Rayne Milner

Spring 2016 Report
Trajectory Design

- Abstract
- Design
  - Trajectory
  - Matlab Interface
- Conclusion
- Appendix:
  - General Setup on New Computer
    - Installing STK
    - Configuring Engine Model
  - STK GUI
    - Resources:
    - Basic STK10 Setup for LCS Purposes
    - Useful Tools in STK GUI
      - Report and Graph Manager
      - Analysis Workbench
      - Results
      - STK Analyzer
      - Targeting Sequence
  - STK Programming Interface

Abstract

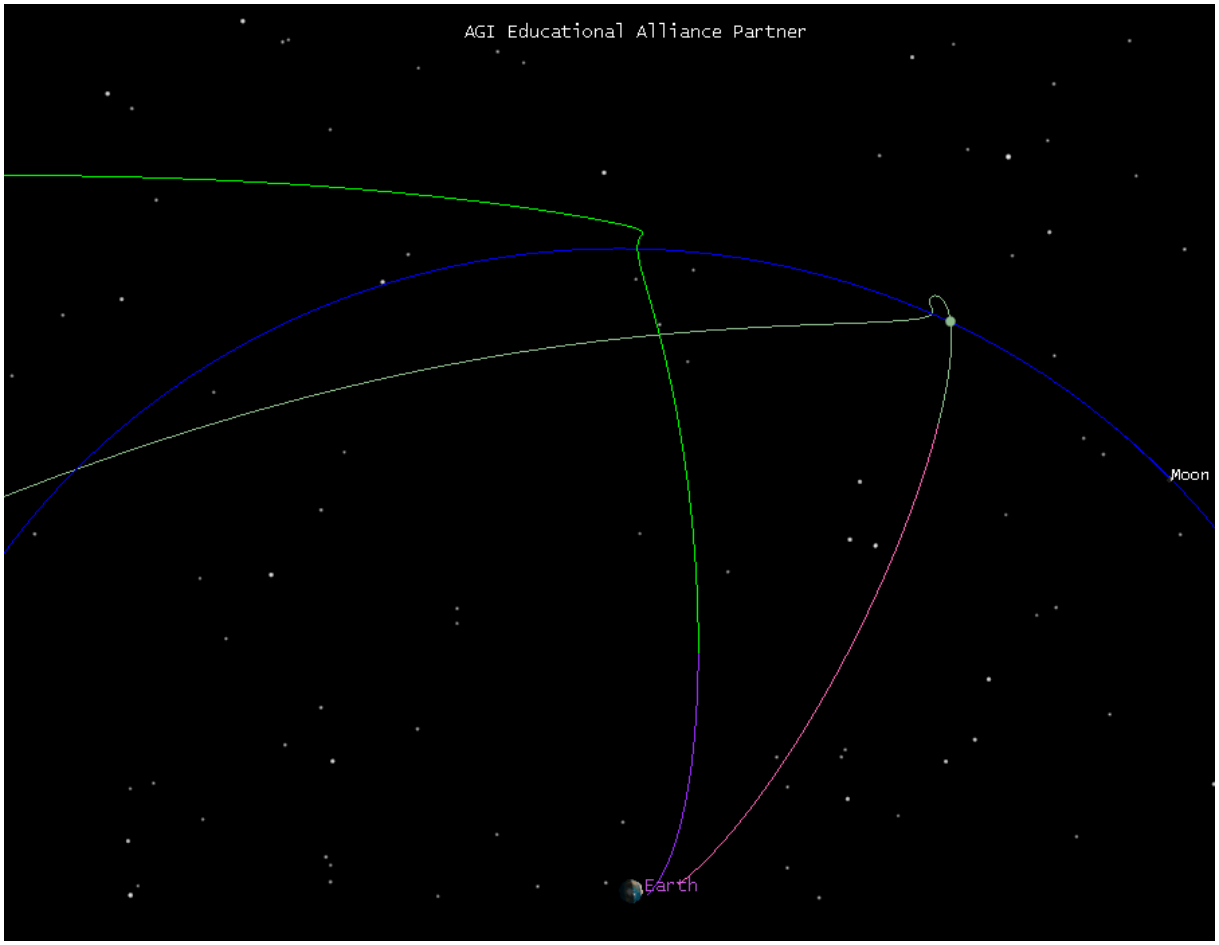
My research this semester was focused on two main projects, in addition two several odd tasks. I used STK's programming interface to connect STK and Matlab, using STK's integrators within the programming interface of Matlab to create a highly customizable tool. I used this tool to bring the attitude control, image processing and trajectory design sub-teams closer to integration. I also worked on analysing a new trajectory provided on May 2nd 2016 with a set launch date of October 2018.

