly the problem statement and solution information, but will be more verbose if a debug mode has been enabled.

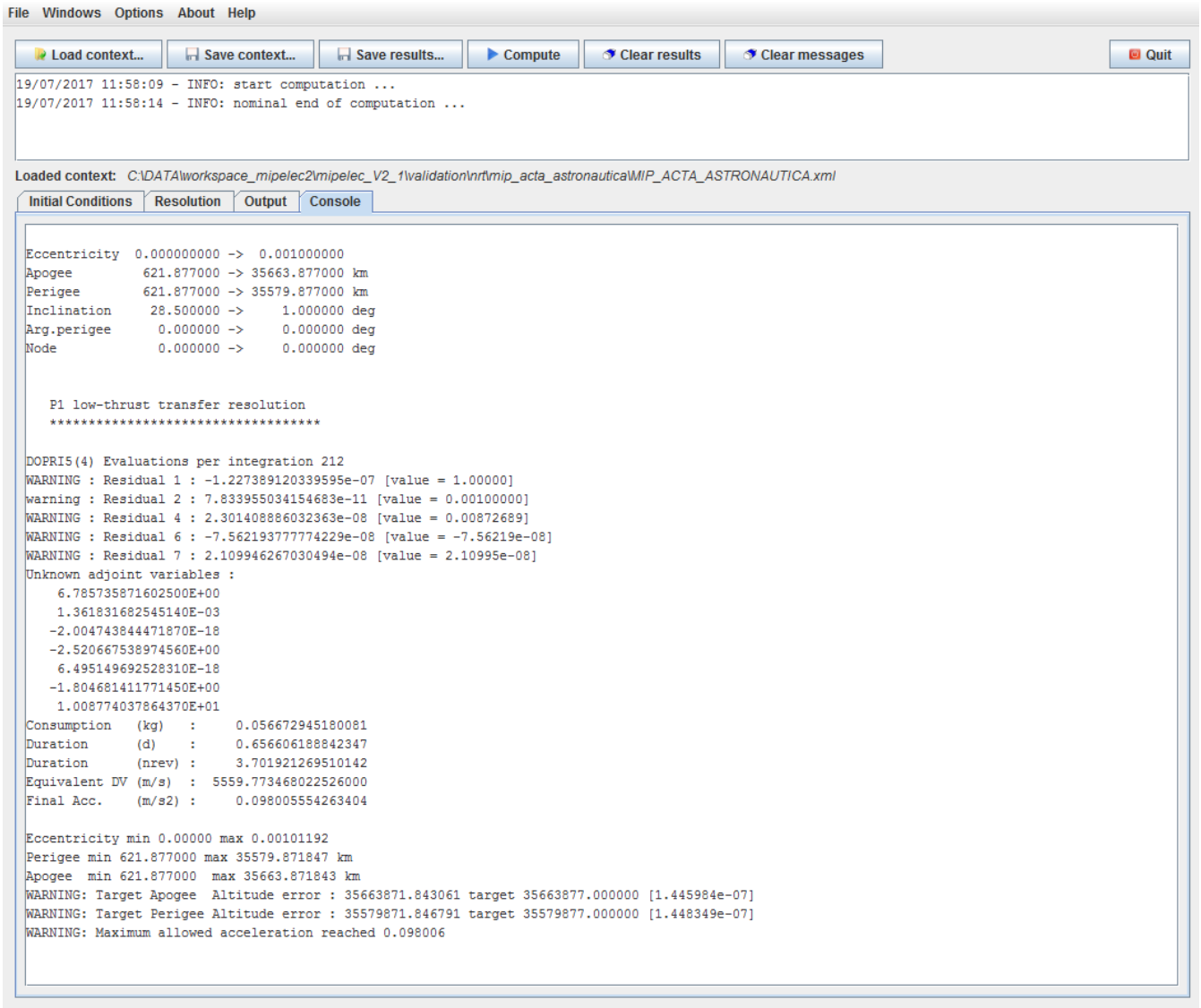

Figure 2: Console tab

The **Output** tab contains the graphic controls (if plots of the solution are desired), so as to display plots with JFreeChart.

