TRACKING OF SPACE DEBRIS FROM PUBLICLY AVAILABLE DATA

Philip Hoddinott

Submitted in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Approved by:
Kurt Anderson, Chair
John Christian
Matthew Oehlschlaeger

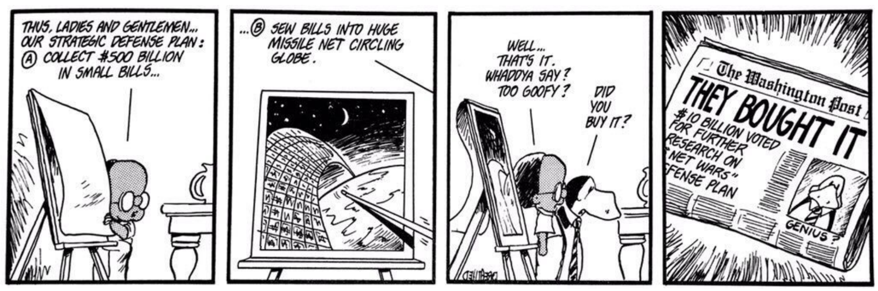
Department of Mechanical, Aerospace, and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York

November 2018

October 30, 2018

But, Captain, I cannot change the laws of physics

-Lt. Commander Montgomery "Scotty" Scott
USS *Enterprise*



Contents

List of Tables

List of Figures

1 Acknowledgments

This paper would never have been possible with out the help of so many people. I am indebted to my parents and my brother for their support. My teachers and professors who have helped me along the way have my deepest gratitude. I am thankful towards Paul McKee for his assistance with Latex and to John Benke for his assistance with Github and PHP. I am incredibly thankful towards my Advisor Professor Kurt Anderson for his support and his many phenomenal stories. Issac Newton once said "If I have seen further it is by standing on the shoulders of giants"[9]. I would like to thank you all for being my giants.

2 Abstract

The purpose of this thesis is to develop software that will acquire information about earth orbiting debris, turn the information into a usable format for someone with basic orbital mechanics experience. The code will take in parameters about the orbital data from the user. Then it checks for existing files and communicates with NORAD servers. The data is downloaded from NORAD and converted from the Two Line Element set format to Keplerian Orbital Elements. From here the data may be sorted according to the user's desire. The code will be able to handle errors that occur during the process.

3 Introduction

Before the methods of this project are explored, the background of the problem must first be presented. Space debris and the threat they pose will be explained, as well as the proposed solution.

3.1 Space Debris

March 17th, 1958 the Vanguard 1 satellite was launched. It was the fourth man made satellite. Despite communication being lost in the 1960s, it is still in orbit to this day.

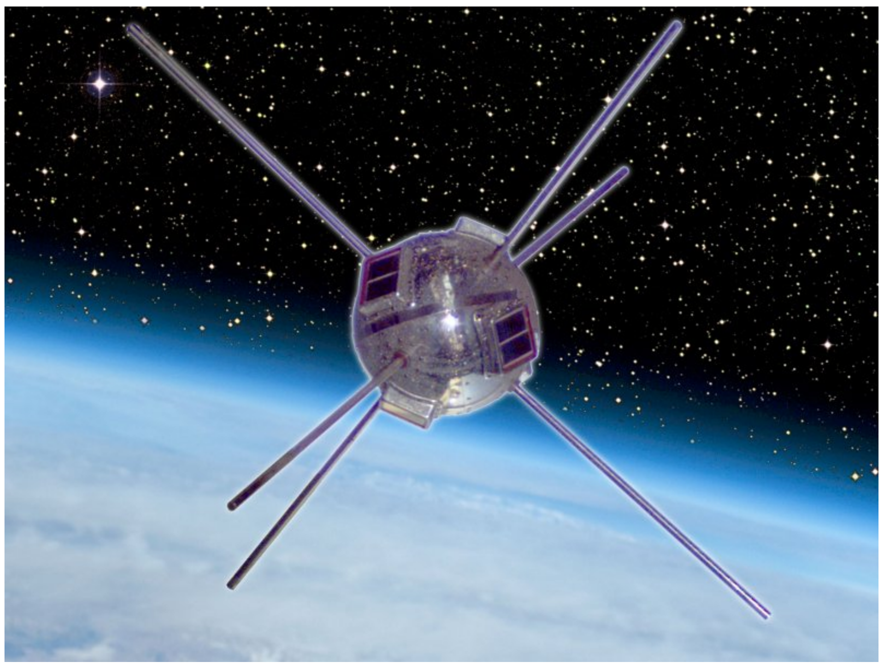


Figure 1: Artist's rendition of Vanguard 1 in orbit[4].

While spacecraft are supposed to be moved to a graveyard orbit, not all are. These large objects would be catastrophic if they collided with another object in orbit. However large objects do not pose the greatest threat. Their size makes them easy to track and there are comparatively little of them.

The debris that poses the greatest threat are the smaller pieces. Their size makes them harder to track and predict trajectories. And they are numerous. As of 2013 it is estimated

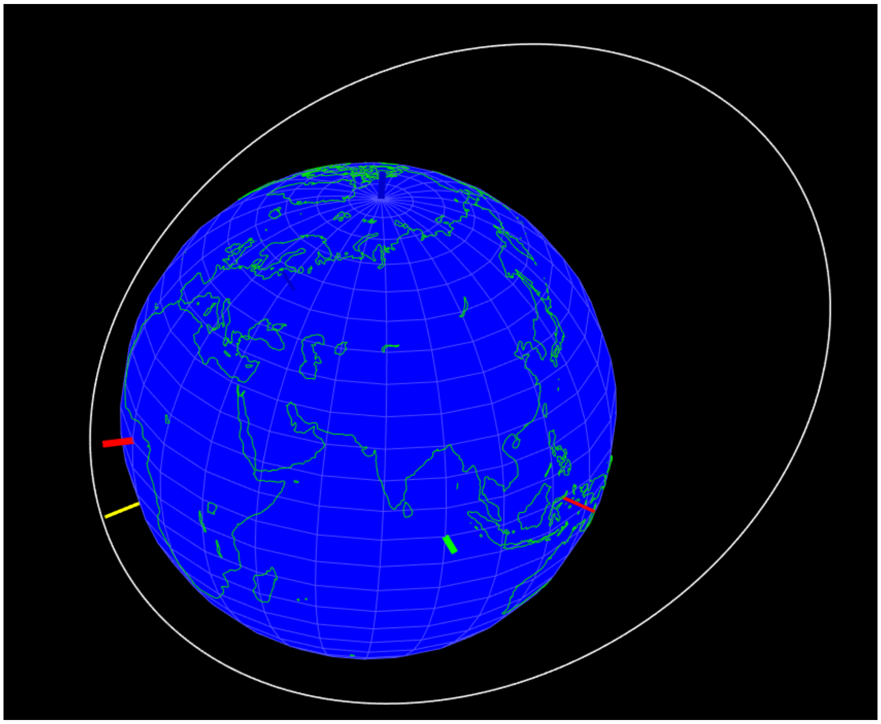


Figure 2: Vanguard's Orbit as of 10/17/2018 UPDATE THIS

that there are:

- 29,000 Objects larger than 10 cm.
- 670,000 Objects larger than 1 cm.
- Over 170,000,000 Objects larger than 1 mm.

These objects can do grievous harm to people or objects in space. A 10 cm object can cause the destruction of a satellite, a 1 cm object could penetrate the ISS's shields putting lives at risk and a 1 mm object could destroy critical subsystems.[10] The impact of one of these objects is shown in figure 3.

