



# Project NextStep

Final Report Appendix

AAE 450 – Spring 2021



## Table of Contents

Mission Design .....	1
I. Launch Windows References .....	2
II. Lunar Transfer Preliminary Analysis & Crewed Trajectory Simulation References .....	2
III. Translunar Trajectory Analysis .....	3
IV. Preliminary Communication Satellite Trajectory Design.....	12
V. GMAT and MATLAB Code .....	16
VI. Lunar Ascent and Descent.....	44
VII. References for Lunar Ascent and Descent .....	50
VIII. References for Lunar Terrain Profile.....	51
IX. Appendix for Lunar and Earth Rendezvous and Initial Free-Return Analysis.....	52
X. Appendix for Lunar Transfer Preliminary Analysis & Crewed Trajectory Simulation .....	106
XI. Appendix for Orbital Stationkeeping for Propellant Depots .....	133
XII. References and Code for Free Return.....	136
Propulsion Appendix .....	142
I. Launch Vehicles .....	144
A. Launch Vehicles Considered - Preliminary Analysis .....	144
1. Light-Lift Launch Vehicles.....	144
2. Medium-Lift Launch Vehicles .....	146
3. Heavy-Lift Launch Vehicles .....	147
4. Super-Heavy-Lift Launch Vehicles .....	149
5. Conclusions .....	151
B. Launch Vehicles Considered - In Depth Analysis .....	152
1. Falcon 9 .....	152
2. Falcon Heavy.....	157
3. Starship.....	164
4. Space Launch System (SLS) .....	167
C. Estimation of $\Delta V$ Splits Between First and Second Stages: Falcon 9 and Falcon Heavy	173
D. Starship Propellant Estimation.....	175
E. Concerns of Falcon 9 or Falcon Heavy Upper Stage Re-ignition in Deep Space .....	180
1. Zero-g Propellant Slosh.....	180
2. In Space Boil-off of Cryogenic Propellants .....	180
II. Launch Mission Tool.....	186
1. Motivation and Purpose .....	186

2. Version 1 .....	186
3. Version 2 .....	189
4. Version 3 .....	190
5. Final Version Updates .....	191
6. Launch Vehicle Function Version 1.....	194
7. Launch Vehicle Function Version 4 (Final Version) .....	199
8. Launch Vehicle Script (Final Version) .....	212
III. Upper Stage Engine Capability Comparison For Use on a Space Tug .....	214
1. Space Tug Sizing Code .....	220
I. Landing Vehicles Appendix .....	243
A. Ice Calculator.....	243
B. Apollo Cargo Sizing .....	245
Propellant Depot Appendix.....	252
I. Starship and Superheavy Performance .....	252
II. Decision on Lunar Station .....	255
III. Propellant Mass Bookkeeping Along the Mission .....	260
IV. Thermal and Power Systems .....	263
V. Boiloff .....	270
VI. Attitude Dynamics/Control .....	273
VII. Attitude Control System (Lorin).....	277
VIII. Costs .....	281
IX. Lunar Mining Background and Need .....	281
X. Code.....	293
A. Propellant Bookkeeping and Refueling Optimization: .....	293
B. Stationkeeping Sizing Calculations .....	297
C. Code for calculating launch numbers for fueling depot.....	300
D. Solar Array Sizing.....	303
E. Code for ACS system in LEO orbit: (Jaxon).....	304
F. Gravity Gradient Stabilization Analysis Code.....	307
G. Tank Sizing Code.....	309
H. Wrapper for finding available delta v for Starship (Trevor Pfiel) .....	312
I. Determining the delta v available from the Super Heavy Booster) .....	313
J. Determining the propellant and delta v requirements to propulsively land the Super Heavy Booster.....	313

K.	Determining the delta v available from Starship .....	315
L.	Determining the propellant and delta v requirements to propulsively land Starship.....	315
M.	Estimating boiloff and tracking propellant mass in propellant depots (Trevor Pfiel)..	316
N.	Settling Burn Mass Requirement .....	323
	Communications .....	330
I.	Data Rate Example Calculation.....	331
II.	Detailed Link Budget Calculation.....	332
III.	Risk Matrix.....	348
IV.	MATLAB Code.....	349
	Vehicles: SatellitesAppendix .....	355
I.	Sizing.....	357
II.	CAD.....	363
III.	Budget and Costs.....	365
A.	CHELL Satellites .....	365
IV.	Mission Design.....	366
A.	FISTO Orbit Design.....	366
B.	FISTO Orbit Stationkeeping Maneuver Strategy .....	367
C.	FISTO Orbit Stationkeeping FreeFlyer Script.....	371
D.	Molniya Orbit.....	376
E.	80°: 4/2/1 Walker Constellation .....	381
F.	Alternative Deployment Strategy Using Single Launch.....	385
V.	Risk and Fault Analysis.....	392
VI.	Subsystems .....	394
A.	Payload.....	394
1.	Camera .....	394
B.	Attitude Control System .....	397
1.	Code for Sizing ACS Systems in Lunar Orbit .....	399
2.	Equations of Motion.....	403
3.	Attitude Control Scripts Mathematical Breakdown .....	407
4.	Nutation and Precession Angle Visualization Considerations .....	408
5.	Attitude Dynamics MATLAB Script .....	419
6.	Attempted PD-Compensator Implementation.....	428
7.	Attempted CubeSat Implementation .....	429
C.	Power and Thermal .....	430

D. Propulsion .....	432
1. FISTO Propulsion System Sizing .....	432
2. propMassGraph.m .....	433
3. propSizeFunc.m.....	435
4. Decision Matrix for Propellant Type .....	437
5. Observation Satellite Propulsion System Analysis .....	438
6. Analysis of Various Electric Propulsion Systems.....	438
7. Code for Chemical Propulsion System Mass and Sizing.....	441
8. Code for Electric Propulsion System Propellant Mass Analysis .....	442
VII. Potential and Future Work.....	444
A. Lunar Positioning System .....	444
References.....	447
Systems: Habitats.....	451
I. Matlab Scripts.....	452
A. Structures .....	452
B. EVA Support.....	491
II. Tables .....	495
1. Structures.....	495
2. Habitat Design Decision.....	495
3. Cost Estimates .....	498
4. Risk and Fault.....	499
III. Design Process and Theory .....	507
A. Airlock Research.....	507
B. Initial Airlock Design Concept .....	508
1. Airlock Design Process .....	511
2. Ingress/Egress Airlock Design (Alternate 1) .....	513
3. Rover Docking .....	515
4. EVA Suit Resource Storage .....	518
C. Alternative Design Study for a Rigid Lunar Habitat .....	521
1. Alternative Design Studies for Regolith Radiation Shielding .....	523
2. Moment of Inertia for Regolith Support Structure I-Beams .....	525
3. Alternate Colonist Makeup Research.....	527
4. Alternate design concepts.....	529
5. EVA suit selection and details .....	530

D.	Inflatable Habitat Material Selection .....	535
E.	Lunar Hotel -Conceptual Drawings .....	544
	Systems - Life Support Appendices.....	549
I.	Appendix A: Medical Equipment Appendix .....	550
1.	Why a Storage Solution of a Medicine Cabinet is Developed.....	550
II.	Appendix B: ATCS Appendix.....	551
III.	Appendix C: ECLSS Layout Simulation Code .....	567
	References.....	579
	Systems: Research Facilities.....	584
I.	Regolith Composition.....	585
	Systems: Agriculture and Food.....	590
I.	Determining Caloric Intake .....	591
II.	Preliminary Food and Diet Research.....	592
1.	Breads and their Components .....	592
2.	Fruits and their Components .....	593
3.	Vegetables and their Components.....	594
4.	Foods to avoid and their Components.....	595
5.	NASA Nutritional Biochemistry Research .....	596
6.	Protein Alternatives.....	597
a.	Alternative Protein #1: Crickets .....	598
b.	Alternative Protein #2: Lab-grown Meats.....	599
III.	Structure Choice .....	600
IV.	Mass/Volume Calculations for the Structure.....	602
1.	Pump Sizing/Power Calculations .....	602
2.	Light Selection and Power Calculations .....	603
V.	Control of the Systems .....	604
2.	Watering.....	604
3.	Humidity.....	606
4.	Temperature .....	608
VI.	Power Calculation Program.....	609
	References.....	612
	Vehicles: Rovers .....	614
I.	Code.....	615
A.	Trajectory Control Parameter Specification .....	615

B.	Solar Array Sizing.....	616
C.	Edge Detection Algorithm .....	618
D.	Clearing Rover Power and Sizing.....	619
E.	Scouting rover range, endurance and power estimation (Ajay Chandra) .....	621
II.	Initial Design Logic/Concepts .....	622
A.	Clearing Rover .....	622
1.	Initial Design Ideas.....	622
2.	Initial Design Concepts .....	625
B.	Collection Rover .....	626
1.	Initial Design Pictures .....	626
2.	Collection Process.....	628
C.	Scouting Rover.....	630
D.	Mining Rover .....	632
1.	Initial Design and Concepts .....	632
E.	General Rover Design.....	633
F.	External Robotic Appendage (ERA) (Ajay Chandra) .....	634
III.	Design Decisions .....	635
A.	Motor Choice .....	635
B.	Sizing .....	635
C.	Sensors .....	635
D.	Tools .....	636
IV.	Rover Risk/Fault Analysis.....	642
A.	Mining Rover .....	642
B.	Clearing Rover.....	644
C.	Construction Rover .....	645
D.	Scouting Rover.....	646
E.	Collection Rover .....	647
V.	CAD Models.....	648
CAD	667	
I.	Food Warmer Briefcase - Entire System.....	668
A.	Food Warmer Briefcase - Heating Element.....	670
B.	Food Warmer Briefcase - Briefcase & Hinge.....	672
C.	Medicine Cabinet - Entire System .....	674
D.	Medicine Cabinet - Drawers & Dividers .....	676

2.	Fig. 16 Wireframe view of the medicine cabinet with 93 dividers in 3 of its 6 drawers.	677
E.	Medicine Cabinet - Housing .....	677
3.	Fig. 18 Isometric view of the medicine cabinet without drawers and wheels. This helps to show the inner mechanisms that allow for the drawers to slide into place. ....	679
4.	Oxygen Regeneration System.....	679
F.	Oxygen Regeneration System - Adsorbent Bed .....	681
G.	Oxygen Regeneration System - Desiccant Bed .....	683
H.	Oxygen Regeneration System - Payload Drawer and Latch.....	684
I.	Oxygen Regeneration System - International Standard Payload Rack.....	685
J.	Full Habitat Design.....	687
K.	Collins Dining & Food Preparation Module.....	688
L.	Crew Module .....	690
M.	Regolith Shielding Cage.....	691
N.	Airlock Module.....	692
O.	Inflatable Habitat Design .....	693
P.	Hard Port Design.....	694
Q.	Hardpoint Supports .....	695
R.	Modular Construction Rover .....	696
S.	Falcon Heavy .....	698
T.	Starship Payload Module .....	704
U.	Deployable Solar Panels .....	705
V.	Kitchen Counter .....	706
W.	Refrigerator .....	707
X.	Computer Case.....	708
Y.	Computer Keyboard.....	709
Z.	Personal Tablet.....	710
AA.	ATCS.....	711
BB.	Communication Satellite .....	714
CC.	Interior Pieces of Living Space - Microwave and Oven/Stovetop.....	721
DD.	Assistant Robot .....	722
EE.	Heavyload Robotic Arm.....	723
FF.	Surgical Mechanical Arm.....	724
GG.	Potable Water Dispenser .....	725

HH.	Human Research Lab Rack (HRL) .....	727
II.	Mass Spectrometer (Research Facilities).....	728
JJ.	Gym Module .....	729
KK.	Kennedy Space Center Launch Pad (KSC).....	733
LL.	Air Filter Drawing.....	734
MM.	Universal Waste Management System.....	735
NN.	Modular Robotic Arm .....	736
OO.	Lunar Lander Cargo Variant .....	739
PP.	Bosch Reactor .....	742
QQ.	Blood Centrifuge.....	745
RR.	Crew Sinks .....	746
SS.	Office Monitors .....	747
TT.	Adjustable TV Mount.....	748
UU.	Observation Satellite FISTO .....	751
VV.	Wheatley Ground Station.....	752
WW.	Repeater Module.....	753
XX.	Communications Relay Tower.....	754
YY.	TREES Agriculture Aeroponics Tower .....	755
ZZ.	TREES Agriculture Lighting system .....	756
AAA.	TREES Seedling Tray .....	757
BBB.	Standard Modular Rack .....	758
CCC.	The Lunar Surface .....	759
DDD.	The Earth-Moon System.....	760
EEE.	Clearing Rover.....	762
FFF.	Regolith Plow .....	764
GGG.	Rocker Bogie .....	765
HHH.	Autonomous Rover Wheel (Solid) .....	766
III.	Large Rover Wheel (Spring Mesh) .....	767
JJJ.	HDA (Heavy-Duty Arm) Module .....	768
KKK.	Construction Rover with HDA (Heavy-Duty Arm) Module.....	769
LLL.	Crane Module Swivel Base and Modular Flatbed Mount .....	770
MMM.	Scouting Rover (STAR).....	771
NNN.	Mining Rover (MARGY) .....	773

OOO. Construction Rover with Crane Module..... 776

## **Mission Design**

Cody Martin, Nikhil Gavini, Daniel Gochenaur, Dale Williams, Alex Maietta, Matthew Popplewell, Eric Gooderham, Mohit Pathak, Stephen Grabowski, Chad Blackwell, Youssef Noureddine, Jarod Utz

*Purdue University, West Lafayette, IN, 47906, United States*

## I. Launch Windows References

- [1] Espenak, F., “Moon at Perigee and Apogee: 2001 to 2100,” *Planetary Ephemeris Data*, [online publication],  
URL: <http://astropixels.com/ephemeris/moon/moonperap2001.html> [retrieved 8 Feb. 2021].
- [2] Espenak, F., NASA GSFC, “Eclipses and The Moon’s Orbit,”[online publication], URL:  
<https://eclipse.gsfc.nasa.gov/SEhelp/moonorbit.html> [retrieved 10 Feb. 2021].
- [3] Hoffmann, T. “Moon Calculator,” [online publication], URL: <https://www.mooncalc.org/#-10,50,2/2021.09.10/15:29/1/3>  
[retrieved 11 Feb. 2021].
- [4] NASA, “Launch Windows for Lunar Landing,” *A Journey to the Moon Through Texas*, [online video], URL:  
[https://old.texasarchive.org/a\\_journey\\_to\\_the\\_moon/portfolio-item/launch-windows-for-lunar-landing/](https://old.texasarchive.org/a_journey_to_the_moon/portfolio-item/launch-windows-for-lunar-landing/)  
[retrieved 11 Feb. 2021].
- [5] Wheeler, R., “Apollo Lunar Landing Launch Window: The Controlling Factors and Constraints,” *Apollo Flight Journal* [online journal], Essay 8, URL:  
<https://history.nasa.gov/afj/launchwindow/lw1.html> [retrieved 8 Feb. 2021].

## II. Lunar Transfer Preliminary Analysis & Crewed Trajectory Simulation References

- [1] Bate, R. R., Mueller, D. D., White, J. E., Saylor, W. W., *Fundamental of Astrodynamics*, 2<sup>nd</sup> ed., Dover Publications, INC., Mineola, New York, 2020, pp. 273.
- [2] Wheeler, R., “Apollo Launch Window: The Controlling Factors and Constraints,” *Apollo Flight Journal* [online journal], URL:  
<https://history.nasa.gov/afj/launchwindow/lw1.html#TLI> [retrieved 04 April 2021].

### III. Translunar Trajectory Analysis

#### Ephemeris to GMAT Script

The ‘ephemerisToGMAT’ MATLAB script is called in order to link to a user-filled excel sheet with 10 years of ephemeris data including the range, right ascension, and declination of the target body at arrival. The excel file name is set up such that we can understand that the Lambert Problem is being solved, the trajectory in this case is from the Earth to the Moon, and the transfer angle is 175 degrees with a desired TOF of 4.00 days. The outputs of this script are shown in the excel sheet, with the six Keplerian orbital parameters being calculated alongside the departure  $\Delta V$  impulsive burn magnitude needed to escape the orbit from the starting body.

##### 1. Set up connection to Excel File Sheet that has Ephemeris Data

```
fileName = 'Lambert_Numbers_Earth_to_Moon_175_400.xlsx'; % Excel File Name in Directory
for year = 1:10
    yearNumber = year;
    if (year == 2 || year == 6) % Year that you want data from
        numLaunches = 14;
        cellRange = 'C3:E16';
    else
        numLaunches = 13;
        cellRange = 'C3:E15';
    end
end
```

##### 2. Read in Ephemeris Data

```
ephemeris = xlsread(fileName, yearNumber, cellRange);
```

##### 3. Create Tables to be filled in by GMAT inputs and Additional Characteristics

```
characteristic_table = cell2table(cell(numLaunches, 6));
characteristic_table_headers = {'a', 'e', 'i', 'RAAN', 'AoP', 'TA_dep'};
characteristic_table.Properties.VariableNames = characteristic_table_headers;

additional_table = cell2table(cell(numLaunches, 2));
```

```
additional_table_headers = {'Type', 'DeltaV'};

additional_table.Properties.VariableNames = additional_table_headers;
```

#### 4. Fill in Table with GMAT Inputs

```
for index = 1:numLaunches
    % Input Data from Ephemeris
    luna_range = ephemeris(index, 1);
    alpha_arr = ephemeris(index, 2);
    delta_arr = ephemeris(index, 3);

    % Get Orbit Characteristics
    [a, e, inclination, RAAN, AoP, theta_star_dep, TRANSFER_TYPE, dv1] = GMAT_inputs(luna_range, alpha_arr, delta_arr);

    % Put Orbit Characteristics into Tables
    characteristic_table{index, 1} = num2cell(a);
    characteristic_table{index, 2} = num2cell(e);
    characteristic_table{index, 3} = num2cell(inclination);
    characteristic_table{index, 4} = num2cell(RAAN);
    characteristic_table{index, 5} = num2cell(AoP);
    characteristic_table{index, 6} = num2cell(theta_star_dep);
    additional_table{index, 1} = cellstr(TRANSFER_TYPE);
    additional_table{index, 2} = num2cell(dv1);

end
```

#### 5. Write Data into Excel Cells in Original File

```
xlswrite(fileName, table2cell(characteristic_table), yearNumber, 'G3');

xlswrite(fileName, table2cell(additional_table), yearNumber, 'P3');
```

#### 6. Let users know of progress

```
fprintf('Year %d is written. \n', yearNumber);
```

```
end
```

```
clear;
```

```
clc;

fprintf('Excel Sheet has been populated with GMAT inputs. \n');
```

## GMAT\_inputs Function

The GMAT\_inputs function is called for each launch in the main ‘For Loop’ in ‘ephemerisToGMAT’. The script takes in the three ephemeris inputs for a specific launch date and returns the six Keplerian orbital parameters to be put into GMAT.

### 1. Function Call

```
function [a, e, inclination, RAAN, AoP, theta_star_dep, TRANSFER_TYPE, dv1] = GMAT_inputs(luna_range, alpha_arr, delta_arr)
```

### 2. Inputs

```
transfer_angle = 175;
desired_TOF_days = 4;
```

### 3. Get Inclination, Argument of Latitude (theta), Right Ascension (Upper Omega)

```
inclination = 28.5; % Minimum 28.5 deg (Cape Canaveral)
AoL = asind(sind(delta_arr)/sind(inclination));

% Solve for RAAN
% fprintf('%4f * c_RAAN - %4f * s_RAAN = %4f \n', cosd(AoL), cosd(inclination)*sind(AoL), cosd(alpha_arr)*cosd(delta_arr));
% fprintf('%4f * c_RAAN + %4f * s_RAAN = %4f \n', cosd(inclination)*sind(AoL), cosd(AoL), sind(alpha_arr)*cosd(delta_arr));

% Solve for RAAN
RAAN_consts = [cosd(AoL), -cosd(inclination)*sind(AoL); cosd(inclination)*sind(AoL), cosd(AoL)];
sol = [cosd(alpha_arr)*cosd(delta_arr); sind(alpha_arr)*cosd(delta_arr)];

RAAN_const_matrix = [RAAN_consts sol];
RAAN_const_solns = rref(RAAN_const_matrix);
```

```

RAAN1           =           acosd(RAAN_const_solns(1,3));
RAAN2           =           360-acosd(RAAN_const_solns(1,3));
RAAN3           =           asind(RAAN_const_solns(2,3));
RAAN4           =           180           -           asind(RAAN_const_solns(2,3));

RAANmatrix      =           [RAAN1           RAAN2;           RAAN3           RAAN4];
if             (floor(abs(RAAN1)))           ==           floor(abs(RAAN3)));
  RAAN           =           RAAN1;
elseif          (floor(abs(RAAN1)))           ==           floor(abs(RAAN4)));
  RAAN           =           RAAN1;
elseif          (floor(abs(RAAN2)))           ==           floor(abs(RAAN3)));
  RAAN           =           RAAN2;
elseif          (floor(abs(RAAN2)))           ==           floor(abs(RAAN4)));
  RAAN           =           RAAN2;
end

```

#### 4. Known Constants and Parameters

mu_earth	=	398600;	%	Gravitation	Parameter	of	Earth
mu_moon	=	4902.8005821478;	%	Gravitation	Parameter	of	Moon
TA	=	transfer_angle;	%	Transfer	angle	(degrees)	
TOF	=	desired_TOF_days*3600*24;	%	Time	of	flight	(seconds)
if		TA	<				180
TRANSFER_NUM	=	1;	%	Transfer	Type	Number	(1 or 2)
else		TRANSFER_NUM	=				2;
end							

#### 5. Departure Quantities and Space Triangle Geometry

%	Departure	Radii			
r_dep	=	6378+600;	%	Earth	Radius
r_arr	=				luna_range;
if	TRANSFER_NUM	==			1
c	=				$\sqrt{r_{dep}^2 + r_{arr}^2 - 2 * r_{dep} * r_{arr} * \cos(TA)}$
else					

c	=	sqrt(r_dep^2+r_arr^2-2*r_dep*r_arr*cosd(360-TA));		
<b>end</b>				
%	Compute	Space	Triangle	Semi-Perimeter
s = (r_arr+r_dep+c)/2;				

## 6. Parabolic TOF and Transfer Type

a_min	=	s/2;	
TOF_min	=	0;	
%	Determine	Parabolic	TOF
if	TRANSFER_NUM	==	1
TOF_par	=	1/3*sqrt(2/mu_earth)*(s^(3/2)-(s-c)^(3/2));	
else			
TOF_par	=	1/3*sqrt(2/mu_earth)*(s^(3/2)+(s-c)^(3/2));	
end			

## 7. Determine Transfer Geometry (ELLIPTIC or HYPERBOLIC)

if	TOF	>	TOF_par
TRANSFER_GEOMETRY	=	"ELLIPTIC";	
alpha_min	=	2*asin(sqrt(s/(2*a_min)));	
beta_min	=	2*asin(sqrt((s-c)/(2*a_min)));	
TOF_min	=	sqrt(a_min^3/mu_earth)*((alpha_min-beta_min)-(sin(alpha_min)-sin(beta_min)));	
if			TOF>TOF_min
TRANSFER_TYPE	=	num2str(TRANSFER_NUM)+"B";	
else			
TRANSFER_TYPE	=	num2str(TRANSFER_NUM)+"A";	
end			
else			"HYPERBOLIC";
TRANSFER_GEOMETRY	=		
TRANSFER_TYPE	=	num2str(TRANSFER_NUM)+"H";	
end			

## 8. Select Appropriate alpha/beta fcns

```

alpha_fcn          =          @(a)          0;
beta_fcn          =          @(b)          0;
lambert_fcn       =          @(a)          0;

switch           TRANSFER_TYPE
case             "1A"
    alpha_fcn      =          @(a)      2*asin(sqrt(s/(2*a)));
    beta_fcn        =          @(a)      2*asin(sqrt((s-c)/(2*a)));
case             "2B"
    alpha_fcn      =          @(a)      2*pi-2*asin(sqrt(s/(2*a)));
    beta_fcn        =          @(a)      -2*asin(sqrt((s-c)/(2*a)));
case             "1B"
    alpha_fcn      =          @(a)      2*pi          -      2*asin(sqrt(s/(2*a)));
    beta_fcn        =          @(a)      2*asin(sqrt((s-c)/(2*a)));
case             "2A"
    alpha_fcn      =          @(a)      2*asin(sqrt(s/(2*a)));
    beta_fcn        =          @(a)      -2*asin(sqrt((s-c)/(2*a)));
case             "1H"
    alpha_fcn      =          @(a)      2*asinh(sqrt(s/(2*abs(a))));
    beta_fcn        =          @(a)      2*asinh(sqrt((s-c)/(2*abs(a))));
case             "2H"
    alpha_fcn      =          @(a)      2*asinh(sqrt(s/(2*abs(a))));
    beta_fcn        =          @(a)      -2*asinh(sqrt((s-c)/(2*abs(a))));
end

```

## 9. Select Appropriate lambert function

```

guess          =          a_min;
switch           TRANSFER_GEOMETRY
case             "ELLIPTIC"
    lambert_fcn   =          @(a)      sqrt(a^3/mu_earth)*((alpha_fcn(a)-sin(alpha_fcn(a))-...
                           (beta_fcn(a)-sin(beta_fcn(a)))-TOF;
    guess          =          a_min;
case             "HYPERBOLIC"
    lambert_fcn   =          @(a)      sqrt(abs(a)^3/mu_earth)*((sinh(alpha_fcn(a))-alpha_fcn(a))-...
                           (sinh(beta_fcn(a))-beta_fcn(a)))-TOF;

```

```

guess = -(r_dep+r_arr)/2;
end

```

## 10. Solve Lambert

```

a = fsolve(lambert_fcn, guess);
alpha_val = alpha_fcn(a);
beta_val = beta_fcn(a);
if abs(lambert_fcn(a)) > 1e-4
    fprintf("ERROR! BAD A");
    5/0;
end

```

## 11. Compute p

### Elliptic Case

```

if TRANSFER_GEOMETRY == "ELLIPTIC"
    p1 = 4*a*(s-r_dep)*(s-r_arr)/c^2*(sin((alpha_val+beta_val)/2))^2;
    p2 = 4*a*(s-r_dep)*(s-r_arr)/c^2*(sin((alpha_val-beta_val)/2))^2;

    if TRANSFER_TYPE == "1A" || TRANSFER_TYPE == "2B"
        p = max(p1, p2);
    else
        p = min(p1,p2);
    end

% Hyperbolic Case
else
    p1 = 4*abs(a)*(s-r_dep)*(s-r_arr)/c^2*(sinh((alpha_val+beta_val)/2))^2;
    p2 = 4*abs(a)*(s-r_dep)*(s-r_arr)/c^2*(sinh((alpha_val-beta_val)/2))^2;
    if TRANSFER_TYPE == "1H"
        p = max(p1, p2);
    else
        p = min(p1,p2);
    end
end

```

## General Orbit Characteristics

```

if TRANSFER_GEOMETRY == "ELLIPTIC"
    e = sqrt(1-p/a);
else
    e = sqrt(1+p/abs(a));
end

spec_E = -mu_earth/(2*a);
v_dep = sqrt(2*(spec_E+mu_earth/r_dep));
v_arr = sqrt(2*(spec_E+mu_earth/r_arr));

[theta_star_dep, theta_star_arr] = get_theta_star(p, e, r_dep, r_arr, TA);
AoP = AoL - theta_star_arr;
if (AoP < 0)
    AoP = AoP + 360;
end
gamma_dep = sign(theta_star_dep)*acosd(sqrt(mu_earth*p)/(r_dep*v_dep));
gamma_arr = sign(theta_star_arr)*acosd(sqrt(mu_earth*p)/(r_arr*v_arr));

rp = a*(1-e);
ra = a*(1+e);

```

## Print Useful Characteristics

```

% fprintf('\nDesired Transfer Angle = %.2f deg \n', transfer_angle);
% fprintf('Desired TOF = %.2f days \n', desired_TOF_days);
% fprintf('Earth-Luna Lambert Type: Type %s Transfer \n\n', TRANSFER_TYPE);
%
% fprintf('GMAT Keplerian Inputs:\n');
% fprintf('a = %.8f km \n', a);
% fprintf('e = %.8f \n', e);
% fprintf('Inclination = %.8f deg \n', inclination);
% fprintf('RAAN (omega) = %.8f deg \n', RAAN);
% fprintf('Argument of Periapsis (w) = %.8f deg \n', AoP);
% fprintf('TA_dep = %.8f deg \n', theta_star_dep);
% fprintf('TA_arr = %.8f deg \n', theta_star_arr);
% fprintf('Argument of Latitude (theta) = %.8f deg \n', AoL);