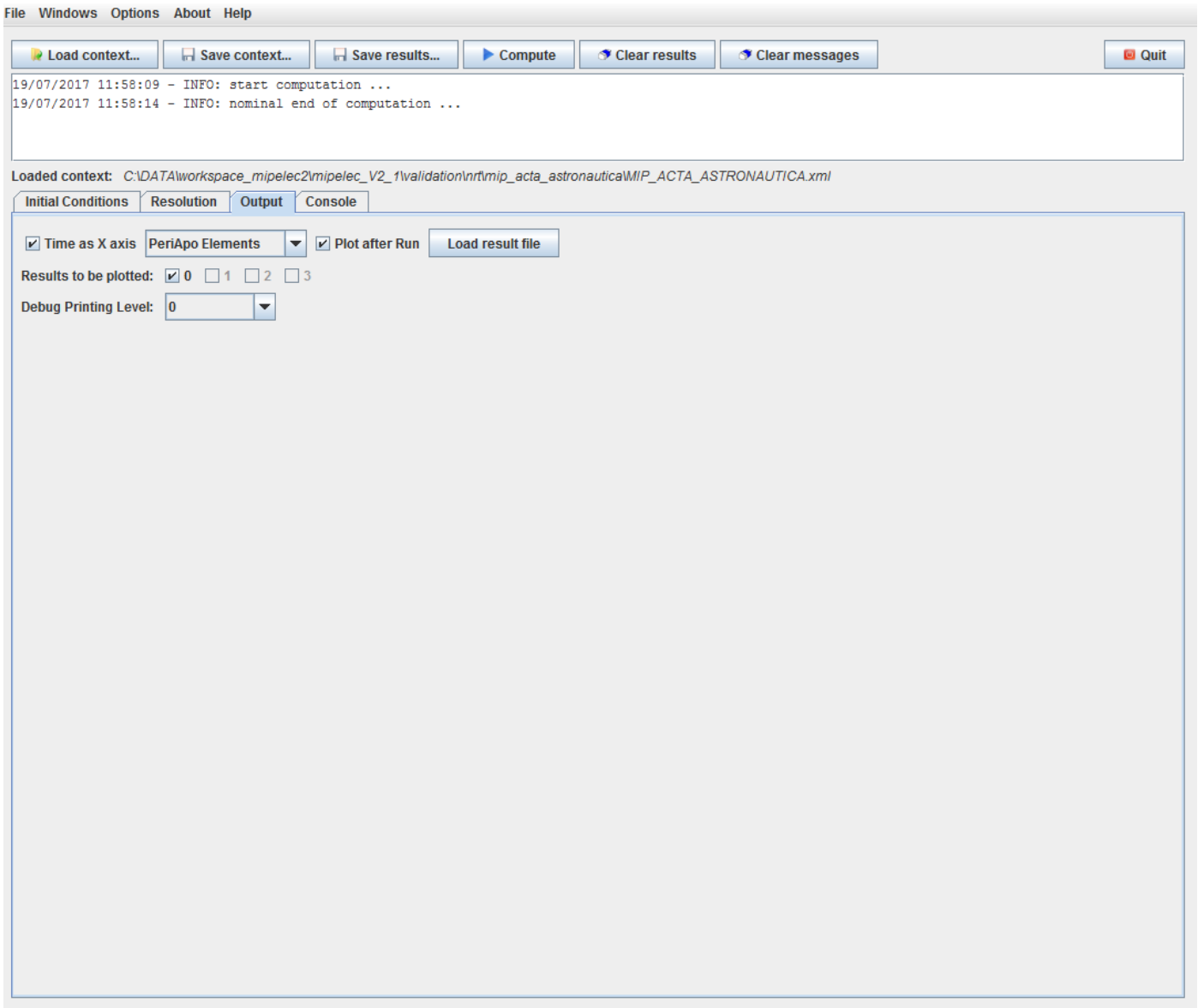

Figure 3 Output tab

- **Time as X Axis**: if chosen the duration (in days) will be used as X axis (of plots using it). If not chosen, the number of revolutions (orbits) will be used instead.
- List choice of **predefined plots**; mostly orbital elements, adjoints and thrust vector direction. A new plot window will be launched directly after changing this selector (if a result file was previously loaded).
- **Plot after run**: if chosen, after program exists results will be automatically loaded and the current selected plot launched.
- **Load result file**: manually select a “resuIP1”/”resuIP23D” file to plot.

