THE COLA COLLISION AVOIDANCE METHOD

K. Aßmann[1], **J. Berger**[2], **and S. Grothkopp**[1,2]

[1]*Technische Universität München, Institute of Astronautics Boltzmannstraße 15, D-85748 Garching / Germany (kaja.assmann@mytum.de)*

[2]*LSE Space Engineering & Operations AG, Münchener Straße 20, D-82234 Wessling / Germany (jens.berger@lsespace.com, stefan.grothkopp@lsespace.com)*

ABSTRACT

In the following we present a collision avoidance method named COLA. The method has been designed to predict collisions for Earth orbiting spacecraft on any orbits, including orbit changes, with other space-born objects. The point in time of a collision and the collision probability are determined. To guarantee effective processing the COLA method uses a modular design and is composed of several components which are either developed within this work or deduced from existing algorithms: A filtering module, the close approach determination, the collision detection and the collision probability calculation. A software tool which implements the COLA method has been verified using various test cases built from sample missions. This software has been implemented in the C++ programming language and serves as a universal collision detection tool at LSE Space Engineering & Operations AG.

1. INTRODUCTION

The increasing amount of space debris and other space-borne objects increases the collision risk for operated spacecraft (S/C). The prediction of possible collisions gets more and more important as it enables the S/C operator to react to the threat and hence avoid collisions by changing the S/C's orbit. For the collision prediction, methods and tools are needed which reliably predict collisions with other space-borne objects. The S/C to be protected is called the main object, often also referred to as primary object. The main object's orbit and position is compared with the orbits and positions of all objects that might produce a collision and are a risk for the main object. These objects are called collision objects, often also referred to as secondary objects. In the following we present the algorithm of the COLA method, the implementation of the method in the COLA software and the verification of this software using various test cases built from sample missions.

2. GENERAL STRUCTURE OF THE METHOD

A diagram of the general structure of the method can be found in Figure 1. The process flow is as follows. First the necessary input data is introduced and checked for validity. If the data format is valid the huge amount of collision objects is reduced by applying a series of three filters that identify irrelevant collision objects and eliminate the corresponding data from the possible collision objects. Subsequently the objects' orbits are propagated for the time period of collision examination. A coordinate transformation follows, to assure the comparability of the orbits by transforming the state vectors of the collision objects into the reference coordinate system (COS). Having determined all objects' state vectors the collision calculation is conducted. The collision calculation consists of three steps: the close approach determination, the collision detection and the collision probability calculation. The results are printed out and the user can decide whether to refine the input data for the detected collision objects using the refinement module or not. If new input data are given, the results will be recalculated. Finally, the results will be written into an output file.

3. ANALYZED METHODS AND ALGORITHMS

Different methods and algorithms have been analyzed for each component of the COLA method to find the most suitable. This lead to the development of new and modifications of existing algorithms. Within the filtering module of COLA two geometrical filters and a time depending filter are available, deduced from filtering methods developed by Vallado [8] and Hoots [6]. The general approaches and the modifications made are described in Section 5.2. When handling Two Line Elements (TLEs) as input data, as it is the case for the collision objects from the NORAD Catalog, a SGP4/SDP4 (Simplified General Perturbations Satellite Orbit Model) propagator is indispensable. For COLA, the Vallado SGP4/SDP4 propagator was chosen [4]. The Vallado propagator is flexible, has a good program structure and is documented through comments in the code. It is assumed to be updated on the basis of the Revisiting Space Track Report #3 [9].

Proc. '5th European Conference on Space Debris', Darmstadt, Germany 30 March – 2 April 2009, (ESA SP-672, July 2009)

