Optimal Strategy For Removing Space Debris

February 1, 2016

Summary

Space debris is a growing problem that could affect many generations to come. Currently there are upwards of 500,000 pieces of junk in orbit around the Earth, which rarely draw the attention of any lay-man. However, recently there have been significant collisions, as pointed out by our problem statement, which have highlighted the level of concern humanity must show towards the space junk problem.

To approach this problem we designed a discrete time and discrete space based model to better understand space debris behavior and the possible solutions we may implement to clean it up. We organize our space debris into a three dimensional cube, and then proceed to approximate its positions one second at a time. This is done by approximating orbits with small segments of parabolas, over and over and over again. In addition, we use our three dimensional cube structure to efficiently check for intersections between our objects.

The two primary solutions we attempted to test were fluid based devices meant to slow debris and lead it to fall into orbit, and net based devices meant to gather debris and bring it out of orbit in one large lump. Both of these devices appeared at first to be relatively inexpensive options which had a possible wide range of results. Unfortunately, due to the large number of computations required to simulate such a large amount of space junk, and the level of precision needed to accurately represent future positions even over a 15 day period (.01 second calculation increments) we were unable to successfully test for the effect of these tools.

From our content research we found that space lasers appear to be the most effective for spot removal of junk in order to protect existing satellites, however very expensive to build and operate. We feel that further simulation and testing of fluid and net based systems will lead to the best solution for space debris clean up.

Table of Contents

1 Problem Statement

Our problem statement presented a collision of two satellites in 2009, and the problem of space debris. Tasked with developing a solution to the problem, and examining the economics of the situation, we began laying the foundations of our Space Junk Model. This object based model is built to simulate the Low Earth Orbit (LOE) which is in the range of 160 to 2000 kilometers [1] from the surface of the earth. In this orbit we create thousands of spherical space debris (called space junk in model) and attribute to them a randomized radius, mass, location and velocity. In this simulated space we created debris removal solutions, including a fluid/ice wall, a ground-based laser, and space balloons to test efficacy. From there the model steps forward in time discretely to generate the results that follow.

2 Overview of Assumptions

To begin this model we made simplifying assumptions, highlighted below and expanded on later. The first simplification, which has already been mentioned, was the limitation of the model to

LOE, rather than the entire space which collects debris around earth. This was made to lead to comfortable calculations mentioned in the discussion of velocity to come. In addition, in our research we found that this area of orbit was the most critical area in need of clean up. According to NASA, a large majority of space debris resides in LOE. Specifically, the amount of debris varies with altitude, with the greatest concentrations of space debris found in the 750-800km range [2]. In regards to the space junk themselves, we made assumptions in regards to attributes described below to allow for their simple generation. Similar assumptions were made in regards to their collisions, and the behavior of our solutions.

3 Model Design

In this section we describe the mathematics based thought process of designing our model, note that mathematical vocabulary is used to describe and not in the precise way for which it was developed. We began thinking of our model as a function (C) on an initial state (S) and the discrete variable (T). This S is a set of N^3 randomly generated 4 tuples (the space debris), each consisting of a location component (P, a vector), a velocity component (V, a vector), a radius and a mass. The discrete variable T is typically one second in our calculations (due to time and processing constraints) and is our largest source of error as our model's accuracy improves as T approaches 0. The function C(S,t) returns the system at time t.

We order the set S by the (x,y,z) coordinates that denote the end of the position vector P, first by z, then y, then x. This ordering allows us to assign a four tuple to a unique position within an oriented NxNxN matrix (the algorithm we use is described under implementation). This matrix serves as our discrete representation of space. C(S,t) returns our NxNxN matrix at time t, and we calculate as follows:

$$C(S,t) = M(B\left(C(S,t-1) + \Delta\big(C(S,t-1)\big)\right))$$

Our delta function takes as an input an NxNxN matrix, C(S,t-1). The delta function also returns an NxNxN matrix containing four tuples, similar to C. However the value of the four tuples are as follows (D(t), A(t), 0, 0). We then add the two matrices using standard matrix addition and addition between the four tuples by element.