3.2 Kessler Syndrome

More than just harm to individual spacecraft or persons there is the threat of Kessler Syndrome. Also known as the Kessler effect or collision cascading, this is a scenario where there are enough objects in low earth orbit that a collision between objects will cause a chain reaction creating more and more space debris. This could result in space being inaccessible for

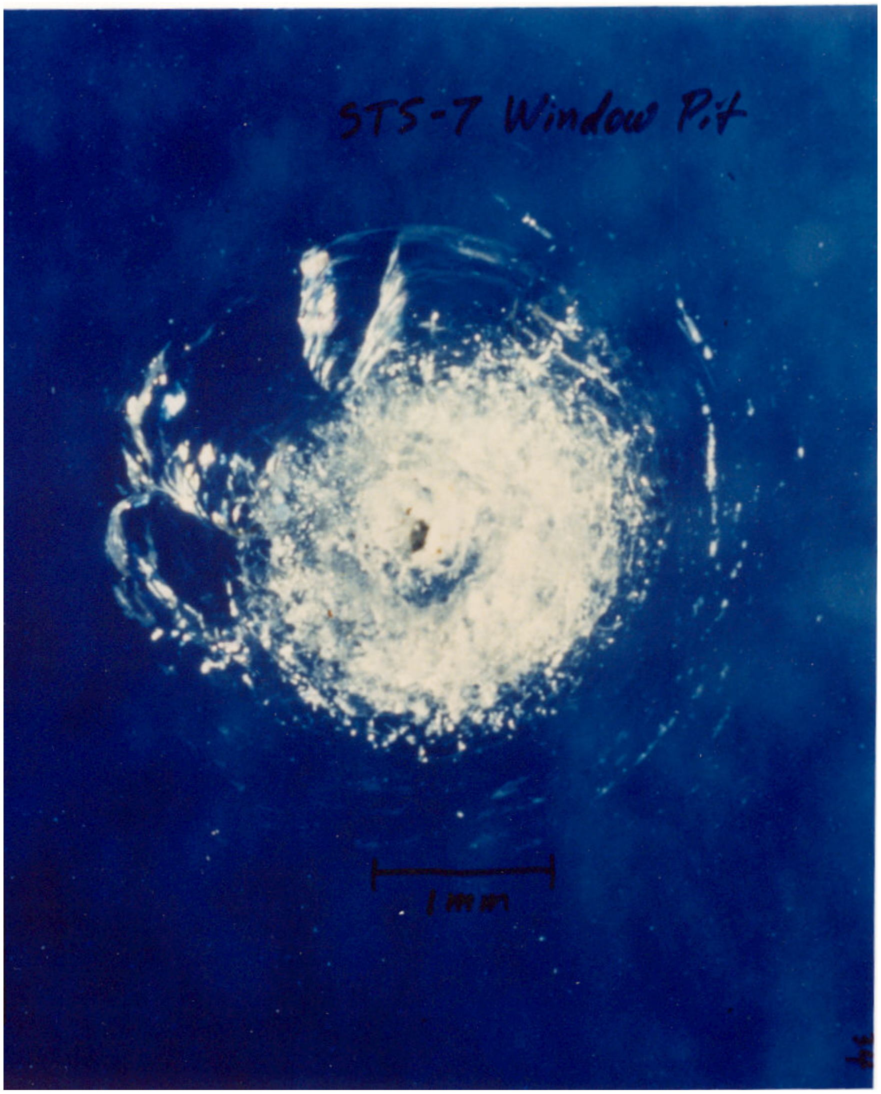


Figure 3: An impact crater on one of the windows of the Space Shuttle Challenger following a collision with a micrometeoroid[5]

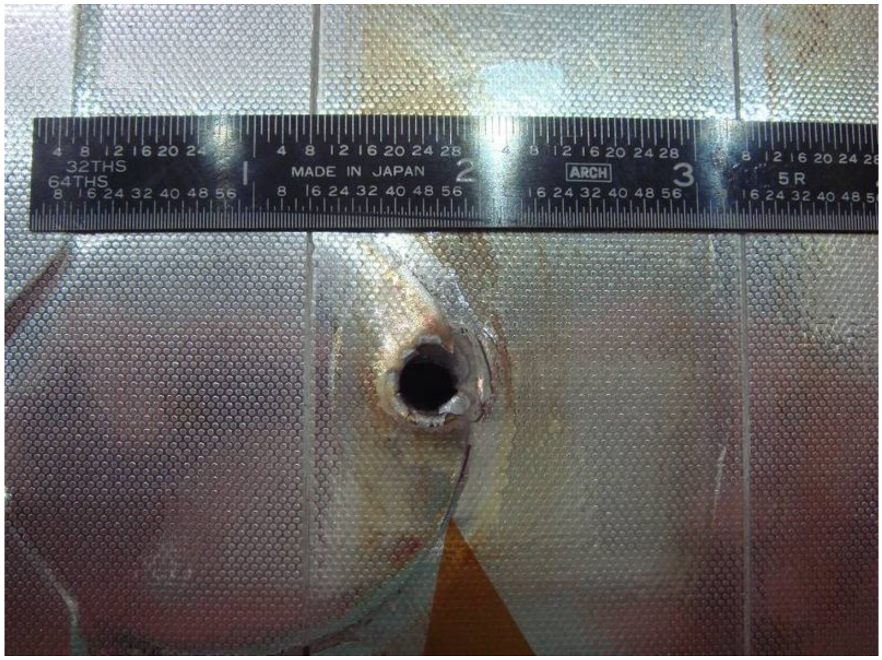


Figure 4: Debris impact on Space Shuttle Endeavorer's radiator panel[6]

years to come, as any object that left the earth's atmosphere would be immediately shredded by debris (thus adding even more debris).

This can be prevented by moving derelict spacecraft to graveyard orbits. However even spacecraft in graveyard orbits are not perfectly safe. Coolant tanks can puncture and coolant droplets freeze adding to the debris.

The alternative is to lower the orbit until it experiences atmospheric effects. But if the debris's orbit keeps it out of the atmosphere, then the debris must be moved.

3.3 CubeSats

For the task of moving debris, a CubeSat will be used. CubeSats are a type of miniaturized satellite. CubeSats get their name from their structure, they are made up of one or more $10 \times 10 \times 10$ cm units. These units have a maximum mass of 1.33 kilograms. First launched in 1998, as of August 2018, there have been 875 CubeSats launched[11].

CubeSats are used for projects that are too risky for a larger more expensive satellite, often demonstrating new technologies. They can take on larger risks due to their low cost.

As such CubeSats are common for experiential missions such as this one.

3.4 OSCAR

There is a CubeSat currently being designed by Rensselaer Polytechnic Institute (RPI). This CubeSat's goal is to deorbit space debris by use of a magnetic tether.

This CubeSat is called O.S.C.A.R, which is short for "Obsolete Satellite Capture and Recovery". It should be noted that this name is only temporary and will likely change in the near future. However for the purposes of this report, the RPI CubeSat will be referred to as OSCAR.

OSCAR will be a three unit ($10 \times 10 \times 30$) CubeSat with a mass of four kilograms. It will be able to remove four pieces of space debris.[12] For OSCAR to be able to deorbit space debris, it must know where the space debris is.

3.5 Orbital Elements

This thesis is concerned with the acquisition of current orbital data of space debris and it's transformation into useful formats.

First orbital elements should be defined. Orbital elements are the parameters that define an orbit. The traditional Keplerian elements found in many textbooks are as follows:

$$\begin{array}{cc} h & \text{specific angular momentum} \\ i & \text{inclination} \\ \Omega & \text{right ascension (RA) of the ascending node} \\ e & \text{eccentricity} \\ \omega & \text{argument of perigee} \\ \theta & \text{True anomaly} \end{array} \tag{1}$$

They may be seen in figure 5.

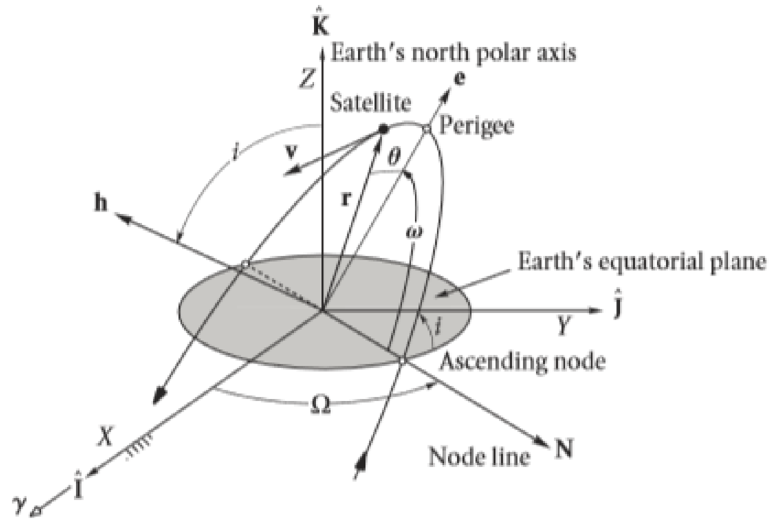


Figure 5: Geocentric equatorial frame and the orbital elements[7].

