LUCA2 - AN ENHANCED LONG-TERM UTILITY FOR COLLISION ANALYSIS

Jonas Radtke, Sven Mueller, Volker Schaus, and Enrico Stoll

Technische Universitaet Braunschweig, Institute of Space Systems, Hermann-Blenk Str. 23, 38108 Braunschweig, Germany, Email: {j.radtke, sv.mueller, v.schaus, e.stoll}@tu-braunschweig.de

ABSTRACT

To assess the impact of different future developments of spacefaring, long-term projections of the space debris environment are performed. In these projections, all major source and sink mechanisms that impact its evolution are considered. They include launches of new objects, propagation of their orbital states, collision rate analysis and performing of fragmentation events (collisions and explosions). Based on these capabilities, different scenarios are simulated and their results are compared. These scenarios consider assumptions for the future launch traffic, solar-activity variations and the implementation of debris mitigation, such as post-mission disposal strategies, and remediation measures, such as active debris removal.
LUCA (Long Term Utility for Collision Analysis) is a software tool developed at the Institute of Space Systems (IRAS) of the Technische Universitaet Braunschweig to perform such simulations. It has been used in several ESA contracts, IADC studies and studies supported by the German Aerospace Center (DLR). The development of this tool started in the late 1990ies. In recent year, the historic implementation of LUCA faced several challenges. A combination of enhanced computational power and the demand for much more detailed simulations lead to the decision of a complete re-implementation of this tool, leading to its latest version, LUCA2. The main goals for this new implementation were to allow for maximum flexibility in the models used for single sinks and sources, the extension of its functionalities as well as the consideration of new standards in software engineering.
In this paper, at first the overall software architecture of the new tool is described. For this, an object-oriented concept in Fortran 2008 has been developed, which allows easy exchange of different models by using plugins via a generic interface. Following, the single functionalities of the tool are introduced, including implemented methods for collision rate determination, propagation, solar cycle forecasting, break-up modeling and the determination of post-mission disposal orbits. Exemplary results of these core functionalities are presented. Lastly, results of long-term projections of the space debris environment used for the validation of LUCA2 are shown. These results furthermore contain a sensitivity analysis of the new tool to user-defined input parameters, such as different solar cycles or launch traffic assumptions, but also to different physical models, for instance using different collision rate determination algorithms.

Key words: LUCA; long-term evolution; CUBE; Orbit Trace; NASA Breakup Model.

1. INTRODUCTION

Long-term projections of the space debris environment are performed to compare the impact of different future scenarios on its evolution. Exemplary, they can be used to analyse the effects of passive mitigation measures (such as post-mission disposal (PMD) [20]) and active remediation (such as active debris removal (ADR) [10]), but also the impact from possibly disruptive events, such as the release of the so-called Mega-Constellations consisting of several hundreds of satellites [2].

To perform this task, several tools exist which are comparable in their general functionality: starting from an initial space debris population at a certain date, the complete environment is propagated over the desired time frame. During this propagation, all major source and sink mechanisms of space debris are simulated. These models include amongst others the consideration of main, most secular, orbital perturbations, the calculation of collision rates and triggering of collisions, inclusion of future launch traffic, and assumptions regarding future space debris mitigation measures. As all the models include uncertainties, depending on the study, certain events are triggered randomly. In all cases, this includes the collisions, but it can be extended to all other models as well. Models to project the long-term space debris environment evolution include, amongst others, the Italian SDM [18], the British Damage [9], ESA's Delta [13], NASA's LEGEND [12], and CNES'MEDEE [4].

LUCA (Long term utility for collision analysis) is a tool to project the long-term evolution of the space debris environment which has been developed at the Institute

of Space Systems (IRAS, former Institute of Aerospace Systems, ILR) since the late 1990s. It has been used in a large number of projects and publications (for example [16], [17], [2]), and more recently contributed to still on-going studies performed in the frame of the Inter-Agency Space Debris Coordination Committee (IADC) on behalf of the German Aerospace Center (DLR).

Being under development for about 20 years now, it was decided to completely re-develop the code. Reasons for this were on the one-hand-side, to be able to easier exploit modern computation techniques, such as GPU computing, by using newer software standards. The main reason though was to increase the overall flexibility and functionality of the tool. In its first version, not much emphasis was put on a modular implementation, making it almost impossible to exchange certain models, for example to calculate the collision rates between on-orbit objects.

2. LUCA2

LUCA2 is a tool to project the long-term evolution of the space debris environment. Using the same starting point in terms of an initial population together with a defined scenario, simulations are performed as Monte-Carlo runs, in which certain events and/or inputs are varied randomly. The random variations performed thereby depend on the scenario to be simulated, but include the triggering of collisions in almost all cases though, based on calculated collision rates. The top level work flow of a LUCA2 simulation is shown in Figure 1. The plugins that are used during a simulation depend on the scenario that is to be simulated. A standard scenario usually calls the following plugins:

- Create solar forecast.
- Read initial population.
- Reliability and lifetime reduction of active satellites.
- Propagate population.
- Determine encounters and collision rates.
- Trigger collisions.
- Perform post-mission disposal manoeuvres.
- Write outputs.

More details on some of the plugins are given in Section 3. To analyse the different scenarios, the results from all single Monte-Carlo runs are taken and statistics of for example the number and distribution of objects, collisions etc. are calculated.

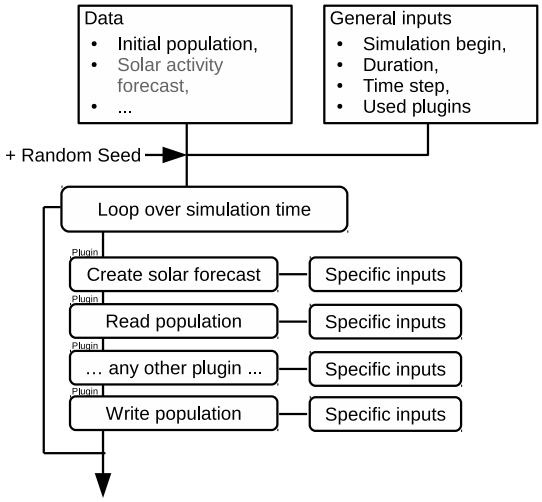

Figure 1. General work flow of LUCA2.