Design

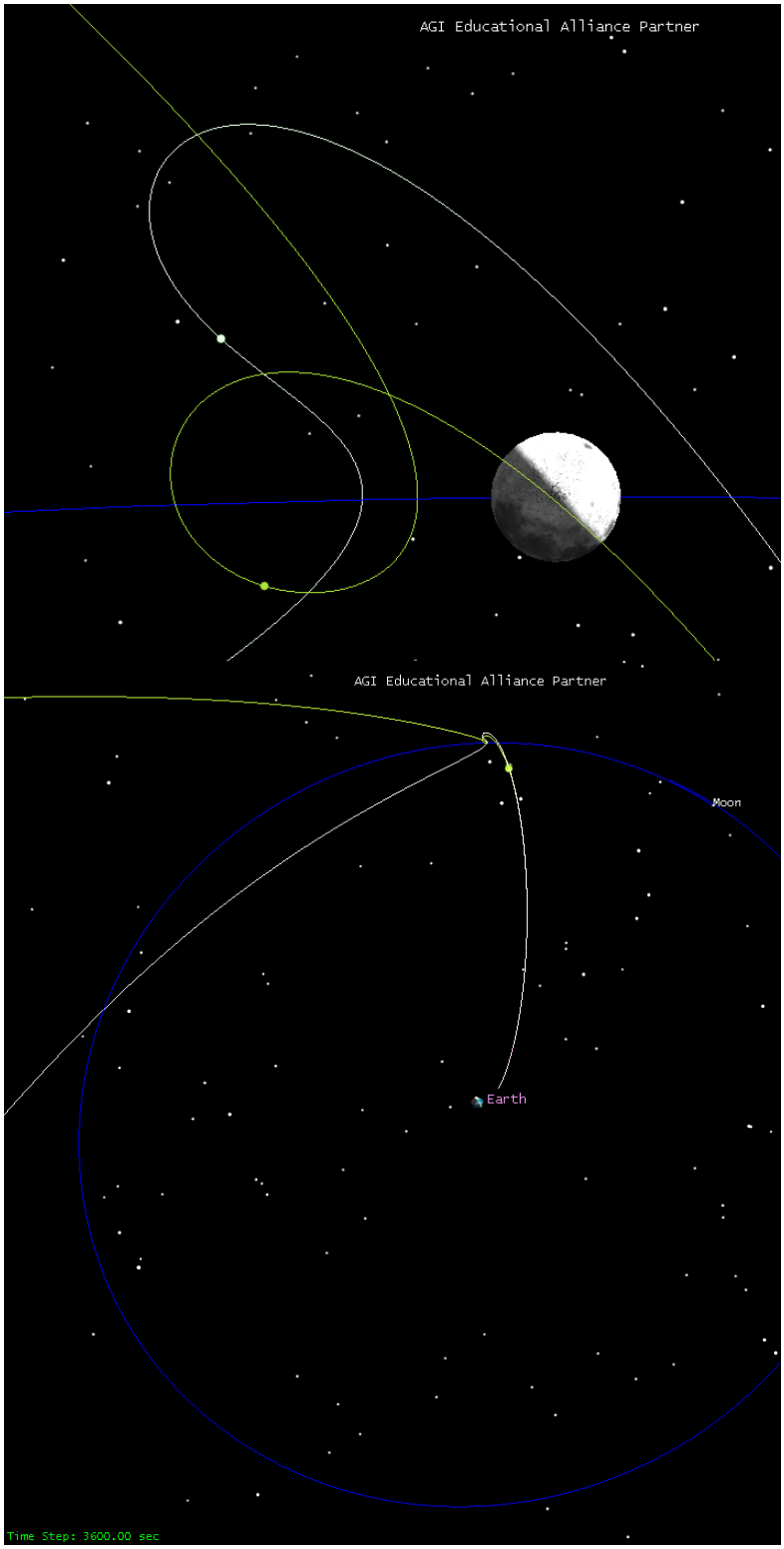
Trajectory

The most recent data on the initial disposal state of the craft was provided on May 2, 2016. The data is planned to become obsolete on September 30, 2016. These initial conditions are unique because they produce a trajectory which intersects the moon.

Pictured below are two trajectories propagated from initial conditions without applied delta-v. On the left is a "3600 km" offset trajectory, this trajectory does not intersect the moon and produces a



hyperbolic lunar swingby maneuver. This can be analyzed using a method similar to what we see in "Gravitational assist in celestial mechanics--a tutorial" by James Allen. On the right (in above image) we see a lunar impact trajectory. Note how the craft (blue dot) has arrived at the orbit of the moon far before the moon reaches the path of the trajectory. If we apply no thrust the craft will intersect the moon. However, with around 10m/s delta-v applied retrograde, the craft can instead loop around the moon. This maneuver is also a hyperbolic orbit and puts the craft in a similar state to that of the "3600km" orbit.