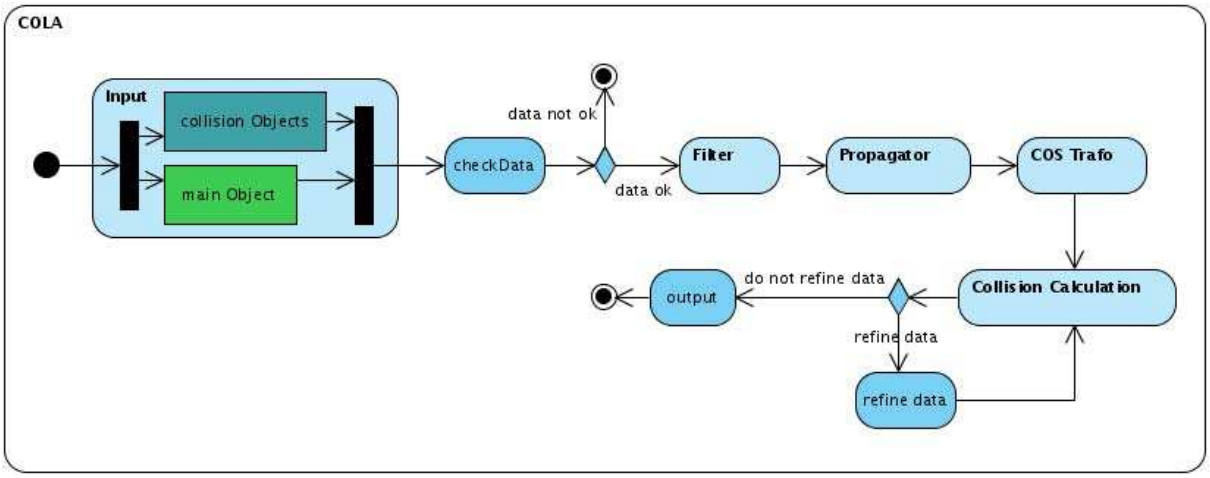

Figure 1. General Structure of the COLA Method

The close approach algorithm within the collision calculation module determines the point of time and the minimum distances between two objects on different orbits. Therefore, two algorithms were evaluated: A method developed by Alfano/Negron [3] [8], which determines close approach cases from the distance of objects, and a method developed by Coppola/Woodburn [5], which uses the distance of ellipsoids. The COLA method includes a modified version of the Alfano/Negron method described in detail in Section 5.4. This algorithm has been chosen because of its low computation time. Since there might, depending on the input data, only limited information available on the error ellipsoids around the objects, the Alfano/Negron algorithm is more adequate than the ellipsoidal approaches.

To decide, whether a collision is likely to occur, three collision detection methods were considered. Two methods use the detection of ellipsoidal intersection developed by Wang/Choi/Chan/Kim/Wang [11] and Alfano/Greer [2]. The third method was developed within this work and compares the minimum distance and the maximum positional error of the objects. This method is used by COLA, because of its simplicity and its considerably lower computation time at no loss of accuracy.

Due to the error introduced by the orbit determination and propagation of the objects, their positions are given as a predicted position surrounded by an error ellipsoid, defined by positional covariances. For the calculation of the collision probability, four algorithms which use this error information were considered and compared: The algorithms by Foster, Chan, Patera and Alfano [1]. After careful consideration and analysis the Alfano algorithm was chosen for the COLA method. It does not require any further mathematical methods because of the transformation of the double integral, used for the probability calculation, into a series expression. This transformation includes error functions and exponential expressions. Additionally, the consumed computation time is low in comparison to most other collision probability algorithms.

4. ASSUMPTIONS AND CONDITIONS

In this section assumptions are listed, that were made in the course of the COLA development. A few simplifications have been made, mostly because more detailed information on the objects is not available yet or the exact calculations would exceed both, the mathematical and computational possibilities of standard computers. In addition, most of the results would not improve much if the simplifications were left out, because of the overall error made at tracking and propagating the objects' orbits. The assumptions made for COLA are:

- The objects are approximated as spheres for the collision detection at the point of close approach
- Linear motion exists for the short time of encounter
- No docking maneuvers will be performed during the calculation period
- The covariance propagation is approximated by a linear growth
- The positional errors are distributed in a zero-mean Gaussian distribution
- The positional errors are uncorrelated and constant for the short time of encounter
- The reference coordinate system is the J2000 COS
- The main orbit data (ephemeris) is given as a comma separated value (CSV) text file or via a C++ application programming interface (API) in the J2000 COS
- The initial covariance matrices are given in the J2000 COS.