```

## Delta-Vs Using F and G

```

v_0_xyz           =      sqrt(mu_earth/r_dep)*[0          1          0];
v_f_xyz           =      sqrt(mu_earth/r_arr)*[cosd(TA+90)   sind(TA+90)  0];

r_dep_xyz        =      r_dep * [1          0          0];
r_arr_xyz        =      r_arr * [cosd(TA)    sind(TA)    0];

[f, g]           =      fg(r_dep, r_arr, TA, p, mu_earth);

v_dep_xyz        =      (r_arr_xyz - r_dep_xyz*f)/g;

[df, dg]         =      dfg(r_dep_xyz, v_dep_xyz, TA, p, mu_earth);

v_arr_xyz        =      df*r_dep_xyz + dg*v_dep_xyz;

dv1_vnb          =      rt2vnb(0)*rth2xyz(0, 0, 0)*(v_dep_xyz-v_0_xyz);
dv1               =      norm(dv1_vnb);
%fprintf('\nDeltaV1 to escape Earth Parking Orbit = %.4f km/s \n', dv1);

dv2_vnb          =      rt2vnb(gamma_arr)*rth2xyz(0, 0, TA)*(v_f_xyz - v_arr_xyz);
dv2               =      norm(dv2_vnb);
%fprintf('DeltaV2 to enter Lunar Parking Orbit = %.4f km/s \n', dv2);

dv_tot            =      dv1+dv2;

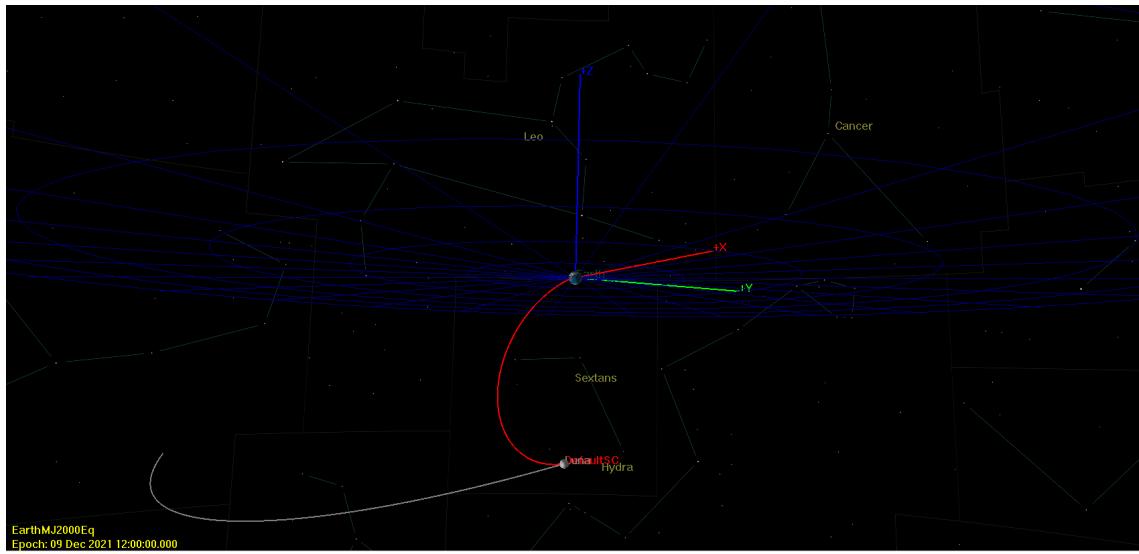
% fprintf('\nTotal Lambert DeltaV = %.4f km/s', dv_tot);

```

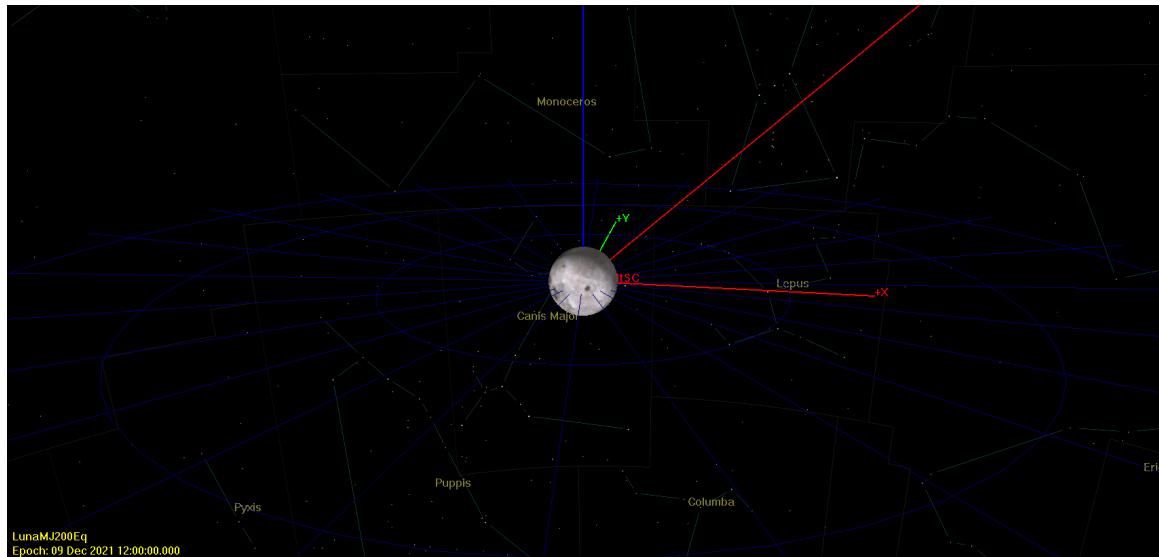
#### **IV. Preliminary Communication Satellite Trajectory Design**

In order to design a successful precision translunar trajectory, a starting point is needed. This can be accomplished through the use of patched conics. In the patched conic model, a spacecraft only interacts with one attracting body at a given time. This allows us to consider the spacecraft in the two-body problem, which has an analytical solution. From this model, we are able to solve what is called the Lambert problem. Namely, we can specify two points in space and a desired time of flight and compute a conic trajectory that will connect these two points. We can also compute a spacecraft maneuver that will allow us to “jump” onto this trajectory.

Typically, the Lambert problem is first treated in two dimensions. However, we can easily “translate” a two-dimensional (2D) Lambert trajectory into three-dimensions (3D). This is done through the use of Lunar ephemeris data, which basically serves as a table of where the Moon will be at any given time. This ephemeris data provides values of right ascension ( $\alpha$ ) and declination ( $\delta$ ), which through geometry can be related to the Keplerian parameters that describe an orbit. The relation between ephemeris data and the Keplerian parameters yields a key observation: if we know where the Moon will be at an instant of time, we can compute a 3D conic trajectory that will arrive at the Moon at the same time. An example of this is shown in Figures 1-2 below.



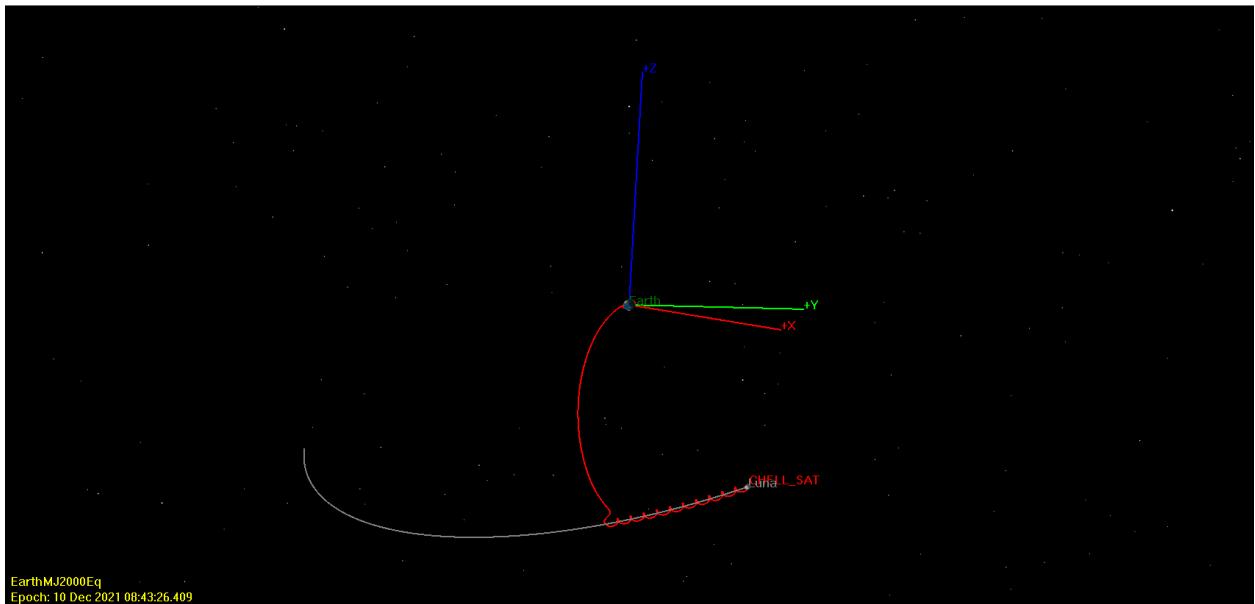
**Fig. 13D-Lambert Trajectory (Earth-Centered View):** a spacecraft follows a conic path to intersect the Moon at a specific time as part of an initial design.



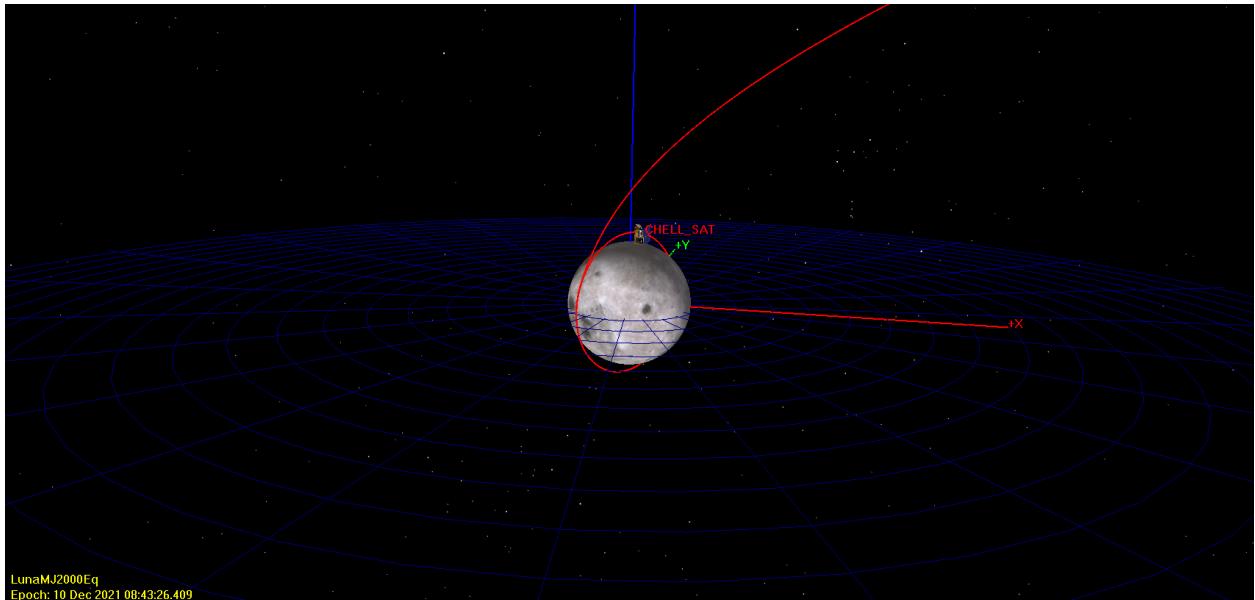
**Fig. 23D-Lambert Trajectory (Lunar-Centered View):** a spacecraft arrives at the Moon at a specific time in the two-body problem.

We note that this model does not include the effects of Lunar gravity, nor any other perturbing forces. Clearly, this is still not “good enough” to meet our design requirements, although it does serve as an important starting point. Namely, we can use a Lambert trajectory such as the one shown in Fig. 1-2 as a starting point for numeric algorithms that can “target” a more precise trajectory. This is a vital step because such algorithms typically require reasonable initial guesses to converge to a desired trajectory.

If we “turn on” Lunar gravity (as well as other perturbing forces) and use the characteristics of this conic orbit as an initial guess, we are able to obtain a trajectory that will pass over the Lunar south pole (corresponding to an inclination of  $90^\circ$ ) by using these targeting algorithms and a course adjustment maneuver during the outbound trajectory. We can then capture into a Lunar orbit as shown in Figures 3-4. This takes us one step closer to solving the problem of flying a CHELL satellite to its desired final Lunar orbit.



**Fig. 3 Targeting Example – Earth View: a spacecraft flies from Earth to the Moon captures into a polar orbit.**



**Fig. 4 Targeting Example – Lunar View: a spacecraft captures into a polar orbit of the Moon.**

The method outlined above is one method for obtaining reasonable initial translunar trajectories. Others exist, some of which omit the conic step and move directly to targeting algorithms. These methods typically employ a “coarse” and “fine” targeting loop in conjunction with one another. However, the use of such algorithms in the design of CHELL translunar trajectories is limited, so they are not discussed further.

## V. GMAT and MATLAB Code

GMAT Script for Multiple CHELL Translunar Insertion:

```
%General Mission Analysis Tool (GMAT) Script
%Created: 2021-02-27 15:02:22

%-----
%----- Spacecraft
%-----

Create Spacecraft CHELL_6;
GMAT CHELL_6.DateFormat = UTCModJulian;
GMAT CHELL_6.Epoch = '29757.25';
GMAT CHELL_6.CoordinateSystem = EarthMJ2000Eq;
GMAT CHELL_6.DisplayStateType = Keplerian;
GMAT CHELL_6.SMA = 215562.0000000678;
GMAT CHELL_6.ECC = 0.9676360000000089;
GMAT CHELL_6.INC = 28.49999999999999;
GMAT CHELL_6.RAAN = 63.75603300000003;
GMAT CHELL_6.AOP = 245.201332;
GMAT CHELL_6.TA = 1.825482000000342;
GMAT CHELL_6.DryMass = 850;
GMAT CHELL_6.Cd = 2.2;
GMAT CHELL_6.Cr = 1.8;
GMAT CHELL_6.DragArea = 15;
GMAT CHELL_6.SRPArea = 1;
GMAT CHELL_6.SPADDragScaleFactor = 1;
GMAT CHELL_6.SPADSRPScaleFactor = 1;
GMAT CHELL_6.NAIFIId = -10000001;
GMAT CHELL_6.NAIFIIdReferenceFrame = -9000001;
GMAT CHELL_6.OrbitColor = Red;
GMAT CHELL_6.TargetColor = Teal;
GMAT CHELL_6.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70
0 0 0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0
; 0 0 0 0 0 1e+70 ];
GMAT CHELL_6.CdSigma = 1e+70;
GMAT CHELL_6.CrSigma = 1e+70;
GMAT CHELL_6.Id = 'SatId';
GMAT CHELL_6.Attitude = CoordinateSystemFixed;
GMAT CHELL_6.SPADSRPInterpolationMethod = Bilinear;
GMAT CHELL_6.SPADSRPScaleFactorSigma = 1e+70;
GMAT CHELL_6.SPADDragInterpolationMethod = Bilinear;
GMAT CHELL_6.SPADDragScaleFactorSigma = 1e+70;
```

```

GMAT CHELL_6.ModelFile = 'aura.3ds';
GMAT CHELL_6.ModelOffsetX = 0;
GMAT CHELL_6.ModelOffsetY = 0;
GMAT CHELL_6.ModelOffsetZ = 0;
GMAT CHELL_6.ModelRotationX = 0;
GMAT CHELL_6.ModelRotationY = 0;
GMAT CHELL_6.ModelRotationZ = 0;
GMAT CHELL_6.ModelScale = 1;
GMAT CHELL_6.AttitudeDisplayStateType = 'Quaternion';
GMAT           CHELL_6.AttitudeRateDisplayStateType      =
'AngularVelocity';
GMAT CHELL_6.AttitudeCoordinateSystem = EarthMJ2000Eq;
GMAT CHELL_6.EulerAngleSequence = '321';

Create Spacecraft CHELL_3;
GMAT CHELL_3.DateFormat = UTCGregorian;
GMAT CHELL_3.Epoch = '26 Jun 2022 18:00:00.000';
GMAT CHELL_3.CoordinateSystem = EarthMJ2000Eq;
GMAT CHELL_3.DisplayStateType = Keplerian;
GMAT CHELL_3.SMA = 215562.0000000678;
GMAT CHELL_3.ECC = 0.9676360000000089;
GMAT CHELL_3.INC = 28.49999999999999;
GMAT CHELL_3.RAAN = 63.75603300000003;
GMAT CHELL_3.AOP = 245.201332;
GMAT CHELL_3.TA = 1.825482000000342;
GMAT CHELL_3.DryMass = 850;
GMAT CHELL_3.Cd = 2.2;
GMAT CHELL_3.Cr = 1.8;
GMAT CHELL_3.DragArea = 15;
GMAT CHELL_3.SRPArea = 1;
GMAT CHELL_3.SPADDragScaleFactor = 1;
GMAT CHELL_3.SPADSRPScaleFactor = 1;
GMAT CHELL_3.NAIFIId = -10000001;
GMAT CHELL_3.NAIFIIdReferenceFrame = -9000001;
GMAT CHELL_3.OrbitColor = [63 189 192];
GMAT CHELL_3.TargetColor = Teal;
GMAT CHELL_3.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70
0 0 0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0
; 0 0 0 0 0 1e+70 ];
GMAT CHELL_3.CdSigma = 1e+70;
GMAT CHELL_3.CrSigma = 1e+70;
GMAT CHELL_3.Id = 'SatId';
GMAT CHELL_3.Attitude = CoordinateSystemFixed;
GMAT CHELL_3.SPADSRPInterpolationMethod = Bilinear;
GMAT CHELL_3.SPADSRPScaleFactorSigma = 1e+70;
GMAT CHELL_3.SPADDragInterpolationMethod = Bilinear;

```

```

GMAT CHELL_3.SPADDragScaleFactorSigma = 1e+70;
GMAT CHELL_3.ModelFile = 'aura.3ds';
GMAT CHELL_3.ModelOffsetX = 0;
GMAT CHELL_3.ModelOffsetY = 0;
GMAT CHELL_3.ModelOffsetZ = 0;
GMAT CHELL_3.ModelRotationX = 0;
GMAT CHELL_3.ModelRotationY = 0;
GMAT CHELL_3.ModelRotationZ = 0;
GMAT CHELL_3.ModelScale = 1;
GMAT CHELL_3.AttitudeDisplayStateType = 'Quaternion';
GMAT           CHELL_3.AttitudeRateDisplayStateType      =
'AngularVelocity';
GMAT CHELL_3.AttitudeCoordinateSystem = EarthMJ2000Eq;
GMAT CHELL_3.EulerAngleSequence = '321';

%-----
%----- ForceModels
%-----

Create ForceModel NearEarth_ForceModel;
GMAT NearEarth_ForceModel.CentralBody = Earth;
GMAT NearEarth_ForceModel.PrimaryBodies = {Earth};
GMAT NearEarth_ForceModel.PointMasses = {Jupiter, Luna, Sun};
GMAT NearEarth_ForceModel.Drag = None;
GMAT NearEarth_ForceModel.SRP = On;
GMAT NearEarth_ForceModel.RelativisticCorrection = Off;
GMAT NearEarth_ForceModel.ErrorControl = RSSStep;
GMAT NearEarth_ForceModel.GravityField.Earth.Degree = 4;
GMAT NearEarth_ForceModel.GravityField.Earth.Order = 4;
GMAT NearEarth_ForceModel.GravityField.Earth.StmLimit = 100;
GMAT NearEarth_ForceModel.GravityField.Earth.PotentialFile =
'JGM2.cof';
GMAT     NearEarth_ForceModel.GravityField.Earth.TideModel      =
'None';
GMAT NearEarth_ForceModel.SRP.Flux = 1367;
GMAT NearEarth_ForceModel.SRP.SRPMODEL = Spherical;
GMAT NearEarth_ForceModel.SRP.Nominal_Sun = 149597870.691;

Create ForceModel NearMoon_ForceModel;
GMAT NearMoon_ForceModel.CentralBody = Luna;
GMAT NearMoon_ForceModel.PrimaryBodies = {Luna};
GMAT NearMoon_ForceModel.PointMasses = {Earth, Sun};
GMAT NearMoon_ForceModel.Drag = None;
GMAT NearMoon_ForceModel.SRP = On;
GMAT NearMoon_ForceModel.RelativisticCorrection = Off;
GMAT NearMoon_ForceModel.ErrorControl = RSSStep;

```

```

GMAT NearMoon_ForceModel.GravityField.Luna.Degree = 4;
GMAT NearMoon_ForceModel.GravityField.Luna.Order = 4;
GMAT NearMoon_ForceModel.GravityField.Luna.StmLimit = 100;
GMAT NearMoon_ForceModel.GravityField.Luna.PotentialFile =
'LP165P.cof';
GMAT NearMoon_ForceModel.GravityField.Luna.TideModel =
'None';
GMAT NearMoon_ForceModel.SRP.Flux = 1367;
GMAT NearMoon_ForceModel.SRP.SRPModel = Spherical;
GMAT NearMoon_ForceModel.SRP.Nominal_Sun = 149597870.691;

%----- Propagators
%-----

Create Propagator NearEarth;
GMAT NearEarth.FM = NearEarth_ForceModel;
GMAT NearEarth.Type = RungeKutta89;
GMAT NearEarth.InitialStepSize = 1;
GMAT NearEarth.Accuracy = 9.99999999999999e-12;
GMAT NearEarth.MinStep = 0.001;
GMAT NearEarth.MaxStep = 2700;
GMAT NearEarth.MaxStepAttempts = 50;
GMAT NearEarth.StopIfAccuracyIsViolated = true;

Create Propagator NearMoon;
GMAT NearMoon.FM = NearMoon_ForceModel;
GMAT NearMoon.Type = RungeKutta89;
GMAT NearMoon.InitialStepSize = 60;
GMAT NearMoon.Accuracy = 9.99999999999999e-12;
GMAT NearMoon.MinStep = 0.001;
GMAT NearMoon.MaxStep = 2700;
GMAT NearMoon.MaxStepAttempts = 50;
GMAT NearMoon.StopIfAccuracyIsViolated = true;

%----- Burns
%-----


Create ImpulsiveBurn PlaneChange;
GMAT PlaneChange.CoordinateSystem = Local;
GMAT PlaneChange.Origin = Earth;
GMAT PlaneChange.Axes = VNB;
GMAT PlaneChange.Element1 = 0;
GMAT PlaneChange.Element2 = 0;
GMAT PlaneChange.Element3 = 0;