2.1. Implementation

One of the main goals for re-development of LUCA was to modularize the program to allow a flexible exchange of functionalities. To be able to re-use parts of the code from LUCA as well as to easily integrate available libraries at the Institute of Space Systems, LUCA2 was written in Fortran (standard 2008). To achieve the desired modularization, a Fortran approach was developed, which consists of three levels: Fortran-modules, object-oriented classes and plugins. LUCA2 consists of all three of theses levels, the schematics of this approach are shown in Figure 2.

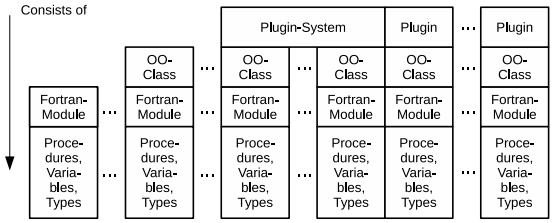

Figure 2. Three levels of modularization in LUCA2. "OO-Class" stands for "object-oriented class".

Fortran modules are directly available in Fortran and used to group procedures, variables and types that belong to the same context. One typical example is a module "files", which contains all code related to file handling. Modules can furthermore be used in other parts of the software, in which the public attributes can be used. All code in LUCA2 is written in Fortran modules.

Object-oriented classes

Object-oriented classes are types of software components used in a software design paradigm called Object-Oriented Programming (OOP). In this paradigm, variables and procedures which belong together are bundled to form classes. As a result, a class can be seen as the def-

inition of a software component which has certain properties, defined through the variables of the class. The component also has certain capabilities, defined through the procedures of the class. This allows to create a model of the part of reality which shall be represented in software. An input file, for example, may be represented then in its own class. A property, or variable, of this class could be its `file_name`. Capabilities, or procedures, of the input file class could be `set_file_name()`, `open()` and `read_line()`. An input file could then be read in a program by using the class as a template to create a concrete object from it. This object would represent the concrete input file that should be read (as opposed to an abstract input file represented by the class). The input file's capabilities in form of the procedures listed above could then be used to read the file in an easy-to-read manner. This approach also leads to an easier way for creating software documentation, demonstrated by Figure 3. For the LUCA2 development, one class consists of a special kind of Fortran module containing a special kind of Fortran type putting Fortran 2008's OOP features to use.

Plugins provide the single functionalities of LUCA2. For this, a plugin system has been designed, which consists of a generic plugin class with a generic interface. The plugins with different functionalities are derived from the generic plugin and can then be extended by plugin specific needs.

Using this approach for modularization, the software itself is split into two parts: the "luca core" and the "plugins". The scheme of this separation together with an excerpt of available classes is shown in Figure 3.

The **LUCA core** uses the first two levels of modularization: All code is written in Fortran modules, which are provided to the programmer in object-oriented classes. Its task is to manage a long-term simulation. For this, most simply, it runs a loop over a time frame with a time step. Always valid for the current time step, it furthermore contains the debris population. A debris population consists of an array of debris objects, which themselves contain all properties needed during the long-term runs, such as an identification, orbital parameters, object characteristics etc. All entities are themselves objects, which can be accessed using get functions. This helps protecting values and keeping all properties of the objects consistent: for example, if the semi major axis is to be changed, it has to be done using the set function "update_orbit", which automatically adopts all other values which depend on the semimajor axis. Additionally to debris objects, a population contains an array of encounters, which contain a pair of debris objects as well as characteristics of the encounter, such as the collision rate and the relative velocity. Alongside with these main functionalities, LUCA core provides a large number of support classes to the programmer. Examples are a class to handle input files, a settings manager to handle all main inputs (simulation time frame, time step etc.), classes for fast access to certain types of debris objects (for example all active objects or all objects of a certain kind), a category rules store, and a sophisticated logging framework. The category rules store is an object-oriented class, which allows defining rules (or rather a certain behaviour during the long-term simulation) of freely definable categories for on-orbit-objects (such as their type, orbit region, mass etc.), without needing to know beforehand, how and which categories will be needed in a simulation.

The **plugin system** provides a generic plugin class that can be used to derive specific plugins. The plugin class provides all needed routines to create and destroy instances of a plugin. It furthermore takes care of the correct call of specified plugins. The plugin basically provides a generic interface between the LUCA core and the specific plugin. This interface was broken down to the minimum similarity between all plugins: their intention to process a debris population. Therefore, the plugin interface always only provides the current debris population (including objects and encounters), the starting point of the simulation and the duration of the current time step. The plugin then receives a local reference list to the population, which can be updated as desired to perform the tasks of the plugin and in the end be returned to LUCA core.

3. PLUGINS

As of date, 17 plugins are available for LUCA2, some of which are alternatives for the same functionalities

- Create solar and geomagnetic forecasts.
- Launch Traffic: Either based on a repeating launch cycle or based on the initial population.
- Include and maintain Mega-Constellations.
- Reliability and lifetime reduction of active payloads.
- Explosion: Either by using a fixed number of explosions per year based on past events or by using explosion rates for freely definable object types.
- Propagation of population.
- Collision rate determination using either Orbit-Trace or Cube.
- Collision avoidance.
- Triggering of collisions using two different implementations of the NASA Breakup Model.
- Post-mission disposal to freely definable remaining lifetime orbits.
- Active Debris Removal.
- Write population and collision rates to file.

A subset of the plugins are presented in this section in more detail.