- **Debug-print level** [0,1,2] (though *most likely not desired*, as it is just a development feature). Note this is not affected by the above buttons, action is direct and not part of the .

- **Results to be plotted**: allow simultaneous plot of previously loaded files (by default only last == 0 is plotted)

  Loaded values are displayed on a time-shifting basis (so when a new one is loaded in [0] previous one takes place [1], etc and oldest one is discarded).

Graphics can be then controlled using standard JFreeChart mouse and menu features (zoom, save to png, etc).

2.3.2. Advanced Expert Settings window

Not to be used nominally. Playing with these values can possibly worsen the integrator and solver convergence!!!

- **Apply**: save current configuration and close window.

  If the fields contain errors (value outside “valid range”; shown in red) user will be requested confirmation, but apply can proceed (program may later fail if they are really bad).

- **Reset Defaults**: discard changes (if any), reset configuration values to predefined ones and close window.

- **Cancel**: discard changes (if any) and close window.

This window contains configuration parameters use to finely tune the algorithms.

2.3.2.1. Physical Parameters tab

The Physical Parameters tab in the Expert Settings window contains the following parameters:

Model parameters:

- **NBTPS**: number of points used in discretization of P23D solution so as to search for maximum thrust used (used for estimating L1 in automatic P1 first initialization)

- **Gravitational Parameter**: Body´s central gravitational attraction GM

- **Standard gravity**: Value of gravity at planet surface; used to relate specific impulse to mass rate.

- **Spherical radius**: Radius of body assuming a spherical s

2.3.2.2. Integration

The Integration tab in the Expert Settings window contains the following parameters:

Model solution warnings:

- **Maximum allowed acceleration**: warn if final acceleration exceeds this value
- **Minimum allowed perigee altitude**: warn if perigee descends below this value
- **Maximum allowed apogee altitude**: warn if apogee exceeds this value
- **Maximum allowed eccentricity**: warn if eccentricity exceeds this value

Note that these warnings often indicate a non-valid solution (either physically or because model assumptions have been violated).

Numerical integrator (Dormand-Prince 5(4)) configuration:

- **Minimum integration step**: Minimum step to be used [1E-8]. If required step needs to be smaller to maintain requested tolerance, the integration will abort.

  Note that while in theory reducing it allows for integrations to proceed, in practice it may lead to extremely long integration times (and often to just throw several steps later). Allowed range is set to [1.E-12, 1E-3], though it better not exit [1.E-8, 1.E-6].
- **Maximum integration step**: Maximum step to be used by integrator [0.05].

  Since integration interval is normalized to [0-1] this (and the above) step value use those same units. This parameter may affect computation speed of well-conditioned cases (at the possible expense of some accuracy). Allowed range is set to [0.001, 0.1].

2.3.2.3. Tolerances tab

The Tolerances tab in the Expert Settings window contains the following parameters:

Numerical integrator (Dormand-Prince 5(4)) configuration:

- **step relative tolerance**: Per-component relative accuracy to achieve at each step [1.E-6].

  Requesting higher accuracy may hinder the integrator ability to converge without going to very small steps (notice it must be kept at EVERY step, so a small “noisy transition” in the differential function may provoke an abort if too much accuracy is required). Allowed range is set to [1.E-16, 1E-3], though better not exit [1.E-8, 1.E-4].
- **step absolute tolerance**: Per-component absolute accuracy to achieve at each step [1.E-16]

  The actual tolerance is computed by the integrator as AbsTOL + RelTOL * ABS(value), so this threshold serves as a guard against values being too close to zero (for a multivariate function it is actually some sort of rms of the estimated error with respect to the above tolerance for each component). Allowed range is set to [1.E-16, 1E-3], though better not exit [1.E-12, 1.E-6].