5. COMPONENTS OF THE COLA METHOD

5.1. Input

The COLA method requires input information for the main object and the collision objects to be able to calculate possible collisions. The main object's data is usually provided by the satellite operator, who wants to protect his satellite. The main object's data is provided to the COLA software via a CSV text file or a C++ API in the J2000 COS. This can be done via LSE's own Mission Design Tool (MDT) or any other software capable to generate those data. Any collision objects' state vector data including collision objects error information can be handled by the COLA method. Currently the only collision data available are data taken from the North American Aerospace Defense Command (NORAD) object catalog, provided as TLEs to the software. The collision objects' state vectors are propagated within the COLA software. The collision object and the main object data are read into the software from two files respectively. The input and the output files can be replaced by a corresponding data interface when integrating the COLA software into another program (e.g. MDT), but for the stand-alone version of the COLA software the data has to be provided via files.
Additionally to the objects' state vectors their positional errors in the form of covariance matrices and the objects' sizes are needed. In the case that these data is not available the COLA software uses default values. The default values are chosen conservatively, so that no collision will be missed. The default values can be changed via the refinement module of the COLA software, then the collision probabilities are recalculated.

5.2. Filtering Module

The COLA filtering module consists of a series of three filters, two geometrical and one time depending filter, which reduce the huge amount of possible collision objects to the relevant ones. Hence, the computation time for the later collision calculation can be reduced by this elimination of irrelevant collision objects for the considered main orbit. The applied filters are deduced from filtering methods developed by Vallado [8] and Hoots [6] and have been extended to be able to handle all kinds of orbits, including orbits where impulsive maneuvers or low thrust transfers are applied. Reducing the number of objects for the actual collision calculation is the main factor in keeping the computation time within a reasonable limit. The three filters are ordered from the roughest to the finest filter to maximize the filtering efficiency. Increasing precision within the filtering mechanism results in increasing computation time as well. Hence the maximum efficiency of the filters is reached by applying all three filters serial.

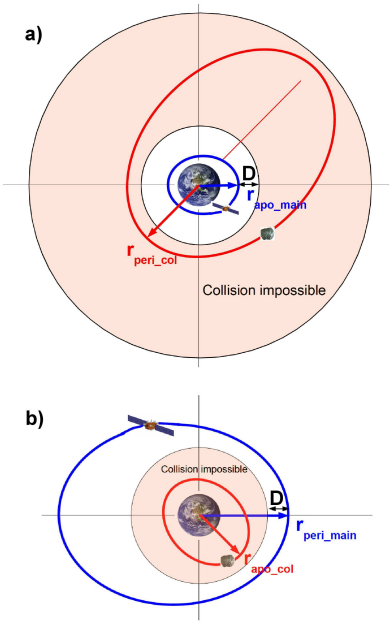

Figure 2. First geometrical filter for a) outer and b) inner collision objects

First Filter: Geometrical

The first filter that is used to reduce the number of possible collision objects is a geometrical filter. It is based on the perigee-apogee filtering algorithm of [8], and includes some expanding modifications. This harsh filter uses the orbits' perigees and apogees to calculate the minimum distance between two orbits without considering their orientation in space or the objects' positions on the orbits. Hence the filter can be applied prior to the orbit propagation of the collision objects. The actual filter is divided into two parts.
First, all objects with an orbit much bigger than the maximum main orbit and then all objects with an orbit much smaller than the minimum main orbit are eliminated from the list of possible collision objects. A collision is considered impossible if the difference between the collision object's radius of perigee ($r_{pericol}$) and the main object's radius of apogee ($r_{apomain}$), which equals the minimum distance (D) between the two orbits, is bigger than a certain threshold (See Figure 2a and (1)). In the second case, objects with a smaller orbit than the minimum main orbit are eliminated. In this case a collision between the two objects is considered impossible if the difference between the collision object's radius of apogee (r_{apocol}) and the main object's radius of perigee ($r_{perimain}$) is smaller than the acceptable close approach threshold (See Figure 2b and (2)). We considered a threshold of $D = 300$ km a save value for both cases.

$$r_{pericol} > r_{apomain} + D \tag{1}$$

$$r_{apocol} < r_{perimain} - D \tag{2}$$

In case of major changes in the main orbit during the propagation period, for example due to a kick-burn of the satellite's thrusters, the filter considers the maximum and the minimum main orbit for the outer and inner collision object filtering respectively.