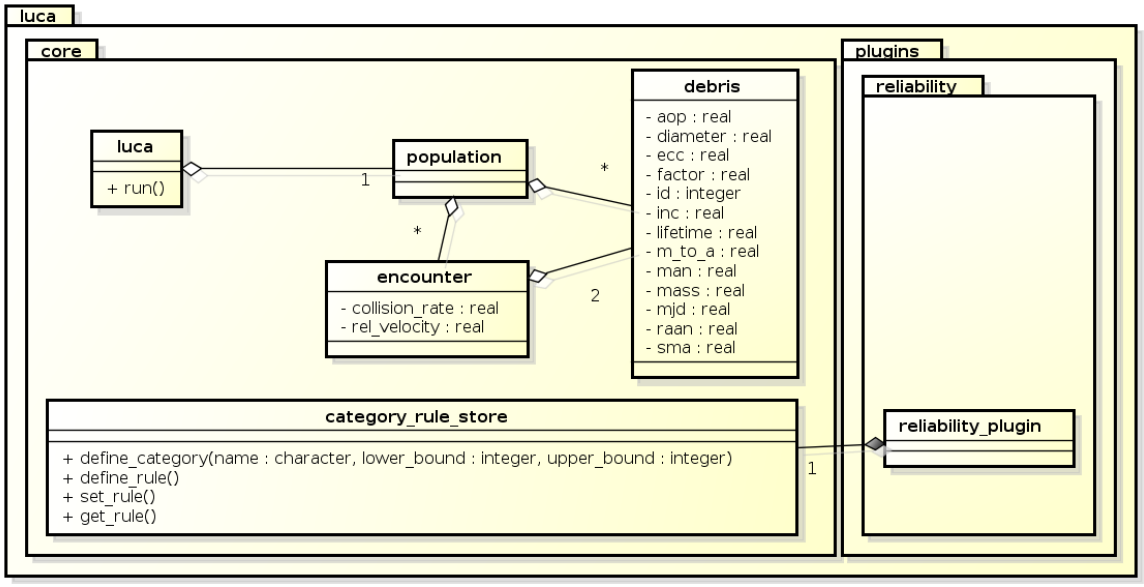

Figure 3. Visualization of central LUCA2 classes and their relationships. `core`, `plugins` *and* `reliability` *are LUCA2 subsystems which contain the classes. The lines beginning with a diamond symbolize a "has x" relationship, with "x" being the cardinality of the relation. For example, one* `luca` *(object) has exactly one* `population` *(object). An asterisk indicates an arbitrary amount of objects.*

3.1. Solar and Geomagnetic Activity Forecast

LUCA2 can be launched with any externally provided solar and geomagnetic activity forecast. Nevertheless, if required for the simulations, it is capable of creating its own forecast. In this case, a different forecast is used for every single Monte-Carlo run. Currently, two different types of projection are supported: either creating a random combination of past solar cycles, or a series of Monte-Carlo sampled cycles, which is recommended for long-term solar flux forecasting by the ISO 27852:2011 standard [6]. In general, for the long-term simulations themselves, random combinations of past cycles are used, to include both cycles with low and those with high activities. An exemplary forecast is shown in Figure 4.

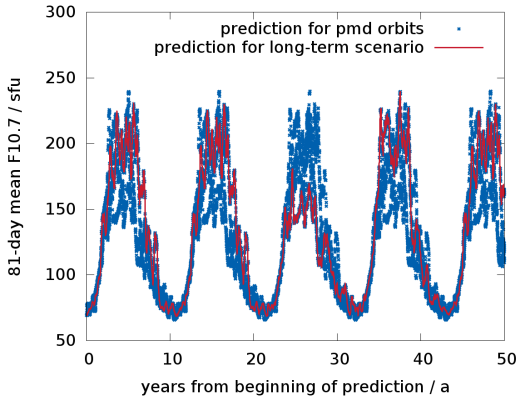

Figure 4. Exemplary solar flux forecast, using both method supported by LUCA2: a random combination of past solar cycles and a random sampling from past solar cycles.

3.2. Collision Rate Determination

For the collision rate determination, two different algorithms are available in LUCA2: One Orbit-Trace algorithm and one CUBE algorithm.

Orbit-Trace The Orbit-Trace method used in LUCA2 is based on a method developed by E.J. Öpik in [14], which was originally derived to calculate collision rates between interplanetary objects. The method basically determines how long space objects stay in a region around the intersection of their orbits to derive a collision rate. This can be calculated using:

$$P = \frac{2 \cdot \sqrt{(r_1 + r_2)^2 - d_{min}^2}}{sin(\alpha) v_2 \cdot T_{u,1} \cdot T_{u,2}}. \quad (1)$$

Here, r_1 and r_2 are the radii of the two objects under investigation, d_{min} is the shortest distance between their two orbits, v_2 the velocity of the impacting object, α the angle between the trajectories at the intersection and $T_{u,1}$ and $T_{u,2}$ are the orbital periods. To also be able to handle objects with identical periods, the implementation has been supplemented with a system of filters to avoid unrealistic encounters between objects within constellations

[15].

Cube The second method available is CUBE, which has been introduced in [11]. The general idea behind this algorithm is to uniformly sample the system in time and determine the collision rates for each of the time steps.

Referring to [11], over a long time step, the number of collisions in a population can be described via:

$$N_c = \int_{t_{begin}}^{t_{end}} P_{i,j}(t)dt = \int_{s=0}^{s=L} \int_{t_s}^{t_{s+1}} P_{i,j}(t)dtds. \tag{2}$$

Here, $P_{i,j}$ describes the collision rate, L the number of time intervals between t_{begin} and t_{end}, and t_s and t_{s+1} the beginning and end of a time interval s. For sufficiently short time intervals, whereas sufficiently means that the collision characteristics do not change significantly, $P_{i,j}$ can be assumed constant during this interval. Therefore, Formula 3.2 can be changed to:

$$N_c = \int_{s=0}^{s=L} [t_{s+1} - t_s] \times P_{i,j}(s)ds. \tag{3}$$

To estimate the collision rate, the approach now assumes that collisions are only possible, if, at one time step, the two objects are located within a small cube. Therefore, the collision rate for two objects at a specific time step is:

$$P_{i,j} = s_i s_j V_{rel} \sigma dU, \tag{4}$$

where s_i and s_j are the spatial densities of the encountering objects in the volume, V_{rel} is the relative velocity between the objects, σ is the collision cross section, and dU is the volume of the cube.

In the implementation used in LUCA2, the Cartesian positions of all objects in the population are calculated at each time step. To avoid aliasing effects between objects, especially in low Earth orbits, the main anomaly of the objects are randomized at each time step [1]. The three coordinates are then converted into integer position indexes to allow a very fast identification of objects within one cube. To additionally fasten this identification progress, the objects are pre-sorted by their orbital altitudes. Currently, in LUCA2 the standard settings from [11] for the cubes' edge length of 10 km and a time step of 5 days are used. A sensitivity analysis on the appropriateness of these value for LUCA2 simulations is outstanding.