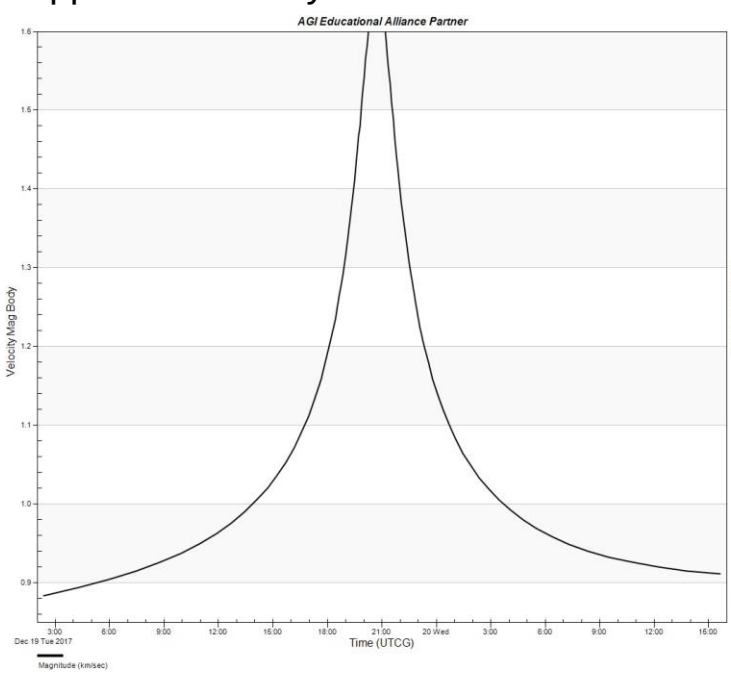


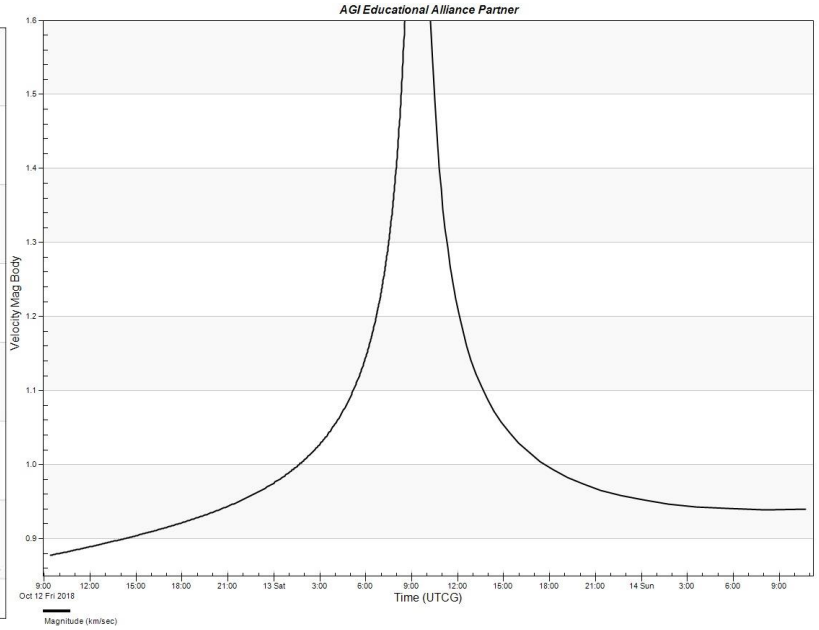
Here to the right we see the change between no thrust and 10m/s delta-v. The darker trajectory plots the craft that applied 10m/s delta-v of 56.3 hours. As you can see in the image, the craft no longer crashes into the moon and instead loops around it, eventually being sent into deep space. This means that this impact trajectory can be implemented in the same manner as the 3600km offset.

On the next page you can see in the two graphs of Approach Velocity wrt to Moon in the ICRF. The right graph is in October 2018 (the lunar impact trajectory) the left graph is in December 2017 (the 3600km offset trajectory). The two crafts' velocity behave very similarly over this maneuver. In the moon's frame both crafts approach and depart with the same speed. However, in the ICRF the departing velocity is greater than the approach velocity, as you can see in the asymmetry of the curves below. These two graphs depict a craft which approaches the moon with a velocity of around 0.88 km/s, flies very close to the moon and then leaves the vicinity of the moon. The 3600 km offset trajectory on the left ends with a velocity around 0.91km/s while the lunar impact trajectory ends with a velocity of around 0.95 km/sec. This discrepancy may inhibit a successful trajectory, seeing that later on, when the craft encounters the moon again, it cannot reduce velocity quick enough to capture.

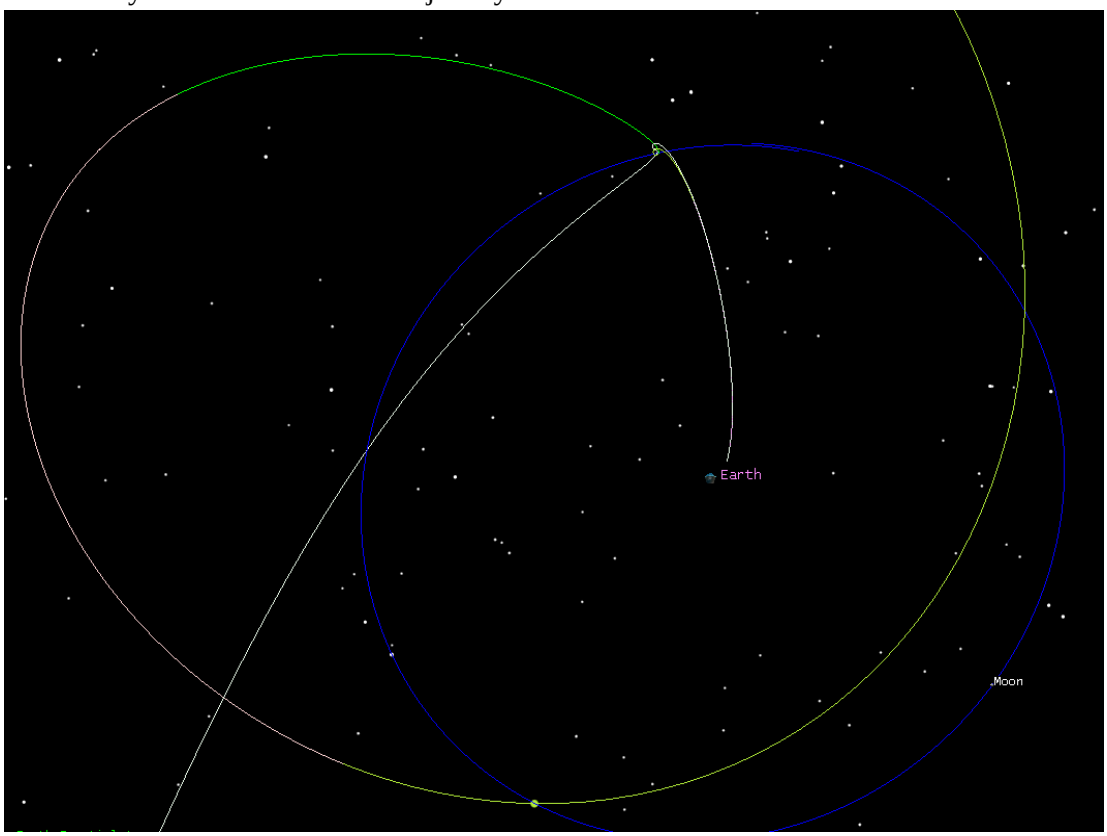
Here the trajectory is depicted with 12m/s delta-v applied over a period of 56.3hr between the earth and the moon and another

Approach Velocity of Satellite wrt Moon





152.6 m/s delta-v applied over 700hr while outside of the moon's orbit (maneuver shown in pink). This is very close to a successful trajectory.