```

```
GMAT PlaneChange.DecrementMass = false;
GMAT PlaneChange.Isp = 300;
GMAT PlaneChange.GravitationalAccel = 9.81;

Create ImpulsiveBurn CaptureBurn;
GMAT CaptureBurn.CoordinateSystem = Local;
GMAT CaptureBurn.Origin = Luna;
GMAT CaptureBurn.Axes = VNB;
GMAT CaptureBurn.Element1 = 0;
GMAT CaptureBurn.Element2 = 0;
GMAT CaptureBurn.Element3 = 0;
GMAT CaptureBurn.DecrementMass = false;
GMAT CaptureBurn.Isp = 300;
GMAT CaptureBurn.GravitationalAccel = 9.81;

Create ImpulsiveBurn AjustRAAN;
GMAT AjustRAAN.CoordinateSystem = Local;
GMAT AjustRAAN.Origin = Luna;
GMAT AjustRAAN.Axes = VNB;
GMAT AjustRAAN.Element1 = 0;
GMAT AjustRAAN.Element2 = 0;
GMAT AjustRAAN.Element3 = 0;
GMAT AjustRAAN.DecrementMass = false;
GMAT AjustRAAN.Isp = 300;
GMAT AjustRAAN.GravitationalAccel = 9.81;

Create ImpulsiveBurn FinalizeINC;
GMAT FinalizeINC.CoordinateSystem = Local;
GMAT FinalizeINC.Origin = Luna;
GMAT FinalizeINC.Axes = VNB;
GMAT FinalizeINC.Element1 = 0;
GMAT FinalizeINC.Element2 = 0;
GMAT FinalizeINC.Element3 = 0;
GMAT FinalizeINC.DecrementMass = false;
GMAT FinalizeINC.Isp = 300;
GMAT FinalizeINC.GravitationalAccel = 9.81;

Create ImpulsiveBurn ReducePeriapsis;
GMAT ReducePeriapsis.CoordinateSystem = Local;
GMAT ReducePeriapsis.Origin = Luna;
GMAT ReducePeriapsis.Axes = VNB;
GMAT ReducePeriapsis.Element1 = 0;
GMAT ReducePeriapsis.Element2 = 0;
GMAT ReducePeriapsis.Element3 = 0;
GMAT ReducePeriapsis.DecrementMass = false;
GMAT ReducePeriapsis.Isp = 300;
```

```
GMAT ReducePeriapsis.GravitationalAccel = 9.81;

Create ImpulsiveBurn FinalizeOrbit;
GMAT FinalizeOrbit.CoordinateSystem = Local;
GMAT FinalizeOrbit.Origin = Luna;
GMAT FinalizeOrbit.Axes = VNB;
GMAT FinalizeOrbit.Element1 = 0;
GMAT FinalizeOrbit.Element2 = 0;
GMAT FinalizeOrbit.Element3 = 0;
GMAT FinalizeOrbit.DecrementMass = false;
GMAT FinalizeOrbit.Isp = 300;
GMAT FinalizeOrbit.GravitationalAccel = 9.81;

Create ImpulsiveBurn FinalizeOrbit2;
GMAT FinalizeOrbit2.CoordinateSystem = Local;
GMAT FinalizeOrbit2.Origin = Luna;
GMAT FinalizeOrbit2.Axes = VNB;
GMAT FinalizeOrbit2.Element1 = 0;
GMAT FinalizeOrbit2.Element2 = 0;
GMAT FinalizeOrbit2.Element3 = 0;
GMAT FinalizeOrbit2.DecrementMass = false;
GMAT FinalizeOrbit2.Isp = 300;
GMAT FinalizeOrbit2.GravitationalAccel = 9.81;

Create ImpulsiveBurn RecuceApoC2;
GMAT RecuceApoC2.CoordinateSystem = Local;
GMAT RecuceApoC2.Origin = Earth;
GMAT RecuceApoC2.Axes = VNB;
GMAT RecuceApoC2.Element1 = 0;
GMAT RecuceApoC2.Element2 = 0;
GMAT RecuceApoC2.Element3 = 0;
GMAT RecuceApoC2.DecrementMass = false;
GMAT RecuceApoC2.Isp = 300;
GMAT RecuceApoC2.GravitationalAccel = 9.81;

%----- Coordinate Systems
%-----

Create CoordinateSystem LunaInertial;
GMAT LunaInertial.Origin = Luna;
GMAT LunaInertial.Axes = BodyInertial;

%----- Solvers
%
```

```
Create DifferentialCorrector DefaultDC;
GMAT DefaultDC.ShowProgress = true;
GMAT DefaultDC.ReportStyle = Normal;
GMAT                               DefaultDC.ReportFile      =
'DifferentialCorrectorDefaultDC.data';
GMAT DefaultDC.MaximumIterations = 100;
GMAT DefaultDC.DerivativeMethod = ForwardDifference;
GMAT DefaultDC.Algorithm = NewtonRaphson;

%----- Subscribers -----
%----- Subscribers -----



Create OrbitView EarthCentral;
GMAT EarthCentral.SolverIterations = Current;
GMAT   EarthCentral.UpperLeft     = [    0.2029411764705882
0.2493857493857494 ];
GMAT   EarthCentral.Size         = [    0.5588235294117647
0.4459459459459459 ];
GMAT EarthCentral.RelativeZOrder = 208;
GMAT EarthCentral.Maximized = false;
GMAT EarthCentral.Add = {CHELL_6, CHELL_3, Earth, Luna};
GMAT EarthCentral.CoordinateSystem = EarthMJ2000Eq;
GMAT EarthCentral.DrawObject = [ true true true true ];
GMAT EarthCentral.DataCollectFrequency = 1;
GMAT EarthCentral.UpdatePlotFrequency = 50;
GMAT EarthCentral.NumPointsToRedraw = 0;
GMAT EarthCentral.ShowPlot = true;
GMAT EarthCentral.MaxPlotPoints = 20000;
GMAT EarthCentral.ShowLabels = true;
GMAT EarthCentral.ViewPointReference = Earth;
GMAT EarthCentral.ViewPointVector = [ 30000 0 0 ];
GMAT EarthCentral.ViewDirection = Earth;
GMAT EarthCentral.ViewScaleFactor = 1;
GMAT EarthCentral.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT EarthCentral.ViewUpAxis = Z;
GMAT EarthCentral.EclipticPlane = Off;
GMAT EarthCentral.XYPlane = Off;
GMAT EarthCentral.WireFrame = Off;
GMAT EarthCentral.Axes = On;
GMAT EarthCentral.Grid = Off;
GMAT EarthCentral.SunLine = Off;
GMAT EarthCentral.UseInitialView = On;
GMAT EarthCentral.StarCount = 7000;
GMAT EarthCentral.EnableStars = On;
```

```
GMAT EarthCentral.EnableConstellations = On;

Create OrbitView Selenocentric;
GMAT Selenocentric.SolverIterations = Current;
GMAT Selenocentric.UpperLeft    = [ 0.1735294117647059 -
40.0036855036855 ];
GMAT Selenocentric.Size        = [ 0.591764705882353
0.4459459459459459 ];
GMAT Selenocentric.RelativeZOrder = 193;
GMAT Selenocentric.Maximized = false;
GMAT Selenocentric.Add = {CHELL_6, CHELL_3, Earth, Luna};
GMAT Selenocentric.CoordinateSystem = LunaInertial;
GMAT Selenocentric.DrawObject = [ true true true true ];
GMAT Selenocentric.DataCollectFrequency = 1;
GMAT Selenocentric.UpdatePlotFrequency = 50;
GMAT Selenocentric.NumPointsToRedraw = 0;
GMAT Selenocentric.ShowPlot = true;
GMAT Selenocentric.MaxPlotPoints = 20000;
GMAT Selenocentric.ShowLabels = true;
GMAT Selenocentric.ViewPointReference = Luna;
GMAT Selenocentric.ViewPointVector = [ 0 0 30000 ];
GMAT Selenocentric.ViewDirection = Luna;
GMAT Selenocentric.ViewScaleFactor = 1;
GMAT Selenocentric.ViewUpCoordinateSystem = LunaInertial;
GMAT Selenocentric.ViewUpAxis = Z;
GMAT Selenocentric.EclipticPlane = Off;
GMAT Selenocentric.XYPlane = On;
GMAT Selenocentric.WireFrame = Off;
GMAT Selenocentric.Axes = On;
GMAT Selenocentric.Grid = Off;
GMAT Selenocentric.SunLine = Off;
GMAT Selenocentric.UseInitialView = On;
GMAT Selenocentric.StarCount = 7000;
GMAT Selenocentric.EnableStars = Off;
GMAT Selenocentric.EnableConstellations = Off;

Create ReportFile ReportFile1;
GMAT ReportFile1.SolverIterations = Current;
GMAT ReportFile1.UpperLeft    = [ 0.07692307692307693
0.1597051597051597 ];
GMAT ReportFile1.Size        = [ 0.5818070818070818
0.7616707616707616 ];
GMAT ReportFile1.RelativeZOrder = 654;
GMAT ReportFile1.Maximized = false;
GMAT ReportFile1.Filename = 'ReportFile1.txt';
GMAT ReportFile1.Precision = 16;
```

```
GMAT ReportFile1.Add      = {CHELL_6.LunaInertial.RAAN,
CHELL_6.LunaInertial.INC};
GMAT ReportFile1.WriteHeader = true;
GMAT ReportFile1.LeftJustify = On;
GMAT ReportFile1.ZeroFill = Off;
GMAT ReportFile1.FixedWidth = true;
GMAT ReportFile1.Delimiter = ' ';
GMAT ReportFile1.ColumnWidth = 23;
GMAT ReportFile1.WriteReport = true;

Create ReportFile FinalCharacteristics;
GMAT FinalCharacteristics.SolverIterations = Current;
GMAT FinalCharacteristics.UpperLeft = [ 0.06043956043956044
0.0687960687960688 ];
GMAT FinalCharacteristics.Size = [ 0.5952380952380952
0.7899262899262899 ];
GMAT FinalCharacteristics.RelativeZOrder = 720;
GMAT FinalCharacteristics.Maximized = false;
GMAT FinalCharacteristics.Filename = 'C:\Users\dale
admin\Documents\AAE\Spring2021\AAE
450\NonCoplanar\SystematicApproach\ConstellationInsert\Using
Ephemerides\ConvergedTrajectories\RetargetWith2\FinalOrbitCh
ars.txt';
GMAT FinalCharacteristics.Precision = 16;
GMAT FinalCharacteristics.WriteHeader = true;
GMAT FinalCharacteristics.LeftJustify = On;
GMAT FinalCharacteristics.ZeroFill = Off;
GMAT FinalCharacteristics.FixedWidth = true;
GMAT FinalCharacteristics.Delimiter = ' ';
GMAT FinalCharacteristics.ColumnWidth = 23;
GMAT FinalCharacteristics.WriteReport = true;

Create ReportFile FinalDV;
GMAT FinalDV.SolverIterations = Current;
GMAT FinalDV.UpperLeft = [ 0.07753357753357754
0.156019656019656 ];
GMAT FinalDV.Size = [ 0.5989010989010989 0.7972972972972973 ];
GMAT FinalDV.RelativeZOrder = 602;
GMAT FinalDV.Maximized = false;
GMAT FinalDV.Filename = 'C:\Users\dale
admin\Documents\AAE\Spring2021\AAE
450\NonCoplanar\SystematicApproach\ConstellationInsert\Using
Ephemerides\ConvergedTrajectories\RetargetWith2\FinalDVs.txt
';
GMAT FinalDV.Precision = 16;
GMAT FinalDV.WriteHeader = true;
```

```
GMAT FinalDV.LeftJustify = On;
GMAT FinalDV.ZeroFill = Off;
GMAT FinalDV.FixedWidth = true;
GMAT FinalDV.Delimiter = ' ';
GMAT FinalDV.ColumnWidth = 23;
GMAT FinalDV.WriteReport = true;

Create OpenFramesInterface OpenFrames1;
GMAT OpenFrames1.SolverIterations = Current;
GMAT OpenFrames1.UpperLeft      = [ 0.1476470588235294
0.2137592137592138 ];
GMAT OpenFrames1.Size = [ 0.52 0.5110565110565111 ];
GMAT OpenFrames1.RelativeZOrder = 244;
GMAT OpenFrames1.Maximized = true;
GMAT OpenFrames1.Add = {CHELL_6, CHELL_3, Earth, Luna};
GMAT OpenFrames1.View = {CoordinateSystemView1, EarthView1,
LunaView1, CommSatView1, CommSatView2};
GMAT OpenFrames1.CoordinateSystem = LunaInertial;
GMAT OpenFrames1.DrawObject = [ true true true true ];
GMAT OpenFrames1.DrawTrajectory = [ true true true true ];
GMAT OpenFrames1.DrawAxes = [ false false false false ];
GMAT OpenFrames1.DrawXYPlane = [ false false false false ];
GMAT OpenFrames1.DrawString = [ true true true true ];
GMAT OpenFrames1.DrawUsePropLabel = [ false false false false
];
GMAT OpenFrames1.DrawCenterPoint = [ true true false false ];
GMAT OpenFrames1.DrawEndPoints = [ true true false false ];
GMAT OpenFrames1.DrawVelocity = [ false false false false ];
GMAT OpenFrames1.DrawGrid = [ false false false false ];
GMAT OpenFrames1.DrawLineWidth = [ 2 2 2 2 ];
GMAT OpenFrames1.DrawMarkerSize = [ 10 10 10 10 ];
GMAT OpenFrames1.DrawFontSize = [ 14 14 14 14 ];
GMAT OpenFrames1.Axes = On;
GMAT OpenFrames1.AxesLength = 1;
GMAT OpenFrames1.AxesLabels = On;
GMAT OpenFrames1.FrameLabel = Off;
GMAT OpenFrames1.XYPlane = On;
GMAT OpenFrames1.EclipticPlane = Off;
GMAT OpenFrames1.EnableStars = Off;
GMAT OpenFrames1.StarCatalog = 'inp_StarsHYGv3.txt';
GMAT OpenFrames1.StarCount = 40000;
GMAT OpenFrames1.MinStarMag = -2;
GMAT OpenFrames1.MaxStarMag = 6;
GMAT OpenFrames1.MinStarPixels = 1;
GMAT OpenFrames1.MaxStarPixels = 10;
GMAT OpenFrames1.MinStarDimRatio = 0.5;
```

```
GMAT OpenFrames1.ShowPlot = true;
GMAT OpenFrames1.ShowToolbar = true;
GMAT OpenFrames1.SolverIterLastN = 1;
GMAT OpenFrames1.ShowVR = false;
GMAT OpenFrames1.PlaybackTimeScale = 3600;
GMAT OpenFrames1.MultisampleAntiAliasing = On;
GMAT OpenFrames1.MSAASamples = 2;
GMAT OpenFrames1.DrawFontPosition = { 'Top-Right', 'Top-Right', 'Top-Right', 'Top-Right' };

%-----
%----- Arrays, Variables, Strings
%-----

Create Variable PropTimeLuna;
GMAT PropTimeLuna = 0;

%-----
%----- User Objects
%-----



Create OpenFramesView CoordinateSystemView1;
GMAT CoordinateSystemView1.ViewFrame = CoordinateSystem;
GMAT CoordinateSystemView1.ViewTrajectory = Off;
GMAT CoordinateSystemView1.InertialFrame = Off;
GMAT CoordinateSystemView1.SetDefaultLocation = Off;
GMAT CoordinateSystemView1.SetCurrentLocation = On;
GMAT CoordinateSystemView1.CurrentEye = [ -36406052.51103072
-26379849.01454974 79846238.19641475 ];
GMAT CoordinateSystemView1.CurrentCenter = [ -34774029.17813017 -7639211.128751844 64085343.13704695 ];
GMAT CoordinateSystemView1.CurrentUp = [ 0.9385620374083761
0.1705740175730837 0.3000096772852757 ];
GMAT CoordinateSystemView1.FOVy = 45;

Create OpenFramesView EarthView1;
GMAT EarthView1.ViewFrame = Earth;
GMAT EarthView1.ViewTrajectory = Off;
GMAT EarthView1.InertialFrame = Off;
GMAT EarthView1.SetDefaultLocation = Off;
GMAT EarthView1.SetCurrentLocation = Off;
GMAT EarthView1.FOVy = 45;

Create OpenFramesView LunaView1;
GMAT LunaView1.ViewFrame = Luna;
GMAT LunaView1.ViewTrajectory = Off;
GMAT LunaView1.InertialFrame = Off;
```

```

GMAT LunaView1.SetDefaultLocation = Off;
GMAT LunaView1.SetCurrentLocation = On;
GMAT   LunaView1.CurrentEye    = [ 29205.02472182056   -
7618.639950823027 1091.222144259993 ];
GMAT   LunaView1.CurrentCenter = [ 7.909236160201544   -
76.50372574529865 -516.9422554307641 ];
GMAT   LunaView1.CurrentUp    = [ -0.06429322790198544   -
0.03624950933749327 0.9972724572145438 ];
GMAT LunaView1.FOVy = 45;

Create OpenFramesView CommSatView1;
GMAT CommSatView1.ViewFrame = CHELL_6;
GMAT CommSatView1.ViewTrajectory = Off;
GMAT CommSatView1.InertialFrame = Off;
GMAT CommSatView1.SetDefaultLocation = Off;
GMAT CommSatView1.SetCurrentLocation = On;
GMAT   CommSatView1.CurrentEye    = [ 12494.54483223128   -
12973.17936002785 -541.1669637809755 ];
GMAT   CommSatView1.CurrentCenter = [ 0 0 -9.094947017729282e-
13 ];
GMAT   CommSatView1.CurrentUp    = [ -0.1688438372895651   -
0.2028497242215219 0.9645432846651245 ];
GMAT CommSatView1.FOVy = 45;

Create OpenFramesView CommSatView2;
GMAT CommSatView2.ViewFrame = CHELL_3;
GMAT CommSatView2.ViewTrajectory = Off;
GMAT CommSatView2.InertialFrame = Off;
GMAT CommSatView2.SetDefaultLocation = Off;
GMAT CommSatView2.SetCurrentLocation = On;
GMAT   CommSatView2.CurrentEye    = [ -17036.40622362892   -
14750.62123424074 1743.858956873945 ];
GMAT   CommSatView2.CurrentCenter = [ 944.5475456207387   -
1589.994981816175 -1296.416657127193 ];
GMAT   CommSatView2.CurrentUp    = [ 0.263004641659021   -
0.1387725968988258 0.9547621299652423 ];
GMAT CommSatView2.FOVy = 45;

%----- Mission Sequence
%-----

BeginMissionSequence;
Propagate 'Prop To Inc Adj' Synchronized NearEarth(CHELL_6)
NearEarth(CHELL_3) {CHELL_6.ElapsedDays = 2.5};

```

```

Target 'Target Inclination' DefaultDC {SolveMode = Solve,
ExitMode = DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Tangential' DefaultDC(PlaneChange.Element1 =
0.02633794514157808, {Perturbation = 0.0001, Lower = -1, Upper
= 1, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Normal' DefaultDC(PlaneChange.Element2 =
0.06953575773554994, {Perturbation = 0.0001, Lower = -1, Upper
= 1, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Binormal' DefaultDC(PlaneChange.Element3 =
0.01873471578917586, {Perturbation = 0.0001, Lower = -1, Upper
= 1, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Maneuver 'ExecuteInclinationBurn' PlaneChange(CHELL_6);
    Maneuver PlaneChange(CHELL_3);
    Propagate 'ProptoPerilune' Synchronized NearEarth(CHELL_6)
NearEarth(CHELL_3) {CHELL_6.Luna.Periapsis, OrbitColor = [134
121 126]};
    Achieve 'Achieve i=90' DefaultDC(CHELL_6.LunaInertial.INC
= 90, {Tolerance = 0.1});
    Achieve 'Achieve r=4500' DefaultDC(CHELL_6.Luna.RMAG =
4500, {Tolerance = 0.1});
EndTarget; % For targeter DefaultDC

Target 'TargetEccentricCapture' DefaultDC {SolveMode = Solve,
ExitMode = DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Tangential' DefaultDC(CaptureBurn.Element1 =
-0.4444230966506997, {Perturbation = 0.0001, Lower = -2, Upper
= 2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Normal' DefaultDC(CaptureBurn.Element2 =
-9.412064252937924e-07, {Perturbation = 0.0001, Lower = -2,
Upper = 2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Binormal' DefaultDC(CaptureBurn.Element3 =
-5.34545440750055e-06, {Perturbation = 0.0001, Lower = -2,
Upper = 2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Maneuver 'ECM' CaptureBurn(CHELL_6);
    Maneuver CaptureBurn(CHELL_3);
    Achieve 'Achieve e=0.5' DefaultDC(CHELL_6.Luna.ECC = 0.5,
{Tolerance = 0.1});
    Achieve 'Achieve i=90' DefaultDC(CHELL_6.LunaInertial.INC
= 90, {Tolerance = 0.1});

```

```

EndTarget; % For targeter DefaultDC

Propagate 'Propagate to Apoapsis' Synchronized
NearMoon(CHELL_6) NearMoon(CHELL_3) {CHELL_6.Luna.Apoapsis,
OrbitColor = [22 17 238]};

Target 'TargetRAAN' DefaultDC {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Tangential' DefaultDC(AjustRAAN.Element1 = -
0.002286743160581306, {Perturbation = 0.0001, Lower = -2,
Upper = 2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Normal' DefaultDC(AjustRAAN.Element2 = -
0.06285722502018203, {Perturbation = 0.0001, Lower = -2, Upper
= 2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'BiNormal' DefaultDC(AjustRAAN.Element3 = -
3.759272113754348e-05, {Perturbation = 0.0001, Lower = -2,
Upper = 2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Maneuver 'RAM' AjustRAAN(CHELL_6);
    Maneuver AjustRAAN(CHELL_3);
    Achieve 'Achieve RAAN' = 240'
DefaultDC(CHELL_6.LunaInertial.RAAN = 240, {Tolerance =
0.001});
    Achieve 'Achieve e = 0.5' DefaultDC(CHELL_6.Luna.ECC = 0.5,
{Tolerance = 0.1});
EndTarget; % For targeter DefaultDC

Propagate 'PropToNode' Synchronized NearMoon(CHELL_6)
NearMoon(CHELL_3) {CHELL_6.LunaInertial.Z = 0, OrbitColor =
[48 234 21]};

Target 'TargetInclination' DefaultDC {SolveMode = Solve,
ExitMode = DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Normal' DefaultDC(FinalizeINC.Element2 = -
0.06797281534335545, {Perturbation = 0.0001, Lower = -0.2,
Upper = 0.2, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Maneuver 'ApplyIncAdjust' FinalizeINC(CHELL_6);
    Maneuver FinalizeINC(CHELL_3);
    Achieve 'Achieve INC' = 80'
DefaultDC(CHELL_6.LunaInertial.INC = 80, {Tolerance = 0.01});
EndTarget; % For targeter DefaultDC

```

```

Propagate 'PropToApo' Synchronized NearMoon(CHELL_6)
NearMoon(CHELL_3) {CHELL_6.Luna.Apoapsis, OrbitColor = [216 39
211]};

Target 'TargetLower' DefaultDC {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Tangential' DefaultDC(ReducePeriapsis.Element1 = -
0.03927210804528232, {Perturbation = 0.0001, Lower = -1, Upper
= 0, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Maneuver 'ApplyPeriluneReduction'
ReducePeriapsis(CHELL_6);
    Maneuver ReducePeriapsis(CHELL_3);
    Achieve 'Achieve rp = 3238.2' DefaultDC(CHELL_6.Luna.RadPer
= 3238.2, {Tolerance = 0.1});
EndTarget; % For targeter DefaultDC

Propagate 'PropToPerilune' Synchronized NearMoon(CHELL_6)
NearMoon(CHELL_3) {CHELL_6.Luna.Periapsis, OrbitColor = [230
215 26]};
GMAT CHELL_3 = CHELL_6;
Target 'ReduceApolune1' DefaultDC {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Tangential' DefaultDC(FinalizeOrbit.Element1 = -
0.3150647031837387, {Perturbation = 0.0001, Lower = -1, Upper
= 0, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Normal' DefaultDC(FinalizeOrbit.Element2 = -
0.005174696005709776, {Perturbation = 0.0001, Lower = -1,
Upper = 1, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Vary 'Binormal' DefaultDC(FinalizeOrbit.Element3 = -
2.812502412014243e-05, {Perturbation = 0.0001, Lower = -1,
Upper = 1, MaxStep = 0.2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
    Maneuver 'AplyApoRedux1' FinalizeOrbit(CHELL_6);
    Achieve 'Achieve e=0' DefaultDC(CHELL_6.Luna.ECC = 0,
{Tolerance = 0.01});
        Achieve 'Achieve RAAN = 240'
DefaultDC(CHELL_6.LunaInertial.RAAN = 240, {Tolerance =
0.01});
EndTarget; % For targeter DefaultDC
GMAT PropTimeLuna = 2.5*CHELL_6.Luna.OrbitPeriod;
Target 'ReduceApolune2' DefaultDC {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgressWindow = true};

```

```

    Vary 'Tangential' DefaultDC(RecuceApoC2.Element1 = 0,
{Perturbation = 0.0001, Lower = -1, Upper = 1, MaxStep = 0.2,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
    Vary 'Normal' DefaultDC(RecuceApoC2.Element2 = 0,
{Perturbation = 0.0001, Lower = -1, Upper = 1, MaxStep = 0.2,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
    Vary 'Binormal' DefaultDC(RecuceApoC2.Element3 = 0,
{Perturbation = 0.0001, Lower = -1, Upper = 1, MaxStep = 0.2,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
    Maneuver 'AplyApoRedux' RecuceApoC2(CHELL_3);
    Achieve 'Achieve RadApo' DefaultDC(CHELL_3.Luna.RadApo =
4.276996480502451e+03, {Tolerance = 0.01});
    Achieve 'Achieve RAAN' DefaultDC(CHELL_3.LunaInertial.RAAN =
240, {Tolerance = 0.01});
EndTarget; % For targeter DefaultDC
Propagate Synchronized NearMoon(CHELL_6) NearMoon(CHELL_3)
{CHELL_6.ElapsedSecs = PropTimeLuna};

Target 'ReducePerilune2' DefaultDC {SolveMode = Solve,
ExitMode = DiscardAndContinue, ShowProgressWindow = true};
    Vary 'Tangential' DefaultDC(FinalizeOrbit2.Element1 = -0.3150647031837387, {Perturbation = 0.0001, Lower = -1, Upper =
1, MaxStep = 0.2, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
    Vary 'Normal' DefaultDC(FinalizeOrbit2.Element2 = -0.005174696005709776, {Perturbation = 0.0001, Lower = -1, Upper =
1, MaxStep = 0.2, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
    Vary 'Binormal' DefaultDC(FinalizeOrbit2.Element3 = -2.812502412014243e-05, {Perturbation = 0.0001, Lower = -1, Upper =
1, MaxStep = 0.2, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
    Maneuver 'FinalizeC2' FinalizeOrbit2(CHELL_3);
    Achieve 'Achieve RadApo' DefaultDC(CHELL_3.Luna.ECC = 0,
{Tolerance = 0.001});
    Achieve 'Achieve RAAN' DefaultDC(CHELL_3.LunaInertial.RAAN =
240, {Tolerance = 0.01});
EndTarget; % For targeter DefaultDC

Propagate Synchronized NearMoon(CHELL_6) NearMoon(CHELL_3)
{CHELL_6.ElapsedSecs = CHELL_6.Luna.OrbitPeriod};
Report 'ReportFinalDV' FinalDV PlaneChange.Element1
PlaneChange.Element2 PlaneChange.Element3
CaptureBurn.Element1 CaptureBurn.Element2
CaptureBurn.Element3 AjustRAAN.Element1 AjustRAAN.Element2
AjustRAAN.Element3 FinalizeINC.Element1 FinalizeINC.Element2

```

```

FinalizeINC.Element3
ReducePeriapsis.Element2
FinalizeOrbit.Element1
FinalizeOrbit.Element3
RecuceApoC2.Element2
FinalizeOrbit2.Element1
FinalizeOrbit2.Element3;
Report      'ReportFinalChars'      FinalCharacteristics
CHELL_6.Luna.SMA  CHELL_6.Luna.ECC  CHELL_6.LunaInertial.RAAN
CHELL_6.LunaInertial.INC          CHELL_6.LunaInertial.RA
CHELL_6.LunaInertial.DEC  CHELL_3.Luna.SMA  CHELL_3.Luna.ECC
CHELL_3.LunaInertial.RAAN          CHELL_3.LunaInertial.INC
CHELL_3.LunaInertial.RA  CHELL_3.LunaInertial.DEC;

```

### MATLAB Script for Generating Initial Conic Trajectories:

```

%%%%%
%%%
% Conic3D:
%
% Description: This script allows the user to input the
% characteristics of
% a desired 2D Lambert transfer (TOF, TA, and departure radius)
% and an
% Earth departure inclination. It then will output the
% corresponding 3D
% orbital characteristics and generate the beginnings of a
% GMAT script with
% a spacecraft in this transfer orbit. The script also outputs
% the delta-v
% values associated with all computed trajectories as a vector
%
% Inputs:
% TOFDays: Time of flight (days)
% TA: Transfer Angle (°)
% INC: Earth departure inclination (default to 28.5°)
% rDep: Earth departure radius (km)
% ephemerisFilename: String representing a csv version of a
% JPL Horizons
%                               database ephemeris file
%
% Outputs:
% GMAT script files corresponding to transfer orbits at various
% Epochs

```

```
% dv_vals: Associated delta-v values for each computed
% trajectory/epoch
%
%
%% Inputs
TOFDays = 4.25;
TOF = TOFDays*3600*24;
TA = 175;
INC = 28.5;
rDep = 600+6378.14;
ephemerisFilename = 'July2022EphemerisCSV.csv';
%% Read in Ephemeris Data and Declare Output File
ephDat = csvread(ephemerisFilename);

arrEpochs = ephDat(:,2);
alpha = ephDat(:,3); % RA
delta = ephDat(:,4); % DEC
rArr = ephDat(:,5); % Arrival RMAG

dv_vals = zeros(length(rArr), 3);

for i = 1:length(rArr)
    [dep_TA, arr_TA, a,e, dv] = GeneralLambert(TA, TOF, rDep,
rArr(i));
    solve_fun = @(x) solve_orb(x, alpha(i), delta(i), INC);
    slowAngles = fsolve(solve_fun, [0 0]);

    RAAN = mod(slowAngles(1),360);
    Theta = mod(slowAngles(2),360);
    AOP = Theta-arr_TA;
    theta_star_dep = dep_TA;
    inc_vals = INC;

    sc_name = strcat("SC", string(arrEpochs(i)));
    scriptFile = strcat(sc_name, ".script");
    sc_name = strcat("SC", string(floor(arrEpochs(i))));
    writeSC(scriptFile, 'w', sc_name, arrEpochs(i)-TOFDays,
a, e, INC, ...
    RAAN, AOP, theta_star_dep);
    dv_vals(i,:) = dv';
end
```

## MATLAB Lambert Problem Solver:

```
%% General Lambert Computation Function
```

```

% Description: General Purpose Lambert Problem solver.
Computes a conic
orbit through two points at different radii from Earth in
the two body
problem.

%
% Inputs:
% TA: Transfer Angle (°)
% TOF: Time of Flight (seconds)
% rDep: Departure Radius (km)
% rArr: Arrival Radius (km)
%
% Outputs:
% theta_star_dep: True anomaly at departure (°)
% theta_star_arr: True anomaly at arrival (°)
% a: Transfer orbit semimajor axis
% e: Transfer orbit eccentricity
% dv1_vnb: Vector containing delta-v components in the VNB
coordinate system
%
% Notes: Other quantities are solved for within the script
that may be of
interest to the user. One can modify the output vector to
include such
quantities.
%
function [theta_star_dep, theta_star_arr, a, e, dv1_vnb] =
GeneralLambert(TA, TOF, rDep, rArr)
    mu = 3.986e5;
    if TA < 180
        TRANSFER_NUM = 1; % Transfer Type Number (1 or 2)
    else
        TRANSFER_NUM = 2;
    end

    %% Departure Quantities and Space Triangle Geometry
    if TRANSFER_NUM == 1
        c = sqrt(rDep^2+rArr^2-2*rDep*rArr*cosd(TA));
    else
        c = sqrt(rDep^2+rArr^2-2*rDep*rArr*cosd(360-TA));
    end

    % Compute Space Triangle Semi-Perimeter

```

```

s = (rArr+rDep+c)/2;

%% Parabolic TOF and Transfer Type
a_min = s/2;
TOF_min = 0;

% Determine Parabolic TOF
if TRANSFER_NUM == 1
    TOF_par = 1/3*sqrt(2/mu)*(s^(3/2)-(s-c)^(3/2));

else
    TOF_par = 1/3*sqrt(2/mu)*(s^(3/2)+(s-c)^(3/2));
end

%% Determine Transfer Geometry (ELLIPIC or HYPERBOLIC)
if TOF > TOF_par
    TRANSFER_GEOMETRY = "ELLIPTIC";
    alpha_min = 2*asin(sqrt(s/(2*a_min)));
    beta_min = 2*asin(sqrt((s-c)/(2*a_min)));
    TOF_min = sqrt(a_min^3/mu)*((alpha_min-beta_min)-
(sin(alpha_min)-sin(beta_min)));

    if TOF>TOF_min
        TRANSFER_TYPE = num2str(TRANSFER_NUM)+"B";
    else
        TRANSFER_TYPE = num2str(TRANSFER_NUM)+"A";
    end

else
    TRANSFER_GEOMETRY = "HYPERBOLIC";
    TRANSFER_TYPE = num2str(TRANSFER_NUM)+"H";
end

%% Select Appropriate alpha/beta fcns
alpha_fcn = @(a) 0;
beta_fcn = @(b) 0;
lambert_fcn = @(a) 0;

switch TRANSFER_TYPE
    case "1A"
        alpha_fcn = @(a) 2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) 2*asin(sqrt((s-c)/(2*a)));
    case "2B"
        alpha_fcn = @(a) 2*pi-2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) -2*asin(sqrt((s-c)/(2*a)));
    case "1B"

```

```

        alpha_fcn = @(a) 2*pi - 2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) 2*asin(sqrt((s-c)/(2*a)));
    case "2A"
        alpha_fcn = @(a) 2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) -2*asin(sqrt((s-c)/(2*a)));
    case "1H"
        alpha_fcn = @(a) 2*asinh(sqrt(s/(2*abs(a))));
        beta_fcn = @(a) 2*asinh(sqrt((s-c)/(2*abs(a))));
    case "2H"
        alpha_fcn = @(a) 2*asinh(sqrt(s/(2*abs(a))));
        beta_fcn = @(a) -2*asinh(sqrt((s-c)/(2*abs(a))));
    end

    %% Select Appropriate lambert function
    guess = a_min;
    switch TRANSFER_GEOMETRY
        case "ELLIPTIC"
            lambert_fcn = @(a) sqrt(a^3/mu)*((alpha_fcn(a)-
sin(alpha_fcn(a)))-...
                (beta_fcn(a)-sin(beta_fcn(a)))-TOF;
            guess = a_min;
        case "HYPERBOLIC"
            lambert_fcn = @(a) sqrt(abs(a)^3/mu)*((sinh(alpha_fcn(a))-alpha_fcn(a))-...
                (sinh(beta_fcn(a))-beta_fcn(a)))-TOF;
            guess = -(rDep+rArr)/2;
    end

    %% Solve Lambert
    a = fsolve(lambert_fcn, guess);
    alpha_val = alpha_fcn(a);
    beta_val = beta_fcn(a);
    if abs(lambert_fcn(a)) > 1e-4
        fprintf("ERROR! BAD A");
        5/0;
    end

    %% Compute p
    % Elliptic Case
    if TRANSFER_GEOMETRY == "ELLIPTIC"
        p1 = 4*a*(s-rDep)*(s-rArr)/c^2*(sin((alpha_val+beta_val)/2))^2;
        p2 = 4*a*(s-rDep)*(s-rArr)/c^2*(sin((alpha_val-beta_val)/2))^2;

```

```

    if TRANSFER_TYPE == "1A" || TRANSFER_TYPE == "2B"
        p = max(p1, p2);
    else
        p = min(p1,p2);
    end

    % Hyperbolic Case
    else
        p1             = 4*abs(a)*(s-rDep)*(s-
rArr)/c^2*(sinh((alpha_val+beta_val)/2))^2;
        p2             = 4*abs(a)*(s-rDep)*(s-
rArr)/c^2*(sinh((alpha_val-beta_val)/2))^2;
        if TRANSFER_TYPE == "1H"
            p = max(p1, p2);
        else
            p = min(p1,p2);
        end
    end

    %% General Orbit Characteristics
    if TRANSFER_GEOMETRY == "ELLIPTIC"
        e = sqrt(1-p/a);
    else
        e = sqrt(1+p/abs(a));
    end

    spec_E = -mu/(2*a);
    v_dep = sqrt(2*(spec_E+mu/rDep));
    v_arr = sqrt(2*(spec_E+mu/rArr));

    [theta_star_dep, theta_star_arr] = get_theta_star(p, e,
rDep, rArr, TA);

    gamma_dep
sign(theta_star_dep)*acosd(sqrt(mu*p)/(rDep*v_dep));
    gamma_arr
sign(theta_star_arr)*acosd(sqrt(mu*p)/(rArr*v_arr));

    rp = a*(1-e);
    ra = a*(1+e);

    %% Delta-Vs Using F and G

    v_0_xyz = sqrt(mu/rDep)*[0 1 0]';
    v_f_xyz = sqrt(mu/rArr)*[cosd(TA+90) sind(TA+90) 0]';