The calculated collision rate within a population is highly sensitive to the number of internal randomizations of the mean anomaly (cf. Figure 5). Therefore, additionally to the external randomizations, the number of randomizations of the mean anomaly can be chosen via input file. As default, currently 30 internal randomizations are used, leading to 1500 overall randomizations in a standard long-term simulation with 50 Monte-Carlo runs.

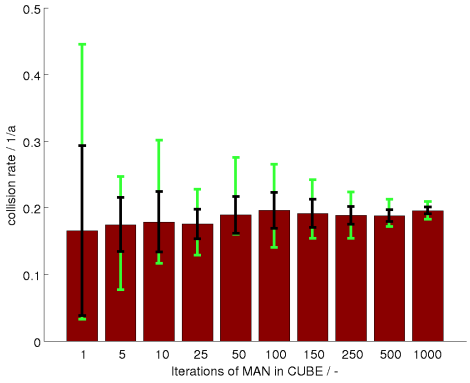

Figure 5. Collision rate calculated with CUBE within a typical space debris population over the number of internal iterations of the mean anomaly. Red boxes show the average, black error bars indicate the standard variation, green error bars the minimum/maximum values. MAN stands for mean anomaly.

Comparison In a simple comparison case, the collision rates calculated with CUBE and Orbit-Trace over time were compared. In this case, a space debris population from MASTER, valid for 1^{st} of January 2013 was propagated with constant solar flux (F10.7 = 130) and geomagnetic activity (AP = 9) over a time frame of 200 years. Figure 6 shows the yearly collision rates using both Orbit-Trace and CUBE. Additionally, a comparison of the same scenario using the old version of LUCA, which uses the Orbit-Trace algorithm, is shown. All three set-ups lead to very similar results. Nevertheless, a larger variance in the collision rates calculated with CUBE can be observed. Furthermore, over the complete time frame, cumulatively, CUBE leads to 16.6 collisions while Orbit-Trace calculates only 15.6 collisions.

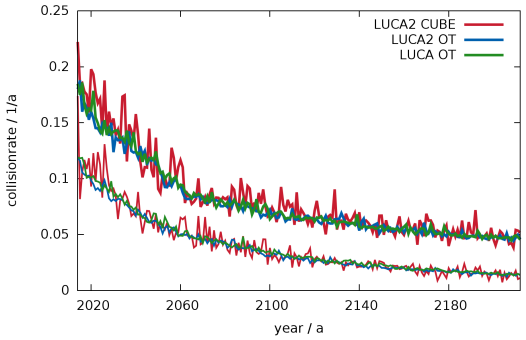

Figure 6. Comparison of collision rates over time calculated with Orbit-Trace in LUCA, and Orbit-Trace and CUBE in LUCA2. Results from one Monte-Carlo run.

3.3. Implementation of the NASA Breakup Model

For LUCA2, the standard NASA Breakup model [7] has been re-implemented, to be in-line with the new coding standards. The advice on the proper implementation given in [8] has been considered. A simplified flow chart of the implementation is given in Figure 7.

In the first step, the number of debris objects to be created is determined using the power law

$$N(L_c) = S6L_c^{-1.6}, \tag{5}$$

for explosion events. Here, L_c is the characteristic length of the fragments and S a unit-less type-dependent scaling factor, to allow account for different explosion objects and types. A set of scaling factors is given in [5]. For catastrophic collisions, the number of fragments is calculated via:

$$N(L_c) = 0.1(M)^{0.75}L_c^{-1.71}. \tag{6}$$

Here, M is the mass (in kg) involved in the collision. If the collision is assumed to be catastrophic, M is simply the sum of the two colliding objects, if it is assumed to be non-catastrophic, it is the product of the mass (in kg) of the smaller object and the relative velocity (in km/s) of that collision. As distinction between catastrophic and non-catastrophic collision, the generally used threshold of $40J/g$ for the energy-to-mass ratio (EMR) is applied. Both equations are valid between 1 mm and 1 m. Once the total number of fragments that are to be created referring to the power laws has been calculated, they are reduced by the number of fragments at 1 m. This is the threshold until where the power laws are applicable.

Starting with the smallest fragment, in a loop all fragments below 1 m are created. This is achieved in five steps:

1. The characteristic length of the fragment is calculated, using rearranged Equations 5 and 6.

2. Using the characteristic length as independent parameter, the area-to-mass ratio is determined. For characteristic lengths larger than 11 cm, the two values are linked with a bi-modal distribution function:

$$\begin{aligned} D_{A/m}(\lambda_c, X) = \\ \alpha(\lambda_c, X)N(\mu_1(\lambda_c), \sigma_1(\lambda_c), X) \\ +(1-\alpha(\lambda_c))N(\mu_2(\lambda_c), \sigma_2(\lambda_c), X), \end{aligned} \tag{7}$$

   for rocket body fragments smaller than 8 cm and payload fragments smaller than 1.7 cm [5]:

$$D_{A/m}(\lambda_c, X) = N(\mu(\lambda_c), \sigma(\lambda_c), X). \tag{8}$$

   Here, $\lambda_c = \log_{10}(L_c)$, $X = \log_{10}(A/m)$, and N refers to a normal distribution. All further needed parameters, which differ for spacecraft and rocket-bodies for the bi-modal distribution for larger objects, are given in [7]. In between the distributions, bridging functions of the form:

$$L_b = 10(\log_1 0(L_c) + 1.76) \tag{9}$$

   for rocket bodies, and

$$L_b = 10(\log_1 0(L_c) + 1.05) \tag{10}$$

   for payloads are used. If $L_c > L_b$ the large particle distribution is used [5].

3. The average cross sectional area of the fragment is determined. For fragments with $L_c < 0.00167m$, the relationship is:

$$A_x = 0.540424 \cdot L_c^2, \tag{11}$$

   for larger fragments it is

$$A_x = 0.556945 \cdot L_c^2.0047077. \tag{12}$$

4. The mass is then determined by simply dividing the cross section by the area-to-mass ratio.

5. Last, the ΔV is needed. It is described by a simple normal distribution. Again, values for different objects types and sizes are given in [7] and [5].