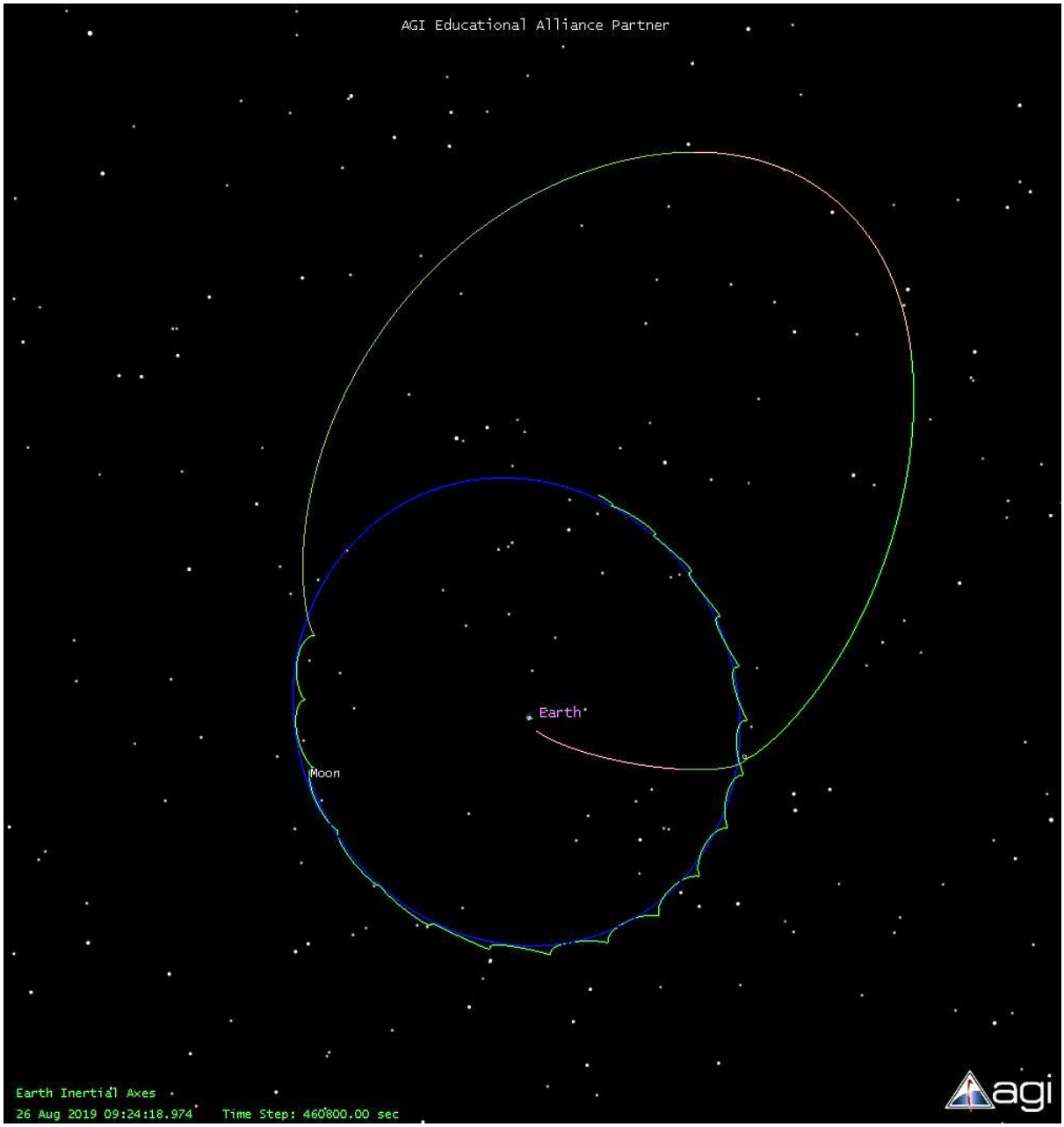


Now with some slight modifications of burn length. We can find a combination of maneuver one and two which allow us to get within the sphere of influence of the moon.

Segment	Segment type	Start Time	End Time	Stopping condition	Duration	Delta-v applied
MCC	Burn	7 Oct 2018 12:53:10.614 UTCG	9 Oct 2018 04:59:10.614 UTCG	Duration	144360 sec	0.01053 km/sec
Prop to Moon	Prop	9 Oct 2018 04:59:10.614 UTCG	13 Oct 2018 09:23:34.782 UTCG	Periapsis	2250000 sec	
Prop to MCC2	Prop	13 Oct 2018 09:23:34.782 UTCG	31 Oct 2018 09:23:34.782 UTCG	Duration	1553956.8 sec	
MCC2	Burn	31 Oct 2018 09:23:34.782 UTCG	26 Nov 2018 10:23:34.782 UTCG	Duration	2.25e+006 sec	0.1350 km/sec
Prop to Moon	Prop	26 Nov 2018 10:23:34.782 UTCG	20 Dec 2018 19:29:09.221 UTCG	Periapsis	2106339.2 sec	
Total						0.14553 km/sec

This total is only for the period before lunar capture. The final maneuver (Prop to Moon) ends at the point of periselene. The altitude of periapsis at this point is 316 km and eccentricity of 1.04. Using STK's built in numerical optimizer, I found 1.04 to be the lowest eccentricity possible for the trajectory set up in this way. This eccentricity is unviable because our engine does not have the power required to capture. More work must be done to determine a trajectory which is viable for capture based on these initial conditions.