$$D(T) = \int_{t-1}^{T} (V + \int_{t-1}^{T} a(x)dx)\, dy$$

$$A(T) = \int_{t-1}^{T} a(x)dx$$

With this discrete and simplified spatial representation we began to look at the local behaviour of our model in regards to collision. To do this we chose to examine “balls” of “radius” one around our four tuples position within the oriented matrix. We defined the distance as follows:

$$d\big(A_{i,j,k}, B_{i',j',k'}\big) = (i+j+k) - (i'+j'+k')$$

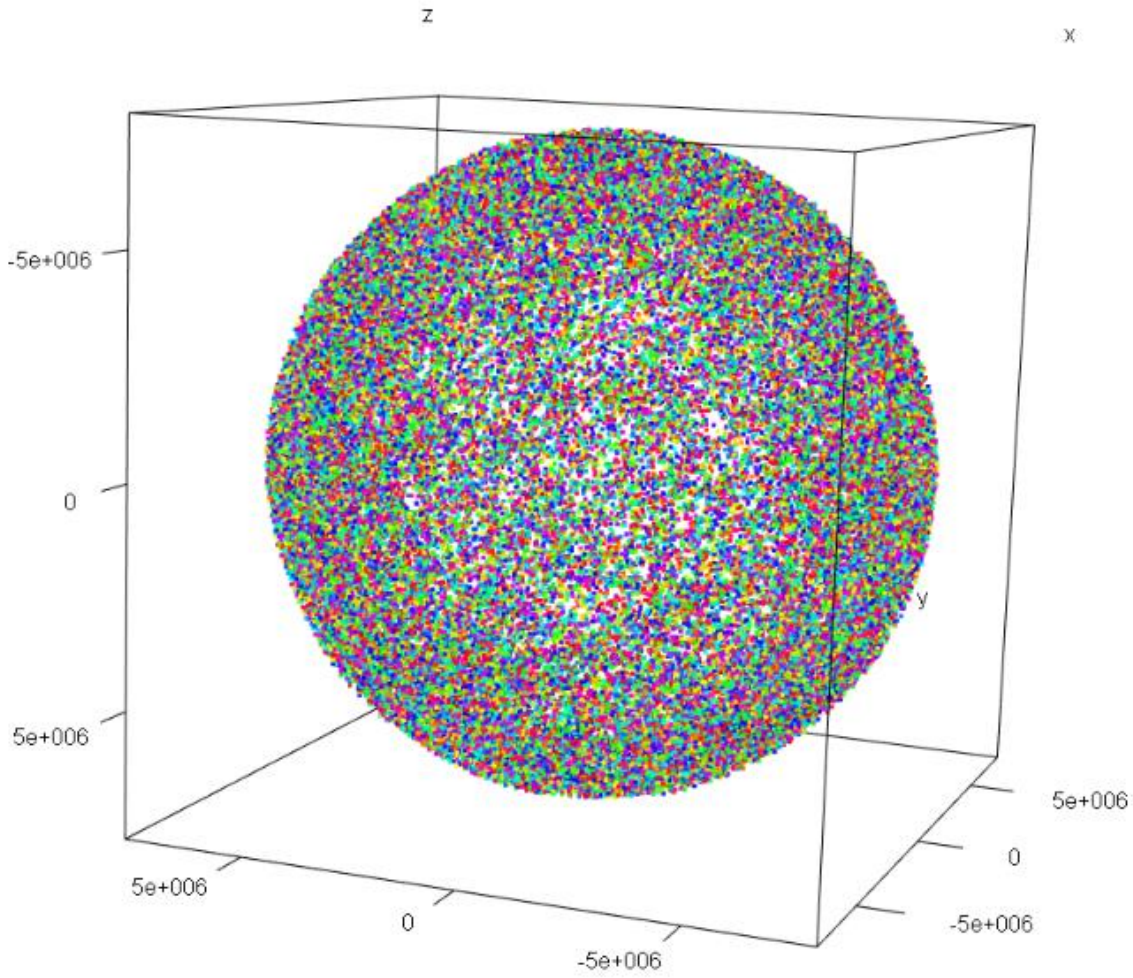
These balls were chosen because their contents are the most likely candidates for collision. Abstractly we define function B, which takes as input an NxNxN matrix, and returns a similar matrix of four tuples. However, this maps the balls centered around some element C_{ijk} to a ball centered around the same element, but with possibly different four tuples. We were able to define this rule explicitly in our numerical approximation, but still struggle to find an accurate description of the function which lies between the realms of complete abstraction and Java code.