```

```

r_dep_xyz = rDep * [1 0 0]';
r_arr_xyz = rArr * [cosd(TA) sind(TA) 0]';

[f, g] = fg(rDep, rArr, TA, p, mu);

v_dep_xyz = (r_arr_xyz - r_dep_xyz*f)/g;

[df, dg] = dfg(r_dep_xyz, v_dep_xyz, TA, p, mu);

v_arr_xyz = df*r_dep_xyz + dg*v_dep_xyz;

dv1_vnb = rt2vnb(0)*rth2xyz(0, 0, 0)'*(v_dep_xyz-
v_0_xyz);
dv2_vnb = rt2vnb(gamma_arr)*rth2xyz(0, 0, TA)'*(v_f_xyz-
v_arr_xyz);

dv1 = norm(dv1_vnb);
dv2 = norm(dv2_vnb);
dv_tot = dv1+dv2;
end

```

## Additional MATLAB Functions:

```

%%%%%
%% function fg
%%
%% Descriptions: Computes f and g coefficients from initial
%% radius, final
%% radius, transfer angle (change in true anomaly), semilatus
%% rectum and
%% gravitational parameter:
%%
%% Inputs:
%% r0: Initial radius (km)
%% rf: Final radius (km)
%% TA: Transfer angle (change in true anomaly) (°)
%% p: Semilatus rectum (km)
%% mu: Gravitaional parameter (km^3/s^2)
%%
%% Outputs
%% f: f-coefficient
%% g: g-coefficient
function [f, g] = fg(r0, rf, TA, p, mu)
    f = 1-rf/p*(1-cosd(TA));
    g = rf*r0/sqrt(mu*p)*sind(TA);

```

```
%%%%%%
% function dfg
%
% Description:
% Determines f_dot and g_dot coefficients from initial
position vector,
% velocity vector, true anomaly, semilatus rectum and
gravitaitonal
% parameter.
%
% Inputs:
% r0_vect: Initial position vector (km)
% v0_vect: Initial velocity vector (km/s)
% TA: Change in true anomaly (generally transfer angle)
% p: Semilatus rectum (km)
% mu: Gravitational paramter (km^3/s^2)
%
% Outputs
% df: f_dot coefficient
% dg: g_dot coefficient
%%%%%
function [df, dg] = dfg(r0_vect, v0_vect, TA, p, mu)
    r0 = norm(r0_vect);
    df      = dot(r0_vect,v0_vect) / (p*r0)*(1-cosd(TA))-
1/r0*sqrt(mu/p)*sind(TA);
    dg = 1-r0/p*(1-cosd(TA));
end

%%%%%
% function get_theta_star
%
% Description: Computes the departure and arrival true
anomalies for an
% orbit given certain characteristics

% Inputs:
% p: Orbit semilatus rectum (km)
% e: Orbit eccentricity
% r_dep: Departure radius (km)
% r_arr: Arrival radius (km)
% TA: Transfer angle (°)
%
```



```

%
% Inputs:
% gamma: flight path angle (°)
%
% Outputs:
% rot_mat: Computed rotation matrix
%
% Note, the matrix rot_mat should be right multiplied by a
vector in the
% rth coordinate system to yield one in the vnb coordinate
system
%%%%%%%%%%%%%%%
%%%%%
function rot_mat = rt2vnb(gamma)
    rot_mat = [sind(gamma) cosd(gamma) 0; 0 0 1; cosd(gamma)
-sind(gamma) 0];

%%%%%
%%%%%
% function rth2xyz
%
% Description: Outputs a rotation matrix between the
rotating orbital unit
% vectors (r_hat, theta_hat, h_hat) and the inertial xyz
system
%
% Inputs:
% Omega: Longitude of the ascending node (°)
% i: Inclination (°)
% theta: True anomaly + argument of periapsis
%
% Outputs:
% rot_mat: Rotation matrix (DCM)
%
% NOTES: The rotation matrix should be right multiplied to
convert from rth
% to the xyz system
%%%%%%%%%%%%%%%
%%%%%
function rot_mat = rth2xyz(Omega, i, theta)
    rot_mat=zeros(3);
    rot_mat(1,1) = cosd(theta)*cosd(Omega) -
cosd(i)*sind(Omega)*sind(theta);
    rot_mat(1,2) = -sind(theta)*cosd(Omega) -
cosd(theta)*cosd(i)*sind(Omega);
    rot_mat(1,3) = sind(i)*sind(Omega);

```

```

    rot_mat(2,                                1)           =
cosd(theta)*sind(Omega)+sind(theta)*cosd(i)*cosd(Omega);
    rot_mat(2,                                2)           =
sind(theta)*sind(Omega)+cosd(theta)*cosd(i)*cosd(Omega);
    rot_mat(2, 3) = -sind(i)*cosd(Omega);

    rot_mat(3,1) = sind(i)*sind(theta);
    rot_mat(3,2) = cosd(theta)*sind(i);
    rot_mat(3,3) = cosd(i);

%%%%%%%%%%%%%
% function solve_orb
%
% Description: Function to be sent to a solver to compute
orbit angles from
% right ascension and declination
%
% This function should NOT be modified. It serves only as
a "helper" for
% Conic3D
%
% Inputs:
% x: guess value for solver
% x(1): guessed RAAN (°)
% x(2): guessed THETA (True anomaly + argument of periapsis)
(°)
% alpha: Right ascension (°)
% delta: Declination (°)
% i: Set inclination (°) - Generally should be 28.5 for
Canaveral launch
%%%%%%%%%%%%%
function orb_angles = solve_orb(x, alpha, delta, i)

    orb_angles(1)           =           cosd(x(1))*cosd(x(2))-
sind(x(1))*cosd(i)*sind(x(2))-...
    cosd(delta)*cosd(alpha);
    orb_angles(2)           =           sind(x(1))*cosd(x(2))+cosd(x(1))*cosd(i)*sind(x(2))-...
    cosd(delta)*sind(alpha);
    orb_angles(3) = sind(i)*sind(x(2))-sind(delta);

%%%%%%%%%%%%%
%%%%%%%%%%%%%

```

```
% function writeSC
%
% Description: Helper function to output beginnings of a
% GMAT script with a
% spacecraft with certain initial orbit characteristics.
% Utilized within
% Conics3D
%%%%%%%%%%%%%%%
%%%%%
function writeSC(fileName, write_mode, sc_name, epoch, a, e,
i, RAAN, AOP, TA)

    write_file = fopen(fileName, write_mode);
    fprintf(write_file, 'Create Spacecraft %s\n', sc_name);
    fprintf(write_file, 'GMAT %s.DateFormat = UTCModJulian;\n', sc_name);
    fprintf(write_file, "GMAT %s.Epoch = '%f'\n", sc_name, epoch);
    fprintf(write_file, "GMAT %s.CoordinateSystem = EarthMJ2000Eq;\n", sc_name);
    fprintf(write_file, "GMAT %s.SMA = %f;\n", sc_name, a);
    fprintf(write_file, "GMAT %s.ECC = %f;\n", sc_name, e);
    fprintf(write_file, "GMAT %s.INC = %f;\n", sc_name, i);
    fprintf(write_file, "GMAT %s.RAAN = %f;\n", sc_name, RAAN);
    fprintf(write_file, "GMAT %s.AOP = %f;\n", sc_name, AOP);
    fprintf(write_file, 'GMAT %s.TA = %f;\n', sc_name, TA);
    fclose(write_file);
end
```

## VI. Lunar Ascent and Descent

The following code was modified for use in both Lunar Ascent and Lunar Descent situations. The changes that are made include changing the initial and final altitudes values as well as the desired initial and final velocities. Switching between lunar Ascent and Descent also requires changing the boundary conditions. For descent y0 and Vx0 will both be one and yf and Vxf will be zero. For ascent y0 and Vx0 will both be 0 and yf and Vxf will be one. These values are non-dimensionalized and correspond to the initial and final values specified at the beginning of the code.

### Ascent/Descent Code

```
%Optimal Ascent Problem with MATLAB's bvp4c
%
% functions and usage of bvp4c originally by Jose J. Guzman, George E.
% Pollock and Peter J. Edelman
%
% Heavily modified by Cody Martin for use in AAE 450
close all; clear all; clc;

global h g_accel Vc hscale thrust m0 A cD rho mdot
%
rM = 1738.2;
mMu = 4902.8005821478;

rp = rM + 15.24;
ra = rM + 100;

a = (rp + ra) / 2;
e = (ra-rp) / (ra+rp);
p = rp*(1+e);
Vc = sqrt(2*mMu/rp - mMu/a) * 1000;

%
% Everything else
h = (rp - rM) * 1000;
hscale = 8440;
g_accel = 1.625;
```

Cody Martin

```

thrust          =          45000;
m0              =          4700;
A               =          7.069;
cD              =          0;
rho             =          1.225;
mdot            =         15.422;

global x0 y0 Vx0 Vy0 yf Vxf Vyf Hf

```

## Boundary Conditions

```

x0              =          0;
y0              =          1;
Vx0             =          1;
Vy0             =          0;

yf              =          0;
Vxf             =          0;
Vyf             =          0;
Hf = -1;

```

## Initial Guesses

```

t0              =          0;
lambda20        =          0;
lambda30        =          1;
lambda40        =          0;

yinit = [x0 y0 Vx0 Vy0 lambda20 lambda30 lambda40];

tf_guess         =         400;

Nt              =         100;
tau             = linspace(0,1,Nt)';

solinit = bvpinit(tau,yinit,tf_guess);

```

## Solution

```

sol      = bvp4c(@ascent_odes_tf,      @ascent_bcs_tf,      solinit);
tf       =                                     sol.parameters(1);
Z       =                                     deval(sol,tau);

solinit_mass           =           solinit;
solinit_mass.y         =           Z;

```

```

delta_tf = 105;
tf = sol.parameters(1);
solinit_mass.parameters(1) = tf - delta_tf;

m0 = 10334 + 4700;
mdot = 15.422;

sol_mass = bvp4c(@ascent_odes_tf, @ascent_bcs_tf, solinit_mass);
tf = sol_mass.parameters(1);
Z = deval(sol_mass, tau);
time = t0 + tau.* (tf-t0);

xbar_sol = Z(1,:);
ybar_sol = Z(2,:);
vxbar_sol = Z(3,:);
vybar_sol = Z(4,:);
lambda2_sol = Z(5,:);
lambda3_sol = Z(6,:);
lambda4_sol = Z(7,:);

x_sol = xbar_sol * h / 1000;
y_sol = ybar_sol * h / 1000;
vx_sol = vxbar_sol * Vc / 1000;
vy_sol = vybar_sol * Vc / 1000;

theta = 180/pi.*atan(lambda4_sol./lambda3_sol); % steering angle, deg

```

## Plots

```

mf = m0 - abs(mdot)*tf;
deltaV = 9.81*311*log(m0/mf);

close all;

%{
figure(1)
axis equal
plot(x_sol,y_sol)
xlabel('x position (km)')
ylabel('y position (km)')
title('Flat Lunar Descent Cody Martin')

figure(2)
plot(time,theta)
xlabel('time (s)')
ylabel('Steering angle (deg)')
title('Steering angle vs time Cody Martin')
%}

```

Cody Martin

```

figure(3)
subplot(2,2,1)
plot(time,x_sol)
xlabel('time'          position      Cody      (s)')
ylabel('x'           vs       time      Martin') (km)')

subplot(2,2,2)
plot(time,y_sol)
xlabel('time'          position      Cody      (s)')
ylabel('y'           vs       time      Martin') (km)')

subplot(2,2,3)
plot(time,vx_sol)
xlabel('time'          velocity     Cody      (s)')
ylabel('x'           vs       time      Martin') (km/s)')

subplot(2,2,4)
plot(time,vy_sol)
xlabel('time'          velocity     Cody      (s)')
ylabel('y'           vs       time      Martin') (km/s)')

%}
%%         Lunar        Descent        Hohmann      Transfer
rM            =                      1738.2;
mMu           =                      4902.8005821478;

%fix          trajectory        for          plotting
xfix          =                      x_sol;
yfix          =                      zeros(1,100);
for          i           =          1:length(y_sol)
    ang          =          asind(x_sol(i)/(rM+y_sol(i)));
    yfix(i)      =          y_sol(i) + rM*cosd(ang);
end

%Hohman      transfer        and          Parking      orbit
rp            =          rM          +          15.24;
ra            =          rM          +          100;

a              =          (rp          +          ra)          /          2;
e              =          (ra-rp) / (ra+rp);
p              =          rp*(1+e);
v              =          sqrt(2*mMu/rp)          -          mMu          /          a;

%delta
deltaVho =  (sqrt(mMu / ra) - sqrt(2*mMu/ra - mMu / a)) * 1000;
period        =          2*pi*sqrt(a^3/mMu); v

```

```

fprintf('The delta v calculated for landing is %.3f m/s.\n',deltaV);
fprintf('The delta v calculated for the hohmann transfer is %.3f
m/s.\n',deltaVho);

%
theta = Big
r      =      [linspace(90,270,361)];
p      =      ./ (1+e.*cosd(theta) - 90));
ehat  =      r.*cosd(theta);
phat  =      r.*sind(theta);

theta = [linspace(0,360,361)];
moonE = rM.*cosd(theta);
moonP = rM.*sind(theta);

r2    = ra;
ehat2 = r2.*cosd(theta);
phat2 = r2.*sind(theta);

%{
figure(4)
hold on
grid on
axis equal
title('Lunar Descent Cody Martin')
xlim([-50,450])
ylim([1600,1850])
xlabel('X position (km)')
ylabel('Y position (km)')
fill(moonE,moonP,.8)
plot(ehat2,phat2,'g')
plot(ehat,phat,'b')
plot(xfix, yfix)
legend({'Lunar Surface','100km Parking Orbit', '100x15.24km Elliptical
Orbit', 'Final Powered Descent'}, 'Location', 'SouthWest')

figure(5)
hold on
grid on
axis equal
title('Lunar Descent Cody Martin')
%xlim([-50,450])
%ylim([1600,1850])
xlabel('X position (km)')
ylabel('Y position (km)')
fill(moonE,moonP,.8)
plot(ehat2,phat2,'g')
plot(ehat,phat,'b')
plot(xfix, yfix)
legend({'Lunar Surface','100km Parking Orbit', '100x15.24km Elliptical
Orbit', 'Final Powered Descent'}, 'Location', 'SouthWest')
```

```

```
Orbit',      'Final'     Powered    Descent'},      'Location',      'Best')
%
```

The delta v calculated for landing is 1932.184 m/s.  
The delta v calculated for the hohmann transfer is 19.386 m/s.

## Functions

```
function      dx_dtau      =      ascent_odes_tf(tau,X,tf)
global   h   Vc   thrust   mass   rho   cD   A   hscale   g_accel   m0   mdot
mass          =           m0          -           abs(mdot)*tau*tf;
k1            =           rho*A*cD          /          (2*mass);

xdot          =           X(3,:)*Vc/h;
ydot          =           X(4,:)*Vc/h;
Vxdot = (thrust / (mass*Vc)) * (-X(6,:) / sqrt(X(6,:)^2 + X(7,:)^2)) -
k1 * exp(-X(2,:)) * h/hscale) * X(3,:)*sqrt(X(3,:)^2 + X(4,:)^2) * Vc;
Vydot = (thrust / (mass*Vc)) * (-X(7,:) / sqrt(X(6,:)^2 + X(7,:)^2)) -
k1 * exp(-X(2,:)) * h/hscale) * X(4,:)*sqrt(X(3,:)^2 + X(4,:)^2) * Vc -
g_accel/Vc;

if      (sqrt(X(3,:)^2 + X(4,:)^2) == 0)
lambda_2_dot      =           0;
lambda_3_dot      =           0;
lambda_4_dot      =           -X(5,:)*Vc/h;
else
lambda_2_dot = (X(6,:)*X(3,:) + X(7,:)*X(4,:)) * (-h/hscale) * exp(
-X(2,:)*h/hscale) * k1 * sqrt(X(3,:)^2 + X(4,:)^2)*Vc;
lambda_3_dot = k1*Vc*exp(-X(2,:)) * h/hscale) * (X(6,:)*((2*X(3,:)^2 + X(4,:)^2)/sqrt(X(3,:)^2 + X(4,:)^2)) + X(7,:)*((X(3,:)*X(4,:))/sqrt(X(3,:)^2 + X(4,:)^2)));
lambda_4_dot = -X(5,:)*Vc / h + k1*Vc*exp(-X(2,:)) * h/hscale) *
(X(7,:)*((2*X(4,:)^2 + X(3,:)^2)/sqrt(X(3,:)^2 + X(4,:)^2)) + X(6,:)*((X(3,:)*X(4,:))/sqrt(X(3,:)^2 + X(4,:)^2)));
end

dX_dtau = tf*[xdot; ydot; Vxdot; Vydot; lambda_2_dot; lambda_3_dot;
lambda_4_dot];
return
```

Published with MATLAB® R2018b

```
end

function      PSI      =      ascent_bcs_tf(Y0,Yf,tf)
global   x0   y0   Vx0   Vy0   yf   Vxf   Vyf   Hf   m0   mdot
```

```

Ydotf = ascent_odes_tf(1,Yf,tf);

PSI = [Y0(1) - x0 % Initial Condition
       Y0(2) - y0 % Initial Condition
       Y0(3) - Vx0 % Initial Condition
       Y0(4) - Vy0 % Initial Condition
       Yf(2) - yf % Final Condition
       Yf(3) - Vxf % Final Condition
       Yf(4) - Vyf % Final Condition
       Yf(5)*Ydotf(2)+Yf(6)*Ydotf(3)+Yf(7)*Ydotf(4) - Hf]; % Final Condition

return
end

```

## VII. References for Lunar Ascent and Descent

### Websites

[1] *Nssdc.gsfc.nasa.gov*. 2021. *Apollo 11 Lunar Module*. [online] Available at: <<https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1969-059C>> [Accessed 30 March 2021].

[2] Longuski, J. M., Guzma'n, J. J., and Prussing, J. E., “Optimal Control with Aerospace Applications”, Springer, 2014, pp. 217-221

### VIII. References for Lunar Terrain Profile

#### *Websites*

- [1] Brill, J., “Lunar Orbiter Laser Altimeter,” Goddard Space Flight Center Available:  
<https://lola.gsfc.nasa.gov/science.html>.

#### *Digital Images*

- [2] Zink, A., Topographic Map of the Moon's South Pole (80S to Pole), Regional Planetary Image Facility, 2019.

## IX. Appendix for Lunar and Earth Rendezvous and Initial Free-Return Analysis

```
% General Mission Analysis Tool (GMAT) Script
% Created: 2021-03-02 16:01:22

% -----
% ----- Spacecraft
% -----



Create Spacecraft starship;
G MAT starship.DateFormat = UTC Gregorian;
G MAT starship.Epoch = '01 Jan 2024 11:59:23.000';
G MAT starship.CoordinateSystem = EarthMJ2000Eq;
G MAT starship.DisplayStateType = Keplerian;
G MAT starship.SMA = 6678.099999999998;
G MAT starship.ECC = 1.190548774799808e-16;
G MAT starship.INC = 28.49999999999999;
G MAT starship.RAAN = 0;
G MAT starship.AOP = 0;
G MAT starship.TA = 1.000000000000265;
G MAT starship.DryMass = 220000;
G MAT starship.Cd = 2.2;
G MAT starship.Cr = 1.8;
G MAT starship.DragArea = 15;
G MAT starship.SRPArea = 1;
G MAT starship.SPADDragScaleFactor = 1;
G MAT starship.SPADS R P ScaleFactor = 1;
G MAT starship.Tanks = {ChemicalTank1};
G MAT starship.Thrusters = {ChemicalThruster1};
G MAT starship.NAIFId = -10000001;
G MAT starship.NAIFIdReferenceFrame = -9000001;
G MAT starship.OrbitColor = [0 255 64];
G MAT starship.TargetColor = [0 255 64];
G MAT starship.OrbitErrorCovariance = [ 1e+70 0 0 0 0 ; 0 1e+70 0 0 0 0 ; 0
0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ; 0 0 0 0 0 1e+70 ];
G MAT starship.CdSigma = 1e+70;
G MAT starship.CrSigma = 1e+70;
G MAT starship.Id = 'SatId';
G MAT starship.Attitude = CoordinateSystemFixed;
G MAT starship.SPADS R P InterpolationMethod = Bilinear;
G MAT starship.SPADS R P ScaleFactorSigma = 1e+70;
G MAT starship.SPAD DragInterpolationMethod = Bilinear;
G MAT starship.SPAD DragScaleFactorSigma = 1e+70;
G MAT starship.ModelFile = 'aura.3ds';
G MAT starship.ModelOffsetX = 0;
```

```

G MAT starship.ModelOffsetY = 0;
G MAT starship.ModelOffsetZ = 0;
G MAT starship.ModelRotationX = 0;
G MAT starship.ModelRotationY = 0;
G MAT starship.ModelRotationZ = 0;
G MAT starship.ModelScale = 1;
G MAT starship.AttitudeDisplayStateType = 'Quaternion'; 1
G MAT starship.AttitudeRateDisplayStateType = 'AngularVelocity';
G MAT starship.AttitudeCoordinateSystem = EarthMJ2000Eq; GMAT
starship.EulerAngleSequence = '321';

Create Spacecraft fuelDepot;
G MAT fuelDepot.DateFormat = UTC Gregorian;
G MAT fuelDepot.Epoch = '01 Jan 2024 11:59:23.000';
G MAT fuelDepot.CoordinateSystem = EarthMJ2000Eq;
G MAT fuelDepot.DisplayStateType = Keplerian;
G MAT fuelDepot.SMA = 6978.1;
G MAT fuelDepot.ECC = 0;
G MAT fuelDepot.INC = 28.49999999999999;
G MAT fuelDepot.RAAN = 0;
G MAT fuelDepot.AOP = 0;
G MAT fuelDepot.TA = 0;
G MAT fuelDepot.DryMass = 850;
G MAT fuelDepot.Cd = 2.2;
G MAT fuelDepot.Cr = 1.8;
G MAT fuelDepot.DragArea = 15;
G MAT fuelDepot.SRPArea = 1;
G MAT fuelDepot.SPADDragScaleFactor = 1;
G MAT fuelDepot.SPADS R P ScaleFactor = 1;
G MAT fuelDepot.NAIFId = -10000001;
G MAT fuelDepot.NAIFIdReferenceFrame = -9000001;
G MAT fuelDepot.OrbitColor = [255 0 0];
G MAT fuelDepot.TargetColor = [255 0 0];
G MAT fuelDepot.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ;
0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ; 0 0 0 0 0 1e+70 ];
G MAT fuelDepot.CdSigma = 1e+70;
G MAT fuelDepot.CrSigma = 1e+70;
G MAT fuelDepot.Id = 'SatId';
G MAT fuelDepot.Attitude = CoordinateSystemFixed;
G MAT fuelDepot.SPADS R P InterpolationMethod =
Bilinear; GMAT fuelDepot.SPADS R P ScaleFactorSigma =
1e+70;
G MAT fuelDepot.SPADDragInterpolationMethod = Bilinear;
G MAT fuelDepot.SPADDragScaleFactorSigma = 1e+70;
G MAT fuelDepot.ModelFile = 'aura.3ds';
G MAT fuelDepot.ModelOffsetX = 0;
G MAT fuelDepot.ModelOffsetY = 0;
G MAT fuelDepot.ModelOffsetZ = 0;

```

```

G MAT fuelDepot.ModelRotationX = 0;
G MAT fuelDepot.ModelRotationY = 0;
G MAT fuelDepot.ModelRotationZ = 0;
G MAT fuelDepot.ModelScale = 1;
G MAT fuelDepot.AttitudeDisplayStateType = 'Quaternion'; GMAT
fuelDepot.AttitudeRateDisplayStateType = 'AngularVelocity'; GMAT
fuelDepot.AttitudeCoordinateSystem = EarthMJ2000Eq; GMAT
fuelDepot.EulerAngleSequence = '321';

```

```

%-----
%----- Hardware Components
%-----

```

2

```

Create ChemicalTank ChemicalTank1;
G MAT ChemicalTank1.AllowNegativeFuelMass =
false; GMAT ChemicalTank1.FuelMass = 150000;
G MAT ChemicalTank1.Pressure = 1500;
G MAT ChemicalTank1.Temperature = 20;
G MAT ChemicalTank1.RefTemperature = 20;
G MAT ChemicalTank1.Volume = 1500000;
G MAT ChemicalTank1.FuelDensity = 800;
G MAT ChemicalTank1.PressureModel = PressureRegulated;

Create ChemicalThruster ChemicalThruster1;
G MAT ChemicalThruster1.CoordinateSystem =
Local; GMAT ChemicalThruster1.Origin = Earth;
G MAT ChemicalThruster1.Axes = VNB;
G MAT ChemicalThruster1.ThrustDirection1 =
1; GMAT ChemicalThruster1.ThrustDirection2 =
0; GMAT
ChemicalThruster1.ThrustDirection3 = 0;
G MAT ChemicalThruster1.DutyCycle = 1;
G MAT ChemicalThruster1.ThrustScaleFactor = 1;
G MAT ChemicalThruster1.DecrementMass = false;
G MAT ChemicalThruster1.Tank = {ChemicalTank1};
G MAT ChemicalThruster1.MixRatio = [ 1 ]; GMAT
ChemicalThruster1.GravitationalAccel = 9.81; GMAT
ChemicalThruster1.C1 = 12000000;
G MAT ChemicalThruster1.C2 = 0;
G MAT ChemicalThruster1.C3 = 0;
G MAT ChemicalThruster1.C4 = 0;
G MAT ChemicalThruster1.C5 = 0;
G MAT ChemicalThruster1.C6 = 0;
G MAT ChemicalThruster1.C7 = 0;
G MAT ChemicalThruster1.C8 = 0;

```

```

G MAT ChemicalThruster1.C9 = 0;
G MAT ChemicalThruster1.C10 = 0;
G MAT ChemicalThruster1.C11 = 0;
G MAT ChemicalThruster1.C12 = 0;
G MAT ChemicalThruster1.C13 = 0;
G MAT ChemicalThruster1.C14 = 0;
G MAT ChemicalThruster1.C15 = 0;
G MAT ChemicalThruster1.C16 = 0;
G MAT ChemicalThruster1.K1 = 380;
G MAT ChemicalThruster1.K2 = 0;
G MAT ChemicalThruster1.K3 = 0;
G MAT ChemicalThruster1.K4 = 0;
G MAT ChemicalThruster1.K5 = 0;
G MAT ChemicalThruster1.K6 = 0;
G MAT ChemicalThruster1.K7 = 0;
G MAT ChemicalThruster1.K8 = 0;
G MAT ChemicalThruster1.K9 = 0;
G MAT ChemicalThruster1.K10 = 0;
G MAT ChemicalThruster1.K11 = 0;
G MAT ChemicalThruster1.K12 = 0;
G MAT ChemicalThruster1.K13 = 0;
G MAT ChemicalThruster1.K14 = 0;
G MAT ChemicalThruster1.K15 = 0;

```

3

```
G MAT ChemicalThruster1.K16 = 0;
```

```
%-----
%----- Force Models
%-----
```

```

Create Force Model DefaultProp ForceModel;
G MAT DefaultProp Force Model.CentralBody = Earth;
G MAT DefaultProp Force Model.PrimaryBodies = {Earth};
G MAT DefaultProp Force Model.PointMasses = {Luna, Sun};
G MAT DefaultProp Force Model.SRP = Off;
G MAT DefaultProp Force Model.RelativisticCorrection = Off;
G MAT DefaultProp Force Model.ErrorControl = RSS Step;
G MAT DefaultProp Force Model.GravityField.Earth.Degree = 4;
G MAT DefaultProp Force Model.GravityField.Earth.Order = 4;
G MAT DefaultProp Force Model.GravityField.Earth.StmLimit = 100; GMAT
DefaultProp Force Model.GravityField.Earth.PotentialFile = 'JGM2.cof'; GMAT
DefaultProp Force Model.GravityField.Earth.TideModel = 'None'; GMAT

```

```

DefaultProp ForceModel.Drag.AtmosphereModel = MSISE90;
G MAT DefaultProp ForceModel.Drag.HistoricWeatherSource = 'ConstantFluxAndGeoMag';
G MAT DefaultProp ForceModel.Drag.PredictedWeatherSource = 'ConstantFluxAndGeoMag';
G MAT DefaultProp ForceModel.Drag.CSSISpace WeatherFile = 'Space Weather All-v1.2.txt';
G MAT DefaultProp ForceModel.Drag.SchattenFile = 'SchattenPredict.txt';
G MAT DefaultProp ForceModel.Drag.F107 = 150;
G MAT DefaultProp ForceModel.Drag.F107A = 150;
G MAT DefaultProp ForceModel.Drag.MagneticIndex = 3;
G MAT DefaultProp ForceModel.Drag.SchattenErrorModel = 'Nominal'; GMAT
DefaultProp ForceModel.Drag.SchattenTimingModel = 'NominalCycle'; GMAT
DefaultProp ForceModel.Drag.DragModel = 'Spherical';

%-----
%----- Propagators
%-----

Create Propagator DefaultProp;
G MAT DefaultProp.FM = DefaultProp ForceModel;
G MAT DefaultProp.Type = RungeKutta89;
G MAT DefaultProp.InitialStepSize = 60;
G MAT DefaultProp.Accuracy = 9.99999999999999e-12;
G MAT DefaultProp.MinStep = 0.001;
G MAT DefaultProp.MaxStep = 2700;
G MAT DefaultProp.MaxStepAttempts = 50;
G MAT DefaultProp.StopIfAccuracyIsViolated = true;