However for space debris the following set of orbital elements is slightly more useful.

$$\begin{array}{cc} e & \text{eccentricity} \\ a & \text{semimajor axis} \\ i & \text{inclination} \\ \Omega & \text{right ascension (RA) of the ascending node} \\ \omega & \text{argument of perigee} \\ \nu & \text{Mean anomaly} \end{array} \tag{2}$$

This is preferable as the semi major axis is used in conjunction with eccentricity with it's relation to perigee and apogee. However this is not a hard rule, and the code could be altered to use the first set of orbital elements. The orbital elements are better demonstrated in figure 6 and figure 7

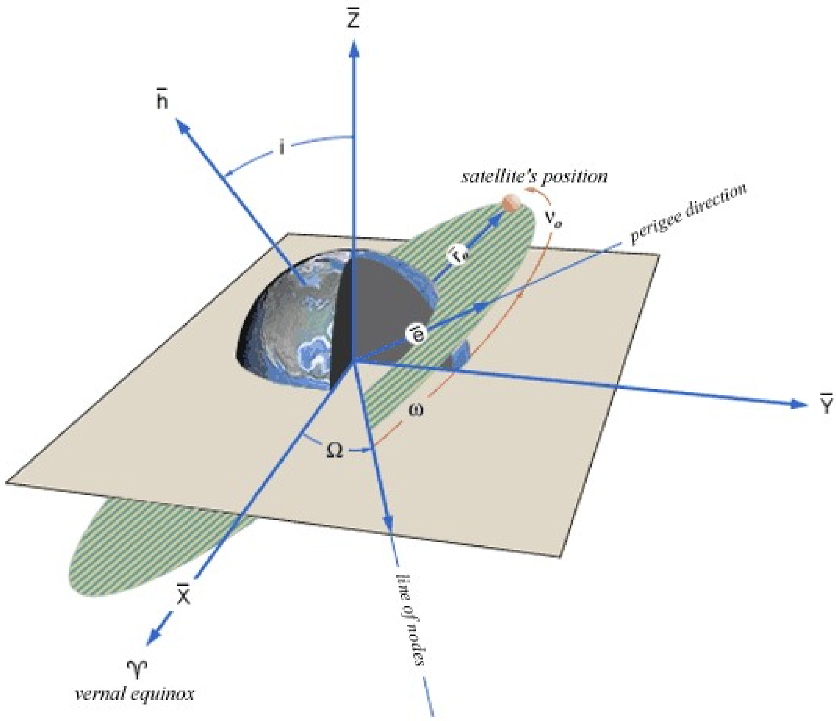


Figure 6: Diagram of Earth and Orbital Elements as well as Cartesian State Vectors[1]

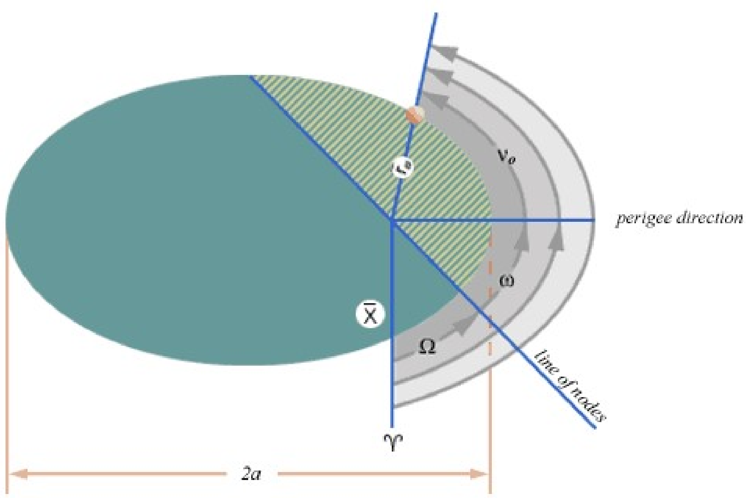


Figure 7: Diagram of Earth and Orbital Elements[1]

Table 1: Orbital Elements and their symbols[1]

Name	Symbol	Description
Semi-major axis	a	Defines the size of the orbit.
Eccentricity	e	Defines the shape of the orbit.
Inclination	i	Defines the orientation of the orbit with respect to the Earth's equator.
Argument of Perigee	ω	Defines where the low point, perigee, of the orbit is with respect to the Earth's surface.
Right Ascension of the Ascending Node	Ω	Defines the location of the ascending and descending orbit locations with respect to the Earth's equatorial plane.
True/Mean Anomaly	ν	Defines where the satellite is within the orbit with respect to perigee.

4 Data Sources and Formats

First data sources for space debris must be found, and the data formats must be understood. The most common data format for orbital elements is the Two Line Element Set (TLE). Despite the large number of objects in space, real time data is surprisingly hard to find. While there are various websites and programs to track the location of objects such as the International Space Station, debris from old spacecraft is not quite as popular. The website space-track.org publishes TLEs for public use. First the TLE format will be explained then their acquisition will be discussed.

4.1 The Two Line Element Format

Orbital elements provide the means to determine a theoretical orbit. Since spacecraft are constantly experiencing forces such as atmospheric drag or solar wind their orbital elements are constantly changing.

A Two Line Element set (TLE) is a data format that encodes a list of orbital elements for an object that orbits Earth at a given time. Using perturbation models TLEs can be used to compute the object's state at a specific time. The TLE format was originally designed for punch cards, now .txt files are used with two 70-column ASCII lines.

The example from figure 8 shows how orbital information is derived from a Two Line

Element.

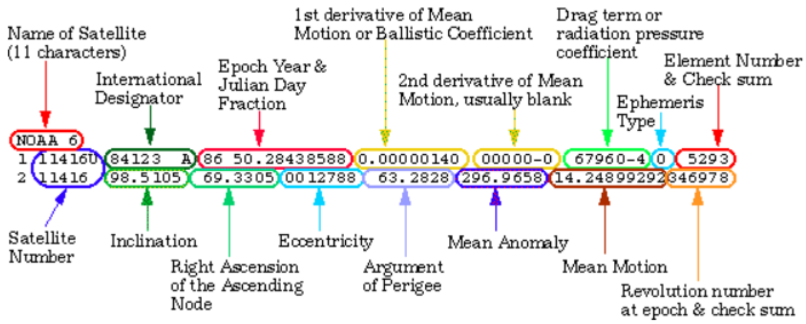


Figure 8: TLE with descriptions[8].

1. **Name of Satellite:** NOAA 6

   This is the name of the satellite. In this example it is NOAA 6.

2. **International Designator:** 84123 A

   This number shows the last two digets of the launch year and what number launch of that year it was. In this example it is 84 123 A. The satellite was launched in the year 1984 and it was the 124th launch of the year. The A means that it was the first object that came from the launch.

3. **Epoch Date and Julian Date Fraction:** 86 50.28438588

   The Epoch date shows the year this TLE was made and the Julian Date fraction is the number of days in said year.

4. **First Derivative of Mean Motion:** 0.00000140

   This is the daily rate of change of number of revolutions the object completes per day divided by two. This is also called the ballistic coefficient.[8] It's units are $\frac{rev}{day^2}$.

5. **Second Derivative of Mean Motion:** 00000-0

   This is the second derivative of mean motion divided by six, with units of $\frac{rev}{day^3}$.

6. **Drag term or B* Term:** 6970-4

   Of the terms given here, the B* term is the least heard of. It is a way of modeling drag on orbiting objects in propagation models. Aerodynamic drag is given by the following equation:

$$F_D = \frac{1}{2}\rho C_d A v^2 \tag{3}$$

   Where A is the area, C_d is the drag coefficient, v the velocity, and ρ is the fluid density. From Newton's second law the acceleration due to the force of drag is

$$F = m \times a \rightarrow a_D = \frac{F_D}{m} = \frac{\rho C_d A v^2}{2m} = \frac{C_d A}{m} \times \frac{\rho v^2}{2} \tag{4}$$

The ballistic coefficient and starred ballistic coefficient are given by:

$$B = \frac{C_d A}{m} \rightarrow B^* = \frac{\rho_0 B}{2} = \frac{\rho_0 C_d A}{2m} \tag{5}$$

This turns the equation for acceleration due to drag into [13][14]

$$a_D = \frac{\rho}{\rho_0} B^* v^2 \tag{6}$$

7. **Element Set Number :** 529 The element set number represents how many TLEs have been generated for this object as of the current TLE.