  Since the integrated values are mostly “normalized” (but for very high inclination, close to 180º, where the inclination vector can become quite large since tan(i/2) is used, all variables remain often no larger than O(1)), this value should likely be much smaller than the relative tolerance (otherwise we are “hiding the

relative tolerance" for all the integration range). This problem is likely to happen when the integrated variable remains always very close to 0 (eg O(E-18), which happens often in the case of equatorial/circular obits).

Since that tolerance (abs+rel*val) is used as a denominator (error/tol)^2 a value <= 0 should NEVER be used. If we want to remove the influence of a particular variable to the integrator step control, we can set the relative tolerance to 0 and the absolute to a value large enough (larger than the highest estimated error).

We can define tolerance for each integrated parameter or use one for all.

2.3.2.4. Solver (Levenberg-Marquadt) configuration:

Configuration of the non-linear least-square solver.

- **Maximum number of solver evaluations** : number of iterations on linear least squares before declaring non-convergence [50].

  If solver does not converge reasonably quickly there is a good chance it will not do it either even when a large number of iterations is given. Allowed range is [10,100] but anywhere between [25, 50] should be better.

- **cost relative tolerance**: Desired relative error in the sum of squares [1E-10].

- **parameter relative tolerance**: Desired relative error in the approximate solution parameters [1E-10].

- **ortho tolerance**: Desired max cosine on the orthogonality between the function vector and the columns of the Jacobian [1E-10]. Allowed range is (0,0.01).

- **weights**: how to weight each of the minimized parameters [1.0].

  If we want to remove the influence of a particular variable to the solver control (so at the end it will not be at minimum, not recommended), we can set the weight to 0. If a variable is very small (but problem is very sensible to small variations in this value) we can give it a large weight.

Other unused (default values always used) LMA parameters are:

initialStepBoundFactor: Positive input variable used in determining the initial step bound. This bound is set to the product of initialStepBoundFactor and the euclidean norm of diag * x if non-zero, or else to initialStepBoundFactor itself. In most cases factor should lie in the interval (0.1, 100.0). 100 is a generally recommended value.

threshold: Desired threshold for QR ranking. If the squared norm of a column vector is smaller or equal to this threshold during QR decomposition, it is considered to be a zero vector and hence the rank of the matrix is reduced

As we are using a least square minimum-search instead of multivariate zero search (gauss-newton) convergence may lead to a minimum that is not a zero (not likely). A warning is issued if final convergence residuals are not very close to zero (>1E-12).

2.3.2.5. Finite difference of Jacobian

The epsilon value used to separate each value from the central one is computed based on the formula δ = *relFD* * MAX(ABS(value), *minFD*), where:

- **Jacobian Relative epsilon**: [relFD] proportional factor to separate FD from value [1E-4]
- **Jacobian value minimum**: [minFD] minimum value to use [1E-6]

  This value needs to be used to avoid too small steps when the variable crosses zero. Note though that when some variable is always very small (in case of circular or equatorial orbits), the minFD value could become much larger than the actual value itself, and the orbit transfer may be very sensitive to such a large variation, rendering the jacobian hard to use.

Special care must be given to introduce coherent parameters; it is dubious that asking too much precision in the jacobian finite difference makes any sense (or is just random noise) when its function value is computed using an integral with a given accuracy. A rule of thumb is thus that the relFD value should not be bigger than the square root of the expected-value-accuracy. Note though that these δ values are applied to each element in the BVP state vector (the initial adjoint elements) while "noise" is found in the residuals; the integrator end conditions (final orbit equinoctial elements) so the translation is not necessarily linear.

2.4. AUTO-ITERATIVE PROCESS

Since the solver needs convenient initial values to converge to an appropriate solution (more so when the transfer requires large changes in the orbit), a convenient way to generate them is to solve simpler problems and then use that solution as the initial value of a modified problem that gets closer to the desired one.

Iteration is done in the given input perigee-apogee orbit, by choosing a "linear vector" between the target and initial orbit and trying intermediate orbits in that representation. Required orbit change is thus reduced N times by a given factor until a valid solution is found, and those adjoint values are used to retry again for M iterations. This is not always the best strategy (some problems may converge better if reducing inclination, so as to make it close to equatorial, or if changing first the orbit shape and then the orbit plane).