%-----
%----- Burns
%-----



Create FiniteBurn FiniteBurn1;

4
G MAT FiniteBurn1.Thrusters = {ChemicalThruster1};
G MAT FiniteBurn1.ThrottleLogicAlgorithm = 'MaxNumberOfThrusters';

Create ImpulsiveBurn DefaultIB;
G MAT DefaultIB.CoordinateSystem = Local;
G MAT DefaultIB.Origin = Earth;
G MAT DefaultIB.Axes = VNB;
G MAT DefaultIB.Element1 = 0;
G MAT DefaultIB.Element2 = 0;
G MAT DefaultIB.Element3 = 0;
G MAT DefaultIB.DecrementMass = false;
G MAT DefaultIB.Isp = 300;
G MAT DefaultIB.GravitationalAccel = 9.81;

```

```

% -----
% ----- Coordinate Systems
% -----



Create CoordinateSystem depot;
GMAT depot.Origin = fuelDepot;
GMAT depot.Axes = MJ2000Eq;

% -----
% ----- Solvers
% -----



Create DifferentialCorrector DefaultDC;
GMAT DefaultDC.ShowProgress = true;
GMAT DefaultDC.ReportStyle = Normal;
GMAT DefaultDC.ReportFile = 'DifferentialCorrectorDefaultDC.data';
GMAT DefaultDC.MaximumIterations = 25;
GMAT DefaultDC.DerivativeMethod =
ForwardDifference; GMAT DefaultDC.Algorithm =
Broyden;

% -----
% ----- Subscribers
% -----



Create OrbitView DefaultOrbitView;
GMAT DefaultOrbitView.SolverIterations = Current;
GMAT DefaultOrbitView.UpperLeft = [
0.2481927710843373 0.09311224489795919 ];
GMAT DefaultOrbitView.Size = [ 0.7048192771084337
0.9502551020408163 ];
GMAT DefaultOrbitView.RelativeZOrder = 3212;
GMAT DefaultOrbitView.Maximized = false;
GMAT DefaultOrbitView.Add = {starship, fuelDepot, Earth};
GMAT DefaultOrbitView.CoordinateSystem = EarthMJ2000Eq;
GMAT DefaultOrbitView.DrawObject = [ true true true ]; GMAT
DefaultOrbitView.DataCollectFrequency = 1;
GMAT DefaultOrbitView.UpdatePlotFrequency = 50;
GMAT DefaultOrbitView.NumPointsToRedraw = 0;
GMAT DefaultOrbitView.ShowPlot = true;
GMAT DefaultOrbitView.MaxPlotPoints = 20000;

5
GMAT DefaultOrbitView.ShowLabels = true;
GMAT DefaultOrbitView.ViewPointReference = Earth;
GMAT DefaultOrbitView.ViewPointVector = [ 30000 0 0 ];

```

```

GMAT DefaultOrbitView.ViewDirection = Earth;
GMAT DefaultOrbitView.ViewScaleFactor = 1;
GMAT DefaultOrbitView.ViewUpCoordinateSystem = 
EarthMJ2000Eq; GMAT DefaultOrbitView.ViewUpAxis = Z;
GMAT DefaultOrbitView.EclipticPlane = Off;
GMAT DefaultOrbitView.XYPlane = On;
GMAT DefaultOrbitView.WireFrame = Off;
GMAT DefaultOrbitView.Axes = On;
GMAT DefaultOrbitView.Grid = Off;
GMAT DefaultOrbitView.SunLine = Off;
GMAT DefaultOrbitView.UseInitialView = On;
GMAT DefaultOrbitView.StarCount = 7000;
GMAT DefaultOrbitView.EnableStars = On;
GMAT DefaultOrbitView.EnableConstellations = On;

Create GroundTrackPlot DefaultGroundTrackPlot;
GMAT DefaultGroundTrackPlot.SolverIterations = Current; GMAT
DefaultGroundTrackPlot.UpperLeft = [ 0.3355421686746988
0.1926020408163265 ];
GMAT DefaultGroundTrackPlot.Size = [ 0.4981927710843373 0.639030612244898 ];
GMAT DefaultGroundTrackPlot.RelativeZOrder = 3013;
GMAT DefaultGroundTrackPlot.Maximized = false;
GMAT DefaultGroundTrackPlot.Add = {starship};
GMAT DefaultGroundTrackPlot.DataCollectFrequency =
1; GMAT DefaultGroundTrackPlot.UpdatePlotFrequency =
50; GMAT DefaultGroundTrackPlot.NumPointsToRedraw
= 0;
GMAT DefaultGroundTrackPlot.ShowPlot = true;
GMAT DefaultGroundTrackPlot.MaxPlotPoints = 20000;
GMAT DefaultGroundTrackPlot.CentralBody = Earth;
GMAT DefaultGroundTrackPlot.TextureMap = 'ModifiedBlueMarble.jpg';

%-----
%----- Arrays, Variables, Strings
%-----

Create Variable BurnDuration propduration;
GMAT BurnDuration = 0;
GMAT propduration = 100;

%-----
%----- Mission Sequence
%-----



BeginMissionSequence;
Target DefaultDC {SolveMode = Solve, ExitMode = DiscardAndContinue,
ShowProgressWindow = true};
Vary DefaultDC(BurnDuration = 11, {Perturbation = 0.0001, Lower = 0.0,

```

```
Upper = 60, MaxStep = 1, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0);
BeginFiniteBurn FiniteBurn1(starship);
```

6

```
Propagate DefaultProp(starship) DefaultProp(fuelDepot) {starship.ElapsedSecs =
BurnDuration, OrbitColor = [228 247 9]};
EndFiniteBurn FiniteBurn1(starship);

Propagate DefaultProp(starship) DefaultProp(fuelDepot) {starship.Earth.Apoapsis}; Achieve
DefaultDC(starship.depot.Y = 0, {Tolerance = 32});
Achieve DefaultDC(starship.Earth.RadApo = 6978.1, {Tolerance = 10});
EndTarget; % For targeter DefaultDC
```

For this code, only the last page will be claimed for credit as this has primarily been written by Daniel and only modified to include a rendezvous

```
% General Mission Analysis Tool(GMAT) Script
% Created: 2021-02-18 16:53:58

% This script developed by GMAT creators
% Edited by Daniel Gochenaur
% Modified by Youssef Noureddine to incorporate Lunar Rendezvous

% -----
% ----- Spacecraft
% -----
```

```
Create Spacecraft MoonSat;
GMAT MoonSat.DateFormat = UTCGregorian;
GMAT MoonSat.Epoch = '10 Oct 2024 20:45:00.000';
GMAT MoonSat.CoordinateSystem = EarthMJ2000Eq;
GMAT MoonSat.DisplayStateType = Keplerian;
GMAT MoonSat.SMA = 6978.1363;
GMAT MoonSat.ECC = 3.083634674369106e-24;
GMAT MoonSat.INC = 28.699999999999999;
GMAT MoonSat.RAAN = 360;
GMAT MoonSat.AOP = 0;
GMAT MoonSat.TA = 1.478779333471098e-06;
GMAT MoonSat.DryMass = 850;
GMAT MoonSat.Cd = 2.2;
```

```

GMAT MoonSat.Cr = 1.8;
GMAT MoonSat.DragArea = 15;
GMAT MoonSat.SRPArea = 20;
GMAT MoonSat.SPADDragScaleFactor = 1;
GMAT MoonSat.SPADSRPScaleFactor = 1;
GMAT MoonSat.NAIFId = -10001001;
GMAT MoonSat.NAIFIdReferenceFrame = -9001001;
GMAT MoonSat.OrbitColor = Red;
GMAT MoonSat.TargetColor = Teal;
GMAT MoonSat.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ;
0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ; 0 0 0 0 0 1e+70 ];
GMAT MoonSat.CdSigma = 1e+70;
GMAT MoonSat.CrSigma = 1e+70;
GMAT MoonSat.Id = 'SatId';
GMAT MoonSat.Attitude = CoordinateSystemFixed;
GMAT MoonSat.SPADSRPInterpolationMethod = Bilinear;
GMAT MoonSat.SPADSRPScaleFactorSigma = 1e+70;
GMAT MoonSat.SPADDragInterpolationMethod =
Bilinear; GMAT MoonSat.SPADDragScaleFactorSigma =
1e+70;
GMAT MoonSat.ModelFile = 'aura.3ds';
GMAT MoonSat.ModelOffsetX = 0;
GMAT MoonSat.ModelOffsetY = 0;
GMAT MoonSat.ModelOffsetZ = 0;
GMAT MoonSat.ModelRotationX = 0;
GMAT MoonSat.ModelRotationY = 0;
GMAT MoonSat.ModelRotationZ = 0;

1
GMAT MoonSat.ModelScale = 1;
GMAT MoonSat.AttitudeDisplayStateType = 'Quaternion'; GMAT
MoonSat.AttitudeRateDisplayStateType = 'AngularVelocity'; GMAT
MoonSat.AttitudeCoordinateSystem = EarthMJ2000Eq; GMAT
MoonSat.EulerAngleSequence = '321';
% This script developed by GMAT creators
% Edited by Daniel Gochenaur

%
%----- Spacecraft
%

Create Spacecraft InitSat;
GMAT InitSat.DateFormat = UTC Gregorian;
GMAT InitSat.Epoch = '01 Oct 2025 01:00:00.000';
GMAT InitSat.CoordinateSystem = EarthMJ2000Eq;
GMAT InitSat.DisplayStateType = Cartesian;

```

```

G MAT InitSat.X = 7100;
G MAT InitSat.Y = 0;
G MAT InitSat.Z = 1300;
G MAT InitSat.VX = 0;
G MAT InitSat.VY = 7.35;
G MAT InitSat.VZ = 1;
G MAT InitSat.DryMass = 850;
G MAT InitSat.Cd = 2.2;
G MAT InitSat.Cr = 1.8;
G MAT InitSat.DragArea = 15;
G MAT InitSat.SRPArea = 1;
G MAT InitSat.SPADDragScaleFactor = 1;
G MAT InitSat.SPADS R P ScaleFactor = 1;
G MAT InitSat.NAIFId = -10002001;
G MAT InitSat.NAIFIdReferenceFrame = -9002001;
G MAT InitSat.OrbitColor = Green;
G MAT InitSat.TargetColor = LightGray;
G MAT InitSat.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ; 0 0
1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ];
G MAT InitSat.CdSigma = 1e+70;
G MAT InitSat.CrSigma = 1e+70;
G MAT InitSat.Id = 'SatId';
G MAT InitSat.Attitude = CoordinateSystemFixed;
G MAT InitSat.SPADS R P InterpolationMethod = Bilinear;
G MAT InitSat.SPADS R P ScaleFactorSigma = 1e+70;
G MAT InitSat.SPADDragInterpolationMethod = Bilinear;
G MAT InitSat.SPADDragScaleFactorSigma = 1e+70;
G MAT InitSat.ModelFile = 'aura.3ds';
G MAT InitSat.ModelOffsetX = 0;
G MAT InitSat.ModelOffsetY = 0;
G MAT InitSat.ModelOffsetZ = 0;
G MAT InitSat.ModelRotationX = 0;
G MAT InitSat.ModelRotationY = 0;
G MAT InitSat.ModelRotationZ = 0;
G MAT InitSat.ModelScale = 0.001;

2
G MAT InitSat.AttitudeDisplayStateType = 'Quaternion'; GMAT
InitSat.AttitudeRateDisplayStateType = 'AngularVelocity'; GMAT
InitSat.AttitudeCoordinateSystem = EarthMJ2000Eq; GMAT
InitSat.EulerAngleSequence = '321';

Create Spacecraft Spacecraft1;
G MAT Spacecraft1.DateFormat = UTC Gregorian;
G MAT Spacecraft1.Epoch = '10 Oct 2024 20:45:00.000';
G MAT Spacecraft1.CoordinateSystem = MoonInertial;
G MAT Spacecraft1.DisplayStateType = Keplerian;

```

```

GMAT Spacecraft1.SMA = 1938.000000000267;
GMAT Spacecraft1.ECC = 1.727406115461129e-13;
GMAT Spacecraft1.INC = 90.00000000000006;
GMAT Spacecraft1.RAAN = 90.89;
GMAT Spacecraft1.AOP = 0;
GMAT Spacecraft1.TA = 1.707547292503188e-06;
GMAT Spacecraft1.DryMass = 850;
GMAT Spacecraft1.Cd = 2.2;
GMAT Spacecraft1.Cr = 1.8;
GMAT Spacecraft1.DragArea = 15;
GMAT Spacecraft1.SRPArea = 1;
GMAT Spacecraft1.SPADDragScaleFactor = 1;
GMAT Spacecraft1.SPADS R P ScaleFactor = 1;
GMAT Spacecraft1.NAIFId = -10018001;
GMAT Spacecraft1.NAIFIdReferenceFrame = -9018001;
GMAT Spacecraft1.OrbitColor = Yellow;
GMAT Spacecraft1.TargetColor = DarkGray;
GMAT Spacecraft1.OrbitErrorCovariance = [ 1e+70 0 0 0 0 ; 0 1e+70 0 0 0
; 0 0 1e+70 0 0 ; 0 0 0 1e+70 0 ; 0 0 0 0 1e+70 ];
GMAT Spacecraft1.CdSigma = 1e+70;
GMAT Spacecraft1.CrSigma = 1e+70;
GMAT Spacecraft1.Id = 'SatId';
GMAT Spacecraft1.Attitude = CoordinateSystemFixed;
GMAT Spacecraft1.SPADS R P InterpolationMethod =
Bilinear; GMAT Spacecraft1.SPADS R P ScaleFactorSigma =
1e+70;
GMAT Spacecraft1.SPAD DragInterpolationMethod = Bilinear;
GMAT Spacecraft1.SPAD DragScaleFactorSigma = 1e+70;
GMAT Spacecraft1.Model File = 'aura.3ds';
GMAT Spacecraft1.Model OffsetX = 0;
GMAT Spacecraft1.Model OffsetY = 0;
GMAT Spacecraft1.Model OffsetZ = 0;
GMAT Spacecraft1.Model RotationX = 0;
GMAT Spacecraft1.Model RotationY = 0;
GMAT Spacecraft1.Model RotationZ = 0;
GMAT Spacecraft1.Model Scale = 1;
GMAT Spacecraft1.AttitudeDisplayStateType = 'Quaternion'; GMAT
Spacecraft1.AttitudeRateDisplayStateType = 'AngularVelocity'; GMAT
Spacecraft1.AttitudeCoordinateSystem = EarthMJ2000Eq; GMAT
Spacecraft1.EulerAngleSequence = '321';

%-----
%----- Force Models
%-----



Create ForceModel NearMoonProp ForceModel;
GMAT NearMoonProp ForceModel.CentralBody = Luna; GMAT
NearMoonProp ForceModel.PointMasses = {Sun, Earth, Luna};

```

```

GMAT NearMoonProp ForceModel.Drag = None;
GMAT NearMoonProp ForceModel.SRP = On;
GMAT NearMoonProp ForceModel.RelativisticCorrection = Off;
GMAT NearMoonProp ForceModel.ErrorControl = RSSStep;
GMAT NearMoonProp ForceModel.SRP.Flux = 1367;
GMAT NearMoonProp ForceModel.SRP.SRPMODEL = Spherical;
GMAT NearMoonProp ForceModel.SRP.Nominal Sun = 149597870.691;

Create ForceModel NearEarthProp ForceModel;
GMAT NearEarthProp ForceModel.CentralBody = Earth; GMAT
NearEarthProp ForceModel.PointMasses = {Earth, Sun, Luna}; GMAT
NearEarthProp ForceModel.Drag = None;
GMAT NearEarthProp ForceModel.SRP = On;
GMAT NearEarthProp ForceModel.RelativisticCorrection = Off;
GMAT NearEarthProp ForceModel.ErrorControl = RSSStep;
GMAT NearEarthProp ForceModel.SRP.Flux = 1367;
GMAT NearEarthProp ForceModel.SRP.SRPMODEL = Spherical;
GMAT NearEarthProp ForceModel.SRP.Nominal Sun = 149597870.691;

Create ForceModel EarthPointMass ForceModel;
GMAT EarthPointMass ForceModel.CentralBody = Earth;
GMAT EarthPointMass ForceModel.PointMasses =
{Earth}; GMAT EarthPointMass ForceModel.Drag = None;
GMAT EarthPointMass ForceModel.SRP = Off;
GMAT EarthPointMass ForceModel.RelativisticCorrection = Off;
GMAT EarthPointMass ForceModel.ErrorControl = RSSStep;

%-----
%----- Propagators
%-----

Create Propagator NearMoonProp;
GMAT NearMoonProp.FM = GMAT
NearMoonProp ForceModel;
GMAT NearMoonProp.Type = RungeKutta89;
GMAT NearMoonProp.InitialStepSize = 60;
GMAT NearMoonProp.Accuracy = GMAT
9.9999999999999e-12;
GMAT NearMoonProp.MinStep = 0.001;
GMAT NearMoonProp.MaxStep = 86400;
GMAT NearMoonProp.MaxStepAttempts = 50;
GMAT NearMoonProp.StopIfAccuracyIsViolated = true;

Create Propagator NearEarthProp;
GMAT NearEarthProp.FM = GMAT
NearEarthProp ForceModel;

```

```

NearEarthProp.Type = RungeKutta89;
G MAT NearEarthProp.InitialStepSize = 60; GMAT
NearEarthProp.Accuracy = 9.99999999999999e-12;
G MAT NearEarthProp.MinStep = 0.001;
G MAT NearEarthProp.MaxStep = 160000;
G MAT NearEarthProp.MaxStepAttempts = 50; GMAT
NearEarthProp.StopIfAccuracyIsViolated = true;

Create Propagator EarthPointMass;
G MAT EarthPointMass.FM =
EarthPointMass ForceModel; GMAT
EarthPointMass.Type = RungeKutta89; GMAT
EarthPointMass.InitialStepSize = 60; GMAT
EarthPointMass.Accuracy = 9.99999999999999e-12;
G MAT EarthPointMass.MinStep = 0.001;
G MAT EarthPointMass.MaxStep = 2700;
G MAT EarthPointMass.MaxStepAttempts = 50; GMAT
EarthPointMass.StopIfAccuracyIsViolated = true;

%-----%
Burns
%-----%

Create ImpulsiveBurn TOI;
G MAT TOI.CoordinateSystem = Local;
G MAT TOI.Origin = Earth;
G MAT TOI.Axes = VNB;
G MAT TOI.Element1 = 3.051;
G MAT TOI.Element2 = 0;
G MAT TOI.Element3 = -0.0204;
G MAT TOI.DecrementMass = false;
G MAT TOI.Isp = 300;
G MAT TOI.GravitationalAccel = 9.81;

Create ImpulsiveBurn LOI;
G MAT LOI.CoordinateSystem = Local;
G MAT LOI.Origin = Luna;
G MAT LOI.Axes = VNB;
G MAT LOI.Element1 = -0.5;
G MAT LOI.Element2 = 0;
G MAT LOI.Element3 = 0;
G MAT LOI.DecrementMass = false;
G MAT LOI.Isp = 300;
G MAT LOI.GravitationalAccel = 9.81;

Create ImpulsiveBurn FinalBurn;

```

```

GMAT FinalBurn.CoordinateSystem =
Local; GMAT FinalBurn.Origin = Earth;
GMAT FinalBurn.Axes = VNB;
GMAT FinalBurn.Element1 = 0;
GMAT FinalBurn.Element2 = 0;
GMAT FinalBurn.Element3 = 0;
GMAT FinalBurn.DecrementMass = false;
GMAT FinalBurn.Isp = 300;
GMAT FinalBurn.GravitationalAccel = 9.81;

Create ImpulsiveBurn ImpulsiveBurn1;
GMAT ImpulsiveBurn1.CoordinateSystem =
Local; GMAT ImpulsiveBurn1.Origin = Luna;
GMAT ImpulsiveBurn1.Axes = VNB;
GMAT ImpulsiveBurn1.Element1 = 0.1;
GMAT ImpulsiveBurn1.Element2 = 0;
GMAT ImpulsiveBurn1.Element3 = 0;
GMAT ImpulsiveBurn1.DecrementMass =
false; GMAT ImpulsiveBurn1.Isp = 300;
GMAT ImpulsiveBurn1.GravitationalAccel = 9.81;

Create ImpulsiveBurn ImpulsiveBurn2;
GMAT ImpulsiveBurn2.CoordinateSystem =
EarthMJ2000Eq; GMAT ImpulsiveBurn2.Element1 = 0.1;
GMAT ImpulsiveBurn2.Element2 = 0;
GMAT ImpulsiveBurn2.Element3 = 0;
GMAT ImpulsiveBurn2.DecrementMass =
false; GMAT ImpulsiveBurn2.Isp = 300;
GMAT ImpulsiveBurn2.GravitationalAccel = 9.81;

%-----%
Coordinate Systems
%-----%

Create CoordinateSystem EarthMoonRot;
GMAT EarthMoonRot.Origin = Earth;
GMAT EarthMoonRot.Axes =
ObjectReferenced; GMAT
EarthMoonRot.XAxis = R;
GMAT EarthMoonRot.ZAxis = N;
GMAT EarthMoonRot.Primary = Earth;
GMAT EarthMoonRot.Secondary = Luna;

Create CoordinateSystem MoonInertial;
GMAT MoonInertial.Origin = Luna;
GMAT MoonInertial.Axes = BodyInertial;

Create CoordinateSystem spacecraft1s;

```

```

G MAT spacecraft1s.Origin = Spacecraft1;
G MAT spacecraft1s.Axes = MJ2000Eq;
%-----
%----- Solvers
%-----

Create DifferentialCorrector DC1;
G MAT DC1.ShowProgress = true;
G MAT DC1.ReportStyle = Normal;
G MAT DC1.ReportFile = 'DifferentialCorrectorDC1.data';
G MAT DC1.MaximumIterations = 200;
G MAT DC1.DerivativeMethod = ForwardDifference;
G MAT DC1.Algorithm = NewtonRaphson;

Create DifferentialCorrector DC2;
G MAT DC2.ShowProgress = true;
G MAT DC2.ReportStyle = Normal;
G MAT DC2.ReportFile = 'DifferentialCorrectorDC2.data';
G MAT DC2.MaximumIterations = 80;
G MAT DC2.DerivativeMethod = ForwardDifference;
G MAT DC2.Algorithm = Broyden;

%-----
%----- Subscribers
%-----

Create OrbitView EarthMoonRotView;
G MAT EarthMoonRotView.SolverIterations = Current;
G MAT EarthMoonRotView.UpperLeft = [ 0.3989637305699482 0.08476658476658476 ];
G MAT EarthMoonRotView.Size = [ 0.5427461139896373 1.224815724815725 ];
G MAT EarthMoonRotView.RelativeZOrder = 5748;
G MAT EarthMoonRotView.Maximized = false;
G MAT EarthMoonRotView.Add = { MoonSat, Spacecraft1, Earth, Luna };
G MAT EarthMoonRotView.CoordinateSystem = EarthMoonRot; GMAT
EarthMoonRotView.Draw Object = [ true true true true ]; GMAT
EarthMoonRotView.DataCollectFrequency = 1;
G MAT EarthMoonRotView.UpdatePlotFrequency = 50;
G MAT EarthMoonRotView.NumPointsToRedraw = 0;
G MAT EarthMoonRotView.ShowPlot = true;
G MAT EarthMoonRotView.MaxPlotPoints = 20000;
G MAT EarthMoonRotView.ShowLabels = true;
G MAT EarthMoonRotView.ViewPointReference = Earth;
G MAT EarthMoonRotView.ViewPointVector = [ 10000 0 30000 ];
G MAT EarthMoonRotView.ViewDirection = Earth;
G MAT EarthMoonRotView.ViewScaleFactor = 40;
G MAT EarthMoonRotView.ViewUpCoordinateSystem =
EarthMoonRot; GMAT EarthMoonRotView.ViewUpAxis = -X;

```

```

G MAT EarthMoonRotView.EclipticPlane = Off;
G MAT EarthMoonRotView.XYPlane = Off;
G MAT EarthMoonRotView.WireFrame = Off;
G MAT EarthMoonRotView.Axes = Off;
G MAT EarthMoonRotView.Grid = Off;
G MAT EarthMoonRotView.SunLine = Off;
G MAT EarthMoonRotView.UseInitialView = On;
G MAT EarthMoonRotView.StarCount = 7000;
G MAT EarthMoonRotView.EnableStars = On;
G MAT EarthMoonRotView.EnableConstellations = Off;

Create OrbitView EarthMoonRotView2;
G MAT EarthMoonRotView2.SolverIterations = Current;
G MAT EarthMoonRotView2.UpperLeft = [ 0.5362694300518135 -0.03071253071253071 ];
G MAT EarthMoonRotView2.Size = [ 0.4572538860103627 1.12039312039312 ];
] ; GMAT EarthMoonRotView2.RelativeZOrder = 5758;
G MAT EarthMoonRotView2.Maximized = false;
G MAT EarthMoonRotView2.Add = { MoonSat, Spacecraft1, Earth, Luna };
G MAT EarthMoonRotView2.CoordinateSystem = EarthMoonRot; GMAT
EarthMoonRotView2.Draw Object = [ true true true true ]; GMAT
EarthMoonRotView2.DataCollectFrequency = 1;
G MAT EarthMoonRotView2.UpdatePlotFrequency = 50;
G MAT EarthMoonRotView2.NumPointsToRedraw = 0;
G MAT EarthMoonRotView2.ShowPlot = true;
G MAT EarthMoonRotView2.MaxPlotPoints = 20000;
G MAT EarthMoonRotView2.ShowLabels = true;
G MAT EarthMoonRotView2.ViewPointReference = Earth;
G MAT EarthMoonRotView2.ViewPointVector = [ 10000 0 30000 ];
G MAT EarthMoonRotView2.ViewDirection = Earth;
G MAT EarthMoonRotView2.ViewScaleFactor = 40;
G MAT EarthMoonRotView2.ViewUpCoordinateSystem =
EarthMoonRot; GMAT EarthMoonRotView2.ViewUpAxis = -X;
G MAT EarthMoonRotView2.EclipticPlane = Off;
G MAT EarthMoonRotView2.XYPlane = Off;
G MAT EarthMoonRotView2.WireFrame = Off;
G MAT EarthMoonRotView2.Axes = Off;
G MAT EarthMoonRotView2.Grid = Off;
G MAT EarthMoonRotView2.SunLine = Off;
G MAT EarthMoonRotView2.UseInitialView = On;
G MAT EarthMoonRotView2.StarCount = 7000;
G MAT EarthMoonRotView2.EnableStars = On;
G MAT EarthMoonRotView2.EnableConstellations = Off;

Create OrbitView MoonInertialView;
G MAT MoonInertialView.SolverIterations = Current;
G MAT MoonInertialView.UpperLeft = [ -0.1172279792746114 0.04791154791154791 ];

```

```

GMAT MoonInertialView.Size = [ 1.091968911917099 1.027027027027027 ];
GMAT MoonInertialView.RelativeZOrder = 5743;
GMAT MoonInertialView.Maximized = false;
GMAT MoonInertialView.Add = {MoonSat, Spacecraft1, Luna, Earth};
GMAT MoonInertialView.CoordinateSystem = MoonInertial; GMAT
MoonInertialView.DrawObject = [ true true true true ]; GMAT
MoonInertialView.DataCollectFrequency = 1;
GMAT MoonInertialView.UpdatePlotFrequency = 50;
GMAT MoonInertialView.NumPointsToRedraw = 150;
GMAT MoonInertialView.ShowPlot = true;
GMAT MoonInertialView.MaxPlotPoints = 20000;
GMAT MoonInertialView.ShowLabels = true;
GMAT MoonInertialView.ViewPointReference = Luna;
GMAT MoonInertialView.ViewPointVector = [ 20000 20000 20000 ];
GMAT MoonInertialView.ViewDirection = Luna;
GMAT MoonInertialView.ViewScaleFactor = 1.5;
GMAT MoonInertialView.ViewUpCoordinateSystem = MoonInertial;
GMAT MoonInertialView.ViewUpAxis = Z;
GMAT MoonInertialView.EclipticPlane = Off;
GMAT MoonInertialView.XYPlane = Off;
GMAT MoonInertialView.WireFrame = Off;
GMAT MoonInertialView.Axes = Off;
GMAT MoonInertialView.Grid = Off;
GMAT MoonInertialView.SunLine = Off;
GMAT MoonInertialView.UseInitialView = On;
GMAT MoonInertialView.StarCount = 7000;
GMAT MoonInertialView.EnableStars = On;
GMAT MoonInertialView.EnableConstellations = Off;

Create OrbitView EarthInertialView;
GMAT EarthInertialView.SolverIterations = Current; GMAT
EarthInertialView.UpperLeft = [ 0.5155440414507773
0.5479115479115479 ];
GMAT EarthInertialView.Size = [ 0.6411917098445595
1.259213759213759 ];
GMAT EarthInertialView.RelativeZOrder = 5753;
GMAT EarthInertialView.Maximized = false;
GMAT EarthInertialView.Add = {MoonSat, Spacecraft1, Earth, Luna};
GMAT EarthInertialView.CoordinateSystem = EarthMJ2000Eq; GMAT
EarthInertialView.DrawObject = [ true true true true ]; GMAT
EarthInertialView.DataCollectFrequency = 1;
GMAT EarthInertialView.UpdatePlotFrequency = 50;
GMAT EarthInertialView.NumPointsToRedraw = 0;
GMAT EarthInertialView.ShowPlot = true;
GMAT EarthInertialView.MaxPlotPoints = 20000;
GMAT EarthInertialView.ShowLabels = true;
GMAT EarthInertialView.ViewPointReference = Earth; GMAT
EarthInertialView.ViewPointVector = [ 0 0 30000 ]; GMAT

```