8. **Check Sum:** 3

   The checksum is a bit of data used to detect errors that occur during transmission.

9. **Satellite Number:** 11416U

   This is the Satellite Catalog Number and designation of the object. A U means unclassified.

10. **Inclination (degrees):** 98.5105

11. **Right Ascension of the Ascending Node (degrees):** 69.3305

12. **Eccentricity:** 0012788

13. **Argument of Perigee (degrees):** 63.2828

14. **Mean Anomaly (degrees):** 296.9658

    These four terms are all orbital elements.

15. **Mean Motion:** 14.24899292

    This is the orbits per day the object completes. There are always 8 digits after the decimal place.

16. **Revolution Number:** 34697

    This is the number of revolutions the object has completed as of the TLE's generation.

17. **Checksum:** 8

    This is the checksum for the second line.

A second example is given below, this time with references to the position of values, for extraction. The line under the dashes is the reference number line. The example TLE is described in table 2. This page is a useful compact summary of a TLE.

```
ISS (ZARYA)
1 25544U 98067A   04236.56031392  .00020137  00000-0  16538-3 0  9993
2 25544  51.6335 344.7760 0007976 126.2523 325.9359 15.70406856328906
----------------------------------------------------------------------
1234567890123456789012345678901234567890123456789012345678901234567890
1         2         3         4         5         6         7
```

Table 2: Description of example TLE[2]

Line 0		
Columns	**Example**	**Description**
1-24	ISS (ZARYA)	The common name for the object based on information from the Satellite Catalog.
Line 1		
Columns	**Example**	**Description**
1	1	Line Number
3-7	25544	Satellite Catalog Number
8	U	Elset Classification
10-11	98	International Designator (Last two digits of launch year)
12-14	067	International Designator (Launch number of the year)
15-17	A	International Designator (Piece of the launch)
19-32	04	Epoch Year (last two digits of year)
21-32	236.56031392	Epoch (day of the year and fractional portion of the day)
34-43	.00020137	1st Derivative of the Mean Motion with respect to Time
45-52	00000-0	2nd Derivative of the Mean Motion with respect to Time (decimal point assumed)
54-61	16538-3	B* Drag Term
63	0	Element Set Type
65-68	999	Element Number
69	3	Checksum
Line 2		
Columns	**Example**	**Description**
1	2	Line Number
3-7	25544	Satellite Catalog Number
9-16	51.6335	Orbit Inclination (degrees)
18-25	344.7760	Right Ascension of Ascending Node (degrees)
27-33	0007976	Eccentricity (decimal point assumed)
35-42	126.2523	Argument of Perigee (degrees)
44-51	325.9359	Mean Anomaly (degrees)
53-63	15.70406856	Mean Motion (revolutions/day)
64-68	32890	Revolution Number at Epoch
69	6	Checksum

5 NORAD Space-Track

Now we understand the format of data that is available (the Two-Line Element set) and the format we want the data to be in (the orbital elements), the next thing to do is get the data. We want a source of TLEs for objects in space. Fortunately, there is one.

As well as tracking one S. Claus every December 24th[15], NORAD also tracks all objects currently in space. This is done through the Space Surveillance Network[16]. The process of TLE gathering and updating is somewhat shadowy. [17] What is known is that observations are collected multiple times per day at the Joint Space Operations Center (JSPOC) which is operated by the US Air Force Space Command (AFSPC). Then the unclassified TLEs are passed on for public release via the website space-track.org. Objects in space are tracked by radar systems (conventional and phased array) as well as the Ground-Based Electro-Optical Deep Space Surveillance system (GEODSS)[18].

5.1 Space-Track.org

In their own words, Space-Track.org "promotes space flight safety, protection of the space environment and the peaceful use of space worldwide by sharing space situational awareness services and information with U.S. and international satellite owners/operators, academia and other entities."[19]

Information about objects in space may be found the website space-track.org. Space-track is the main source for orbital data, though some are also available from the website celestrak.com. Of the two space-track has far more information and better methods of access. "Space-Track.org is managed, maintained and administered by JFSCC"[20]. Space-track allows information to be downloaded manually from the TLE search [2] or the satellite catalog search [21]. While these are useful tools the code needs a way to automatically download data. Fortunately space-track also has an API that allows queries.

5.2 Space-Track Query

An application programming interface or API is a set of methods that allow simple communication between different codes. The Space-Track API allows for programs to send requests for data on objects in space. It is limited to less than 20 requests per minute and less than 200 requests per hour.

The API works by building an API similar to the following URL:

`https://www.space-track.org/basicspacedata/query/class/boxscore/`

Where:

- Base URL: https://www.space-track.org/
- Request Controller: basicspacedata
- Request Action: query
- Predicate Value Pairs: class/boxscore/

The only publicly available Request Controller is basicspacedata. The expandedspacedata controller requires SSA Sharing Agreements [3]. The first two sections of the URL will never change. What will changes is the query, the class, and the boxscore. The request classes that are of interest to this thesis are in table 3.

Table 3: Request Classes of Interest[3]

Class Name	Class Description
tle	Historical record of orbital element sets (Two-Line Elements). Additional metadata columns are available so that users can filter their queries by these values, downloading only the data they need. Shown in html, csv, and xml formats, these additional columns include: object_name, object_type, semi-major axis, period, apogee, perigee, and file. NOTE: There are over 97 million TLEs in the database. The user is strongly advised to limit their API queries by OBJECT_NUMBER / NORAD_CAT_ID AND an epoch range such as ">now-30" to avoid "Query range out of bounds" errors.
satcat	Satellite Catalog Information. The "CURRENT" predicate indicates the most current catalog record with a 'Y'. All older records for that object will have an 'N'.
omm	Orbit Mean-Elements Message (OMM)Keplerian element format that complies with CCSDS Recommended Standard 502.0-B-2

Question for OGE/Professor Anderson: I am citing these tables word for word, what is the proper way to do that for a table?

The Orbit Mean-Elements Message was not used in this thesis because it could not handle large groups of requests. However because it directly returns the Keplerian elements, it may be useful if the API functionality is improved in the future. As a result the TLE and satcat classes are the only classes used in this code. The predicate values pairs may be specified by using REST Operators. The REST Operators that can be used are seen in table 4

Table 4: REST Operators[3]

Operator	Description
>	Greater Than (alternate is %3E)
<	Less Than (alternate is %3C)
<>	Not Equal (alternate is %3C%3E)
,	Comma Delimited 'OR' (ex. 1,2,3)
–	Inclusive Range (ex. 1–100 returns 1 and 100 and everything in between)Date ranges are expressed as YYYY-MM-DD%20HH:MM:SS–YYYY-MM-DD%20HH:MM:SS or YYYY-MM-DD–YYYY-MM-DD
null-val	Value for 'NULL', can only be used with Not Equal (<>) or by itself.
∼∼	"Like" or Wildcard search. You may put the ∼∼before or after the text; wildcard is evaluated regardless of location of ∼∼in the URL.For example, ∼∼OB will return 'OBJECT 1', 'GLOBALSTAR', 'PROBA 1', etc.
ˆ	Wildcard after value with a minimum of two characters. (alternate is %5E) The wildcard is evaluated after the text regardless of location of ˆin the URL. For example, ˆOB will return 'OBJECT 1', 'OBJECT 2', etc. but not 'GLOBALSTAR'
now	Variable that contains the current system date and time. Add or subtract days (or fractions thereof) after 'now' to modify the date/time, e.g. now-7, now+14, now-6.5, now+2.3. Use <,>,and – to get a range of dates; e.g. >now-7, now-14–now

The REST Predicates are seen in table 5

Table 5: REST Predicates[3]

REST Predicates	REST Description
class/classname	Class of data message requested from the API. Required for all operations.
predicates/p1, p2,...	Comma-separated list of predicates to be returned as the result. Can also be set to 'all' or omitted entirely for default behavior of returning all predicates.
metadata/true	Includes data about the request (total # of records, request time, etc); ignored when used with tle, 3le, and csv formats.
limit/x(,x?)	Specifies the number of records to return. Takes an integer for an argument, with an optional second argument to specify offset. For example, /limit/10/ will return the first 10 records; /limit/10,5/ will show 10 records starting after record #5 (rows 6 through 15)
orderby/predicate (asc?—desc?)	Allows ordering by a predicate (column), either ascending or descending, with asc/desc separated from the predicate by a space. Multiple Orderings are supported, separating the pairs of predicate and asc/desc by commas (e.g. http://.../DECAY desc,APOGEE desc/...)
distinct/true	Removes duplicate rows.
format/xxxx	Specifies return format, can be: json, xml, html, csv, tle, 3le, kvn, or stream. See below for additional information. If no format is specified, the default is JSON
emptyresult/show	Queries that return no data will show 'NO RESULTS RETURNED' instead of the default blank page