The final element of our model is the function M, which consumed a large swath of time but refused to yield any logical solution to us. This function takes in an NxNxN matrix and permutes its elements so that they are once again ordered as described above, within the given orientation. We began our search for some kind of algorithm or function to perform this task with the hope of defining it locally. Even after our failed search, we still feel there is likely a way to take advantage of the fact that the original matrix C(S,t-1) was ordered and permute the balls we defined within the matrix, or 3x3x3 submatrices, in order to reorder the overall NxNxN matrix. Due to the fact we were unable to define this function, we describe the method we use to sort our four tuples below, under implementation.

4 Model Implementation

4.1 Overview

In order to implement our model for testing we created a computer simulation on a discrete time interval utilizing physics equations to calculate the state of our junk system in time T (in seconds).



4.2 Space Junk

The focus of much of our efforts was the space junk (an object of that name in our java based model). Our thought process in this area began with having objects of different shapes, spinning at different speeds, with randomized center of gravities. This was not possible in our given time frame and remains an area for future exploration. In addition we hoped to allow for collisions that lead to debris gathering together or splitting into pieces (Kessler Syndrome). This also was beyond our scope for this project, but would allow better analysis of impact focused removal options and allow for examining the impact of debris scattering on the overall situation. Regardless of our time necessitated simplifications, our junk when displayed visually through 3D graphing in R creates slightly elliptical orbits all concentrated in LOE and with a foci at the origin, the location of the center of the earth.

This space junk is an object defined in Java, endowed with some attributes. The first is the center of mass (CoM). This is what determines the endpoint of our position vector, the length of which is used in many of our calculations. The position has three degrees of freedom (x,y,z), which we generate randomly. We have a constraint on the length of this vector for it to fall within the lower bound (LB) and upper bound (UB) of a LOE. First we assign a random value to x, sampled from the uniform distribution over the interval [-LB,LB]. Next we assign a random value to y

from the uniform distribution over [-(LB*LB-x*x)^(1/2), (LB*LB-x*x)^(1/2)]. Finally we assign a value to z randomly from uniform distributions guaranteeing the length of the position vector falls in [LB,UB]. Finally we shuffle these three