```

EarthInertialView.ViewDirection = Earth;
G MAT EarthInertialView.ViewScaleFactor = 45;
G MAT EarthInertialView.ViewUpCoordinateSystem = 
EarthMJ2000Eq; GMAT EarthInertialView.ViewUpAxis = Z;
G MAT EarthInertialView.EclipticPlane = Off;
G MAT EarthInertialView.XYPlane = Off;
G MAT EarthInertialView.WireFrame = Off;
G MAT EarthInertialView.Axes = Off;
G MAT EarthInertialView.Grid = Off;
G MAT EarthInertialView.SunLine = Off;
G MAT EarthInertialView.UseInitialView = On;
G MAT EarthInertialView.StarCount = 7000;
G MAT EarthInertialView.EnableStars = On;
G MAT EarthInertialView.EnableConstellations = Off;

Create ReportFile ReportFile1;
G MAT ReportFile1.SolverIterations = Current;
G MAT ReportFile1.UpperLeft = [ 0.02352941176470588 0.2275 ];
G MAT ReportFile1.Size = [ 0.9 0.78875 ];
G MAT ReportFile1.RelativeZOrder = 33;
G MAT ReportFile1.Maximized = false;
G MAT ReportFile1.Filename = 'ReportFile1.txt';
G MAT ReportFile1.Precision = 16;
G MAT ReportFile1.Add = { MoonSat.EarthMJ2000Eq.RAAN,
MoonSat.EarthMJ2000Eq.AOP, MoonSat.ElapsedDays,
MoonSat.Earth.RadPer, MoonSat.Luna.RadPer,
MoonSat.MoonInertial.BVectorAngle,
MoonSat.Luna.RMAG, MoonSat.Earth.RMAG,
MoonSat.MoonInertial.INC, TOI.Element1, TOI.Element2, TOI.Element3,
LOI.Element1,
LOI.Element2, LOI.Element3, FinalBurn.Element1, FinalBurn.Element2,
FinalBurn.Element3, TOIBurn, LOIBurn, CircBurn, DV,
Spacecraft1.Luna.RMAG};

% (TOI.Element1+TOI.Element2+TOI.Element3+LOI.Element1+LOI.Element2+LOI.Element3
+Fi FinalBurn.Element3)

G MAT ReportFile1.WriteHeader = true;
G MAT ReportFile1.LeftJustify = On;
G MAT ReportFile1.ZeroFill = Off;
G MAT ReportFile1.FixedWidth = true;
G MAT ReportFile1.Delimiter = ' ';
G MAT ReportFile1.ColumnWidth = 23;
G MAT ReportFile1.WriteReport = true;

Create ReportFile ReportFile2;
G MAT ReportFile2.SolverIterations = Current;
G MAT ReportFile2.UpperLeft = [ 0.02352941176470588 0.2275 ];

```

```

G MAT ReportFile2.Size = [ 0.9 0.78875 ];
G MAT ReportFile2.RelativeZOrder = 33;
G MAT ReportFile2.Maximized = false;
G MAT ReportFile2.Filename = 'ReportFile2.txt';
G MAT ReportFile2.Precision = 16;
G MAT ReportFile2.Add = {LOIBurn, TOIBurn, CircBurn,
ImpulsiveBurn2.Element1, rendezvousBurn};

% (TOI.Element1+TOI.Element2+TOI.Element3+LOI.Element1+LOI.Element2+LOI.Element3
+Fi FinalBurn.Element3)

G MAT ReportFile2.WriteHeader = true;
G MAT ReportFile2.LeftJustify = On;
G MAT ReportFile2.ZeroFill = Off;
G MAT ReportFile2.FixedWidth = true;
G MAT ReportFile2.Delimiter = ' ';
G MAT ReportFile2.Column Width = 23;
G MAT ReportFile2.WriteHeader = true;

Create ReportFile ReportFile3;
G MAT ReportFile3.SolverIterations = Current;
G MAT ReportFile3.UpperLeft = [ 0.02352941176470588 0.2275 ];
G MAT ReportFile3.Size = [ 0.9 0.78875 ];
G MAT ReportFile3.RelativeZOrder = 33;
G MAT ReportFile3.Maximized = false;
G MAT ReportFile3.Filename = 'ReportFile3.txt';
G MAT ReportFile3.Precision = 16;

```

10

```

G MAT ReportFile3.Add = { MoonSat.MoonInertial.RAAN, DV,
MoonSat.MoonInertial.X, Spacecraft1.MoonInertial.X,
MoonSat.MoonInertial.Y, Spacecraft1.MoonInertial.Y,
MoonSat.MoonInertial.Z, Spacecraft1.MoonInertial.Z,
MoonSat.MoonInertial.VX, Spacecraft1.MoonInertial.VX,
MoonSat.MoonInertial.VY, Spacecraft1.MoonInertial.VY,
Spacecraft1.MoonInertial.VZ, MoonSat.MoonInertial.VZ,
MoonSat.ElapsedDays};

% (TOI.Element1+TOI.Element2+TOI.Element3+LOI.Element1+LOI.Element2+LOI.Element3
+Fi FinalBurn.Element3)

G MAT ReportFile3.WriteHeader = true;
G MAT ReportFile3.LeftJustify = On;
G MAT ReportFile3.ZeroFill = Off;
G MAT ReportFile3.FixedWidth = true;
G MAT ReportFile3.Delimiter = ' ';

```

```

GMAT ReportFile3.ColumnWidth = 23;
GMAT ReportFile3.WriteReport = true;

Create OrbitView MoonInertialView2;
GMAT MoonInertialView2.SolverIterations = Current;
GMAT MoonInertialView2.UpperLeft = [-0.1269430051813472
-0.01965601965601966];
GMAT MoonInertialView2.Size = [1.090673575129534
0.5933660933660934];
GMAT MoonInertialView2.RelativeZOrder = 5767;
GMAT MoonInertialView2.Maximized = false;
GMAT MoonInertialView2.Add = {MoonSat, Spacecraft1, Luna, Earth};
GMAT MoonInertialView2.CoordinateSystem = MoonInertial;
GMAT MoonInertialView2.DrawObject = [true true true true];
GMAT MoonInertialView2.DataCollectFrequency = 1;
GMAT MoonInertialView2.UpdatePlotFrequency = 50;
GMAT MoonInertialView2.NumPointsToRedraw = 150;
GMAT MoonInertialView2.ShowPlot = true;
GMAT MoonInertialView2.MaxPlotPoints = 80000;
GMAT MoonInertialView2.ShowLabels = true;
GMAT MoonInertialView2.ViewPointReference = Luna;
GMAT MoonInertialView2.ViewPointVector = [20000 20000 20000];
GMAT MoonInertialView2.ViewDirection = Luna;
GMAT MoonInertialView2.ViewScaleFactor = 1.5;
GMAT MoonInertialView2.ViewUpCoordinateSystem = MoonInertial;
GMAT MoonInertialView2.ViewUpAxis = Z;
GMAT MoonInertialView2.EclipticPlane = Off;
GMAT MoonInertialView2.XYPlane = Off;
GMAT MoonInertialView2.WireFrame = Off;
GMAT MoonInertialView2.Axes = Off;
GMAT MoonInertialView2.Grid = Off;
GMAT MoonInertialView2.SunLine = Off;
GMAT MoonInertialView2.UseInitialView = On;
GMAT MoonInertialView2.StarCount = 7000;
GMAT MoonInertialView2.EnableStars = On;
GMAT MoonInertialView2.EnableConstellations = Off;

%-----
%----- Arrays, Variables, Strings
%-----

Create Variable RAAN AOP DV TOIBurn LOIBurn CircBurn proptime DVX DVY DVZ;
Create Variable xcomp ycomp zcomp Dist rendezvousBurn RetroBurn; GMAT
proptime = 100;
GMAT DVX = 0;
GMAT DVY = 0;
GMAT DVZ = 0;

```

```

G MAT xcomp = 0;
G MAT ycomp = 0;
G MAT zcomp = 0;
G MAT Dist = 1000000;
G MAT rendezvousBurn = 0;
G MAT RetroBurn = 0;
%-----
%----- Mission Sequence
%-----

BeginMissionSequence;

%-----
% First Target RAAN and AOP to get close to the moon
%-----

% Set spacecraft required for reinitialization after coarse
targeting.
G MAT InitSat = MoonSat;

Toggle 'Turn Off Selected
Plots' EarthInertialView EarthMoonRotView2 MoonInertialView MoonInertialView2 Off

% This target loop varies the injection orbit RAAN and AOP to align the line
of apsides with the moon.
% This is a coarse target loop that ensures the s/c is in the vicinity of
the Moon at orbit apogee.

```

## 12

```

% The RA and DEC constraints are applied in the Earth-Moon rotating frame.
The x-axis points from Earth to Moon and
% the z-axis is normal to the plane of the lunar orbit about Earth. In this frame,
we RA = 0 and DEC = 0;

Target 'Coarse Lunar Target' DC1 {SolveMode = Solve, ExitMode =
SaveAndContinue, ShowProgressWindow = true};

% Vary parking orbit orientation
Vary 'Vary RAAN' DC1(MoonSat.RAAN = 45.1, {Perturbation = .00001,
Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = 5,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
Vary 'Vary AOP' DC1(MoonSat.AOP = 2.5, {Perturbation = .00001, Lower =
-9.999999e300, Upper = 9.999999e300, MaxStep = 5, AdditiveScaleFactor
= 0.0, MultiplicativeScaleFactor = 1.0});

% Apply TOI
Maneuver 'Apply TOI' TOI(MoonSat);

```

```

% Save variables for use in fine targeting loop
GMAT 'Save RAAN' RAAN = MoonSat.RAAN;
GMAT 'Save AOP' AOP = MoonSat.AOP;

Propagate 'Prop To
Moon' EarthPointMass(MoonSat) {MoonSat.Earth.Apoapsis,
MoonSat.ElapsedDays = 3.24};

% Define the constraints that the line of apsides is aligned with moon
Achieve 'RA = 0' DC1(MoonSat.EarthMoonRot.RA = 0, {Tolerance = 1});
Achieve 'DEC = 0' DC1(MoonSat.EarthMoonRot.DEC = 0, {Tolerance = 1});

EndTarget; % For targeter DC1

%----- % Next Target
RAAN,,AOP, and Transfer orbit Maneuvre to % achieve desired
B-plane
%-----


GMAT MoonSat = InitSat;
GMAT MoonSat.OrbitColor = 'Yellow';
Toggle MoonInertialView EarthInertialView EarthMoonRotView2 On;

Target 'Fine Lunar Target' DC1 {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgress Window = true};

Vary 'Vary RAAN' DC1(MoonSat.RAAN = RAAN, {Perturbation = .000001,
Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = 2,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0}); Vary 'Vary AO P'
DC1(MoonSat.AOP = AOP, {Perturbation = .000001, Lower = -9.999999e300,
Upper = 9.999999e300, MaxStep = 2, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});

13
Vary 'Vary TOI' DC1(TOI.Element1 = TOI.Element1, {Perturbation =
.0000001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .01,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0}); Vary 'Vary TOI'
DC1(TOI.Element3 = TOI.Element3, {Perturbation = .0000001, Lower = -9.999999e300,
Upper = 9.999999e300, MaxStep = .01, AdditiveScaleFactor =
0.0, MultiplicativeScaleFactor = 1.0});

Maneuver 'Apply TOI' TOI(MoonSat);

GMAT 'Save RAAN' RAAN = MoonSat.RAAN;
GMAT 'Save AOP' AOP = MoonSat.AOP;

```

```

Propagate 'Prop To
Moon' NearEarthProp(MoonSat) {MoonSat.Luna.Periapsis};

Achieve 'Achieve RadPer' DC1(MoonSat.Luna.RMAG = 3535.8, {Tolerance =
0.1});

Propagate 'Prop Distance' NearMoonProp(MoonSat) {MoonSat.Luna.RMAG
= 66000}; %Edge of Lunar Sphere of Influence

Propagate 'Prop Distance' NearEarthProp(MoonSat) {MoonSat.Luna.RMAG =
250000}; %Edge of Lunar Sphere of Influence

Achieve 'Achieve Return' DC1(MoonSat.Earth.RadPer = 6478,
{Tolerance = 10});

EndTarget; % For targeter DC1

Propagate 'Prop 3.24 days' NearEarthProp(MoonSat) {MoonSat.Earth.RMAG
= 6900};

%----- % Next Target
RAAN,,AOP, and Transfer orbit Maneuvre to % achieve desired
B-plane
%-----


G MAT MoonSat = InitSat;

Maneuver 'Apply TOI' TOI(MoonSat);

G MAT MoonSat.RAAN = RAAN;
G MAT MoonSat.AOP = AOP;
G MAT MoonSat.OrbitColor = 'Cyan';

% Maneuver 'Apply TOI' TOI(MoonSat);

% Propagate 'Prop To Moon' NearEarthProp(MoonSat) {MoonSat.Luna.RMAG
= 66000};
Propagate 'Prop To Moon' NearEarthProp(MoonSat) {MoonSat.Luna.RMAG
= 66000};
Propagate NearMoonProp(Spacecraft1) {Spacecraft1.UTCModJulian =
MoonSat.UTCModJulian};

Toggle MoonInertialView2 On;

Target 'Lunar Orbit Target' DC1 {SolveMode = Solve, ExitMode =

```

```

DiscardAndContinue, ShowProgress Window = true};

Vary 'Vary LOI' DC1(LOI.Element1 = LOI.Element1, {Perturbation = .00001,
Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Vary 'Vary LOI' DC1(LOI.Element2 = LOI.Element2, {Perturbation = .00001,
Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Vary 'Vary LOI' DC1(LOI.Element3 = LOI.Element3, {Perturbation = .00001,
Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Maneuver 'Apply LOI' LOI(MoonSat);
Propagate NearMoonProp(MoonSat,
Spacecraft1) {MoonSat.Luna.Periapsis};

Achieve 'Achieve RadPer' DC1(MoonSat.Luna.RadPer = 2500, {Tolerance =
0.1});

Achieve 'Achieve Lunar INC' DC1(MoonSat.MoonInertial.INC = 90,
{Tolerance = 0.1});
%Achieve 'Achieve BvectorAngle'
DC1(MoonSat.MoonInertial.BVectorAngle = -89, {Tolerance = 0.1});
EndTarget; % For targeter DC1
Target 'Lunar Orbit Circularization' DC1 {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgress Window = true};

Vary 'Vary LOI' DC1(FinalBurn.Element1 = FinalBurn.Element1, {Perturbation
= .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
Vary 'Vary LOI' DC1(FinalBurn.Element2 = FinalBurn.Element2, {Perturbation
= .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
Vary 'Vary LOI' DC1(FinalBurn.Element3 = FinalBurn.Element3, {Perturbation
= .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Maneuver 'Apply Final Burn' FinalBurn(MoonSat);

Achieve 'Achieve ECC' DC1(MoonSat.Luna.ECC = 0.00, {Tolerance =
0.01});
GMAT TOIBurn = sqrt(TOI.Element1^2 + TOI.Element2^2 +
TOI.Element3^2);
GMAT LOIBurn = sqrt(LOI.Element1^2 + LOI.Element2^2 +
LOI.Element3^2);
GMAT CircBurn = sqrt(FinalBurn.Element1^2 + FinalBurn.Element2^2 +
FinalBurn.Element3^2);

```

```

FinalBurn.Element3^2);
GMAT DV = TOIBurn + LOIBurn + CircBurn;

EndTarget; % For targeter DC1
G MAT ImpulsiveBurn1.Element1 = (sqrt(4904.9*(2/MoonSat.Luna.RMAG-1/
(Spacecraft1.Luna.RMAG*0.5+MoonSat.Luna.RMAG*0.5))-
MoonSat.MoonInertial.VMAG)*1.05;
Target DC1 {SolveMode = Solve, ExitMode = DiscardAndContinue,
ShowProgressWindow = true};
Vary DC1 (proptime = 4000, {Perturbation = 0.001, Lower =
0, Upper = 10000, MaxStep = 20, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
Propagate NearMoonProp(MoonSat, Spacecraft1) {MoonSat.ElapsedSecs =
proptime};
Maneuver ImpulsiveBurn1(MoonSat);
Propagate NearMoonProp(MoonSat, Spacecraft1) {MoonSat.Luna.RMAG =
Spacecraft1.Luna.RMAG };
GMAT Dist = sqrt((MoonSat.MoonInertial.X
Spacecraft1.MoonInertial.X)^2+(MoonSat.MoonInertial.Y
Spacecraft1.MoonInertial.Y)^2+(MoonSat.MoonInertial.Z
Spacecraft1.MoonInertial.Z)^2);
Achieve DC1(Dist = 10, {Tolerance = 10});
EndTarget; % For targeter DC1

G MAT ImpulsiveBurn2.Element1 = Spacecraft1.EarthMJ2000Eq.VX
MoonSat.EarthMJ2000Eq.VX;
G MAT ImpulsiveBurn2.Element2 = Spacecraft1.EarthMJ2000Eq.VY
MoonSat.EarthMJ2000Eq.VY;
G MAT ImpulsiveBurn2.Element3 = Spacecraft1.EarthMJ2000Eq.VZ
MoonSat.EarthMJ2000Eq.VZ;
GMAT RetroBurn = sqrt(ImpulsiveBurn1.Element1^2);
GMAT rendezvousBurn = sqrt(ImpulsiveBurn2.Element1^2+ImpulsiveBurn2.Element2^2+ImpulsiveBurn2.Element3^2);
GMAT DV = DV+rendezvousBurn+RetroBurn;
Maneuver ImpulsiveBurn2(MoonSat);
Propagate NearMoonProp(MoonSat, Spacecraft1) {MoonSat.ElapsedSecs
= 12000};

clear all
inclination = 5*pi/180
% stuff you might want to change

```

```
% starting conditions for s/c
xvel(1) = 0;
yvel(1) = 10.601*cos(inclination);
zvel(1) = 10.601*sin(inclination);
xpos(1) = 6978.1;
ypos(1) = 0;
zpos(1) = 0;

% starting conditions for moon
moonzpos(1) = 377346*sin(inclination);
moonypos(1) = 377346*cos(inclination); %363229 at perigee
moonxpos(1) = -93384;
moonxvel(1) = -0.9896; % 1.07602 for perigee
moonyvel(1) = -0.18802*cos(inclination);
moonzvel(1) = -0.18802*sin(inclination);
TAFind = 103.9

% start end and increment of time
inc = 0.1;
time = 0;
uppertime = 86400*12;

% stuff you likely won't want to change
earthMu = 3.986e5; %earth mu
moonMu = 4902.8; %moon mu
count = 1; %does the counting

%plots earth
figure(1)
[earthx,earthy,earthz] = orbitfind(earthMu, 6378.1, 0, 0,inclination);
plot3(earthx,earthy,earthz)
hold on
zlabel("$$\hat{Z}$$ (km)", "Interpreter", "Latex")
xlabel("$$\hat{X}$$ (km)", "Interpreter", "Latex");
ylabel("$$\hat{Y}$$ (km)", "Interpreter", "Latex");
title('Spacecraft trajectories in a 3-body problem') xlim([-525000 500000])
ylim([-500000 500000])
zlim([-100000 100000])

h = animatedline('MaximumNumPoints',(2300000/inc),'Color','black'); %h is the spacecraft orbit
moon = animatedline('MaximumNumPoints',1,'Marker','o');
% moon is the position of the moon

%reset time and count
time = 0;
```

```

1
timer(1) = 0;
count = 1;
a = 2.648e5;
ecc = (xpos(1)/a-sqrt((xpos(1)/a)^2-4*(xpos(1)-a)/a))/2; % %required
parameters
w = 0; % %angle of peripasis
[xPos2BP,yPos2BP,zPos2BP] = orbitfind(earthMu,a,ecc,w,inclination);
plot3(xPos2BP,yPos2BP,zPos2BP,'Color','red')

closest = 100000;
while time < uppertime
    moonr =
        sqrt(moonxpos(count)*moonxpos(count)+moonypos(count)*moonypos(count)+moonzpos(cou
        moonxacc = -earthMu * moonxpos(count) / (moonr*moonr*moonr);
        moonyacc = -earthMu * moonypos(count) / (moonr*moonr*moonr);
        moonzacc = -earthMu * moonzpos(count) / (moonr*moonr*moonr);

        moonxvel(count+1) = moonxvel(count) + moonxacc * inc;
        moonyvel(count+1) = moonyvel(count) + moonyacc * inc;
        moonzvel(count+1) = moonzvel(count) + moonzacc * inc;

        moonxpos(count+1) = moonxpos(count) + moonxvel(count+1) * inc;
        moonypos(count+1) = moonypos(count) + moonyvel(count+1) * inc;
        moonzpos(count+1) = moonzpos(count) + moonzvel(count+1) * inc;

        r =
            sqrt(xpos(count)*xpos(count)+ypos(count)*ypos(count)+zpos(count)*zpos(count)); %find
        distance to earth
        scmoonr = sqrt( (moonxpos(count)-xpos(count))^2
            +(moonypos(count)-ypos(count))^2+(moonzpos(count)-
            zpos(count))^2); %find distance to moon
        if scmoonr<closest
            closest = scmoonr;
        end

        xacc = -earthMu * xpos(count) / (r*r*r);
        %acc due to earth (x-axis)
        xacc = xacc - moonMu * (xpos(count)-moonxpos(count)) /
        (scmoonr^3); %acc due to moon (take out for 2BP) yacc = -earthMu
        *ypos(count) / (r*r*r);
        %acc due to earth (y-axis)
        yacc = yacc - moonMu * (ypos(count)-moonypos(count)) /

```

```

(scmoonr^3); %acc due to moon (take out for 2BP) zacc = -earthMu
*zpos(count) / (r^2);
%acc due to earth (z-axis)
zacc = zacc - moonMu * (zpos(count)-moonzpos(count)) /
(scmoonr^3); %acc due to moon (take out for 2BP)

xvel(count+1) = xvel(count) + xacc * inc;
yvel(count+1) = yvel(count) + yacc * inc;
zvel(count+1) = zvel(count) + zacc * inc;

2
count = count + 1;
time = time + inc;
timer(count) = time;
xpos(count) = xpos(count-1) + (2*xvel(count) + xacc*inc) * inc / 2;
ypos(count) = ypos(count-1) + (2*yvel(count) + yacc*inc) * inc / 2;
zpos(count) = zpos(count-1) + (2*zvel(count) + zacc*inc) * inc / 2;

%check for crash
if (scmoonr < 1730 | r < 6378)
time
time = uppertime*3;
end
end

[moonxPos2BP, moonyPos2BP, moonzPos2BP] =
orbitfind(earthMu, 384400, 0.055076, 0, inclination);
plot3(moonxPos2BP, moonyPos2BP, moonzPos2BP, 'Color','blue')

%plots every xth number
counter = 1;
points = size(xpos);
every = 5000;
while counter < points(2)
if rem(counter,every) == 0
addpoints(h,xpos(counter),ypos(counter),zpos(counter))
addpoints(moon, moonxpos(counter), moonypos(counter), moonzpos(counter))
drawnow
end
counter = counter + 1;
end

legend('moon trajectory','spacecraft trajectory','moon location','2 body
orbit','FontSize',16)
[xvelocity, yvelocity, zvelocity, xposition, yposition, zposition] =
TAconditions(moonxpos, moonxvel, moonypos,

```

```

moonyvel, moonzpos, moonzvel, TAfind) % 20 would mean 20 degrees above x
axis

closest;
%if there is a crash, just put a * at the position
if time > uppertime*2
plot3(xpos(count), ypos(count), zpos(count), '*', 'MarkerSize', 20) end

% plots velocity profile
figure(2)
hold on
plot(timer/86400, xvel, 'Color', 'blue')
plot(timer/86400, yvel, 'Color', 'red')
plot(timer/86400, zvel, 'Color', 'green')

3
xlabel('Elapsed time (days)')
ylabel('velocity (kms^-1)')
title('x,y,z components of velocities over time')
xlim([0 12])
legend('x velocity','y velocity','z velocity','FontSize',18)
closest

function [xPos, yPos, zPos] = orbitfind(mu, a, ecc, w, inclination)
% mu = 42828; a = 18040, ecc = 0.6929; %% required parameters
lowertheta = -pi; uppertheta = pi; %% theta ranges in radians count = 1;
p = a * (1-ecc*ecc);
omega = 0;
for theta = lowertheta:0.001:uppertheta
r(count) = p / (1+ecc*cos(theta)); %% finds the radius at every theta increment
thetafake = theta+w; %% ensures all thetas are within a 0 to 2*pi range
while thetakfae<0;
thetakfae=thetakfae+6.283185307;
end
if thetakfae<pi/2 %% finds positions when theta is in first quadrant
pPos(count) = r(count)*sin(thetakfae);
ePos1(count) = r(count)*cos(thetakfae);
elseif thetakfae<pi %% finds positions when theta is in the second quadrant
pPos(count) = r(count)*sin(thetakfae);
ePos1(count) = r(count)*cos(thetakfae);
elseif thetakfae<1.5*pi %% finds positions when theta is in the third quadrant
pPos(count) = -r(count)*sin(thetakfae-pi);
ePos1(count) = -r(count)*cos(thetakfae-pi);
elseif thetakfae<2*pi %% finds positions when theta is in the fourth quadrant
pPos(count) = -r(count)*sin(2*pi-thetakfae);
ePos1(count) = r(count)*cos(2*pi-thetakfae);

```

```

end
xPos(count)=r(count)*((cos(omega)*cos(thetafake))-  

sin(omega)*cos(inclination)*sin(thetafake));

yPos(count)=r(count)*((sin(omega)*cos(thetafake))+cos(omega)*cos(inclination)*sin  

zPos(count)=r(count)*sin(inclination)*sin(thetafake);
count = count + 1;
end
nodelocation = round((pi-w)/0.001);
if nodelocation>3142
nodelocation2 = nodelocation-3142;
else
nodelocation2 = nodelocation+3142;
end
end

```

4

```

function [xvelocity,yvelocity,zvelocity,xposition,yposition,zposition] =  

TAconditions(xpos,xvel,ypos,yvel,zpos,zvel,TA)

tancurver = 0;
acc = 1;
xposition = 0;
yposition = 0;
xvelocity = 0;
yvelocity = 0;
zposition = 0;
zvelocity = 0;
tanang = 0;
points = size(xpos);
count = 1;
ypositions = sqrt(ypos.^2+zpos.^2).*sign(ypos);
yvelocitys = sqrt(yvel.^2+zvel.^2).*sign(yvel);

while count < points(2)
tanang=abs(atand(ypositions(count)/xpos(count))-TA); if
xpos(count) < 0
if ypos(count)<0
tanang = tanang - 180;
else
tanang = 180 - tanang;
end
elseif ypos(count)<0
tanang = -tanang;
end

```

```
%use below only if you want to find conditions at a given true anomaly
if abs(tanang)<acc;
acc = abs(tanang);
tancurver =
sqrt(xpos(count)^2+ypos(count)^2+zpos(count)^2);
xposition = xpos(count);
yposition = ypos(count);
zposition = zpos(count);
xvelocity = xvel(count);
yvelocity = yvel(count);
zvelocity = zvel(count);
end
count = count + 1;
end
end
```

## I.Appendix Alex Maietta

### A. Lambert Algorithm

The ‘Lambert Algorithm’ solves the two-dimensional Lambert Problem, where, given a time-of-flight and the space geometry of the problem, a semi-major axis of a transfer arc is calculated and outputted. For the higher-fidelity models, this script is updated to solve a three-dimensional problem and output six orbital characteristics ( $a, e, i, \omega, \Omega, \theta^*$ ), which will define the transfer arc and be used in GMAT as a preliminary trajectory.

```

function [a] = Lambert_Alg(amin,TOF_desired,TOF_min,s,c,mu,type)

a = amin;
TOF = TOF_min;

if type == '1A' | type == '2A'
    while TOF >= TOF_desired
        % calculate alpha0 and beta0
        alpha0 = 2*asin(sqrt(s/2/a));
        beta0 = 2*asin(sqrt((s-c)/2/a));

        if type == '1A'
            % for type 1A
            TOF = sqrt(a^3/mu)*((alpha0-sin(alpha0))-(beta0-sin(beta0)));
        elseif type == '2A'
            % for type 2A
            TOF = sqrt(a^3/mu)*((alpha0-sin(alpha0))+(beta0-sin(beta0)));
        else
            fprintf('error in determining type of transfer orbit')
            return
        end
    end
    a = a + 100;
end

```

```

elseif type == '1B' | type == '2B'
    while TOF <= TOF_desired
        % calculate alpha0 and beta0
        alpha0 = 2*asin(sqrt(s/2/a));
        beta0 = 2*asin(sqrt((s-c)/2/a));

        if type == '1B'
            % for type 1B
            TOF = sqrt(a^3/mu)*(2*pi-(alpha0-sin(alpha0))-(beta0-sin(beta0)));
        elseif type == '2B'
            % for type 2B
            TOF = sqrt(a^3/mu)*(2*pi-(alpha0-sin(alpha0))+(beta0-sin(beta0)));
        else
            fprintf('error in determining type of transfer orbit')
            return
        end

        a = a + 1;

    end

elseif type == '1H' | type == '2H'
    a = 1;
    TOF = 0;
    while TOF <= TOF_desired
        % calculate alpha0 and beta0
        alpha0 = 2*asinh(sqrt(s/2/a));
        beta0 = 2*asinh(sqrt((s-c)/2/a));

        if type == '1H'
            % for type 1H
            TOF = sqrt(a^3/mu)*((sinh(alpha0)-alpha0)-(sinh(beta0)-beta0));
        elseif type == '2H'
            % for type 2H
            TOF = sqrt(a^3/mu)*((sinh(alpha0)-alpha0)+(sinh(beta0)-beta0));
        else
            fprintf('error in determining type of transfer orbit')
            return
        end

        a = a + 100;

    end

```

```

else
    fprintf('error in determining type of transfer orbit')
    return
end

end

```

### B. Lambert Arc Script for TA: 120 degrees and TOF: 3 days

This script calculates the necessary inputs for the Lambert Algorithm, and then calculates the total  $\Delta V$  required for the transfer between the Earth and Moon.