Although this trajectory is not viable for our engine, you can still see below that it can lead to lunar capture. When a 150m/s impulsive maneuver is inserted at the point of periapsis, the craft captures around the moon with a periapsis and apoapsis within the required range of (10,000km to 300km). Some possible issues: the new swingby maneuver imparts too much velocity to the craft near the moons orbit (where a 2- body approximation is relevant). The new swingby maneuver has the trajectory rotated in the plane of the moon's orbit, this affects timing and solar perturbation effects.



The other work I completed this semester is focused on creating a more universal tool for solving these problems in the STK programming interface.

Matlab Interface

This semester, with the help of Ryan Elandt, I designed a Matlab script which can interface STK. This Matlab code uses STK's numerical integrators, but allows a greater freedom. The code can be found in the Box under ElectrolysisPropulsion\Orbit_Analysis\STK\Matlab_STK_Interface. The code has two main functions at this time. It can propagate a satellite while imparting delta-v and can create state vectors at every iterative step. It can create a point at an arbitrary position with respect to the craft, point the STK display out from the craft toward the point, and save a image of this view.

1. When the .m file runs, it first opens an instance of STK 10 (only two instances of STK can be opened at once).

2. It creates a satellite and specifies the initial state and time in J2000, Keplerian coordinates.
3. The code interfaces with STK to create data providers for Cartesian position and velocity in J2000 coordinates.
4. The annotations are removed from STK's view (3D window)
5. A point is created in space with respect to the satellite's body, that point is modified to spin around the craft in the y-z plane (x is velocity) at an angular speed of 63.123 rad/sec
6. Within a for loop set to iterate every second, an impulsive delta v can be applied to the craft and a screenshot of STK's view can be saved
7. The state of the craft is displayed in cartesian coordinates at the beginning of the "mission", end of the "mission" and every step of iteration.

Conclusion

The ability to take snapshots of STK's view and then apply delta-v has wide applications for this project. Using this code we could combine the work of Hunter Adams, Brian Wang and me to fully integrate the subsystems.

The lunar impact trajectory provided by NASA as the interim trajectory for 2016 is problematic, entering into a low perigee near parabolic orbit is possible, but the craft cannot deliver delta-v quickly enough to reduce apogee to acceptable levels. Other trajectories may be possible, but STK's built in numerical optimizer cannot find such a solution.

The work left uncompleted is as follows:

1. The Matlab STK interface must be combined with Brian Wang's circle finder and Hunter Adam's attitude filter. The images received from STK must be analyzed for celestial bodies. The state vector data in conjunction with information from the circle finder must be run through the attitude filter, where the position and velocity produced by the filter can be compared against STK's recorded data.
2. More work must be done to solve the trajectory problem. It is unclear whether the best course of action is to continue working in the STK GUI, knowing that a new trajectory is pending and more manual GUI work will be required, or to develop further the Matlab interface which may result in a more general solution.

Appendix:

General Setup on New Computer

Installing STK

Navigate to https://www.agi.com/ and download the newest version of STK available. With this you can use a few of the propagators in STK. This is useful if you want to exclusively use Matlab to command STK, however it is not useful if you need to use the STK GUI.

To gain access to STK Astrogator (the most powerful propagator) and use the STK GUI, you will need to obtain a STK Pro license. Kyle Doyle and Lorraine Weis are responsible right now for

getting this from AGI. Contact them and give them the following information: 1. STK version 2. Operating System 3. Host ID 4. Registration ID.
The Host ID and Reg. ID can be found in the application "License Manager" which should download with STK

Additionally, you should contact support@agi.com **with a non-edu email address** and ask them to provide you with a copy of STK Analyzer. This is a powerful tool that allows you to run Monte-Carlo like simulations. They will create a login ID so that you can access the file transfer web interface at https://files.agi.com

Configuring Engine Model

If you plan on using STK Astrogator/GUI you should configure the engine model. I have written a script in VBscript which plugs into STK's engine model and allows more control over how an individual maneuver in Astrogator produces thrust and ISP. The plugin consists of three key files, all of which can be found in the Cornell Box under ElectrolysisPropulsion\Orbit_Analysis\STK\WaterElectrolysisScript.

1. *.VBS file*: This contains the actual script. Here you can modify the script to change thrust ISP and mass flow rate calculations.
2. *WCS component*: This must be registered with the .net framework, once per computer. See the "Plugins.txt" file in the folder for instructions on how to do this. Note that you may have to disable the computer's firewall and/or gain administrator privileges for this registration to work.
3. *XML file*: this must be put into the folder C:\Program Files\AGI\STK 10\Plugins. If you change the names of any of the files or create any new engine models you will have to follow this process for every model, carefully changing the names wherever present in each of the three files.
4. **Important Note:** If you plan on using this engine model in any astrogator maneuver sequence, you must modify the standard propagator within the WSC browser. A more detailed description of this process will follow, but quickly, you must edit a propagator (like HPOP or Cislunar) so that it only uses a fixed step size of sixty seconds. Otherwise the engine model will not work properly; STK's clock and the Script's clock will missalign.