Second Filter: Geometrical

The second filter to reduce the number of possible collision objects is a geometrical filter, like the first filter. The filter is deduced from an algorithm developed by Hoots et al. [6]. Unlike the first filter, this filter considers the orientation of the orbits in space. Two elliptical paths with the same focus, in this case the Earth, that are not coplanar, have two distances d_1 and d_2 of close approach. If the distances d_1 and d_2 are greater than the acceptable approach distance D, the two satellites do not approach each other close enough for a collision. Hence, these objects can be eliminated from the list of possible collision objects. Hoots at al. form a system of equations depending only on the true anomaly ν of the main and the collision object. Solving this system of equations, for example using Newton's method, the resulting true anomalies define the part of the orbit where the relative distance is minimal and a close approach takes place. Within the COLA method, the Newton iteration is stopped when the difference between two iteration values is less than $\nu = 0.1^\circ$ or a maximum of 1000 iteration steps is reached in the infrequent case of a non converging solution.
The filtering is repeated for all main orbits that result of possible orbit maneuvers within the propagation period. Only collision objects that do not approach any of the main orbits with a distance smaller than D are eliminated.

Third Filter: Time Depending

The third filter considers not only the geometrical characteristics of the orbits but also the objects' positions on their orbits. Even though two orbits may share a zone of possible close encounters which is within the minimum approach distance allowed, this does not mean, that the satellites pass this part of the orbit simultaneously. Considering the objects' positions the crossing times for the critical section of the orbit can be calculated and objects that pass this section with a certain time-lag to the main object can be eliminated from the list of possible collision objects. Like the second geometrical filter, this filter is deduced from an algorithm developed by Hoots et al. [6]. According to Hoots et al. angular windows for the close approaches are determined and subsequently can be transformed into time windows as indicated in Figure 3. These time windows are used directly for the filtering.
As for the second filter the filtering is repeated for all main orbits that result of possible orbit maneuvers within the propagation period. Only collision objects that do not approach any of the main orbits with a distance smaller than the acceptable approach distance D are eliminated.

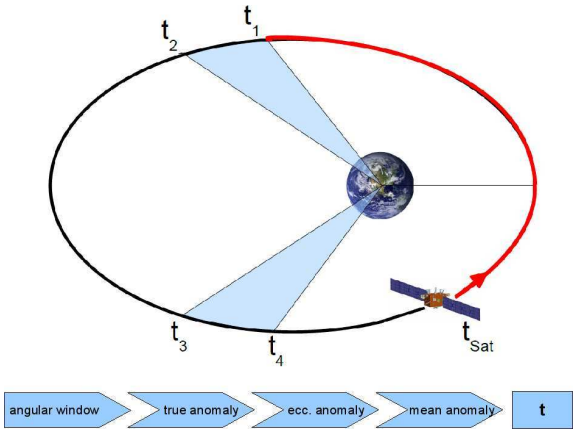

Figure 3. Third Filter, time windows

Filtering Efficiency

The filtering module consisting of the three filters described above reduces the number of possible collision objects in a most effective way. Depending on the input data and naturally varying on a case by case basis for the main object's trajectory, the number of collision objects can be reduced to $\approx$ 10-15 % of the original amount in case of the NORAD object catalog. Table 1 gives an example of the filtering efficiency of each filter for a main object on a circular orbit of 800 km altitude and an inclination of 28.5° for the 10^{th} of July 2008 and a period of calculation of one day.

Table 1. Efficiency of the filtering module

	# of col. obj.	% of total # of col. obj.
before filtering	11572	100.0 %
after filter 1	7876	68.1 %
after filter 2	5728	49.5 %
after filter 3	1212	10.5 %

5.3. Propagation and Coordinate Transformation

For the collision calculation the state vectors and positional covariance matrices of both, the main object and all possible collision objects, are needed for the whole calculation period. The main object's data is given in the J2000 COS. The collision object's state vectors and covariance matrices are propagated and predicted by COLA. For the propagation of the NORAD objects, the Vallado SGP4/SDP4 propagator [4] was chosen.
The state vectors calculated by the SGP4/SDP4 propagator are given in the True Equator Mean Equinox (TEME)

COS. To be able to compare the collision objects' orbits with the main object the propagated state vectors have to be transformed from the TEME into the J2000 COS. This transformation is made using a function implemented and published by Vallado [8] [10]. The collision and the main object's covariance propagation is approximated by using linear increasing position errors depending on the minutes from epoch of the state vector and the start propagation time in order to compensate the increasing error for longer propagation periods.

5.4. Collision Calculation

Close Approach Determination

The close approach determination of the COLA method is based on the Alfano/Negron [3] [8] close approach algorithm mentioned in Section 3. The algorithm requires not only the input of the position and velocity vectors, but also of the acceleration vector to interpolate the distance function. The algorithm has been modified for the COLA software, so that it does not require the accelerations any more. The close approach is determined via the distance vector only, using smaller propagation steps without interpolating the distance function.