Once all small fragments are calculated, it is checked if the cumulative mass of all fragments is equal to or lower than the mass involved in the fragmentation event. If this is not the case, the loop is repeated until this condition is fulfilled. Else, the remaining mass is distributed along different numbers of random large objects with characteristic lengths above a 1 m. To achieve this, the characteristic lengths is randomly drawn from a uniform distribution between 1 m and the characteristic length of the fragmented object. For catastrophic collisions, this procedure is repeated until all mass is used. For non-catastrophic collisions, only one large fragment is created, during explosion events two to eight.

Figure 8 shows as example the area-to-mass ratio over the characteristic length for the catastrophic collision of a 1000 kg spacecraft. The results resemble very well those shown in [7] and [21].

3.4. Determination of End-of-Life manoeuvres

LUCA2 offers a wide selection of possible end-of-life manoeuvres, depending on the orbital regions objects reside within. For LEO objects, the options are:

- Disposal to an either eccentric or circular orbit with an arbitrary remaining lifetime (max. 50 years).

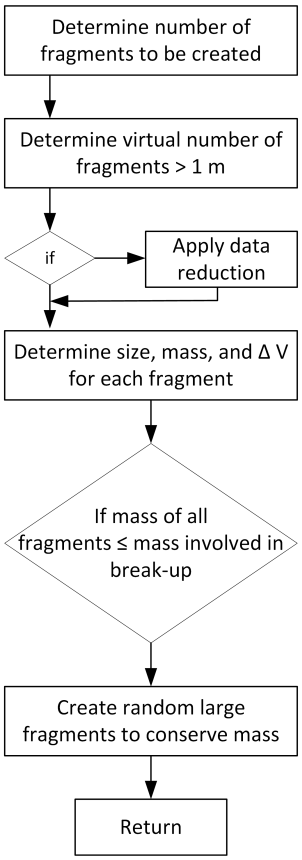

Figure 7. Simplified flow chart of the implementation of the NASA Breakup Model for LUCA2.

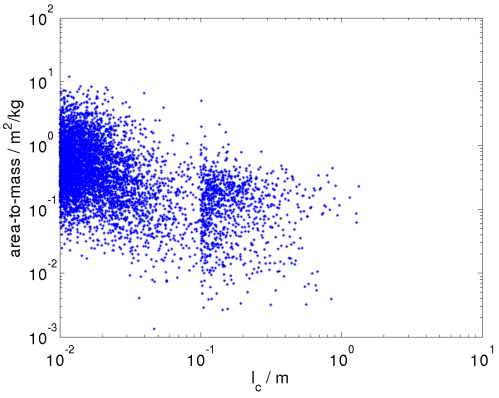

Figure 8. Area-to-mass ratios over characteristic length for the catastrophic collision of a 1000 kg payload with a 10 kg impactor at 10 km/s, computed using the NASA Breakup Model implementation from LUCA2.

- Disposal to an graveyard orbit at an arbitrary altitude.
- Direct re-entry.

For GEO objects, the options are:

- Disposal to an IADC guidelines compliant graveyard orbit.
- Disposal to a graveyard orbit at an arbitrary altitude above or below the GEO region.

The options to be used can be freely defined for each object type separately using the category-rules store. Additionally, a maximum available fuel ratio can be defined, as well as a rule for what to do, in case the fuel is insufficient. In this case, the options "perform manoeuvre as far as possible" or "perform no manoeuvre at all" are available. For the determination of the fuel needed, in all cases Hohmann-Transfers without any losses are assumed.

While the calculation of most disposal orbits is straight forward, the computation of the arbitrary lifetime orbits is more complex. It is achieved by an iterative propagation of the objects to be disposed. In [3], using Regula Falsi for the iteration lead to best results, therefore this method is also used in LUCA2. To ensure that a correct disposal orbit is found for most of the objects, the sensitivity of the implemented algorithm to the number of iterations performed has been tested by calculating eccentric disposal manoeuvres with different remaining lifetimes for all payloads within the population (both active and passive). Figure 9 shows the percentage of correctly calculated orbits over iterations. A disposal orbit is assumed to be correct, if the re-entry occurs within $\pm$ 3 month of the defined disposal lifetime. After 57 iterations, for 99.9% of all objects a correct orbit could be calculated, therefore this value is used in LUCA2. If after these 57 iterations no disposal orbit is found, the manoeuvre is counted as failed.

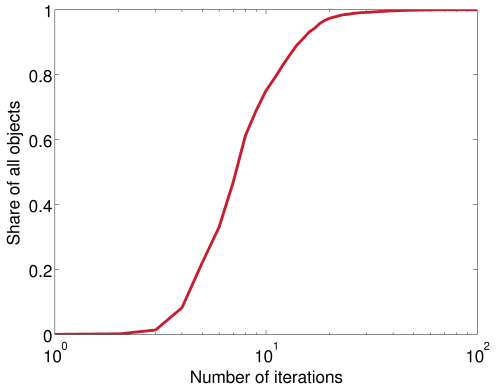

Figure 9. Share of objects for which a satisfying disposal orbit was found over number of iterations needed.

As the disposal orbits are very sensitive to the provided solar and geomagnetic activity, the user can choose between two different options: Using either the same values as for the complete long-term run, or using Monte-Carlo sampled cycles. The first options always leads to precise orbits, which might be wanted in some

simulations. The second option has the advantage of leading to a realistic spread within the actual disposal lifetimes of payloads. This is shown in Figure 10: here, 173000 objects were disposed to 25-year orbits at different times over a 100 year simulation time frame. The red bars show the actually needed re-entry times, if the forecasts for propagation and orbit iteration were identical. The blue bars show needed re-entry times, if for the propagation a random combination of past cycles, and for the orbit iteration a randomly sampled cycle was used. In these results, a clear trend of shorter disposal lifetimes for the Monte-Carlo sampled cycle can be made out. Note that the results shown here only indicate the spread and the distribution valid for the test case and should not be used as a general assessment of the solar activity forecast for mitigation orbit determination.