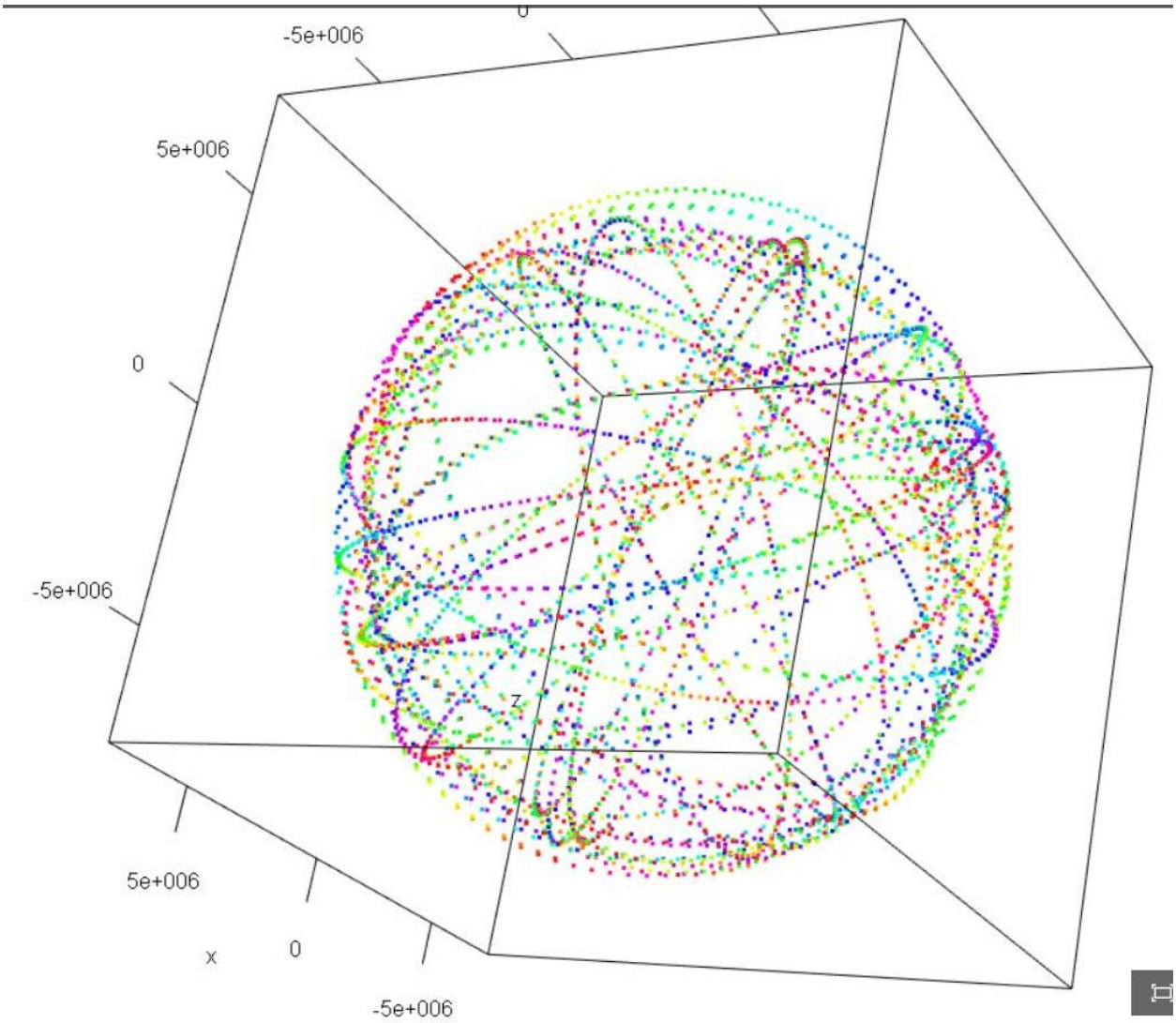
With this position determined three of the six degrees of freedom that determine a keplerian orbit are decided. Next we determine the other three, belonging to our velocity vector. The magnitude is calculated as the speed required to maintain a circular orbit at the given distance from the center of the earth. This is a simplification chosen to be able to more easily solve the problem of creating a velocity which keeps a given piece of junk in orbit.

We randomly select a dimension and assign that component of velocity a value from the uniform distribution [-speed,speed]. Using this speed we also create a constraint, with the magnitude of the velocity equal to the speed. In addition we demand the dot product of the position and velocity vectors be 0, which gives orthogonality (chosen to decrease the number of space junks created which do not remain in orbit). This system of equations first results in a quadratic equation, giving two possible roots, which we choose from randomly. This is used with the randomly chosen velocity component to solve for the final component. The velocity and position vectors determine the orbit of the space junk created.

In addition, each space junk is created with a radius (shape is assumed to be spherical) and a mass. These two components are necessary for many calculations, and to work with collisions.

4.3 Orbits

The space junk orbits are determined as a function of discrete time steps. Our time step is designed to be one second. The space junks move forward at their velocity and have an acceleration towards the earth from gravitational force. This is approximated nearly linearly, over one second. This process is repeated, each time updating the velocity and positions according to the displacement determined by the one second approximation. The outcome is as shown.



4.4 Discrete Representation of Space

In order to work with collisions within our large collection of space debris we mapped it to an NxNxN matrix. In order to do this we first generated N^3 space junks, and lined them up in an array. This array was sorted by the z coordinate of each space junk, and then divided into N arrays of N^2 length. These N arrays are then sorted by their y-coordinate and divided into N arrays of length N. Once these arrays are sorted each individual piece of debris is linked to a specific location in an NxNxN matrix, which is oriented as shown.