In the COLA software the distance vectors $\vec{d}$ of all collision objects to the main object are determined for each propagation step, using the propagated position vectors of the main object $\vec{r}_{main}$ and all collision objects $\vec{r}_{col}$ (See (3) and Figure 4).

$$\vec{d} = \vec{r}_{main} - \vec{r}_{col} \tag{3}$$

The propagation step is recommended to be of $\Delta t = 10$ sec or less, and is not allowed to exceed a maximum of $\Delta t = 20$ sec. Subsequently the modulus of all distance vectors is calculated. The minimum distance can then be found by selecting the smallest of these. Thus, the close approach is determined (See Figure 4). After determining the close approaches, the calculation is repeated with $\Delta t/2$ in the areas around the determined close approaches.

Collision Detection

The next step to perform is the collision detection. It has to be determined, whether the collision objects and the main object can collide at the time of close approach, or not. Two objects are deemed to have a possible collision when their positional covariance ellipsoids intersect. Determining the intersection between two ellipsoids is a rather complicated and time consuming process. To facilitate the collision detection, the error ellipsoids are approximated as spheres with a diameter that equals the maximum ellipsoid expansion. The spheres intersect when the distance between the two objects is

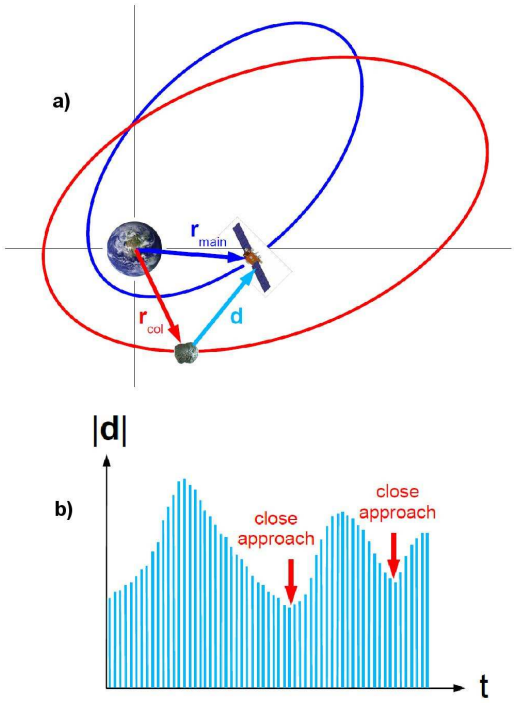

Figure 4. COLA close approach determination. a) Position and distance vectors b) Discrete distance function with close approach detection

smaller than the sum of the radii. This is a simple comparison and does not take much computation time. By approximating the ellipsoids as spheres a few objects are selected wrongly. They are considered collision objects because the error spheres intersect, but the ellipsoids do not. However, the number of wrongly selected objects is really small and it is easy to discard them again because their collision probability will result to zero.

Collision Probability Calculation

Only in the case that a collision is detected, the collision probability calculation is carried out. As method to conduct the collision probability calculation, an algorithm by Alfano [1] has been chosen.

The collision probability P can be calculated by evaluating the path of a combined object, defined by the sum of the main and the collision object's radii, through a combined covariance ellipsoid, defined by the sum of the main and the collision object's covariances σ (See [1]). At this the combined covariance ellipsoid represents the probability density function (pdf). The collision probability can then be calculated by solving the integral of the three dimensional pdf within the collision tube, which is a circular cylinder due to the assumed linear motion. Alfano poses that "this is equivalent to evaluating the integral of the two-dimensional pdf within a circle on a plane perpendicular to the relative velocity at closest approach"

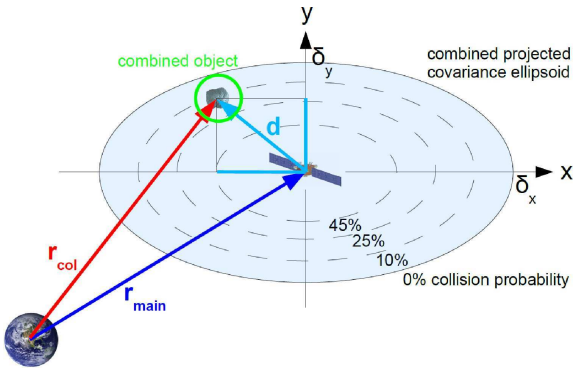

Figure 5. Alfano's collision probability determination

[1] as displayed in Figure 5. The Alfano method transforms this double integral into a series expression including error functions and exponential expressions.