For harder problems, the default strategy may not be good enough or may be too long. In those cases the program can generate intermediate wayPoints (so solution search process is directed from one to the next). Note that those waypoints are NOT intermediate transfer points, just direction helps to the iterative convergence problem (which is always from initial to target orbit).

2.5. SOLVER AND INTEGRATOR CONVERGENCE ISSUES

During the convergence process (mostly in iterative runs), and especially when activating the debug 1 level (debug 2 is highly not recommended as too much output often useless is printed) failed iterations show a given number of reasons for the failure. This section tries to explain the reasons behind any of those cases (though most likely the unaware user should neither activate any debug level nor be too much concerned about these issues).

Many of these problems (often with weird cases) are also problems without a real solution or whose solution violates some of the hypothesis of the underlying averaging problem; they will most likely be affected by the warning on "*Maximum allowed acceleration ratio reached*". In this case it is not due to a computing or numerical issue but a case where the stated problem is simply not feasible (though there is no guarantee as solver could simply be searching in a space solution leading to invalid transfers).

Using some weird advanced parameters could improve or greatly worsen the occurrence of these problems, as well as have a significant influence on the computation time (which will be often small for easy or good input adjoint values, but could become quite large (even up to ~5 min) on some of the unsolved ones.

The differential equations used to compute the optimal transfer can become singular, noisy or discrete for certain

combination of orbit and adjoint state vector (especially if the problem is ill formed and violates the underlying averaging assumptions). Thus, when they are integrated, the DOPRI will abort ("*Integration needs smaller step"*) and we cannot compute the solution.

It may also happen from time to time that the solver goes towards a space solution that violates some constraints on the state vector (mostly that for P1 problem we want to hold positive longitude rate and negative mass rate). In this case, we will get either "*Constraint violation in solution*" or "*Solver bounds reached*".

And due to the convergence mechanism it may happen that the solver converges or not to a solution ("*Solver does not converge*"), and in some extreme cases it may converge on a local minima of the least-squares that is NOT a zero of the solution ("*Optimization failed (Residual too large)*").

Most of these error cases will depend on the advanced parameters used and, mostly, on the difficulty of the problem (easy problems will converge with most reasonable parameters, harder ones might not even with any combination).

Note than using auto-iterative mode, a single failure is not quite critical (we will retry with easier conditions). However some particular problems get harder when the LMA tries to converge to either a non-zero local minima or to a solution close to zero where jacobians become close to singular (in some cases Q-R errors solving the linear system might also be seen).

Typical problems where solver has issues converging are either those of transfers to/from retrograde orbits (close to 180º as equinoctial elements become singular) and strong changes in ascending node´s right ascension at medium to high inclination. Note that nevertheless those problems involve transfer strategies (when they are computed by the solver) which are often not physically feasible (such as descending the perigee below the Earth´s surface) or weird (increasing the apogee to very large values); many times implying intermediate eccentricities close to 1. Though a warning is provided for those cases the numerical methods implemented rely on unconstrained optimization so no "valid" alternative strategy can be found by MIPELEC (not even wayPoints will help; the optimal solution to the transfer is just not physically realistic).

2.6. SUB-ROUTINE MODE

The provided jar file is mostly meant as a stand-alone program (including all required dependencies). It can however be also used as part of the Java CLASSPATH for another Java program, by calling the reentrant static method Mipelec.compute

```
BVSolution compute(final double[] periApo0, double[] periApo1, final double[] initParam,
            double fmax, double Isp, double mass0, int tabStep, double LD, int pb,
            int ini, int mode, double[][] wayPoints);
```

See javadoc for Mipelec class for more detailed information on public methods and constants to be used, content of return class BVSolution, configuration of global variables via static setters (notably advanced parameters via the Configuration class; but in this mode it is also likely the caller wants to initially disable stdout and result file printing with the *setCreateResultFile(false)* and *setDebugPrint(DEBUG_SILENT)* method calls)