Finally the data returned may be in the formats seen in table 6

Table 6: Available Data Formats[3]

Format	Request Rules
eXtensible Markup Language (xml)	No special request rules. For CDM request class, shows CCSDS-compliant format.
JavaScript Object Notation (json)	Preferred format, no special request rules.
HyperText Markup Language (html)	Not recommended for machine parsing. No special request rules.
Comma-Separated Values (csv)	Due to the limitations of the CSV specification, requesting CSV formatted data does not allow for transmission of metadata; metadata concerning the request will need to be gathered through one of the other formats prior to requesting CSV.
Two-Line Element Set (tle)	To get traditionally-formatted TLE data, request the 'tle', 'tle_latest', or 'tle_publish' class. We recommend that you omit predicates from URLs that use tle format. If you do include predicates, it must include line 1 & 2 like this: "/predicates/-TLE_LINE1,TLE_LINE2/". Like CSV, tle format ignores /metadata/true/ REST Predicate.
Three-Line Element Set (3le)	The format adds TLE_LINE0 or the "Title line", a twenty-four character name, before the traditional Two Line Element format. The 3le format defaults to include a leading zero so that you can easily find the object name via scripts. If you do not want the leading zero, you should include this in your URL to show the OBJECT_NAME instead of TLE_LINE0: "/predicates/OBJECT_NAME,TLE_LINE1,TLE_LINE2/". Like CSV and tle, 3le format ignores /metadata/true/ REST Predicate.
Key=Value Notation (kvn)	Key=Value Notation format, currently for use exclusively with CDM Class. Incompatible with /metadata/true/.
File Stream (stream)	File Stream format, currently for use exclusively with download class.

5.3 Code Queries

Using the following rules, the code generates URLs that query space-track for the relevant information. The first URL is used to acquire the relevant satcat IDs for space debris. It is as follows:

`https://www.space-track.org/basicspacedata/query/class/satcat/OBJECT_TYPE/debris/RCS_SIZE/small/LAUNCH_YEAR/>1990/orderby/DECAYasc/format/csv/metadata/false`

Where 1990 is the launch year, and should be replaced by the year the debris was launched in. This is the best way of controlling how much debris the user gets. Setting a launch year of 1960 will obviously included all small debris (debris that is less than 10 cm^2 cross section), while setting a launch year of 2018 would return very little debris. It is advisable to run test cases with small debris before performing operations on all available debris. A detailed walkthrough of the URL may be seen in table 7.

Table 7: Detailed Description of First URL

Section of URL	Description
https://www.space-track.org/basicspacedata/query/	This is the base URL.
class/satcat/	The class of the data will be the satcat Ids.
OBJECT_TYPE/debris/	The type of object this URL is looking for is debris
/CS_SIZE/small/	The URL is looking for debris classified as small
LAUNCH_YEAR/>**launchYear**/	The debris is limited by the provided launch year.
orderby/DECAY asc/	The debris are ordered by their decay. This order is not relevant, and only included because it is a necessary part of the query. The IDs are sorted small to large after the post request.
format/csv/metadata/false	Finally the format of the data returned is set to be a csv. The metadata of the request is not given.

The second URL is later used to return the TLEs from given satcat Ids. It follows the following format:

`https://www.space-track.org/basicspacedata/query/class/tle_latest/NORAD_CAT_ID/43543,43544,43555/orderby/ORDINAL%20asc/limit/3/format/tle/metadata/false`

A detailed description of this URL may be seen in table 8

Table 8: Detailed Description of Second URL

Section of URL	Description
https://www.space-track.org/basicspacedata/query/	This is the base URL, unchanged from the previous example.
class/tle_latest/	The class of the data will be the latest TLEs.
NORAD_CAT_ID/	The data will come from the following NORAD CAT IDs.
43543,43544,43555/	The NORAD CAT IDs that have been requested. Normally the code will request many of them, to make this easy to read only three IDs have been requested in this example.
orderby/ORDINAL%20asc/	The TLEs will be in ascending order. Ascending or descending order does not matter as the order is changed by MATLAB soon after.
limit/**numberOfTLEs**/	This is the number of TLEs to be returned. The number of TLEs should be the same as the number of NORAD CAT IDs provided. This ensures that only one TLE per CAT ID is returned and only the most recent TLE for a piece of debris is returned. In this case it will be 3.
format/tle/metadata/false	Finally the format of the data returned is set to be TLEs. The metadata of the request is not given.

5.4 Post Requests

Once the query URL has been created the next step is using the URL to obtain data. A human may simply type the query URL into a browser. For example typing the following URL into an Internet browser:

`https://www.space-track.org/basicspacedata/query/class/tle/format/tle/NORAD_CAT_ID/25544/orderby/EPOCH%20desc/limit/1`

Provides the result seen in figure 9. This is of course the most recent TLE for the ISS.

```
1 25544U 98067A   18298.51635846  .00001514  00000-0  30392-4 0  9998
2 25544  51.6406  89.2479 0003892 336.3122 134.3245 15.53861856138782
```

Figure 9: TLE for ISS from query URL

However space-track requires a username and password. A browser that humans use saves this information. Typing this URL in without the username and password saved will redirect to the front page of space-track. For MATLAB to use these query URLs and provied a username and password a POST request is needed.

A POST method is a way of requesting data by providing information that would be entered into HTML fields, such as a a username or password [22]. In effect the POST request is able to to go to a website, fill out and fields needed (such as username and password fields) and retrive the desired data. In MATLAB a post request looks like this:

```
function postOutput = examplePostRequest(username,password,timeOutVal)
    baseURL='https://www.space-track.org/'; % base URL
    logURL=[baseURL,'ajaxauth/login']; % login URL
    querySatcatURL=[baseURL,'basicspacedata/query/class/tle/format/', ...
        'tle/NORAD_CAT_ID/25544/orderby/EPOCH%20desc/limit/1']; % query URL
    post={'identity',username, 'password', password, 'query', ...
        querySatcatURL}; % create post request
    postOutput=urlread(logURL,'Post',post,'Timeout',timeOutVal); % runs ...
        and gets the output of the post request
end
```

The output of this post request is

```
>> postOutput = examplePostRequest('exampleUser','examplePass',30)

postOutput =

    '1 25544U 98067A   18298.51635846  .00001514  00000-0  30392-4 0  9998
     2 25544  51.6406  89.2479 0003892 336.3122 134.3245 15.53861856138782
     '

>>
```

Figure 10: Output of examplePostRequest.m

This POST request uses the MATLAB function urlread to send the username, password, query URL, and a timeout value. The time out value is the length of time in seconds the code will wait for a response from the server. It returns a char with a TLE for the ISS. This type of POST request is implemented in this thesis's code.

6 Code Overview

The code obtains orbital data for space debris depending on what variables the user has inputted, such as the launch year of the debris or how recent the data should be. First the code ascertains what data is already available. It checks to see if there is a .mat file for the SATCAT. This is simply a file that contains the catalog ids of the debris. If this file does not exist then the code has not been run for the current parameters, and the code will be run.

The code will also check to see if there is a TLE .mat file. If this file exists the code then checks when the file was created. Depending on user variables, the code may consider the information to be out of date, will rerun everything. However if the TLE .mat file is recent, then the code will load the current debris data into the workspace.

Assuming that the code detects that it has no data, or the data is out of date the code will work in the following steps

1. The desired NORAD satellite catalog ids will be collected with the get_SATCAT.m file. These ids are determined by user variables such as launch date. The code downloads them as a csv, strips things like quotation marks away, and sorts the ids in order.

   - if a SATCAT .mat file exists and is recent this step is skipped.

2. The code then begins to loop through the array of Ids. It enters an error catch to handle url time outs.

   - Within this loop the code then downloads the TLEs from the satellite ids. The TLEs are stored in a string. It does this in groups that have their size determined by the user. A large group size may risk url timeouts, but a small group size will result in a long run time.
   - The TLEs are then parsed, the TLEs are turned into usable orbital information, and saved as an array in MATLAB.