5.5. Output

Finally, the obtained collision data is written to an output file, including: The collision object's identification number, the time of collision given in two variants (time of flight of the main object and UTC), the close approach distance and the collision probability. The output data is provided through this file interface, but for the implementation of the COLA software into other software this file interface can be easily changed into a pure data interface.

6. IMPLEMENTATION

The COLA method has been implemented in the C++ programming language building the COLA software. The COLA software has been integrated as a module into the MDT of LSE, but can also be used stand alone by/with other software via a file interface or a C++ API. The software is available for UNIX/Linux, Mac OS and Windows.

7. VERIFICATION

The COLA software has been verified using various test cases. For these tests sample missions for the main object have been generated using LSE's MDT. The results of the COLA software verification are discussed in the following.

7.1. Test: ISS Retrograde

The first test case is generated by simulating the ISS orbit and another object, a fictitious S/C, flying retrograde on the same orbit. It is obvious that the two stations are deemed to collide twice on every revolution around the Earth. The fictitious S/C is considered as the main object. Its orbit data is generated using the MDT. Thus the objects properties (size and covariance data) are the standard values used by the COLA software. The collision object is the real ISS, TLE's taken from the NORAD catalog of the 27^{th} of September 2008.
Running the COLA software with this input data, collisions are detected approximately every 45 min. This result is expected, since there are two collisions on each ISS orbit having a period of $T \approx 90$ minutes. The collision probabilities result to reasonable values between $10^{-5} < P < 10^{-13}$ using the default covariance data of the COLA software. Introducing a higher positional error via the refinement module, the collision probabilities increase to values between $10^{-5} < P < 10^{-8}$, as expected.
The purpose of this test case was to check the software's results by introducing an example with known results. The collisions were calculated by the COLA software as expected. Additionally the option of covariance variation within the refinement interface has been verified as well.

7.2. Test: Delfi-C3 Retrograde

The second test case that has been evaluated with the COLA software is quite similar to the first one, except for the modification of the covariance and object size data. The purpose of this test case is to test the collision calculation for an object smaller than the standard collision object within the COLA software. Instead of the ISS the triple CubeSat Delfi-C3 has been chosen as collision object, which has a maximum dimension of $d = 0.3$ m, using TLEs taken from the NORAD catalog of the 27^{th} of September 2008. The calculation will be conducted using the standard collision object diameter of $d = 110$ m. Afterwards, the option to change the collision objects data within the refinement interface will be used to repeat the calculation with the much smaller real object diameter. The main object is an object of the standard main object size within the COLA software flying retrograde on the Delfi-C3 orbit, whose orbit data is generated using the MDT. Just like in the previous test the two objects should collide twice at each revolution. The orbits' altitude is about $h = 600$ km so that the orbits' period will be longer than the ISS's. The collisions are expected to occur approximately every 48 minutes.
The results given by the COLA software for this input data are as expected. There are collisions detected every 48 minutes. The collision probabilities result to values between $10^{-4} < P < 10^{-18}$ for the default collision object size of $d = 110$ m. Since the CubeSat is much smaller than the standard collision object of the COLA software, the option to change the collision object's data has to be used to get reliable results. For a smaller object the collision probabilities will be smaller as calculated before. In this test case, the covariance data will not be changed, to be able to observe the results that are obtained by a varying object diameter. The diame-

ter was changed from $d = 0.110$ km to $d = 0.0003$ km for each detected collision. As expected, the only values that change are the collision probabilities. For decreased object size, the collision probability decreases to values between $10^{-7} < P < 10^{-20}$.