STK GUI

Resources:

AGI's vimeo channel: https://vimeo.com/analyticalgraphics

Useful tool for any level of experience, the STK DIY series is particularly helpful but they also have a series of mini lectures (2-5minutes) that explain how to do specific things.
Astrogators guild: http://astrogatorsguild.com/

A blog that has a ton of information on how to use astrogator including the astrogator training scenarios. These can guide you through the learning of astrogator once you've learned the basics of STK
Agi's resources: http://www.agi.com/resources/

Not great. Code samples and example scenarios. most of the example scenarios that are pertinent to this lab are downloaded in the STK folder. The forum has some trouble shooting help
Professor Selva: http://www.cornell.edu/search/people.cfm?netid=ds925

Professor Selva just arrived at cornell in 2013, he seems interested in talking to students and he is well versed in STK

Low Experience Starting Point:

http://www.colorado.edu/engineering/ASEN/asen3200/labs/proTutorial.pdf

Some STK experience Low astrogator experience:

Watch some STK DIY videos, specifically the ones on trajectory planning with astrogator (https://vimeo.com/68702797)

https://vimeo.com/76960075

Basic STK10 Setup for LCS Purposes

-Open STK and create a new scenario.

-Create a new folder with the same name as your scenario and set that as the location.

-Make sure the start and stop time spans the duration of the mission (I would assume 6 months from launch date).

-Now we have reached the STK GUI. First let's make sure the licences and engine models are working:

1. *Insert new satellite*: Select "Insert" a new object, "Satellite", "define properties", "Insert"… What appears is the properties window for the new object (note on the left hand sidebar has a new satellite object).
2. *Change propagator astrogator*: Under "Basic", "Orbit" you should see an option for "Propagator" and in the drop down box, "Astrogator" should be available, select it.
3. *Insert new finite maneuver*: Now we are looking at the maneuver control sequence. Click the "new" button (it looks like the new document button in Microsoft office, hover over it and you should see "insert segment after"), click "Maneuver", "OK". You have just inserted a new maneuver segment into the sequence. The default maneuver type is "Impulsive", we want it to be "Finite", so change that now. When you change the maneuver type, a new tab appears labeled "Propagator".
4. *Change engine model*: First, switch tabs in the maneuver segment to "Engine". Make sure "Engine Model" is selected and click the "..." button. You should see at least six engine models, the one we want is "water electrolysis engine" (if this is not available, try restarting, make sure the engine model is properly registered in the .net framework). Click "water electrolysis engine", "OK".
5. *Change Maneuver propagator*: Near the new button there is a button with a blue symbol on it named "Component Browser" (sometimes it is hidden but if you adjust the horizontal width of the mission control sequence it should become visible). The component browser should open. Here you can view all of the individual components that makes up STK. Select "Show", "Astrogator Components" then click the folder labeled "Propagators", you should see six propagators. Make two duplicates, one of cislunar and one of Earth HPOP Default. Edit the new propagators, navigate to the "Numerical Integrators" tab and select "use fixed step", verify that the initial step is sixty seconds and click "OK". Repeat for the new duplicate. Now navigate back to the mission control sequence, left click the satellite and select "Properties". Click the maneuver you just created and click the "Propagate" tab, click the "..." button next to propagator and select "copy of Earth HPOP Default v10" (or whatever you named the new component).
6. *Verify success*: Finally, let us verify that the engine is delivering thrust. First propagate the maneuver using the green arrow button labeled "run entire mission control sequence" then

right click the maneuver and select summary. The summary provides a breakdown of the maneuver, including delta-v applied if the engine model is running properly the delta-v applied should be on the order of 10-100 m/s.

-Modifying initial state to reflect the spacecraft parameters is the next step. The initial state can be found in the mission control sequence. There are several steps to verify correctness:

1. *Elements*: the "Coordinate System" should be the same as the provided initial state vector. The default setting is "Earth Inertial" but it is more likely you will be working with "J2000" or "TOD" (true of date). Also the "Orbit Epoch" may be given in "UTCG" (Gregorian UTC, looks like 22 May 2016 16:00:00.000 UTCG) and sometimes given in "JDate" (Julian Date 2457531.16666667 JDate) **Note:** that the orbit epoch must correspond to the coordinate system you are using, A date in TOD is not the same as a date in J2000.
2. *Spacecraft Parameters*: All to do here is make sure that the "Dry Mass" and every instance of "Area" is accurate
3. *Fuel Tank*: This is unique to the water electrolysis engine. Make sure the values are as follows: "Tank Pressure:" 1e+006 Pa, "Tank Volume:" (ask structures), "Tank Temperature:" 290K, "Fuel Density:" 1000 kg/m^3, "Fuel Mass" (ask structures), "Maximum Fuel Mass:" (ask structures)
4. *User Variables*: no action required.

-Now you just have to insert the appropriate sequences into the mission control sequence, including maneuver segments, propagate segments, and targeting sequences. A further description of the targeting sequence is in the next section.

Useful Tools in STK GUI

Report and Graph Manager

The report and graph manager can be found on the "Data Providers" toolbar, you can also specify the object directly by right clicking a satellite (or ground station) and selecting "Report & Graph Manager". In this tool you can produce reports and graphs on an unbelievable range of parameters. Click "Create New Report" or "Create New Graph", then "Properties". The new window that opens contains all of the possible data that you can pull from STK about the mission. This includes delta-v, vector operations, ephemeris data, range rates, astrogator values, etc. Once you add these to an axis or column, you specify the time period and produce a graph or chart.

Analysis Workbench

Here we can create and modify vector geometry, time, and calculations. This is useful for creating vectors between celestial bodies, or creating points in space for calculation.