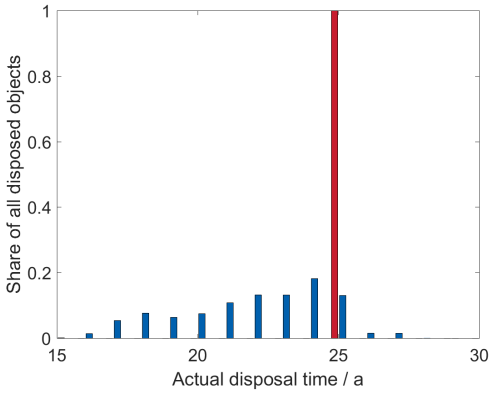

Figure 10. Histogram of actually disposal time, when identical forecasts have been used for orbit propagation and disposal orbit iteration (red bars), and when different forecasts have been used for orbit propagation and disposal orbit iteration.

4. RESULTS FROM LONG-TERM SIMULATIONS

In the frame of an IADC internal task, a large set of long-term simulations to analyse the sensitivity of long-term simulations to different input parameters and assumptions has been performed. Some of the results from this study are shown in this paper.

4.1. Baseline scenario

All simulations were performed by varying different parameters compared to a baseline scenario. The baseline scenario was defined as follows:

- Initial population based on MASTER-2009, including all LEO crossing objects on January 1^{st} 2013.
- Repeated launch cycle including all LEO crossing launches between the years 2005 and 2012.
- 8-year nominal lifetime for spacecraft.
- Assume post-mission disposal for both spacecraft and rocketbodies to 25-year orbits with a 60% success rate.
- Assume no future explosions.
- Allow no stationkeeping for spacecraft and no collision avoidance.
- Use a simulation time frame of 200 years.

For all scenarios, 100 Monte-Carlo runs were performed.

The results of the baseline scenario are shown in Figure 11. For the results shown, as underlying solar and geomagnetic activity forecast, a different random combination of past cycles has been used. The algorithm for collision rates used in the results shown was CUBE.

4.2. Variation of the solaractivity

Following, the solar and geomagnetic was varied in four scenarios, low, medium, high, and random. In the low scenario, the F10.7 value moved between 60 and 120, the AP between 6.5 and 15. In the medium scenario, the F10.7 moved between 65 and 160, the AP between 10 and 18.6, and in the high activity scenario, the F10.7 value moved between 71 and 220, the AP between 13 and 22. The solar and geomagnetic activity forecast used in the random scenario was very similar to that used in the baseline scenario a random combination of past cycles, but provided externally by ESA. The results for all four scenarios, together with those from the baseline scenario, are shown in Figure 12.

In these results, it can be made out that the solar and geomagnetic activity forecast has strong impact on the space debris environment evolution: both the median number of objects as well as the median cumulative number of collisions differ by a factor of three, after 200 years of simulation. Therefore, it is vital to always use a forecast appropriate for the study. This is especially the case for long-term studies performed using different long-term evolutionary models, as different solar activity forecasts might lead to strong differences in results and complicate all further analyses. Furthermore, using too low solar and geomagnetic forecasts might lead to an exaggeration of the environment evolution. From the approach, using random combinations of past solar cycles appears to be the most appropriate for general studies, as in these all features of past cycles (in term of maximum and minimum values per cycle) are covered.

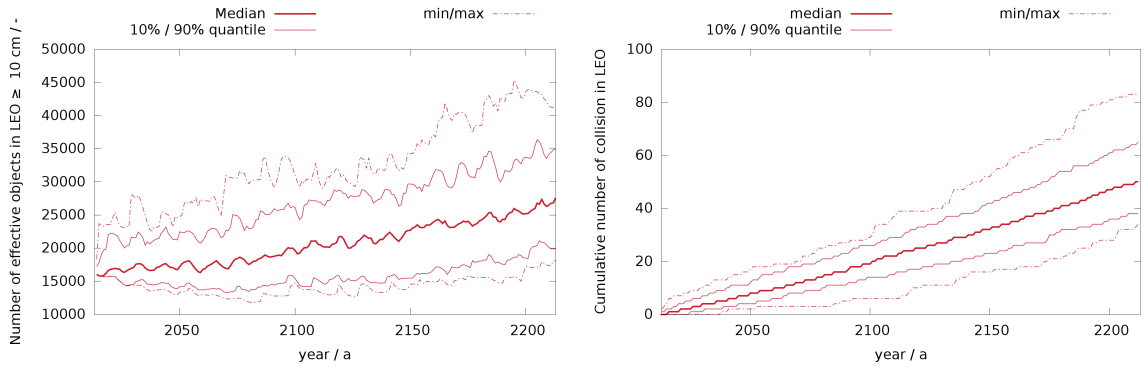

Figure 11. Results of the baseline scenario computed from 100 Monte-Carlo runs, collision rates computed with CUBE. ***Left:*** *Effective number of objects in LEO over time.* ***Right:*** *Cumulative number of catastrophic collisions.*

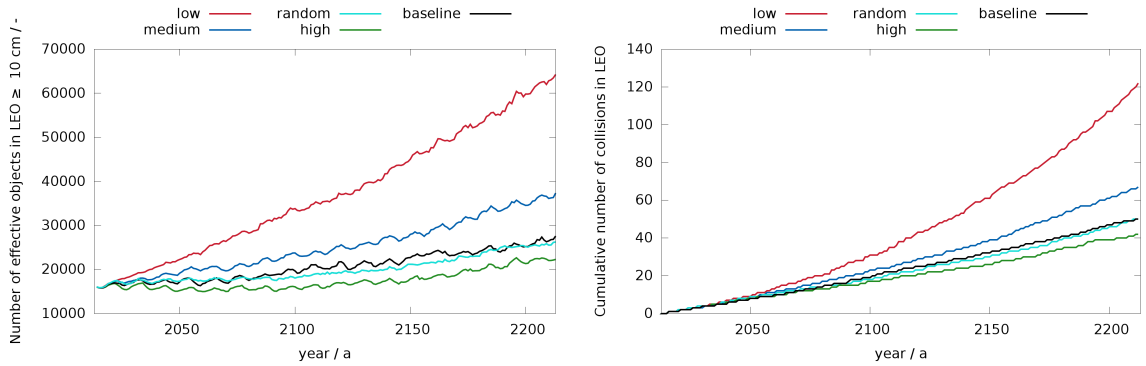

Figure 12. Median results of the solar activity forecast variation scenarios computed from 100 Monte-Carlo runs, collision rates computed with CUBE. ***Left:*** *Effective number of objects in LEO over time.* ***Right:*** *Cumulative number of catastrophic collisions. Note that quantiles and min/max values only have been omitted for clarity. All simulations showed spreads similar or larger than in the baseline scenario.*

4.3. Impact of the collision rate algorithm

Additionally, all simulations and variations have been performed using Orbit-Trace for the collision rate determination. The median results of all scenarios are shown in Figure 13.