7.3. Example: ISS 5 Day Analysis

A typical use case for later operational application of the COLA software will be to predict possible collisions for an operated satellite a few days in advance. The third example demonstrates using the ISS as main object. The ISS orbit data is generated using the MDT. A propagation time of 5 days has been chosen, with a time step of $\Delta t = 5$ seconds. It is not recommended to choose a longer propagation period, because the results get fairly imprecise due to propagation errors. In addition, keeping a small time step, which guarantees accurate results, the amount of data that is produced increases strongly with longer propagation periods. Using a 5 day period with a time step of 5 seconds is a reasonable trade-off between a long propagation time and a small time step. The ISS orbit data will be compared with the full catalog of possible collision objects, here the NORAD object catalog of the 27^{th} of September 2008. The COLA software determined only one collision with a collision probability of $P = 0$ within a 5 days propagation period. This seems reasonable, since the ISS has to perform a maneuver to avoid collisions every few month or less. So the probability is low that this takes places in the analyzed time interval of 5 days. The collision probability means, as explained above, that the positional covariance ellipsoids do not intersect, but the approximated spheres used for the collision detection do. Thus the operator would not have to perform an avoidance maneuver yet, but it might be wise to gather more information on the debris object and to perform more detailed calculations when the time of collision draws closer.

7.4. Example: Hohmann Transfer LEO-GTO-GEO

Here a Hohmann transfer from a low Earth orbit (LEO) with an altitude of $h = 600$ km and inclination $i = 0°$ via a geostationary transfer orbit (GTO) to a geostationary orbit (GEO) is simulated. The main object's data is generated with the MDT as before. Its orbit data will be compared to the full catalog of possible collision objects, here the NORAD object catalog of the 27^{th} of September 2008.
COLA detects three possible collision objects with this input data (# 20905, 21268, 30030). These are classified as a CZ-4 debris, a DELTA 1 debris and a FENGYUN 1C debris object, respectively. The collision probabilities are distributed around values of $P = 10^{-6}$, which are values high enough to consider an avoidance maneuver or to adjust the transfer's start time to avoid a collision. As in the previous example, for real time operation more information should be acquired on the debris objects and the objects' orbits should be kept tracked. The detection of three collisions when crossing from a LEO to a GEO on an elliptical transfer orbit seems a reasonable result.

7.5. Example: Low Thrust Transfer LEO-GEO

This example demonstrates the handling of a low thrust transfer from a circular LEO ($h = 600$ km, $i = 0°$) to GEO. The transfer orbit is helical, as is typical for low thrust propulsion. The main object's data was generated using the MDT (See Figure 6b). This orbit data is compared with the full catalog of possible collision objects, here the NORAD object catalog of the 27^{th} of September 2008. The low thrust transfer usually is very slow, which means that there is a huge amount of propagated orbit data that has to be stored and compared, especially on orbits close to the Earth, because of the slow orbit changes. To reduce the data that has to be compared and to maintain a relatively small propagation step at the same time (here $\Delta t = 5$ seconds), the propagation period is split into five parts. Each of these parts is handed separately with the COLA software.
Running this example in the COLA software 106 possible collisions with 17 different objects are calculated (See Figure 6a). It should be noted that many collisions were detected for the first two days of the propagation time. This is due to the higher concentration of objects in low Earth orbits. Additionally, the change of the main orbit is very slow in the beginning of the low thrust transition. The more the propagation time progresses, the more the orbit is widened and changed more quickly. Regions with a lower concentration of space debris are reached and fewer collisions are detected. As expected there are a lot of possible collisions when performing a low thrust transfer from LEO to GEO no matter which transfer strategy is chosen. There might be an optimal solution where a smaller number of collisions occurs but the S/C operator always has to be aware of possible collisions and be prepared to perform collision avoidance maneuvres.

7.6. Computation Time

In order to make a collision avoidance tool capable of real time analysis and simulations, the computation time needed for the determination of possible collision shall not exceed a reasonable time. The COLA software has been developed by all means to be as fast as possible. This goal has been achieved, since the computation time for all of the examples described above is below 5 min for each example on a standard PC with a $f = 2$ GHz clocked single core processor.

8. CONCLUSION AND OUTLOOK

In this paper the collision avoidance method COLA including it's development, implementation and verification has been presented. The resulting COLA software