```

%% AAE 450 - Lambert Arc Analysis with TA of 120 deg and TOF of 3 days
% Author(s): Alex Maietta

mu_earth = 398600.4415; % gravitation parameter for the Earth (km^3/s^2)
mu_moon = 4902.8005821478; % gravitation parameter for the Moon (km^3/s^2)
R_earth = 6378.1363; % radius of the Earth (km)
R_moon = 1738.2; % radius of the Moon (km)
r_parking_moon = R_moon + 110; % radius of the circular parking orbit around the Moon
r_parking_earth = 600; % altitude of the circular parking orbit around the Earth (km)
a_moon = 384400; % semi-major axis of the Moon's orbit around Earth (km)

r1 = r_parking_earth+R_earth; % km - distance from Earth to circular parking orbit around Earth
r2 = a_moon; % km - distance from Earth to Moon
TA = 120; % transfer angle of the orbit

% we have a type 1 orbit since the TA is less than 180 degrees

c = sqrt(r1^2+r2^2-2*r1*r2*cosd(TA)); % cosine law to find chord of space triangle
s = 1/2*(r1+r2+c); % semi perimeter
amin = s/2; % min energy semi-major axis

% calculate alpha0 and beta0 for the min energy case to find the TOF of the
% min energy case (same for type 1 and 2)
alpha0 = 2*asin(sqrt(s/2/amin));
beta0 = 2*asin(sqrt((s-c)/2/amin));

TOFmin = sqrt(amin^3/mu_earth)*((alpha0-sin(alpha0))+(beta0-sin(beta0))); % TOF for the
min energy case

TOFpar = 1/3*sqrt(2/mu_earth)*(s^(3/2)-(s-c)^(3/2)); % use minus sign for type 1 orbit

```

```

TOF = 3600*24*3; % desired 3 day TOF

% comparing the TOF to the TOFpar we see that with the desired TOF we need
% an elliptical orbit that is a type A since the TOF is less than the
% TOFmin

% calculate the required semi-major axis of the transfer orbit
a = Lambert_Alg(amin,TOF,TOFmin,s,c,mu_earth,'1A');

% recalculate alpha0 and beta0 with the new 'a' value
alpha0 = 2*asin(sqrt(s/2/a));
beta0 = 2*asin(sqrt((s-c)/2/a));

% determine the appropriate quadrant for the alpha and beta values
% for a type 1A orbit we have
alpha = alpha0;
beta = beta0;

% calculate the two roots for the semi-latus rectum
p1 = 4*a*(s-r1)*(s-r2)/c^2 * (sin((alpha+beta)/2))^2;
p2 = 4*a*(s-r1)*(s-r2)/c^2 * (sin((alpha-beta)/2))^2;

% for a 1A orbit, the second foci is closer than in a 1B orbit so we have a
% smaller eccentricity and therefore a larger p value
p = p1;

% calculate the orbital characteristics for the transfer orbit
e = sqrt(1-p/a);
rp = a*(1-e);
ra = a*(1+e);
energy = -mu_earth/2/a;
vdep = sqrt(2*mu_earth/r1-mu_earth/a);
varr = sqrt(2*mu_earth/r2-mu_earth/a);
theta_dep = acosd((p/r1-1)/e);
theta_arr = acosd((p/r2-1)/e);
gamma_dep = acosd(sqrt(mu_earth*p)/r1/vdep);
gamma_arr = acosd(sqrt(mu_earth*p)/r2/varr);

v1 = sqrt(mu_earth/r1); % initial circular velocity of the spacecraft in the parking orbit
v2 = sqrt(mu_earth/r2); % final circular velocity of the spacecraft orbiting the moon wrt Earth

% calculate the deltv required for departure
deltav_dep = sqrt(v1^2+vdep^2-2*v1*vdep*cosd(gamma_dep));
% beta_dep = 180 - asind(vdep/deltav_dep*sind(gamma_dep));

```

```
% alpha_dep = 180 - beta_dep;

% express the deltv departure in vector format in the VNB frame
% deltv_dep_vector = deltv_dep * [cosd(alpha_dep) -sind(alpha_dep)];

% calculate the deltv required for arrival
deltv_arr = sqrt(v2^2+varr^2-2*v2*varr*cosd(gamma_arr));
% beta_arr = 180 - asind(v2/deltv_arr*sind(gamma_arr));
% alpha_arr = 180-beta_arr;

% express the deltv arrival in vector format in the VNB frame
% deltv_arr_vector = deltv_arr * [cosd(alpha_arr) sind(alpha_arr)];

total_deltav = deltv_dep + deltv_arr;

%% PLOTS

% Earth parking orbit
theta = 0:360;
r_temp = zeros(1,length(theta));
ehat1 = zeros(1,length(theta));
phat1 = zeros(1,length(theta));

a_temp = r1;
e_temp = 0;
p_temp = a_temp*(1-e_temp^2);

for i = 1:length(theta)
    r_temp(i) = p_temp/(1 + e_temp*cosd(theta(i)));
    ehat1(i) = r_temp(i) * cosd(theta(i));
    phat1(i) = r_temp(i) * sind(theta(i));
end

clear r_temp a_temp e_temp p_temp theta

% Lunar orbit
theta = 0:360;
r_temp = zeros(1,length(theta));
ehat2 = zeros(1,length(theta));
phat2 = zeros(1,length(theta));

a_temp = a_moon;
e_temp = 0;
p_temp = a_temp*(1-e_temp^2);
```

```

for i = 1:length(theta)
    r_temp(i) = p_temp/(1 + e_temp*cosd(theta(i)));
    ehat2(i) = r_temp(i) * cosd(theta(i));
    phat2(i) = r_temp(i) * sind(theta(i));
end

clear r_temp a_temp e_temp p_temp theta

% Transfer Arc
theta = 0:TA;
r_temp = zeros(1,length(theta));
ehat3 = zeros(1,length(theta));
phat3 = zeros(1,length(theta));

a_temp = a;
e_temp = e;
p_temp = a_temp*(1-e_temp^2);
t = theta_dep;

for i = 1:length(theta)
    r_temp(i) = p_temp/(1 + e_temp*cosd(theta(i)+t));
    ehat3(i) = r_temp(i) * cosd(theta(i));
    phat3(i) = r_temp(i) * sind(theta(i));
end

figure(1)
plot(ehat1,phat1,'k-',ehat2,phat2,'b-',ehat3,phat3,'r-',0,0,'ko',ehat2(TA+1),phat2(TA+1),'go')
title('Lambert Arc with TA 120 deg and TOF 3 days - Alex Maietta')
xlabel('e axis (km)')
ylabel('p axis (km)')
axis([-5*10^5 5*10^5 -5*10^5 5*10^5])
legend('Earth Parking Orbit (600 km alt)', 'Lunar Orbit', 'Lambert Transfer Trajectory', 'Earth', 'Moon')
grid on

```

### C. GMAT Moon to Earth Mission Sequence, Resource Tree, and Script

The GMAT script created to analyze the return trajectory from the Moon to the Earth is uploaded to the Github under the Mission Design folder and is labeled ‘PreliminaryMoon2Earth\_AlexMaietta’.

#### Mission Sequence GUI and code

The mission sequence involves two target sequences that are designed to produce the required  $\Delta V$ 's to return back to Earth. By implementing the ‘Vary’ commands, each component direction of the maneuver is perturbed slightly until the solution converges to the desired conditions set in the ‘Achieve’ commands.



Figure 6. Mission Sequence GUI for the Return Trajectory GMAT Script



```
BeginMissionSequence;
```

```
Target 'Target Earth' DC1 {SolveMode = Solve, ExitMode = SaveAndContinue, ShowProgressWindow = true};
```

```
Vary 'Vary EOI_1.V' DC1(EOI_1.Element1 = 0.7642600000000001, {Perturbation = 0.00001, Lower = -10e300, Upper = 10e300, MaxStep = 0.002, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
```

```
Vary 'Vary EOI_1.N' DC1(EOI_1.Element2 = -0.1117623868415196, {Perturbation = 0.00001, Lower = -10e300, Upper = 10e300, MaxStep = 0.002, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
```

```
Vary 'Vary EOI_1.B' DC1(EOI_1.Element3 = 0.0479, {Perturbation = 0.00001, Lower = -10e300, Upper = 10e300, MaxStep = 0.002, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
```

```
Maneuver 'Apply EOI_1' EOI_1(Satellite);
```

```
Propagate 'Prop to Periapsis' Propagator(Satellite) {Satellite.Periapsis};
```

```
Achieve 'Achieve INC' DC1(Satellite.EarthMJ2000Eq.INC = 28.5, {Tolerance = 0.1});
```

```
Achieve 'Achieve RMAG' DC1(Satellite.Earth.RMAG = 6978.1363, {Tolerance = 0.1});
```

```
EndTarget; % For targeter DC1
```

```
GMAT           'Calc'          DeltaV'          DeltaV_EOI_1          =
(EOI_1.Element1^2+EOI_1.Element2^2+EOI_1.Element3^2)^{(1/2)};
```

```
Report 'Report Initial EOI_1' firstTarget DeltaV_EOI_1 EOI_1.Element1 EOI_1.Element2
EOI_1.Element3;
```

```
Target 'Target Circular Orbit' DC1 {SolveMode = Solve, ExitMode = DiscardAndContinue, ShowProgressWindow = true};
```

```
Vary 'Vary EOI_2.V' DC1(EOI_2.Element1 = -2.9, {Perturbation = 0.0001, Lower = -10, Upper = 10, MaxStep = 0.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
```

```
Vary 'Vary EOI_2.N' DC1(EOI_2.Element2 = 0, {Perturbation = 0.0001, Lower = -10, Upper = 10, MaxStep = 0.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
```

```
Vary 'Vary EOI_2.B' DC1(EOI_2.Element3 = 0, {Perturbation = 0.0001, Lower = -10, Upper = 10, MaxStep = 0.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
```

```
Maneuver 'Apply EOI_2' EOI_2(Satellite);
```

```
Propagate 'Prop One Period' Propagator(Satellite) {Satellite.ElapsedSecs = Satellite.Earth.OrbitPeriod};
```

```
Achieve 'Achieve ECC' DC1(Satellite.Earth.ECC = 0, {Tolerance = 0.1});
```

```
EndTarget; % For targeter DC1
```

```
GMAT           'Calc'          DeltaV'          DeltaV_EOI_2          =
(EOI_2.Element1^2+EOI_2.Element2^2+EOI_2.Element3^2)^{(1/2)};
```

Report 'Report Final EOI\_2' secondTarget DeltaV\_EOI\_2 EOI\_2.Element1 EOI\_2.Element2 EOI\_2.Element3;

### Resource Tree

The resource tree is illustrated below outlining the parameters that are set for the preliminary GMAT analysis. There are two impulsive burns that GMAT calculates to enter into the desired Earth orbit. Also, the propagator includes the force model parameters where the solar and lunar gravity are added, along with solar radiation pressure forces. The boundary value solver is necessary for GMAT's target and vary commands in the mission sequence that allow specific parameters to be set and achieved

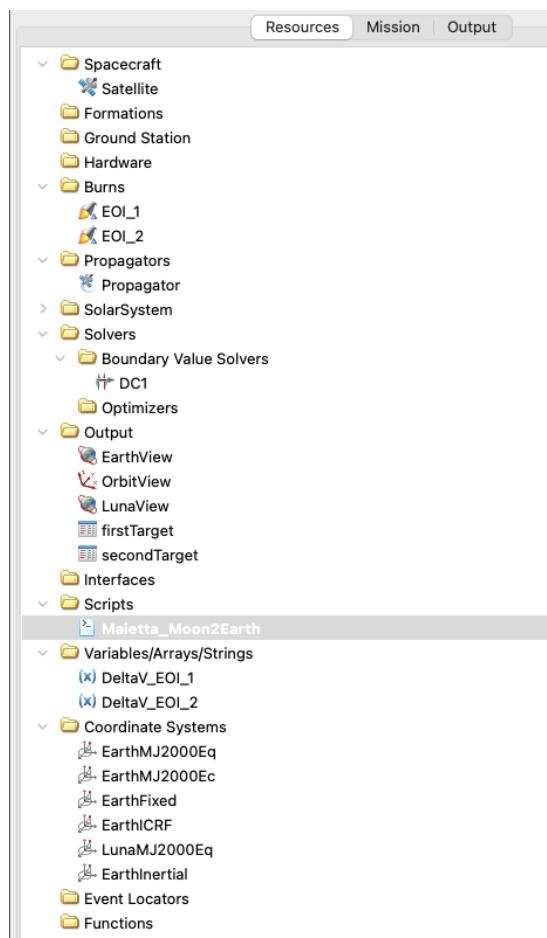


Figure 7. Resource Tree for Return Trajectory GMAT Script

GMAT Script

```
%General Mission Analysis Tool(GMAT) Script
%Created: 2021-02-10 14:57:22
%Author: Alex Maietta

%-----
%----- Spacecraft
%-----

Create Spacecraft Satellite;
GMAT Satellite.DateFormat = UTCGregorian;
GMAT Satellite.Epoch = '30 Sep 2022 00:00:00.000';
GMAT Satellite.CoordinateSystem = LunaMJ2000Eq;
GMAT Satellite.DisplayStateType = Keplerian;
GMAT Satellite.SMA = 2000;
GMAT Satellite.ECC = 0;
GMAT Satellite.INC = 89.9999999999488;
GMAT Satellite.RAAN = 0;
GMAT Satellite.AOP = 0;
GMAT Satellite.TA = 0;
GMAT Satellite.DryMass = 850;
GMAT Satellite.Cd = 2.2;
GMAT Satellite.Cr = 1.8;
GMAT Satellite.DragArea = 15;
GMAT Satellite.SRPArea = 1;
GMAT Satellite.SPADDragScaleFactor = 1;
GMAT Satellite.SPADSRPScaleFactor = 1;
GMAT Satellite.NAIFId = -10000001;
GMAT Satellite.NAIFIdReferenceFrame = -9000001;
GMAT Satellite.OrbitColor = Red;
GMAT Satellite.TargetColor = Teal;
GMAT Satellite.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0 0 0 0 ; 0 0 1e+70 0 0 0 ;
0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ; 0 0 0 0 0 1e+70 ];
GMAT Satellite.CdSigma = 1e+70;
GMAT Satellite.CrSigma = 1e+70;
GMAT Satellite.Id = 'SatId';
GMAT Satellite.Attitude = CoordinateSystemFixed;
GMAT Satellite.SPADSRPInterpolationMethod = Bilinear;
GMAT Satellite.SPADSRPScaleFactorSigma = 1e+70;
GMAT Satellite.SPADDragInterpolationMethod = Bilinear;
```

```
GMAT Satellite.SPADDragScaleFactorSigma = 1e+70;  
GMAT Satellite.ModelFile = 'aura.3ds';  
GMAT Satellite.ModelOffsetX = 0;  
GMAT Satellite.ModelOffsetY = 0;  
GMAT Satellite.ModelOffsetZ = 0;  
GMAT Satellite.ModelRotationX = 0;  
GMAT Satellite.ModelRotationY = 0;  
GMAT Satellite.ModelRotationZ = 0;  
GMAT Satellite.ModelScale = 1;  
GMAT Satellite.AttitudeDisplayStateType = 'Quaternion';  
GMAT Satellite.AttitudeRateDisplayStateType = 'AngularVelocity';  
GMAT Satellite.AttitudeCoordinateSystem = EarthMJ2000Eq;  
GMAT Satellite.EulerAngleSequence = '321';
```

```
%-----  
%----- ForceModels  
%-----
```

```
Create ForceModel Propagator_ForceModel;  
GMAT Propagator_ForceModel.CentralBody = Earth;  
GMAT Propagator_ForceModel.PrimaryBodies = {Earth};  
GMAT Propagator_ForceModel.PointMasses = {Luna, Sun};  
GMAT Propagator_ForceModel.Drag = None;  
GMAT Propagator_ForceModel.SRP = On;  
GMAT Propagator_ForceModel.RelativisticCorrection = Off;  
GMAT Propagator_ForceModel.ErrorControl = RSSStep;  
GMAT Propagator_ForceModel.GravityField.Earth.Degree = 4;  
GMAT Propagator_ForceModel.GravityField.Earth.Order = 4;  
GMAT Propagator_ForceModel.GravityField.Earth.StmLimit = 100;  
GMAT Propagator_ForceModel.GravityField.Earth.PotentialFile = 'JGM2.cof';  
GMAT Propagator_ForceModel.GravityField.Earth.TideModel = 'None';  
GMAT Propagator_ForceModel.SRP.Flux = 1367;  
GMAT Propagator_ForceModel.SRP.SRPModel = Spherical;  
GMAT Propagator_ForceModel.SRP.Nominal_Sun = 149597870.691;
```

```
%-----  
%----- Propagators  
%-----
```

```
Create Propagator Propagator;  
GMAT Propagator.FM = Propagator_ForceModel;  
GMAT Propagator.Type = RungeKutta89;
```

```
GMAT Propagator.InitialStepSize = 600;
GMAT Propagator.Accuracy = 1e-13;
GMAT Propagator.MinStep = 0;
GMAT Propagator.MaxStep = 600;
GMAT Propagator.MaxStepAttempts = 50;
GMAT Propagator.StopIfAccuracyIsViolated = true;

%-----
%----- Burns
%-----

Create ImpulsiveBurn EOI_1;
GMAT EOI_1.CoordinateSystem = Local;
GMAT EOI_1.Origin = Luna;
GMAT EOI_1.Axes = VNB;
GMAT EOI_1.Element1 = 0.6860000000000001;
GMAT EOI_1.Element2 = 0;
GMAT EOI_1.Element3 = -0.9408;
GMAT EOI_1.DecrementMass = false;
GMAT EOI_1.Isp = 300;
GMAT EOI_1.GravitationalAccel = 9.81;

Create ImpulsiveBurn EOI_2;
GMAT EOI_2.CoordinateSystem = Local;
GMAT EOI_2.Origin = Earth;
GMAT EOI_2.Axes = VNB;
GMAT EOI_2.Element1 = 0;
GMAT EOI_2.Element2 = 0;
GMAT EOI_2.Element3 = 0;
GMAT EOI_2.DecrementMass = false;
GMAT EOI_2.Isp = 300;
GMAT EOI_2.GravitationalAccel = 9.81;

%-----
%----- Coordinate Systems
%-----

Create CoordinateSystem LunaMJ2000Eq;
GMAT LunaMJ2000Eq.Origin = Luna;
GMAT LunaMJ2000Eq.Axes = MJ2000Eq;

Create CoordinateSystem EarthInertial;
GMAT EarthInertial.Origin = Earth;
GMAT EarthInertial.Axes = BodyInertial;
```

```
%-----
%----- DataInterfaces
%-----

Create ThrustSegment ThrustSegment1;
GMAT ThrustSegment1.ThrustScaleFactor = 1;
GMAT ThrustSegment1.ThrustScaleFactorSigma = 1e+70;
GMAT ThrustSegment1.ApplyThrustScaleToMassFlow = false;
GMAT ThrustSegment1.MassFlowScaleFactor = 1;

%-----
%----- Solvers
%-----

Create DifferentialCorrector DC1;
GMAT DC1.ShowProgress = true;
GMAT DC1.ReportStyle = Normal;
GMAT DC1.ReportFile = 'DifferentialCorrectorDC1.data';
GMAT DC1.MaximumIterations = 300;
GMAT DC1.DerivativeMethod = ForwardDifference;
GMAT DC1.Algorithm = NewtonRaphson;

%-----
%----- Subscribers
%-----

Create OrbitView EarthView;
GMAT EarthView.SolverIterations = Current;
GMAT EarthView.UpperLeft = [ 0.1090277777777778 0.0311111111111111 ];
GMAT EarthView.Size = [ 0.805555555555556 0.8744444444444445 ];
GMAT EarthView.RelativeZOrder = 680;
GMAT EarthView.Maximized = false;
GMAT EarthView.Add = {Satellite, Earth, Luna};
GMAT EarthView.CoordinateSystem = EarthMJ2000Eq;
GMAT EarthView.DrawObject = [ true true true ];
GMAT EarthView.DataCollectFrequency = 1;
GMAT EarthView.UpdatePlotFrequency = 50;
GMAT EarthView.NumPointsToRedraw = 0;
GMAT EarthView.ShowPlot = true;
GMAT EarthView.MaxPlotPoints = 20000;
GMAT EarthView.ShowLabels = true;
GMAT EarthView.ViewPointReference = Earth;
GMAT EarthView.ViewPointVector = [ 30000 0 0 ];
```

```
GMAT EarthView.ViewDirection = Earth;
GMAT EarthView.ViewScaleFactor = 1;
GMAT EarthView.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT EarthView.ViewUpAxis = Z;
GMAT EarthView.EclipticPlane = Off;
GMAT EarthView.XYPlane = On;
GMAT EarthView.WireFrame = Off;
GMAT EarthView.Axes = On;
GMAT EarthView.Grid = Off;
GMAT EarthView.SunLine = Off;
GMAT EarthView.UseInitialView = On;
GMAT EarthView.StarCount = 7000;
GMAT EarthView.EnableStars = On;
GMAT EarthView.EnableConstellations = Off;

Create OpenFramesInterface OrbitView;
GMAT OrbitView.SolverIterations = Current;
GMAT OrbitView.UpperLeft = [ -1.502777777777778 0.0522222222222223 ];
GMAT OrbitView.Size = [ 1.282638888888889 1.3466666666666667 ];
GMAT OrbitView.RelativeZOrder = 684;
GMAT OrbitView.Maximized = false;
GMAT OrbitView.Add = {Satellite, Earth, Luna};
GMAT OrbitView.View = {CoordinateSystemView1, EarthView1, LunaView1,
Crewed_SatView1};
GMAT OrbitView.CoordinateSystem = EarthMJ2000Eq;
GMAT OrbitView.DrawObject = [ true true true ];
GMAT OrbitView.DrawTrajectory = [ true true true ];
GMAT OrbitView.DrawAxes = [ false false false ];
GMAT OrbitView.DrawXYPlane = [ false false false ];
GMAT OrbitView.DrawLineWidth = [ 2 2 2 ];
GMAT OrbitView.DrawMarkerSize = [ 10 10 10 ];
GMAT OrbitView.DrawStringSize = [ 14 14 14 ];
GMAT OrbitView.Axes = On;
GMAT OrbitView.AxesLength = 1;
GMAT OrbitView.AxesLabels = On;
GMAT OrbitView.FrameLabel = Off;
GMAT OrbitView.XYPlane = On;
GMAT OrbitView.EclipticPlane = Off;
```

```
GMAT OrbitView.EnableStars = On;
GMAT OrbitView.StarCatalog = 'inp_StarsHYGv3.txt';
GMAT OrbitView.StarCount = 40000;
GMAT OrbitView.MinStarMag = -2;
GMAT OrbitView.MaxStarMag = 6;
GMAT OrbitView.MinStarPixels = 1;
GMAT OrbitView.MaxStarPixels = 10;
GMAT OrbitView.MinStarDimRatio = 0.5;
GMAT OrbitView.ShowPlot = true;
GMAT OrbitView.ShowToolbar = true;
GMAT OrbitView.SolverIterLastN = 1;
GMAT OrbitView.ShowVR = false;
GMAT OrbitView.PlaybackTimeScale = 3600;
GMAT OrbitView.MultisampleAntiAliasing = On;
GMAT OrbitView.MSAASamples = 2;
GMAT OrbitView.DrawFontPosition = {'Top-Right', 'Top-Right', 'Top-Right'};

Create OrbitView LunaView;
GMAT LunaView.SolverIterations = Current;
GMAT LunaView.UpperLeft = [ 0.02638888888888889 0.0333333333333333 ];
GMAT LunaView.Size = [ 0.764583333333333 0.893333333333333 ];
GMAT LunaView.RelativeZOrder = 675;
GMAT LunaView.Maximized = false;
GMAT LunaView.Add = {Satellite, Earth, Luna};
GMAT LunaView.CoordinateSystem = LunaMJ2000Eq;
GMAT LunaView.DrawObject = [ true true true ];
GMAT LunaView.DataCollectFrequency = 1;
GMAT LunaView.UpdatePlotFrequency = 50;
GMAT LunaView.NumPointsToRedraw = 0;
GMAT LunaView.ShowPlot = true;
GMAT LunaView.MaxPlotPoints = 20000;
GMAT LunaView.ShowLabels = true;
GMAT LunaView.ViewPointReference = Luna;
GMAT LunaView.ViewPointVector = [ 0 0 30000 ];
GMAT LunaView.ViewDirection = Satellite;
GMAT LunaView.ViewScaleFactor = 1;
GMAT LunaView.ViewUpCoordinateSystem = LunaMJ2000Eq;
GMAT LunaView.ViewUpAxis = Z;
GMAT LunaView.EclipticPlane = Off;
GMAT LunaView.XYPlane = Off;
GMAT LunaView.WireFrame = Off;
GMAT LunaView.Axes = On;
GMAT LunaView.Grid = Off;
GMAT LunaView.SunLine = Off;
```

```
GMAT LunaView.UseInitialView = On;
GMAT LunaView.StarCount = 7000;
GMAT LunaView.EnableStars = Off;
GMAT LunaView.EnableConstellations = Off;

Create ReportFile firstTarget;
GMAT firstTarget.SolverIterations = Current;
GMAT firstTarget.UpperLeft = [ 0 0 ];
GMAT firstTarget.Size = [ 0 0 ];
GMAT firstTarget.RelativeZOrder = 0;
GMAT firstTarget.Maximized = false;
GMAT firstTarget.Filename = 'firstTarget.txt';
GMAT firstTarget.Precision = 16;
GMAT firstTarget.WriteHeaders = true;
GMAT firstTarget.LeftJustify = On;
GMAT firstTarget.ZeroFill = Off;
GMAT firstTarget.FixedWidth = true;
GMAT firstTarget.Delimiter = ' ';
GMAT firstTarget.ColumnWidth = 23;
GMAT firstTarget.WriteReport = true;

Create ReportFile secondTarget;
GMAT secondTarget.SolverIterations = Current;
GMAT secondTarget.UpperLeft = [ 0 0 ];
GMAT secondTarget.Size = [ 0 0 ];
GMAT secondTarget.RelativeZOrder = 0;
GMAT secondTarget.Maximized = false;
GMAT secondTarget.Filename = 'secondTarget.txt';
GMAT secondTarget.Precision = 16;
GMAT secondTarget.WriteHeaders = true;
GMAT secondTarget.LeftJustify = On;
GMAT secondTarget.ZeroFill = Off;
GMAT secondTarget.FixedWidth = true;
GMAT secondTarget.Delimiter = ' ';
GMAT secondTarget.ColumnWidth = 23;
GMAT secondTarget.WriteReport = true;

%-----
%----- Arrays, Variables, Strings
%-----

Create Variable DeltaV_EOI_1 DeltaV_EOI_2;
```

```
%-----  
%----- User Objects  
%-----  
  
Create OpenFramesView CoordinateSystemView1;  
GMAT CoordinateSystemView1.ViewFrame = CoordinateSystem;  
GMAT CoordinateSystemView1.ViewTrajectory = Off;  
GMAT CoordinateSystemView1.InertialFrame = Off;  
GMAT CoordinateSystemView1.SetDefaultLocation = Off;  
GMAT CoordinateSystemView1.SetCurrentLocation = On;  
GMAT CoordinateSystemView1.CurrentEye = [ -218326.2702697742 -353455.4688939032 -  
135159.8190914017 ];  
GMAT CoordinateSystemView1.CurrentCenter = [ 1.746229827404022e-10 -  
2.91038304567337e-10 -3.201421350240707e-10 ];  
GMAT CoordinateSystemView1.CurrentUp = [ -0.1766190472587181 -0.2544897719519128  
0.9508105321867685 ];  
GMAT CoordinateSystemView1.FOVy = 45;  
  
Create OpenFramesView EarthView1;  
GMAT EarthView1.ViewFrame = Earth;  
GMAT EarthView1.ViewTrajectory = Off;  
GMAT EarthView1.InertialFrame = Off;  
GMAT EarthView1.SetDefaultLocation = Off;  
GMAT EarthView1.SetCurrentLocation = Off;  
GMAT EarthView1.FOVy = 45;  
  
Create OpenFramesView LunaView1;  
GMAT LunaView1.ViewFrame = Luna;  
GMAT LunaView1.ViewTrajectory = Off;  
GMAT LunaView1.InertialFrame = Off;  
GMAT LunaView1.SetDefaultLocation = Off;  
GMAT LunaView1.SetCurrentLocation = On;  
GMAT LunaView1.CurrentEye = [ 4370.947432655566 5034.789600001575 -  
3307.972616218983 ];  
GMAT LunaView1.CurrentCenter = [ -9.094947017729282e-13 -9.094947017729282e-13  
4.547473508864641e-13 ];  
GMAT LunaView1.CurrentUp = [ -0.009917626470163199 0.5550806090455309  
0.8317374334168451 ];  
GMAT LunaView1.FOVy = 45;  
  
Create OpenFramesView Crewed_SatView1;  
GMAT Crewed_SatView1.ViewFrame = Satellite;  
GMAT Crewed_SatView1.ViewTrajectory = Off;  
GMAT Crewed_SatView1.InertialFrame = Off;
```