It can be seen that the results are very similar: both the trends as well as the overall median numbers agree well when using either method. Nevertheless, an offset in the cumulative number of collisions can be found in that the collisions computed with Orbit-Trace are in all cases slightly lower than those computed with CUBE. Therefore, the significance of this difference was tested, using a Wilcoxon rank sum test [19].

This test is a non-parametric hypothesis test, which tests if the medians of tested samples from independent populations originate from statistical distributions with identical medians at an defined significance level α (null-hypothesis). The test then returns the probability p of the two samples being from distributions fulfilling this hypothesis. The hypothesis is assumed to be fulfilled, if $p > \alpha$, else it is rejected and the difference between the populations is assumed to be significant. For the analyses presented here, the standard significance level of $\alpha = 0.05$ has been chosen. In the comparison, always the results computed with CUBE are compared against those computed with Orbit-Trace, for both the number of objects on orbit as well as the cumulative number of collisions.

The results of the rank sum test are shown in Figure 14. They support the findings from the general results: the difference in number of objects is at most time steps not found to be significant, only in the very beginning as well around the year 2060 in the medium activity scenario, differences are large enough to reject the null-hypothesis. These might be effects from the low number of performed Monte-Carlo runs. The cumulative number of collisions shows a different behaviour: Although the results are very similar, the differences are significant for most scenarios over a large share of the simulation time-frame. Again, one reason might be the number of performed Monte-Carlo runs, another possibility is the filer system for objects on orbits with similar periods, which is used in Orbit-Trace, but not CUBE (cf. Section 3.2). But as the difference between CUBE and Orbit-Trace appears to be systematic, this should be investigated in more details.

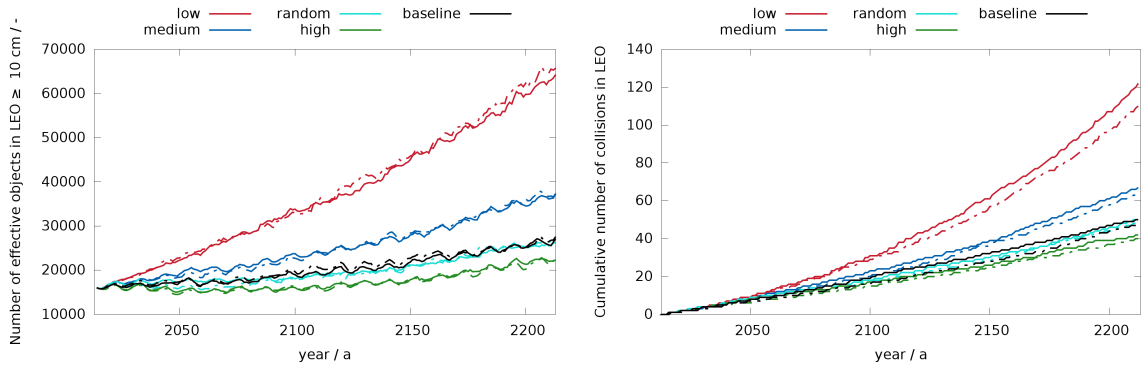

Figure 13. Median results of all scenarios computed from 100 Monte-Carlo runs, collision rates computed with both CUBE (solid line) and Orbit-Trace (dashed line). ***Left:*** *Effective number of objects in LEO over time.* ***Right:*** *Cumulative number of catastrophic collisions. Note that quantiles and min/max values only have been omitted for clarity. All simulations showed spreads similar or larger than in the baseline scenario.*

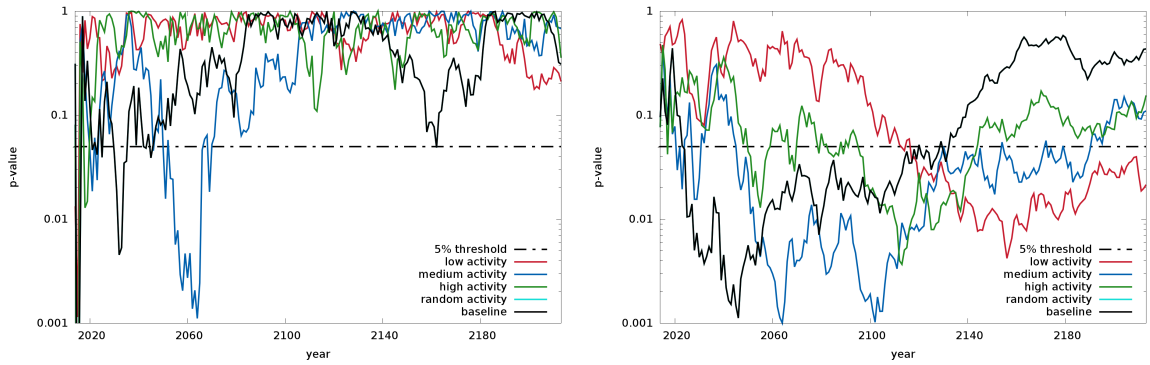

Figure 14. Outputs of a Wilcoxon rank sum test performed with results from all varied scenarios using both CUBE and Orbit-Trace. The p-value indicates the probability that the medians of the underlying distributions are identical. For $p < 5\%$, it is assumed that the difference is significant. ***Left:*** *Test performed with the number of objects on orbit.* ***Right:*** *Test performed using the cumulative collisions as parameter.*

5. SUMMARY AND OUTLOOK

Over the last year, LUCA2 has been completely re-implemented and by now reached a state of being fully usable. During this phase, the main goals of the re-implementation could be achieved: LUCA2 is now a modern, modular software, which allows an easy exchange of functionalities. For some of these, such as the collision rate determination, already exchangeable plugins were developed and are in use.