This structure given to our data allows for improved collision detection, which is expanded upon in the following section. In our implementation, after movement along the orbit and collision checking, the entire matrix is resorted and ordered. This was convenient to implement, but we feel there is likely a more efficient way to reorder the matrix. The primary idea we explored is employing an algorithm that locally swaps elements of the matrix, taking advantage of the knowledge that the matrix was previously ordered and each piece of debris has travelled for one

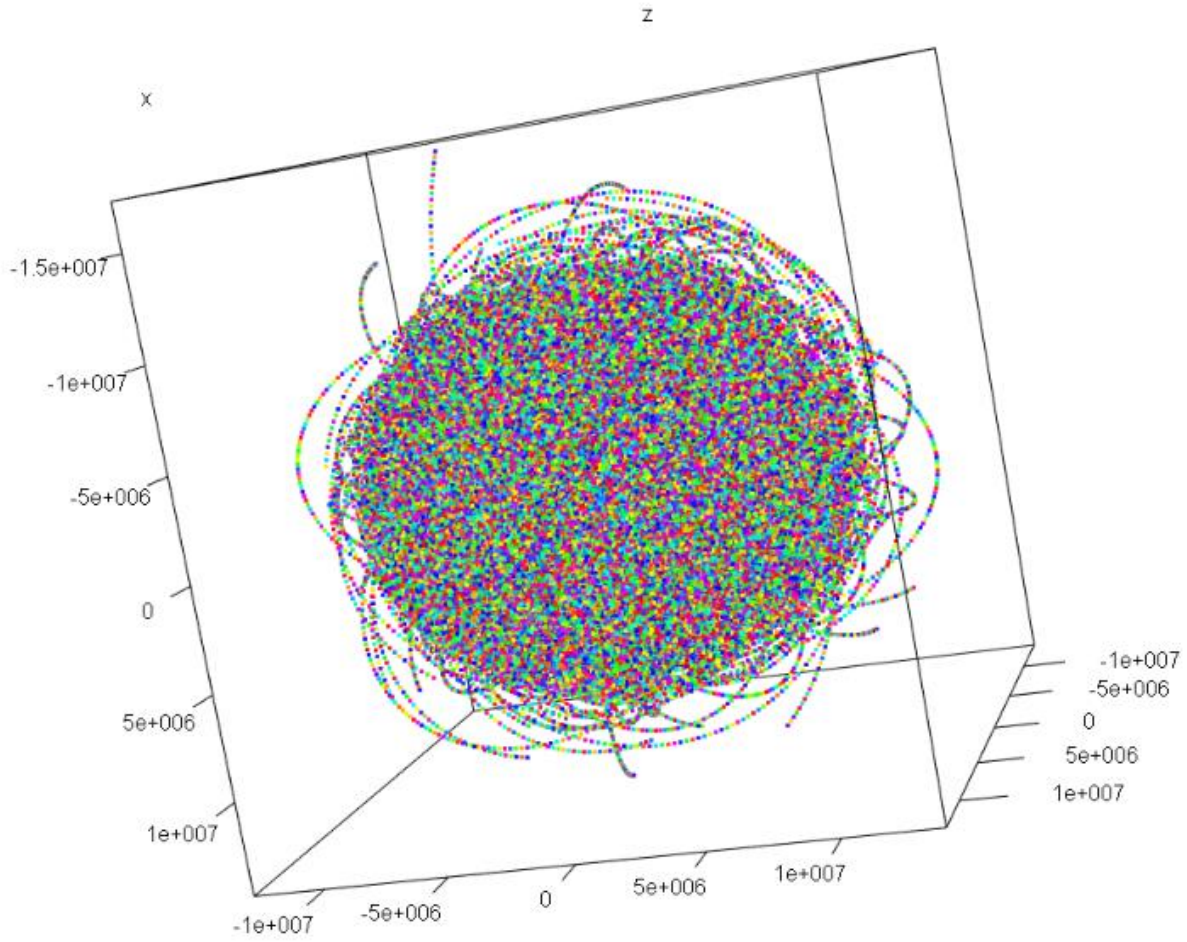
second. It also may be possible to take advantage of the fact that there are a number of central entries that are next to the earth, not actually their neighbors in the matrix, and thus do not need to worry about swapping with elements on the opposite side of the planet (but which are stored adjacently in the matrix).