- When the loop has run through all the Ids, the array of data is saved as a .mat file wit the date of it's creation.

3. Now the array of debris data is ready. A copy is saved with a header listing what each collum represents. Another is sent to an orbit visualization file.

4. It should be noted that not all TLEs requested will be provided. So if the code requests TLEs of objects with a id from 1000 to 1100 and expects 100 TLEs, but some are classified, then duplicated TLEs less than 100 TLEs will be returned. The user now has a usable list of space debris and their orbital elements.

What now follows is a more in depth explanation of each file.

6.1 VarStore.m

This is a file where a few important variables are stored. It allows the user to only have to edit one file to change the code's operation.

6.2 UserPass.m

This file is where the username and password for space-track.org are stored. This is kept separate from the other code to ensure privacy of username and password.

6.3 MASTER_TLE.m

This file is the master file for the Two Line Element MATLAB files. Running it will run all of the associated MATLAB files. These MATLAB files take some time to run, so it may be convenient to alter the range of data operated on by MASTER_TLE.m in VarStore.m

6.4 get_SATCAT.m

Get_SATCAT.m is the MATLAB file that gets the satellite catalog numbers of all orbital debris launched after a given year and with the "RCS_SIZE" value equal to "SMALL". The first URL described previously is used here. The Launch Year can be set by the values given by the user in VarStore.m The default launch year is set to 1990. Note that an earlier launch year will provide more data, and thus it will take more time to process. This may cause a time out error. Should this happen the timeOutVal in VarStore.m should be adjusted to be longer.

Once the SATCAT csv file has been acquired the file then formats it. The quotation marks that are around every entry are removed. The debris that have already deorbited are also removed. Finally the debris is sorted by NORAD Catoluge Id, and saved as a .mat file. The post request in the get_SATCAT.m file was originally developed by Joao Encarnacao[23]

6.5 get_TLE_from_ID_Manager.m

The purpose of get_TLE_from_ID_Manager.m is to act as a handler for function get_TLE_from_NorID.m. The code first checks to see if there exists a folder to store the data in, if there is not a folder is created. Then the code begins to run through the list of NORAD Ids. It sends these Ids to get_TLE_from_NorID.m, receives the orbital element data from those Ids.

The code also contains an error handler. The most common problem with downloading large numbers of TLEs is a URL timeout may occur. Timeouts are easy for a human to deal with, they may be fixed by simply refreshing the web page. However for MATLAB, it is more difficult. The code deals with a URL timeout as follows:

First the code detects the specific error thrown for URL timeouts. This error is caught, so the code does not stop running. Then the code gets the current location in the Id array the time out occurred at, and restarts at that spot. The effect of this is the same as refreshing a webpage that is not loading.

Once all the data has been collected for the given Ids, it is saved as a .mat file with the time that it was created.

6.6 get_TLE_from_NorID.m

The purpose of get_TLE_from_NorID.m is to download the TLEs from NORAD given Ids, and to save the TLEs as useful orbital elements. This function may be divided into two parts: the web interface for TLEs and the TLE to orbital element section.

The first part of the code creates a URL from the current set of Ids. This url is then used to download the TLEs. The TLEs are saved as a string within MATLAB.

In the next part of the code a loop goes through the TLE string. The information from these TLEs is saved as a matrix. Earlier versions of this projects code first downloaded all TLEs as .txt files and then parsed through the .txt files to get the orbital elements. However that was unnecessarily slow as MATLAB had to open each .txt file and read it's contents as opposed to reading the contents of a string already in the workspace. By constantly saving the results as a .mat file through out the operation of the code, there is no threat of data loss.

From each TLE the inclination (i), RA of ascending node (Ω), eccentricity (e), argument of perigee (ω), mean anomaly, and mean motion (n) may be acquired directly. From these values the other orbital elements may be calculated.

Period of rev:

$$T = \frac{24 \times 60 \times 60}{n} \tag{7}$$

Semi-major axis:

$$a = \left(\left(\frac{T}{2\pi} \right)^2 \times \mu_{\text{earth}} \right)^{(1/3)} \tag{8}$$

Where μ_{earth} is the standard gravitational parameter of earth: 3.986×10^{14}. Semi-minor axis:

$$b = a \times \sqrt{1 - e^2} \tag{9}$$

Perigee and Apogee:

$$r_{\text{per}} = (1 - e) \times a \tag{10}$$

$$r_{\text{ap}} = (1 + e) \times a \tag{11}$$

The code to parse the elements from the TLE was based off of Brett Pantalone's parse TLE code[24], modified to handle multiple TLEs from within the MATLAB workspace.

7 Conclusion

The goal of this thesis was to create a program that can download the latest orbital elements needed to calculate the real time locations of space debris, given user inputted parameters. The information would be usable to someone who has taken a basic spaceflight mechanics class.

7.1 Results of Code

When the code is run the result is a matrix of orbital elements, as seen in figure 11.

	1	2	3	4	5	6	7	8	9	10	11
1	norad_cat_id	Epoch time	Inclination (deg)	RAAN (deg)	Eccentricity (deg)	Arg of perigee(deg)	Mean anomaly (de...	Mean motion (rev/...	Period of rev (s/rev)	Semi-major axis (...	Semi-minor axis (...
2	27007	18299.4397506	99.2226	100.4889	0.0013079	81.9395	278.3238	13.74495904	6285.94088557	7361608.44642	7361602.15002
3	27097	18299.3548568	97.5158	76.6664	0.0033083	227.6313	132.213	15.38575327	5615.58465704	6828455.8388	6828418.47048
4	27115	18298.8709667	97.6567	9.8344	0.0007545	280.6391	142.2202	15.6827731	5509.22974203	6741964.19073	6741962.27172
5	27119	18298.9671141	97.8301	296.438	0.003318	348.404	11.6422	15.11329322	5716.82152541	6910279.42017	6910241.382
6	27121	18299.3978791	97.2825	347.8566	0.0041776	6.6697	353.5083	15.15278367	5701.92262238	6898268.04239	6898207.84666
7	27123	18299.2143144	97.3291	142.3428	0.0045235	32.4822	327.9152	14.63932908	5901.90981621	7058637.81697	7058565.59949
8	27128	18299.3993871	97.5909	63.4017	0.00552	301.9181	57.6686	15.10697985	5719.21064686	6912204.53968	6912099.23006
9	27129	18299.4841821	97.9426	291.0433	0.0038129	215.9247	261.6164	14.89454517	5800.78136082	6977773.21494	6977722.4926
10	27132	18299.4645447	97.7506	277.0071	0.0045767	296.43	187.8394	15.07519444	5731.26936066	6921917.18921	6921844.69496
11	27133	18299.1685771	97.7414	17.7721	0.0046777	132.6129	227.9034	14.63390309	5904.098139	7060382.52042	7060305.27631
12	27136	18299.4549753	98.0312	274.4571	0.0041204	190.3024	290.87	14.8879565	5803.3484985	6979831.74033	6979772.48935
13	27142	18299.3458385	97.3846	70.8587	0.0289079	133.4967	229.0634	14.10117193	6127.15031268	7237105.03342	7234080.49764
14	27146	18298.9543694	98.1887	274.8654	0.005008	311.3073	114.221	14.7740379	5848.09654509	7015665.51439	7015577.53717

Figure 11: MATLAB matrix resulting from code execution

Compared to the data in figure 9 this is far easier for a person with basic orbital mechanics experience to use. A number of parameter sets have been provided for users to experiment with.

7.2 Future Work

Now that data can be obtained the next step is to look at possible orbits to place OSCAR. Due to OSCAR's limited delta V inclination and plane changes are out of the question. Thus OSCAR should be inserted into a orbit which shares a inclination and RAAN with the most debris. The number of debris sorted by inclination and RAAN are seen in figure 12.