Using LUCA2, a set of simulations as part of an IADC sensitivity analysis have been performed and presented in this paper. The results from these underline the necessity for a proper choice of the solar and geomagnetic forecast used for the simulations. With the availability of an automated creation of random combination from past solar cycles, LUCA2 provides a good solution to this problem. Furthermore, using the modularity of LUCA2, the significance of the difference between using either CUBE or Orbit-Trace as collision rate determination method was investigated. Generally, it was found that in all simulated scenarios, both methods lead to simulations with very similar results in both median number of objects and cumulative number of collisions. Nevertheless, the difference in the number of collisions was found to be significant in most cases. The reason for this difference should be further investigated.

Now that the new tool is operational, next steps will be to use it for dedicated long-term studies. Upcoming simulation campaigns will be performed to provide answers to the question on "how to" achieve a certain goal. Additionally, the applicability of used collision rate determination algorithms for large constellations within long-term simulations will be analysed.

6. ACKNOWLEDGEMENTS

The re-implementation of LUCA2 and the computation of all results shown in this paper where achieved during the research project "Langzeitauswirkung und Kostenanalyse aktiver Weltraummll-Entfernungsmanahmen

(ELKE)" (Fkz.: 50 LZ 1501) funded by German Federal Ministry for Economic Affairs and Energy. All responsibilities for the contents of this publication resides with the authors.

Furthermore, the author would like to thank ESA's Space Debris Office for providing space debris populations used as inputs to the simulations.

REFERENCES

1. R. Blake, H. G. Lewis, The Effect of Modelling Assumptions on Predictions of the Space Debris Environment 65th International Astronautical Congress, 2014. Paper ID IAC-14-A6.2.4.
2. B. Bastida Virgili, J.-C. Dolado-Perez, H. G. Lewis, J. Radtke, H. Krag, B. Revelin, C Cazaux, C. Colombo, R. Crowther, M. Metz, Risk to space sustainability from large constellations of satellites, Acta Astronautica , 2016, 126, 154 - 162
3. V. Braun, S. Flegel, J. Gelhaus, M. Moeckel, C. Kebschull, J. Radtke, C. Wiedemann, H. Krag, P. Voersmann, Orbital lifetime estimation using ESA's OSCAR tool. 5th European Conference for Aerospace Sciences (EUCASS), Munich, 2013.
4. J.-C. Dolado-Perez, R. D. Costanzo, B. Revelin, Introducing medee a new orbital debris evolutionary model, in: Proceedings of the 6th EuropeanConference on Space Debris, volume ESA SP-723,2013.
5. S. Flegel, Maintenance of the ESA MASTER Model Institute of Aerospace Systems, TU Braunschweig, 2011
6. International Organization for Standardization. Space systems - Estimation of orbit lifetime, 2011.
7. N. Johnson, P. Krisko, J.-C. Liou, P. Anz-Meador, NASA's new breakup model of evolve 4.0, Advances in Space Research, 2001, 28, 1377 - 1384
8. P. Krisko, Proper Implementation of the 1998 NASA Breakup Model Orbital Debris Quartely News, NASA Orbital Debris Program Office, 2011, Issue 15.4
9. H. G. Lewis, G. G. Swinerd, N. Williams, G. Gittins, Damage: A dedicated geo debris model framework, in: Proceedings of the 3rd European Conference on Space Debris, number ESA SP473, 2001.
10. H. G. Lewis, G. G. Swinerd, J. R. Newland, A. Saunders, Active Removal Study for On-Orbit Debris Using Damage, Proceedings of the Fifth European Conference on Space Debris, 2009
11. J.-C. Liou, Collision activities in the future orbital debris environment Advances in Space Research, 2006, 38, 2102 - 2106
12. J.-C. Liou, D. Hall, P. Krisko, J. Opiela, Legend a three-dimensional leo-to-geo debris evolutionary model, Adv. SpaceRes. 34(5)(2004)981986, Space Debris.
13. C. Martin, R. Walker, H. Klinkrad, The sensitivity of the ESA DELTA model, Adv. Space Res. 34 (5) (2004) 969974.
14. E. J. Öpik, (1952). Collision probabilities with the planets and the distribution of interplanetary matter. *Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences*, 54:165–199.
15. J. Radtke, S. Flegel, J. Gelhaus, M. Moeckel, V. Braun, C. Kebschull, C. Wiedemann, H. Krag, K. Merz, K., P. Voersmann, Revision of Statistical Collision Analysis for Objects Inside of Satellite Constellations International Astronautical Congress, 2013
16. J. Radtke, R. Dominguez-Gonzalez, S. Flegel, N. Sanchez-Ortiz, K. Merz, Impact of eccentricity build-up and graveyard disposal strategies on MEO navigation constellations, Adv. Space Res. 56 (11) (2015) 26262644.
17. J. Radtke, E. Stoll, Comparing long-term projections of the space debris environment to real world data Looking back to 1990. Acta Astronautica , 2016, 127, 482 - 490.
18. A. Rossi, L. Anselmo, C. Pardini, R. Jehn, G. B. Valsecchi, The new space debris mitigation (sdm4.0)long term evolution code, in: Fifth European Conference on Space Debris,ESA/ESOC,2009.
19. S. Siegel (Ed.), Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill, New York, 1956.
20. C. Wiedemann, S. Flegel, M. Moeckel, J. Gelhaus, J. Bendisch, M. Metz, P. Voersmann, The effectiveness of space debris mitigation measures, Proceedings of the 61st International Astronautical Congress, Prague, Czech Republic, 2010
21. B. Zhang, Z. Wang, Y. Zhang. An analytic method of space debris cloud evolution and its collision evaluation for constellation satellites Advances in Space Research, 2016, 58, 903 - 913