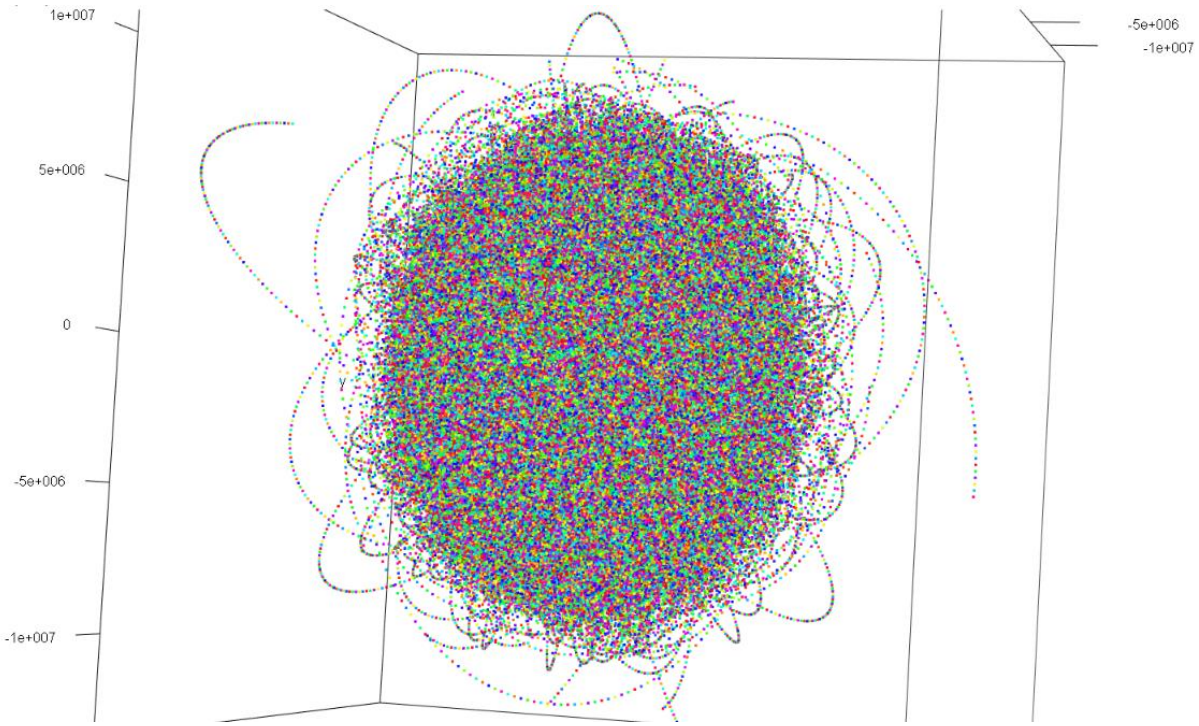
One issue with this representation is demonstrated above in two dimensions. Due to the relative ordering, the A_{ijk} element always is the ith in z coordinate, the jth in y coordinate, and kth in x coordinate. However the A(i+1)jk element may actually be closer to the to an element with different j and k in floor i than A_{ijk}. The results of this are dealt with under collision detection.

4.5 Collision Detection

Collision detection is performed on a sub-cube of our cube of junk. We created a formula, which at time T (a fraction of our usual interval) returns the distance between two center of masses of debris, minus the sum of the two objects radii. We then find zeroes of this function by checking the sign of the output at discrete subintervals of T, and looking for sign change. With these zeroes found we aim to apply the appropriate collision calculations at that time, and recalculate the end position and velocity of the two objects. The formulae are shown below, and featured is a picture in which we simulated approximately 3,400 space junks with extended radii to guarantee frequent collision. Rather than forming a tight sphere of orbits, one can see the orbits entering and leaving the sphere caused by collision.

$$S(t) = Rcos(t)\hat{d} + Rsin(t)(\hat{n} \times \hat{d}) + C$$


z
x
-1.5e+007
-1e+007
-5e+006
0
5e+006
1e+007
-1e+007
-5e+006
0
5e+006
1e+007
-1e+007
-5e+006
0
5e+006
1e+007