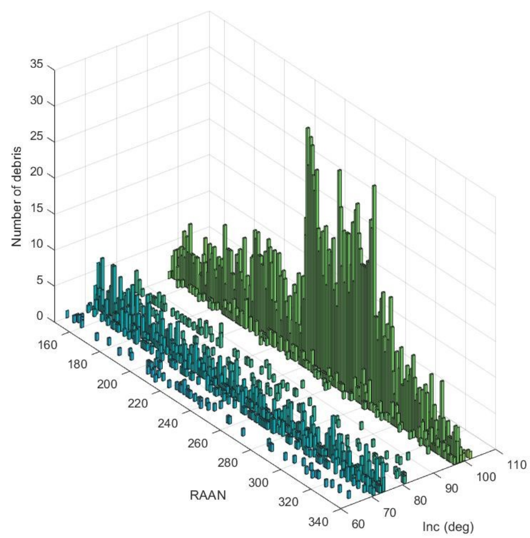


Figure 12: Number of debris by inclination and RAAN

The next step from this code would be to include propagation. The TLEs are for a given epoch, so to get the real time location prorogation from sgp4 is needed. Additionally the code should be developed to run autonomously on RPI computers, so there is a database of space debris that RPI controls.

References

[1] W. H. Tracy. https://spaceflight.nasa.gov/realdata/elements/, 2012. [Online; accessed 23-October-2018].

[2] Space-Track, "Basic description of the two line element (tle) format." https://www.space-track.org/documentation#/tle, 2013. [Online; accessed 31-July-2018].

[3] Space-Track, "Introduction to the space track api." https://www.space-track.org/documentation#api-Overview, 2013. [Online; accessed 31-July-2018].

[4] Usnrl, "Vanguard i celebrates 50 years in space." https://www.eurekalert.org/pub_releases/2008-03/nrl-vic031308.php, Mar 2008. [Online; accessed 16-October-2018].

[5] NASA, "Space debris impact on space shuttle window." http://www.orbitaldebris.jsc.nasa.gov/photogallery/gallarypage/sts7crack.jpg. [Online; accessed 19-October-2018].

[6] NASA, "Image of the entry hole created on space shuttle endeavour's radiator panel by the impact of unknown space debris.." http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080010742_2008009999.pdf. [Online; accessed 19-October-2018].

[7] H. D. CURTIS, *ORBITAL MECHANICS FOR ENGINEERING STUDENTS*. Elsevier Butterworth Heinemann, 2013.

[8] A. K. Kim Dismukes, "Definition of two-line element set coordinate system." https://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/SSOP_Help/tle_def.html, 2011. [Online; accessed 6-August-2018].

[9] I. Newton, "Letter from sir isaac newton to robert hooke." https://discover.hsp.org/Record/dc-9792/Description#tabnav. [Online; accessed 30-October-2018].

[10] "How many space debris objects are currently in orbit?." http://www.esa.int/Our_Activities/Space_Engineering_Technology/Clean_Space/How_many_space_debris_objects_are_currently_in_orbit, Jul 2014. [Online; accessed 16-October-2018].

[11] E. Kulu, "Nanosatellite and cubesat database." https://www.nanosats.eu. [Online; accessed 16-October-2018].

[12] P. McKee, "Cubesat dynamics and attitude control: Kane's method, lqr, and kalman filtering," Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, 2018.

[13] B. D. T. Kelso, "Frequently asked questions: Two-line element set format." http://celestrak.com/columns/v04n03/, 1998. [Online; accessed 31-July-2018].

[14] C. S. Tracking and O. R. (CASTOR), "B-star drag term." http://www.castor2.ca/03_Mechanics/03_TLE/B_Star.html, 2010. [Online; accessed 31-July-2018].

[15] NORAD, "Norad tracks santa." http://www.norad.mil/About-NORAD/NORAD-Tracks-Santa/. [Online; accessed 23-October-2018].

[16] D. T. Kelso, "Space surveillance." https://www.celestrak.com/columns/v04n01/, 1997. [Online; accessed 23-October-2018].

[17] D. A. Vallado and P. J. Cefola, "Two-line element sets–practice and use," in *63rd International Astronautical Congress, Naples, Italy*, 2012.

[18] U. S. A. Force, "Ground-based electro-optical deep space surveillance." https://www.afspc.af.mil/About-Us/Fact-Sheets/Article/249016/ground-based-electro-optical-deep-space-surveillance/, 2017. [Online; accessed 26-October-2018].

[19] Space-Track, "Space-track.org." https://www.space-track.org/, 2018. [Online; accessed 23-October-2018].

[20] Space-Track, "Space track legend." https://www.space-track.org/documentation#/legend, 2018. [Online; accessed 6-August-2018].

[21] Space-Track, "Space-track satellite catalog search." https://www.space-track.org/#/catalog, 2018. [Online; accessed 6-August-2018].

[22] J. F. R. Roy T. Fielding, "Hypertext transfer protocol (http/1.1): Semantics and content." https://tools.ietf.org/html/rfc7231#section-4.3.3. [Online; accessed 25-October-2018].

[23] J. Encarnacao, "matlab-sgp4." https://github.com/jgte/matlab-sgp4, 2015.

[24] B. Pantalone, "Read two-line element ephemeris files." https://www.mathworks.com/matlabcentral/fileexchange/56904-read-two-line-element-ephemeris-files, 2016.

Appendix 1 - MATLAB code

7.3 Master_TLE.m

```
%% Master Program
% By Philip Hoddinott
% This is the Master program to aquire TLEs and turn them into usable
% keplerian elements.
%% Setup
close all; clear all; clc; % clear workspace
%% get data
VarStore % run var store for stored variables, ugly but it works


%% check for existing data
strNam = ['mat_files/TLE_',num2str(launchYear),'.mat']; % get strNam
strNam_SC = ['mat_files/SATCAT_',num2str(launchYear),'.mat']; % get strNam

try % check the satcat tog;
    load(strNam_SC);
    tog_SC=0;

    try % try for TLE file
        load(strNam, 'tle_final','dateCreated'); % load in file
        cTime =datetime;
        if cTime>(dateCreated+calweeks(1)) % file out of date, rerun ...
            not SATCAT
            fprintf('File is one week out of date, auto running ...
                program\n');
            tog_All=1;
            tog_SC=1;
        else % file recent, no need to run any
            fprintf('The file %s, was created within the last ...
                week\n',strNam);
            tog_All=0;
            tog_SC=0;
        end
    catch ME % if tle not found or no date run all
        switch ME.identifier
            case 'MATLAB:load:couldNotReadFile'
                warning('TLE File does not exist, auto running program');
            case 'MATLAB:UndefinedFunction'
                warning('dateCreated not found, auto running program');
        end
        tog_All=1;
        tog_SC=0;
    end

catch ME
    switch ME.identifier
        case 'MATLAB:load:couldNotReadFile'
            warning('SATCAT File does not exist, auto running program');
```

```
            tog_SC=1;
            tog_All=1;
    end
end
% assign values
get_SATCAT_tog =tog_SC; % toggle for get_SATCAT 1 = run, 0 = don't run
get_Multiple_TLE_from_Id_tog =tog_All;
readTLE_txt_tog=tog_All;
check_TLE_Edit_TLE_tog=tog_All;



%% func get_SATCAT
% Function to get a .mat file from the SATCAT

if get_SATCAT_tog==1 % if the satcat file is out of date, run this
    get_SATCAT % get SATCAT, comment out if already run
    fprintf('get_SATCAT.m has finished running\n'); % output SATCAT has run
else % if the satcat file is recent then no need to run get_SATCAT
    load(strNam_SC,'all_TLE','decayEnd'); % load mat of SATCAT
    fprintf('get_SATCAT.m was not run\n'); % output SATCAT was not run
end

relDeb=str2num(char(all_TLE(2:decayEnd,2))); % get NORAD CAT ID

%% func get_Multiple_TLE_from_Id
% Function to get tle txt files from the norat_cat_ids in the SATCAT
VarStore % run var store for stored variables, ugly but it works

if get_Multiple_TLE_from_Id_tog==1
   get_TLE_from_ID_Manager
   fprintf('get_TLE_from_ID_Manager.m has finished running\n');
else
    fprintf('get_TLE_from_ID_Manager.m was not run\n');
end

%close all; clear all; % clear out everything
VarStore % run var store for stored variables, ugly but it works
strNam = ['mat_files/TLE_',num2str(launchYear),'.mat']; % get strNam

load(strNam, 'tle_final')

%tle_low=sortrows(tle_final(:,:),11);
%save('Orbits_MOD_1/tle_low2high.mat','tle_low');
%{
tle_high=sortrows(tle_final(:,:),11,'descend');
save('Orbits_MOD_1/tle_high2low.mat','tle_high');
% Note that these files could be made functions in MATLAB. For debuggin
% purposes they currently are not

tle_high=sortrows(tle_final(:,:),4,'descend');
save('Orbits_MOD_1/tle_RANN.mat','tle_high');

tle_INC=sortrows(tle_final(:,:),3,'descend');
```

```
save('Orbits_MOD_1/tle_INC.mat','tle_INC');
%}
tle_view=tle_final;
tle_view_temp=["norad_cat_id","Epoch time","Inclination (deg)","RAAN ...
    (deg)","Eccentricity (deg)","Arg of perigee(deg)","Mean anomaly ...
    (deg)","Mean motion (rev/day)","Period of rev (s/rev)","Semi-major ...
    axis (meter)","Semi-minor axis (meter)"];

tle_view_str = [tle_view_temp;tle_view]; % useful for looking at numbers

save('master_output','tle_view_str')
```