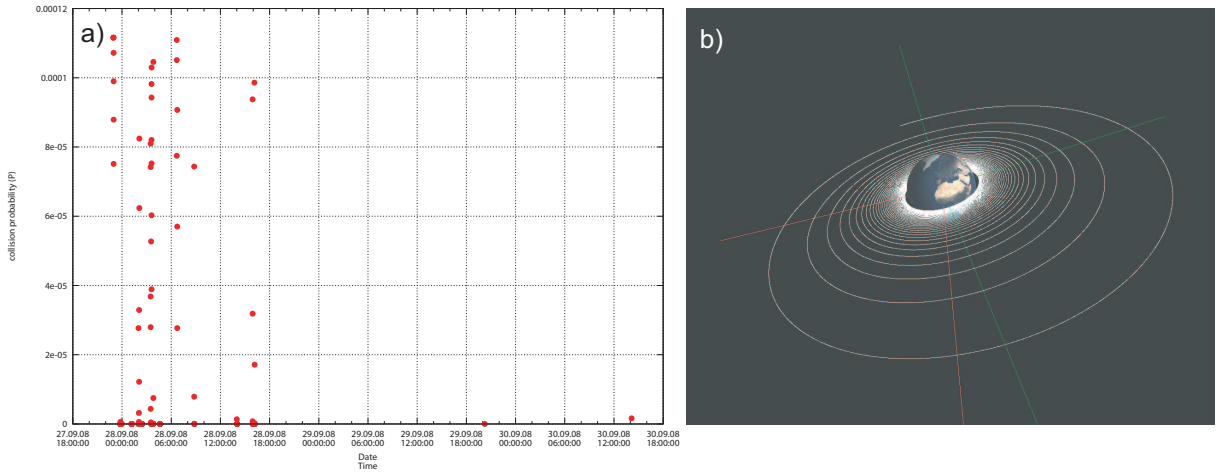

Figure 6. Low thrust transfer LEO to GEO. a) Collision probability vs. time of flight. b) Path of the transfer

is capable of determining collisions between Earth orbiting objects. The method considers orbit changes like high impulse and low thrust transfers. The following parameters are determined by the COLA software: The point of time of the collision, the distance between the centers of the positional error ellipsoids of the objects at their closest approach and the collision probability. A number of different software modules, designed to carry out the necessary calculations, have been developed and have then been joined to build the COLA software. The combination of different algorithms were to create a universal collision detection software for all kinds of Earth orbiting objects. This tool can be used for simulation purposes as well as real time operations. An additional verification of the software against some other collision detecting software would increase the level of confidence of predicted collisions. This is planned in the near future.

Nevertheless, there are some possible future enhancements. The COLA software is restricted to Earth orbiting objects. A possible enhancement could be to handle deep space missions. Another possible enhancement of the COLA software could include docking maneuvers. Here, the close approach determination would have to be extended to non linear motion. These two enhancements give rise to new applications of the software. However, there might be possible enhancements to improve the current functionality of the COLA software as well. First, the collision object data base will be updated and enlarged whenever there is more data available, especially collision objects' data with tracking error information and a precise covariance propagator model would be valuable. Secondly a Kalman filter could be included to reduce the overestimation of the positional error when propagating a data set of TLEs [7]. This might increase the propagation accuracy.

REFERENCES

[1] Salvatore Alfano (2007). Review of Conjunction Probability Methods for Short-term Encounters, AAS-07-148

[2] Salvatore Alfano, Meredith L. Greer (2003). Determining If Two Solid Ellipsoids Intersect, JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS, **26**.

[3] Salvatore Alfano (1994). Determining Satellite Close Approaches, Part II, Journal of the Astronautical Sciences, **42**, 143.

[4] http://www.centerforspace.com/downloads/

[5] Vincent Coppola, James Woodburn (1999). Determination of Close Approaches Based on Ellipsoidal Threat Volumes, AAS 99-170.

[6] Felix R. Hoots, Linda L. Crawford, and Ronald L. Roehrich (1984). An Analytical Method to Determine Future Close Approaches Between Satellites, Celestial Mechanics, **33**, 143.

[7] T. S. Kelso (2007). Validation of SGP4 and IS-GPS-200D Against GPS Precision Ephemerides, AAS 07-127.

[8] David A. Vallado (2007). Fundamentals of Astrodynamics and Applications, 3rd Edition, STL.

[9] David A. Vallado, Paul Crawford, Richard Hujsak, and T. S. Kelso (2006). Revisiting Spacetrack Report #3, AIAA 2006-6753.

[10] Jaohn H. Seago and David A. Vallado (2000). The Coordinate Frames of the US Space Object Catalog, AIAA 2000-4025.

[11] W.Wang, Y-K. Choi, B. Chan, M-S. Kim, and J. Wang (2004). Efficient Collision Detection for Moving Ellipsoids Using Separating Planes, Computing, **72**, 235.