$t = time$

$R = Radius\ of\ sphere$

$$\hat{d} = \frac{< x' - x\ , y' - y, z' - z >}{\sqrt{(x' - x)^2 + (y' - y)^2 + (z' - z)^2}} = unit\ vector\ of\ \vec{d}$$

$$\hat{n} = \frac{< -y, x, 0 >}{\sqrt{(y^2 + x^2}} = unit\ vector\ of\ \vec{n}$$

$C = D(t) = Displacement\ vector\ at\ time\ t$

$S(t) = Parametric\ equation\ of\ a\ sphere\ as\ a\ function\ of\ time$

$Starting\ point\ at\ t = 0: < x, y, z >$

$Ending\ point\ at\ time\ t: < x', y', z' >$

$\vec{d} = < x' - x,\ y' - y,\ z' - z > = displacement\ vector$

$\vec{n} = < -y, x, 0 > = vector\ orthogonal\ to\ vector\ \vec{d}$

5 Solution Analysis

5.1 Water Walls Solution

The first solution we explored is an idea called the Ballistic Orbital Removal System, which was proposed by James Hollopeter of GIT Satellite [3]. In this method, decommissioned missiles filled with water are fired into space. The rockets then release the water into orbit and create a field of crystallized water that slow down orbiting space debris, thus causing the space debris to fall out of orbit. This is a relatively inexpensive solution, with the only real costs coming from fossil fuel (as water can be taken from the ocean).

5.2 Space Net Solution

The second solution we explored is the launching of a giant space net attached to a rocket. Ideally, the space net would span a large distance, rotate relatively little (or not at all), and be composed of a durable and lightweight material. Any space debris that hits the net would be slowed and lose velocity, eventually falling to earth. The costs that we would take into consideration for this model include the cost of the balloon material and fossil fuel.

5.3 Ground-based Laser Solution

The third solution that we explored is the use of a ground-based laser to create plasma jets on space debris for 1-2 hours. This would cause the space debris to slow down gradually until it re-enters and either burns up in the atmosphere or falls into the ocean. The laser technology needed for such a venture has been around for 15 years and as a result this is our most practical option. The costs associated with this venture include $1 million for each laser (each of which can take down up to 10 objects per day) and the cost of operating the laser, which is 5kW.

Analysis

All of these solutions are of one off in nature and thus are relatively costly. Due to time and computing constraints we were not able to fully explore their efficacy, but we did consider the costs. Obviously construction and engineering of both projects is a large endeavour. In this competition we have learned personally how complicated orbital mechanics are when aiming for a high degree of accuracy. If a reusable launch platform (likely a rocket) was developed for these two solutions then costs would decline rapidly on a per junk basis over the life of the tool.

Another difficulty is the sparse nature of space junk. It is spread out over a very large area and in our simulations we found it difficult to guarantee a collision with another object. However this can be countered with high quality planning and precalculation.

A final risk is the additional junk and debris these methods both put into orbit, and burn up in our atmosphere. If the permanent method of debris removal is atmosphere based, we did not find any information on the long term affects to our atmosphere. Secondly these systems have the risk of malfunctioning and creating more debris of even greater magnitude. These facts combine to demand much further research before any solution is implemented.

Conclusion

Due to the difficulty of computing such a simulation we ran out of time to properly test our model and solutions at an accurate level. However we did greatly expand our understanding of the space debris problem and orbital mechanics in the process. From these developments we have two primary conclusions. The first is in regards to the often proposed laser solution. This solution is great for spot removal but not cost effective at a large scale, due to the volume and

types of junk in orbit. The second conclusion is that this is a relatively deterministic process, that definitely has an optimal solution. Based upon the fact that we were able to model the skies, albeit simply, over a single weekend on budget laptops, we are confident that with appropriate attention and resources this is a problem which humanity can solve.

References

[1] Riebeek, Holly. "Catalog of Earth Satellite Orbits : Feature Articles." *Catalog of Earth Satellite Orbits : Feature Articles*. NASA, 4 Sept. 2009. Web. 01 Feb. 2016.
[2] "NASA Orbital Debris FAQs." *NASA Orbital Debris FAQs*. NASA Orbital Debris Program Office, n.d. Web. 01 Feb. 2016.
[3] "Space Safety Is No Accident." *Google Books*. Ed. Tommaso Sgobba and Isabelle Rongier. International Association for the Advancement of Space Safety, n.d. Web. 01 Feb. 2016.

Appendix

Model Implementation and code: https://github.com/tameravci/COMAP-MCM-SpaceJunk