7.4 get_SATCAT.m

```
    %% get_SATCAT.m
% By Philip Hoddinott
% This code gets SATCAT from NORARD Query via post requests.
% This code was based off of the following code: ...
    https://github.com/jgte/matlab-sgp4/blob/master/get_tle.m

load('UserPass.mat') % load in username and password
baseURL='https://www.space-track.org/';
logURL=[baseURL,'ajaxauth/login'];
querySatcatURL=[baseURL,'basicspacedata/query/class/satcat/OBJECT_TYPE', ...
    '/debris/RCS_SIZE/small/LAUNCH_YEAR/>'];
querySatcatURL=[querySatcatURL,num2str(launchYear),'/orderby/DECAY ...
    asc/format/csv/metadata/false'];

URL='https://www.space-track.org/ajaxauth/login'; % URL for login

post={'identity',username, 'password',password,'query',querySatcatURL}; ...
    % Create Post Request

out=urlread(logURL,'Post',post,'Timeout',timeOutVal); % gets the output
outStr=convertCharsToStrings(out); % coverts output to string


newStr = strsplit(outStr,["\n"]); % split string by line break
for i=1:length(newStr) % split string by commas
    all_TLE(i,:) = strsplit(newStr(i),',');
end


[m,n] = size(all_TLE); % get size of matrix
%% Remove " marks
for i=1:m % remove the " marks
    for j=1:n
        all_TLE(i,j)=strip(all_TLE(i,j),'left','"');
        all_TLE(i,j)=strip(all_TLE(i,j),'right','"');
    end
end
```

```

%%  find where the last decayed item is and remove it
decay_loc=8;

for i=1:length(all_TLE(1,:)) % find the columm decay is located in, it ...
        should be in 8
    if all_TLE(1,i)=="DECAY"
        decay_loc = i;  % save decay loc
    end
end


decayEnd=m+10;
for i=2:length(newStr) % find the columm decay is located in, it should ...
        be in 8
    if all_TLE(i,decay_loc)≠[""] && decayEnd==m+10
        decayEnd=i−1; % get the last location where there is no yet decay
    end
end
all_TLE=all_TLE(1:decayEnd,:); % trim to only have debris that has not ...
        yet decayed


all_TLES=sortrows(all_TLE(2:end,:),2); % sort rows by NORAD ID
all_TLE=[all_TLE(1,:);all_TLES]; % save the sorted row by NORAD ID

%% save to a file
strNam = ['mat_files/SATCAT_',num2str(launchYear),'.mat'];
save(strNam,'all_TLE','decayEnd');
```

7.5 get_TLE_from_ID_Manager.m

```
%% get_TLE_from_ID_Manager
% By Philip Hoddinott
% This code handles getMultiple_TLE_from_ID and deals with url read ...
        time outs. urlread is an older function, however it works best with ...
        post requets.


load('UserPass.mat') % load in username and password
if exist(tle_folder)≠7 % if the folder does not exist it will make it
    mkdir(tle_folder)% tle_text_files % note this will give a warning ...
            if folder already exists
end
tle_final=[];
strNam = ['mat_files/TLE_',num2str(launchYear),'.mat']; % save the TLE ...
        as a .mat
save(strNam,'tle_final');
tleA = 1:tle_inc:decayEnd;
tleA=[tleA,decayEnd];

```

```
jStart =1; % starting value, will get reset if the connection times out
jEnd=length(tleA)-1; % end val

j_GMTFI=jStart;

while j_GMTFI≠jEnd % ensures that try catch keeps running untill it has ...
    gone all the way though
    fprintf('While loop, j = %d, jE = %d\n',j_GMTFI,jEnd);
    try
        for j_GMTFI = jStart:jEnd % try normal loop
            fprintf('Try for, j = %d, jE = %d\n',j_GMTFI,jEnd);
            load(strNam,'tle_final');
            tle_current=tle_final;
            get_TLE_from_NorID
            tle_final=[tle_current;tle_final];
            save(strNam,'tle_final'); % save the tle every loop through
        end

    catch ME      % time out catch
        fprintf('catch ME, j = %d, jE = %d\n',j_GMTFI,jEnd);
       switch ME.identifier
            case 'MATLAB:urlread:Timeout'
               warning('connection timed out at j = %d, trying ...
                   again',j_GMTFI);
               jStart=j_GMTFI;
       end
    end

end
load(strNam,'tle_final');
[C,ia,ic]=unique(tle_final(:,1),'rows'); % sort by row by norard cat id
tle_final_N = tle_final(ia,:);
dateCreated=datetime;
save(strNam,'tle_final','dateCreated');
```

7.6 get_TLE_from_NorID.m

```
%% get_TLE_from_NorID.m
% By Philip Hoddinott
% This code gets the TLEs from NORARD IDs and then extracts the ...
    keplerian elements from the TLEs. This code is orginally based off ...
    of this code: https://github.com/jgte/matlab-sgp4/blob/master/get_tle.m

%% Create string for post request
strTLE='/';
for i = tleA(j_GMTFI):tleA(j_GMTFI+1)-2
    strTLE=[strTLE, num2str(relDeb(i)),','];
end
if tleA(j_GMTFI+1)==decayEnd
    strTLE=[strTLE, ...
        num2str(relDeb(tleA(j_GMTFI+1)-1)),num2str(relDeb(tleA(j_GMTFI))),'/'];
```

```
else
    strTLE=[strTLE, num2str(relDeb(tleA(j_GMTFI+1)-1)),'/'];
end
fprintf('j = %d, tle %d to tle %d',j_GMTFI,tleA(j_GMTFI), tleA(j_GMTFI+1));
c=clock;
fprintf(' H = %d, Min = %d, Sec = %.1f\n',c(4),c(5),c(6));
%fprintf(strTLE);fprintf('\n'); %uncomment if you need to see %NORAD_CAT_ID
strTLE=[strTLE,'orderby/ORDINAL%20asc/limit/',num2str(tle_inc),...
    '/format/tle/metadata/false'];
baseURL='https://www.space-track.org/basicspacedata';
strTLE=[baseURL,'/query/class/tle_latest/NORAD_CAT_ID',strTLE];
URL='https://www.space-track.org/ajaxauth/login';

post={'identity',username,'password',password, 'query',strTLE};

out3TLE=urlread(URL,'Post',post,'Timeout',timeOutVal); % gets the output
outStr=convertCharsToStrings(out3TLE); % coverts output to string

j=1;
k=1;
C = splitlines(outStr);
while j<length(C)-1
    LineOne=convertStringsToChars(C(j,1)); % get first line
    LineTwo=convertStringsToChars(C(j+1,1)); % get second line

    satnum = str2num(LineOne(3:7)); % get sat num
    inc = str2num(LineTwo(9:16)); % get inc
    Omega = str2num(LineTwo(18:25)); % get Omega
    ecc = str2num(['.' LineTwo(27:33)]); % get ecc
    w = str2num(LineTwo(35:42));
    M = str2num(LineTwo(44:51));
    n = str2num(LineTwo(53:63));
    T = 86400/n;
    a = ((T/(2*pi))^2*398.6e12)^(1/3);
    b = a*sqrt(1-ecc^2);
    % store in array
    tle_stor(k,1)=satnum;
    tle_stor(k,2)=str2num(LineOne(19:32));
    tle_stor(k,3)=inc;
    tle_stor(k,4)=Omega;
    tle_stor(k,5)=ecc;
    tle_stor(k,6)=w;
    tle_stor(k,7)=M;
    tle_stor(k,8)=n;
    tle_stor(k,9)=T;
    tle_stor(k,10)=a;
    tle_stor(k,11)=b;

    j=j+2; % increment j
    k=k+1; % increment k
end
tle_final=tle_stor;
```