Results

This is available in Astrogator in the mission control sequence. To use it, select any of the segments (like the maneuver) and click results. This opens up a window where you can select from the various parameters that STK can calculate. Do this for a maneuver segment and select "Vx", "Vy" and "Vz".

STK Analyzer

This is a very powerful tool, it is a plugin, meaning that you must download it separately to STK. If you have a STK Pro license, you can contact support@agi.com to get a download link. Analyser has the ability to iterate values and then collect data from the mission. Let's take a look at an example:

1. *Open analyzer*: This can be done from the Analyzer toolbar, or by right clicking the satellite. There are several available options on the analyzer toolbar, but we just want the one labeled "Analyzer…"
2. *Add Variable*s: First, make sure that the correct satellite is selected, now select some STK Property Variables. For example, select Initial State, "X", "Y", "Z". The three inputs should move over to the right hand side of the window under "Analyzer Variables" under "Inputs". Now we need some outputs, first you must use the results function to select some parameters, select "Vx", "Vy", "Vz" for the maneuver after "Initial State". Then, in Analyzer open the folders until you reach the maneuver, then results. Add the results "Vx","Vy","Vz" to the Analyzer Variables column. They should appear under "Outputs".
3. *Probabilistic Analysis Tool/Design Variables:* After specifying your variables, open the probabilistic analysis tool. You should have a few variables available on the left column. You can drag them from the left to the right. If you drag an input variable (green) you are prompted to select a distribution for that variable. This determines how analyzer iterates over the variable. Chose "Normal" with a standard deviation of ten percent. Do this for "X", "Y" and "Z". Now Drag the output variables (red) over to the "responses" section. Select the number of runs you would like to see. And click "Run..."
4. *Data analysis*: Once you hit run, a new window appears labeled "Data Explorer" here you can look at the data that the monte carlo simulation is creating. The table feature just lists out the data points, you can copy it into other programs like Excel and MatLab. There is a histogram feature and a scatter matrix feature. The scatter matrix feature produces plots which are a little confusing at first but can be very revealing; it plots all the variables against each other to create a huge matrix of plots.
5. *Exporting Data*: The export data feature can be useful, but often I just choose to copy and paste the data points. There is a big issue with exporting epoch's from STK, the best way around this is to change all epoch units into Jdate which is a lot simpler for calculations.

Targeting Sequence

The targeting sequence in STK's Astrogator is also a powerful tool for iterating over parameters, however it also has the ability to change parameters in reaction to results of the iteration. Using this tool is difficult and requires experience for success. I would strongly suggest watching this video and following along. I will go over some of the basics below

1. *Inserting initial state segment*: First you must have a few sequences in your mission control sequence. To start, insert an Initial state with the following parameters.

Julian Date (UTC)	2458102.98578642
Epoch of True of Date Frame	2458102.91143017
Semi Major Axis (km)	219204.3507
Eccentricity	0.968827176

Inclination (deg) TOD Frame	28.38115326
RAAN (deg) TOD Frame	58.26318437
Argument of Perigee (deg)	49.58505528
True Anomaly (deg)	67.43138182

2. *Inserting segments*: Now insert three segments after initial state. First the targeting sequence, then nested underneath insert a maneuver and a propagate segment. In the maneuver segment, first ensure that the "Maneuver Type" is "Finite" go to "Attitude" and change "Attitude Control" to "Thrust Vector" make sure "Spherical" is selected. You should see a little bull's eye next to "Azimuth", click it. This bull's eye is a selector for the targeter, when you check the bull's eye you are indicating that you would like this parameter to be available for iteration within the targeting sequence. If the bulls eye is not available, that means you need to make sure the segment is nested under the targeter. Now move to the "Engine" tab, change the engine model to water electrolysis engine. Check the bull's eye next to thrust efficiency. Now move to the "Propagator" tab. Change the "Propagator" to "copy of HPOP/Cislunar..." (the propagator that works with the water electrolysis engine). Finally, check the bull's eye by "Trip Value". Now let's move onto the Propagator. Add a new stopping condition using the "New…" button, make the condition periapsis and insure that the "Central Body" for that stopping condition is "Moon". Delete the other (duration) stopping condition. While still on the propagate segment click "Results"(or right click on propagator and click "Results") and open the "MultiBody" folder. Move both "BdotT" and "BdotR" over to the right. Page seven this paper has a good explanation of Bplane Geometry.
3. *Setting up targeter*: navigate to the targeting segment and double click the default "Differential Corrector". A new window should open on a "Variables" tab. There should be a few "Control Parameters" and two "Equality Constraints" (if not, something is not nested correctly). Make sure "FiniteMnvr.Spherical.Azimuth", "...ThrustEfficiency", "BdotR", and "BdotT" all have the "Use" column checked. Under equality constraints set the "Desired Value" of BDotT to 7500km and BDotR to 0km for a good starting point.
4. *Experimentation*: What we have set up in this small tutorial is a basic differential corrector. The targeter will iterate through the control parameters until the equality constraints are met or the targeting fails. Every targeting scenario is different so you will have to experiment with the settings to get the targeter to converge. Here are some settings to experiment with:
    a. *"Perturbation"*: this is a setting which can be applied to control parameters, this affects how the targeter finds solutions. It is the amount by which the targeter perturbs around the current solution to determine which "direction" the correct solution might be. This is an important parameter to experiment with.
    b. *"Max Step"*: This is similar to perturbation but this is a limit on maximum distance away from the original solution. The targeter will not exceed this amount when perturbing a solution on a single iteration.
    c. *"Tolerance"*: This is a setting which can be applied to equality constraints. This is the plus or minus allowance on error for the constraint. If a targeting sequence is not converging, you can try increasing the tolerance and slowly decreasing it until you determine where the constraint is limited or you find a solution.

d. *"Maximum Iterations"*: Under the "Convergence" tab. self explanatory, increasing this will increase the time for the targeter to determine that there is no solution. Increasing this will also increase the chance that the targeter does find a solution.
e. *"Advanced" settings*: this tab contains a lot of useful settings. Here you can change the root finding algorithm (I've never had any different results, but might be useful). The "Line Search" function is useful when you need to iterate more systematically. Homotopy makes the targeter only focus on one control parameters at once. Clearing corrections before each run is useful when the motion is chaotic, or any other time you want to start the iteration from the same place every time.

STK Programming Interface

The STK GUI is powerful tool for quick data collection, trajectory visualization, and much more. However, it can be slow and constrictive. Fortunately, there is a way to utilize all the integrators and detailed models of spaceflight mechanics contained within STK while still having the freedom of Matlab programming. This solution is the STK programming interface and documentation can be found here (http://help.agi.com/resources/help/online/stkdevkit/10.1.3/index.html). I will walk you through the basics of interfacing STK through Matlab and explain some pitfalls.

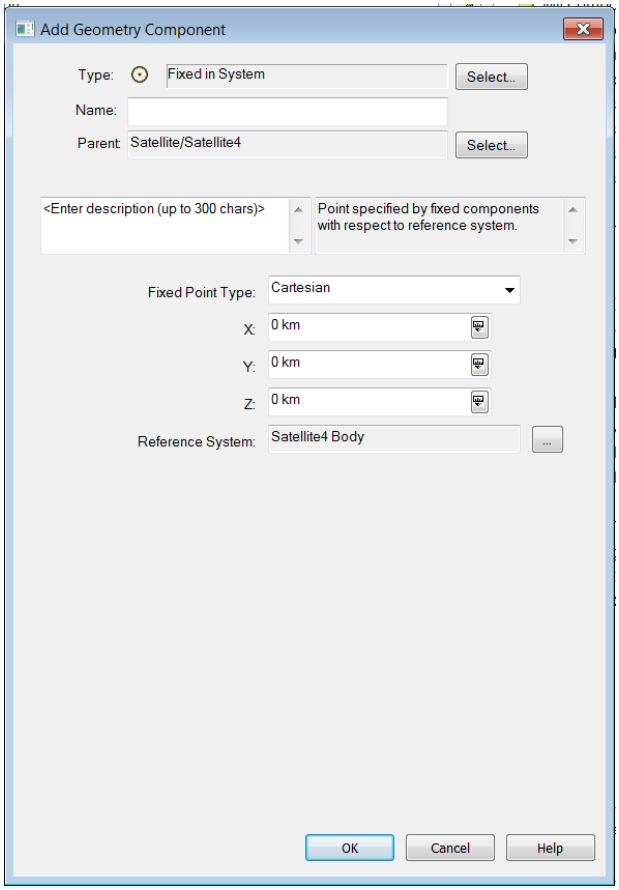
The link above takes us to AGI's documentation on the STK Matlab interface. I suggest immediately looking at the alphabetical listing of connect commands. Navigate to this using the left hand sidebar on the webpage, the listing I am referring to can be found under STK_Programming_Interface\Library_reference\Connect_Command_Library\Alphabetical_Listing here you can find every possible action available in STK's GUI, but in a poorly documented series of Matlab commands.

Looking at the code I have written in the box under ElectrolysisPropulsion\Orbit_Analysis\STK\Matlab_STK_Interface we can see example of how to use these commands.

1. First thing in the code we define the "app" this simply opens STK and is the first step to commanding STK within Matlab. App.delete is a method which deletes the instance of STK
2. Next we define the "root" and "scenario". These are objects within STK, they have methods which we can call.
3. *Calling Methods*: most commands sent to STK will be methods of root. root.ExecuteCommand() calls most functions that we will use, including "SetState",

"Annotation", "VectorTool", "ViewFromTo". One command which is a method of scenario is "SetTimePeriod"

4. *Defining Commands*: each command (listed in the alphabetical listing) has a specific syntax, the command must be perfectly constructed or you will receive STK's only error return "error on line …". Each command has pretty detailed documentation describing syntax, but the example are sometimes lacking so you must learn to experiment. The best method for understanding this syntax is to look into the GUI for the GUI window that this command is mimicking. For example the "VectorTool Modify Point" command. In the STK window we see this window. In Matlab we have the corresponding command: ***['VectorTool * Satellite/Satellite4 Modify Point PosXPoint "Fixed in System" Cartesian ',thx,thy,' ',thz,' "Satellite/Satellite4 Body"']***



   a. Vector Tool is the GUI tool which we are using
   b. * indicates "use default" this is specifying the scenario which we should already have loaded in the Matlab script
   c. Satellite/Satellite4 is the "Parent"
   d. Modify is more unique to Matlab and specifies that we are modifying the point (but the window looks the same whether we are modifying or creating a new point)
   e. "Fixed in System" refers to the "Type"
   f. Cartesian refers to the "Fixed point type"
   g. Thx, thy and thz are the values intended for "X:", "Y:" and "Z:"
   h. Satellite/Satellite4 Body refers to the "Reference System"

The order of commands in Matlab is not implicit from the GUI so you will have to check the documentation to figure out the specifics. Additionally, having the right white space is important for STK so I would reference my code for example if you are having trouble.

Finally, there are some bugs in the STK programming interface. Ryan developed methods for working around them. There is problem with memory leaks and STK crashing, this is solved with an "if" statement and a "try catch" statement and the end of the code. If you are having trouble with bugs, pay close attention to the statements at the beginning and end of the code.