```

GMAT Crewed_SatView1.SetDefaultLocation = Off;
GMAT Crewed_SatView1.SetCurrentLocation = Off;
GMAT Crewed_SatView1.FOVy = 45;
%-----
%----- Mission Sequence
%-----

BeginMissionSequence;

Target 'Target Earth' DC1 {SolveMode = Solve, ExitMode = SaveAndContinue,
ShowProgressWindow = true};
  Vary 'Vary EOI_1.V' DC1(EOI_1.Element1 = 0.7642600000000001, {Perturbation =
0.00001, Lower = -10e300, Upper = 10e300, MaxStep = 0.002, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
  Vary 'Vary EOI_1.N' DC1(EOI_1.Element2 = -0.1117623868415196, {Perturbation =
0.00001, Lower = -10e300, Upper = 10e300, MaxStep = 0.002, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
  Vary 'Vary EOI_1.B' DC1(EOI_1.Element3 = 0.0479, {Perturbation = 0.00001, Lower = -
10e300, Upper = 10e300, MaxStep = 0.002, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});
  Maneuver 'Apply EOI_1' EOI_1(Satellite);
  Propagate 'Prop to Periapsis' Propagator(Satellite) {Satellite.Periapsis};
  Achieve 'Achieve INC' DC1(Satellite.EarthMJ2000Eq.INC = 28.5, {Tolerance = 0.1});
  Achieve 'Achieve RMAG' DC1(Satellite.Earth.RMAG = 6978.1363, {Tolerance = 0.1});
EndTarget; % For targeter DC1

GMAT           'Calc          DeltaV'          DeltaV_EOI_1      =
(EOI_1.Element1^2+EOI_1.Element2^2+EOI_1.Element3^2)^(1/2);

Report 'Report Initial EOI_1' firstTarget DeltaV_EOI_1 EOI_1.Element1 EOI_1.Element2
EOI_1.Element3;

Target 'Target Circular Orbit' DC1 {SolveMode = Solve, ExitMode = DiscardAndContinue,
ShowProgressWindow = true};
  Vary 'Vary EOI_2.V' DC1(EOI_2.Element1 = -2.9, {Perturbation = 0.0001, Lower = -10,
Upper = 10, MaxStep = 0.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
  Vary 'Vary EOI_2.N' DC1(EOI_2.Element2 = 0, {Perturbation = 0.0001, Lower = -10, Upper
= 10, MaxStep = 0.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
  Vary 'Vary EOI_2.B' DC1(EOI_2.Element3 = 0, {Perturbation = 0.0001, Lower = -10, Upper
= 10, MaxStep = 0.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
  Maneuver 'Apply EOI_2' EOI_2(Satellite);
  Propagate 'Prop One Period' Propagator(Satellite) {Satellite.ElapsedSecs =
Satellite.Earth.OrbitPeriod};
  Achieve 'Achieve ECC' DC1(Satellite.Earth.ECC = 0, {Tolerance = 0.1});

```

```
EndTarget; % For targeter DC1
```

```
GMAT          'Calc          DeltaV'          DeltaV_EOI_2          =
(EOI_2.Element1^2+EOI_2.Element2^2+EOI_2.Element3^2)^(1/2);
```

```
Report 'Report Final EOI_2' secondTarget DeltaV_EOI_2 EOI_2.Element1 EOI_2.Element2
EOI_2.Element3;
```

#### D. Launch Date Analysis Spreadsheet for Year 1

Table 3. Launch and Arrival Dates and Corresponding Ephemeris Data

| <b>Launch Date</b> | <b>Arrival Date</b> | <b>Moon Range<br/>(km)</b> | <b>Alpha Arrival<br/>(deg)</b> | <b>Delta Arrival<br/>(deg)</b> |
|--------------------|---------------------|----------------------------|--------------------------------|--------------------------------|
| 10/8/21            | 10/12/21            | 3.71E+05                   | 101.9747                       | 26.21507                       |
| 11/5/21            | 11/9/21             | 3.69E+05                   | 113.19429                      | 25.79                          |
| 12/4/21            | 12/8/21             | 3.71E+05                   | 138.18927                      | 21.40214                       |
| 1/1/22             | 1/5/22              | 3.69E+05                   | 146.5312                       | 18.85285                       |
| 1/30/22            | 2/3/22              | 3.75E+05                   | 167.50788                      | 10.59056                       |
| 2/26/22            | 3/2/22              | 3.74E+05                   | 162.05104                      | 12.88633                       |
| 3/23/22            | 3/27/22             | 3.73E+05                   | 129.87582                      | 23.47827                       |
| 4/19/22            | 4/23/22             | 3.72E+05                   | 126.58626                      | 24.30186                       |
| 5/17/22            | 5/21/22             | 3.71E+05                   | 137.5719                       | 21.76723                       |
| 6/14/22            | 6/18/22             | 3.68E+05                   | 147.26301                      | 18.70372                       |
| 7/13/22            | 7/17/22             | 3.72E+00                   | 169.14443                      | 9.6017                         |

|         |         |          |           |         |
|---------|---------|----------|-----------|---------|
| 8/10/22 | 8/14/22 | 3.71E+05 | 176.62486 | 5.80118 |
| 9/7/22  | 9/11/22 | 3.74E+05 | 183.86061 | 2.02492 |

Table 4. Lambert Arc Orbital Characteristics using Ephemeris Data

| $a$ (km) | $e$      | $i$ (deg) | $\Omega$ (deg) | $\omega$ (deg) | $\theta^*$ (deg) |
|----------|----------|-----------|----------------|----------------|------------------|
| -4900.56 | 1.357628 | 90        | 41.02425       | 237.357        | -38.264          |
| -4742.27 | 1.368181 | 90        | 113.1943       | 249.5121       | -38.7221         |
| -4675.12 | 1.372851 | 90        | 138.1893       | 245.3236       | -38.9215         |
| -4731.27 | 1.368938 | 90        | 146.5312       | 242.6074       | -38.7546         |
| -4563.25 | 1.380905 | 90        | 167.5079       | 234.8512       | -39.2606         |
| -4587.47 | 1.379131 | 90        | 162.051        | 237.0728       | -39.1864         |
| -4625.6  | 1.376374 | 90        | 129.8758       | 247.5488       | -39.0705         |
| -4644.83 | 1.374998 | 90        | 126.5863       | 248.3143       | -39.0124         |
| -4689.92 | 1.371812 | 90        | 137.5719       | 245.6445       | -38.8773         |
| -4764.73 | 1.366646 | 90        | 147.263        | 242.3598       | -38.6561         |
| -4657.5  | 1.374097 | 90        | 169.1444       | 233.576        | -38.9743         |
| -4666.12 | 1.373487 | 90        | 176.6249       | 229.7496       | -38.9484         |
| -4605.68 | 1.377809 | 90        | 183.8606       | 226.1559       | -39.131          |

Table 5: Additional Lambert Arc Details

| <b>Transfer Angle<br/>(deg)</b> | <b>TOF<br/>(Days)</b> | <b>Transfer Type</b> | <b>Lunar Escape Delta V (km/s)</b> |
|---------------------------------|-----------------------|----------------------|------------------------------------|
| 175                             | 4                     | 1H                   | 1.14351946                         |
| 175                             | 4                     | 1H                   | 1.15694107                         |
| 175                             | 4                     | 1H                   | 1.16286525                         |
| 175                             | 4                     | 1H                   | 1.15790207                         |
| 175                             | 4                     | 1H                   | 1.1730609                          |
| 175                             | 4                     | 1H                   | 1.17081773                         |
| 175                             | 4                     | 1H                   | 1.16732781                         |
| 175                             | 4                     | 1H                   | 1.16558522                         |
| 175                             | 4                     | 1H                   | 1.16154777                         |
| 175                             | 4                     | 1H                   | 1.15499141                         |
| 175                             | 4                     | 1H                   | 1.16444452                         |
| 175                             | 4                     | 1H                   | 1.1636706                          |
| 175                             | 4                     | 1H                   | 1.16914498                         |

Table 6. GMAT Earth Orbit Insertion 1 and 2  $\Delta V$  Results

| <b>EOI 1 V (km/s)</b> | <b>EOI 1 N (km/s)</b> | <b>EOI 1 B (km/s)</b>    | <b>EOI 1 Total (km/s)</b> | <b>EOI 2 (km/s)</b> |
|-----------------------|-----------------------|--------------------------|---------------------------|---------------------|
| -0.031331             | 0.01146301            | 0.02118915<br>0.01465844 | 0.0395                    | 2.9                 |
| -0.020806             | 0.01112052            |                          | 0.0278                    | 2.9                 |
| -0.019661             | 0.01397166            | 0.01703834               | 0.0295                    | 2.9                 |

|           |            |            |        |     |
|-----------|------------|------------|--------|-----|
| -0.019286 | 0.0159865  | 0.01942117 | 0.0317 | 2.9 |
| -0.025586 | 0.00234745 | -0.0251112 | 0.0359 | 2.9 |
| -0.029934 | 0.00410387 | -0.0305755 | 0.0430 | 2.9 |
| -0.033097 | 0.02247476 | 0.00841746 | 0.0409 | 2.9 |
| -0.028406 | 0.01788775 | 0.00887955 | 0.0347 | 2.9 |
| -0.022885 | 0.01482423 | 0.01073514 | 0.0293 | 2.9 |
| -0.018241 | 0.01387809 | 0.01447299 | 0.0271 | 2.9 |
| -0.019958 | 0.00194215 | -0.0152924 | 0.0252 | 2.9 |
| -0.022354 | 0.00682308 | -0.0094654 | 0.0252 | 2.9 |
| -0.027975 | 0.0124213  | -0.0080643 | 0.0317 | 2.9 |

Table 7. Final  $\Delta V$  and TOF Results

| Actual TOF (days) | Total Delta V (km/s) |
|-------------------|----------------------|
| 3.032498897       | 4.0830               |
| 3.018394227       | 4.0847               |
| 2.989902824       | 4.0924               |
| 2.995899299       | 4.0896               |
| 2.999621634       | 4.1090               |
| 3.015662984       | 4.1138               |
| 3.021402355       | 4.1082               |
| 3.009226483       | 4.1003               |
| 3.00002451        | 4.0909               |

|             |        |
|-------------|--------|
| 2.997067011 | 4.0821 |
| 2.993440066 | 4.0897 |
| 2.999908976 | 4.0889 |
| 3.008849877 | 4.1008 |

## X. Appendix for Lunar Transfer Preliminary Analysis & Crewed Trajectory Simulation

### GMAT Script: Lunar Free Return with Optimization

```
%This script created by Daniel Gochenaur
%Method and initializations based on GMAT sample script:
"Ex_GivenEpochGoToTheMoon.script"

%-----
%----- Spacecraft
%-----

Create Spacecraft MoonSat;
GMAT MoonSat.DateFormat = UTCGregorian;
GMAT MoonSat.Epoch = '10 Oct 2024 01:00:00.000';
GMAT MoonSat.CoordinateSystem = EarthMJ2000Eq;
GMAT MoonSat.DisplayStateType = Keplerian;
GMAT MoonSat.SMA = 6978.1363;
GMAT MoonSat.ECC = 3.083634674369106e-24;
GMAT MoonSat.INC = 28.699999999999999;
GMAT MoonSat.RAAN = 360;
GMAT MoonSat.AOP = 0;
GMAT MoonSat.TA = 1.478779333471098e-06;
GMAT MoonSat.DryMass = 850;
GMAT MoonSat.Cd = 2.2;
GMAT MoonSat.Cr = 1.8;
GMAT MoonSat.DragArea = 15;
GMAT MoonSat.SRPArea = 20;
GMAT MoonSat.SPADDragScaleFactor = 1;
GMAT MoonSat.SPADSRPSCscaleFactor = 1;
GMAT MoonSat.NAIFId = -10001001;
GMAT MoonSat.NAIFIdReferenceFrame = -9001001;
GMAT MoonSat.OrbitColor = Red;
GMAT MoonSat.TargetColor = Teal;
GMAT MoonSat.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0
0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ; 0 0
0 0 1e+70 ];
GMAT MoonSat.CdSigma = 1e+70;
GMAT MoonSat.CrSigma = 1e+70;
GMAT MoonSat.Id = 'SatId';
GMAT MoonSat.Attitude = CoordinateSystemFixed;
GMAT MoonSat.SPADSRPInterpolationMethod = Bilinear;
GMAT MoonSat.SPADSRPSCscaleFactorSigma = 1e+70;
GMAT MoonSat.SPADDragInterpolationMethod = Bilinear;
GMAT MoonSat.SPADDragScaleFactorSigma = 1e+70;
GMAT MoonSat.ModelFile = 'aura.3ds';
GMAT MoonSat.ModelOffsetX = 0;
```

```

GMAT MoonSat.ModelOffsetY = 0;
GMAT MoonSat.ModelOffsetZ = 0;
GMAT MoonSat.ModelRotationX = 0;
GMAT MoonSat.ModelRotationY = 0;
GMAT MoonSat.ModelRotationZ = 0;
GMAT MoonSat.AttitudeDisplayStateType = 'Quaternion';
GMAT MoonSat.AttitudeRateDisplayStateType = 'AngularVelocity';
GMAT MoonSat.AttitudeCoordinateSystem = EarthMJ2000Eq;
GMAT MoonSat.EulerAngleSequence = '321';

Create Spacecraft InitSat;
%----- ForceModels
%-----

Create ForceModel NearMoonProp_ForceModel;
GMAT NearMoonProp_ForceModel.CentralBody = Luna;
GMAT NearMoonProp_ForceModel.PointMasses = {Sun, Earth, Luna};
GMAT NearMoonProp_ForceModel.Drag = None;
GMAT NearMoonProp_ForceModel.SRP = On;
GMAT NearMoonProp_ForceModel.RelativisticCorrection = Off;
GMAT NearMoonProp_ForceModel.ErrorControl = RSSStep;
GMAT NearMoonProp_ForceModel.SRP.Flux = 1367;
GMAT NearMoonProp_ForceModel.SRP.SRPMODEL = Spherical;
GMAT NearMoonProp_ForceModel.SRP.Nominal_Sun = 149597870.691;

Create ForceModel NearEarthProp_ForceModel;
GMAT NearEarthProp_ForceModel.CentralBody = Earth;
GMAT NearEarthProp_ForceModel.PointMasses = {Earth, Sun, Luna};
GMAT NearEarthProp_ForceModel.Drag = None;
GMAT NearEarthProp_ForceModel.SRP = On;
GMAT NearEarthProp_ForceModel.RelativisticCorrection = Off;
GMAT NearEarthProp_ForceModel.ErrorControl = RSSStep;
GMAT NearEarthProp_ForceModel.SRP.Flux = 1367;
GMAT NearEarthProp_ForceModel.SRP.SRPMODEL = Spherical;
GMAT NearEarthProp_ForceModel.SRP.Nominal_Sun = 149597870.691;

Create ForceModel EarthPointMass_ForceModel;
GMAT EarthPointMass_ForceModel.CentralBody = Earth;
GMAT EarthPointMass_ForceModel.PointMasses = {Earth};
GMAT EarthPointMass_ForceModel.Drag = None;
GMAT EarthPointMass_ForceModel.SRP = Off;
GMAT EarthPointMass_ForceModel.RelativisticCorrection = Off;
GMAT EarthPointMass_ForceModel.ErrorControl = RSSStep;

%----- Propagators
%-----

Create Propagator NearMoonProp;

```

```
GMAT NearMoonProp.FM = NearMoonProp_ForceModel;
GMAT NearMoonProp.Type = RungeKutta89;
GMAT NearMoonProp.InitialStepSize = 60;
GMAT NearMoonProp.Accuracy = 9.99999999999999e-12;
GMAT NearMoonProp.MinStep = 0.001;
GMAT NearMoonProp.MaxStep = 86400;
GMAT NearMoonProp.MaxStepAttempts = 50;
GMAT NearMoonProp.StopIfAccuracyIsViolated = true;

Create Propagator NearEarthProp;
GMAT NearEarthProp.FM = NearEarthProp_ForceModel;
GMAT NearEarthProp.Type = RungeKutta89;
GMAT NearEarthProp.InitialStepSize = 60;
GMAT NearEarthProp.Accuracy = 9.99999999999999e-12;
GMAT NearEarthProp.MinStep = 0.001;
GMAT NearEarthProp.MaxStep = 160000;
GMAT NearEarthProp.MaxStepAttempts = 50;
GMAT NearEarthProp.StopIfAccuracyIsViolated = true;

Create Propagator EarthPointMass;
GMAT EarthPointMass.FM = EarthPointMass_ForceModel;
GMAT EarthPointMass.Type = RungeKutta89;
GMAT EarthPointMass.InitialStepSize = 60;
GMAT EarthPointMass.Accuracy = 9.99999999999999e-12;
GMAT EarthPointMass.MinStep = 0.001;
GMAT EarthPointMass.MaxStep = 2700;
GMAT EarthPointMass.MaxStepAttempts = 50;
GMAT EarthPointMass.StopIfAccuracyIsViolated = true;

%----- Burns
%-----

Create ImpulsiveBurn TOI;
GMAT TOI.CoordinateSystem = Local;
GMAT TOI.Origin = Earth;
GMAT TOI.Axes = VNB;
GMAT TOI.Element1 = 3.047;
GMAT TOI.Element2 = 0;
GMAT TOI.Element3 = 0.048;
GMAT TOI.DecrementMass = false;
GMAT TOI.Isp = 300;
GMAT TOI.GravitationalAccel = 9.81;

Create ImpulsiveBurn LOI;
GMAT LOI.CoordinateSystem = Local;
GMAT LOI.Origin = Luna;
GMAT LOI.Axes = VNB;
GMAT LOI.Element1 = -0.5;
GMAT LOI.Element2 = 0;
```

```
GMAT LOI.Element3 = 0;
GMAT LOI.DecrementMass = false;
GMAT LOI.Isp = 300;
GMAT LOI.GravitationalAccel = 9.81;

Create ImpulsiveBurn FinalBurn;
GMAT FinalBurn.CoordinateSystem = Local;
GMAT FinalBurn.Origin = Earth;
GMAT FinalBurn.Axes = VNB;
GMAT FinalBurn.Element1 = -0.77323;
GMAT FinalBurn.Element2 = 0.35998;
GMAT FinalBurn.Element3 = 0.00998;
GMAT FinalBurn.DecrementMass = false;
GMAT FinalBurn.Isp = 300;
GMAT FinalBurn.GravitationalAccel = 9.81;

%-----
%----- Coordinate Systems
%-----

Create CoordinateSystem EarthMoonRot;
GMAT EarthMoonRot.Origin = Earth;
GMAT EarthMoonRot.Axes = ObjectReferenced;
GMAT EarthMoonRot.XAxis = R;
GMAT EarthMoonRot.ZAxis = N;
GMAT EarthMoonRot.Primary = Earth;
GMAT EarthMoonRot.Secondary = Luna;

Create CoordinateSystem MoonInertial;
GMAT MoonInertial.Origin = Luna;
GMAT MoonInertial.Axes = BodyInertial;

%-----
%----- Solvers
%-----

Create DifferentialCorrector DC1;
GMAT DC1.ShowProgress = true;
GMAT DC1.ReportStyle = Normal;
GMAT DC1.ReportFile = 'DifferentialCorrectorDC1.data';
GMAT DC1.MaximumIterations = 80;
GMAT DC1.DerivativeMethod = ForwardDifference;
GMAT DC1.Algorithm = NewtonRaphson;

Create FminconOptimizer NLP;
GMAT NLP.MaximumIterations = 50;

%-----
%----- Subscribers
%-----
```

```

Create OrbitView EarthMoonRotView;
GMAT EarthMoonRotView.SolverIterations = Current;
GMAT EarthMoonRotView.UpperLeft      = [    0.05294506949040371
0.003603603603603604 ];
GMAT EarthMoonRotView.Size         = [    0.5433487756452681
1.243243243243243 ];
GMAT EarthMoonRotView.RelativeZOrder = 23;
GMAT EarthMoonRotView.Maximized = false;
GMAT EarthMoonRotView.Add = {MoonSat, Earth, Luna};
GMAT EarthMoonRotView.CoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView.DrawObject = [ true true true ];
GMAT EarthMoonRotView.DataCollectFrequency = 1;
GMAT EarthMoonRotView.UpdatePlotFrequency = 50;
GMAT EarthMoonRotView.NumPointsToRedraw = 0;
GMAT EarthMoonRotView.ShowPlot = true;
GMAT EarthMoonRotView.MaxPlotPoints = 20000;
GMAT EarthMoonRotView>ShowLabels = true;
GMAT EarthMoonRotView.ViewPointReference = Earth;
GMAT EarthMoonRotView.ViewPointVector = [ 10000 0 30000 ];
GMAT EarthMoonRotView.ViewDirection = Earth;
GMAT EarthMoonRotView.ViewScaleFactor = 40;
GMAT EarthMoonRotView.ViewUpCoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView.ViewUpAxis = -X;
GMAT EarthMoonRotView.EclipticPlane = Off;
GMAT EarthMoonRotView.XYPlane = Off;
GMAT EarthMoonRotView.WireFrame = Off;
GMAT EarthMoonRotView.Axes = Off;
GMAT EarthMoonRotView.Grid = Off;
GMAT EarthMoonRotView.SunLine = Off;
GMAT EarthMoonRotView.UseInitialView = On;
GMAT EarthMoonRotView.StarCount = 7000;
GMAT EarthMoonRotView.EnableStars = On;
GMAT EarthMoonRotView.EnableConstellations = Off;

Create OrbitView EarthMoonRotView2;
GMAT EarthMoonRotView2.SolverIterations = Current;
GMAT EarthMoonRotView2.UpperLeft      = [    0.1403044341495698
0.003603603603603604 ];
GMAT EarthMoonRotView2.Size         = [    0.456651224354732
1.243243243243243 ];
GMAT EarthMoonRotView2.RelativeZOrder = 15;
GMAT EarthMoonRotView2.Maximized = false;
GMAT EarthMoonRotView2.Add = {MoonSat, Earth, Luna};
GMAT EarthMoonRotView2.CoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView2.DrawObject = [ true true true ];
GMAT EarthMoonRotView2.DataCollectFrequency = 1;
GMAT EarthMoonRotView2.UpdatePlotFrequency = 50;
GMAT EarthMoonRotView2.NumPointsToRedraw = 0;
GMAT EarthMoonRotView2.ShowPlot = true;

```

```

GMAT EarthMoonRotView2.MaxPlotPoints = 20000;
GMAT EarthMoonRotView2.ShowLabels = true;
GMAT EarthMoonRotView2.ViewPointReference = Earth;
GMAT EarthMoonRotView2.ViewPointVector = [ 10000 0 30000 ];
GMAT EarthMoonRotView2.ViewDirection = Earth;
GMAT EarthMoonRotView2.ViewScaleFactor = 40;
GMAT EarthMoonRotView2.ViewUpCoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView2.ViewUpAxis = -X;
GMAT EarthMoonRotView2.EclipticPlane = Off;
GMAT EarthMoonRotView2.XYPlane = Off;
GMAT EarthMoonRotView2.WireFrame = Off;
GMAT EarthMoonRotView2.Axes = Off;
GMAT EarthMoonRotView2.Grid = Off;
GMAT EarthMoonRotView2.SunLine = Off;
GMAT EarthMoonRotView2.UseInitialView = On;
GMAT EarthMoonRotView2.StarCount = 7000;
GMAT EarthMoonRotView2.EnableStars = On;
GMAT EarthMoonRotView2.EnableConstellations = Off;

Create OrbitView MoonInertialView;
GMAT MoonInertialView.SolverIterations = Current;
GMAT MoonInertialView.UpperLeft      = [ 0.3170086035737922
0.1153153153153153 ];
GMAT MoonInertialView.Size         = [ 1.101257445400397
1.052252252252252 ];
GMAT MoonInertialView.RelativeZOrder = 10;
GMAT MoonInertialView.Maximized = false;
GMAT MoonInertialView.Add = {MoonSat, Luna, Earth};
GMAT MoonInertialView.CoordinateSystem = MoonInertial;
GMAT MoonInertialView.DrawObject = [ true true true ];
GMAT MoonInertialView.DataCollectFrequency = 1;
GMAT MoonInertialView.UpdatePlotFrequency = 50;
GMAT MoonInertialView.NumPointsToRedraw = 150;
GMAT MoonInertialView>ShowPlot = true;
GMAT MoonInertialView.MaxPlotPoints = 20000;
GMAT MoonInertialView>ShowLabels = true;
GMAT MoonInertialView.ViewPointReference = Luna;
GMAT MoonInertialView.ViewPointVector = [ 20000 20000 20000 ];
GMAT MoonInertialView.ViewDirection = Luna;
GMAT MoonInertialView.ViewScaleFactor = 1.5;
GMAT MoonInertialView.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT MoonInertialView.ViewUpAxis = Z;
GMAT MoonInertialView.EclipticPlane = Off;
GMAT MoonInertialView.XYPlane = Off;
GMAT MoonInertialView.WireFrame = Off;
GMAT MoonInertialView.Axes = Off;
GMAT MoonInertialView.Grid = Off;
GMAT MoonInertialView.SunLine = Off;
GMAT MoonInertialView.UseInitialView = On;
GMAT MoonInertialView.StarCount = 7000;

```

```

GMAT MoonInertialView.EnableStars = On;
GMAT MoonInertialView.EnableConstellations = Off;

Create OrbitView EarthInertialView;
GMAT EarthInertialView.SolverIterations = Current;
GMAT EarthInertialView.UpperLeft = [ 0.6042356055592323 0 ];
GMAT EarthInertialView.Size      =      [      0.6419589675711449
1.277477477477478 ];
GMAT EarthInertialView.RelativeZOrder = 5;
GMAT EarthInertialView.Maximized = false;
GMAT EarthInertialView.Add = {MoonSat, Earth, Luna};
GMAT EarthInertialView.CoordinateSystem = EarthMJ2000Eq;
GMAT EarthInertialView.DrawObject = [ true true true ];
GMAT EarthInertialView.DataCollectFrequency = 1;
GMAT EarthInertialView.UpdatePlotFrequency = 50;
GMAT EarthInertialView.NumPointsToRedraw = 0;
GMAT EarthInertialView.ShowPlot = true;
GMAT EarthInertialView.MaxPlotPoints = 20000;
GMAT EarthInertialView.ShowLabels = true;
GMAT EarthInertialView.ViewPointReference = Earth;
GMAT EarthInertialView.ViewPointVector = [ 0 0 30000 ];
GMAT EarthInertialView.ViewDirection = Earth;
GMAT EarthInertialView.ViewScaleFactor = 45;
GMAT EarthInertialView.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT EarthInertialView.ViewUpAxis = Z;
GMAT EarthInertialView.EclipticPlane = Off;
GMAT EarthInertialView.XYPlane = Off;
GMAT EarthInertialView.WireFrame = Off;
GMAT EarthInertialView.Axes = Off;
GMAT EarthInertialView.Grid = Off;
GMAT EarthInertialView.SunLine = Off;
GMAT EarthInertialView.UseInitialView = On;
GMAT EarthInertialView.StarCount = 7000;
GMAT EarthInertialView.EnableStars = On;
GMAT EarthInertialView.EnableConstellations = Off;

Create ReportFile ReportFile1;
GMAT ReportFile1.SolverIterations = Current;
GMAT ReportFile1.UpperLeft = [ 0.02352941176470588 0.2275 ];
GMAT ReportFile1.Size = [ 0.9 0.78875 ];
GMAT ReportFile1.RelativeZOrder = 33;
GMAT ReportFile1.Maximized = false;
GMAT ReportFile1.Filename = 'ReportFile1.txt';
GMAT ReportFile1.Precision = 16;

GMAT ReportFile1.WriteHeader = true;
GMAT ReportFile1.LeftJustify = On;
GMAT ReportFile1.ZeroFill = Off;
GMAT ReportFile1.FixedWidth = true;
GMAT ReportFile1.Delimiter = ' ';

```

```

GMAT ReportFile1.ColumnWidth = 23;
GMAT ReportFile1.WriteReport = true;

%-----
%----- Arrays, Variables, Strings
%-----

Create Variable RAAN AOP DV DV1 DV2 DV3, bError, incError,
eccError, radPerError, ...
burnLoc, burnLocError;

GMAT ReportFile1.Add = {MoonSat.ElapsedDays,
MoonSat.Earth.RadPer, MoonSat.Luna.RadPer, ...
MoonSat.MoonInertial.BVectorAngle, MoonSat.MoonInertial.INC,
MoonSat.Luna.RMAG, ...
MoonSat.Earth.RMAG, MoonSat.MoonInertial.INC, TOI.Element1,
TOI.Element2, ...
TOI.Element3, LOI.Element1, LOI.Element2, LOI.Element3,
FinalBurn.Element1, ...
FinalBurn.Element2, FinalBurn.Element3, DV1, DV2, DV3, DV,
FinalBurn.Element1, ...
FinalBurn.Element2, FinalBurn.Element3};

%-----
%----- Mission Sequence
%-----

BeginMissionSequence;

GMAT InitSat = MoonSat;
Toggle 'Turn Off Selected Plots' MoonInertialView
EarthInertialView EarthMoonRotView2 Off;

%% Calculate Coarse LOI Burn

Target 'Coarse Lunar Target' DC1 {SolveMode = Solve, ExitMode =
SaveAndContinue, ShowProgressWindow = true};

Vary 'Vary RAAN' DC1(MoonSat.RAAN = 45.1, {Perturbation =
.00001, Lower = -9.99999e300, Upper = 9.99999e300, MaxStep = 5,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
Vary 'Vary AOP' DC1(MoonSat.AOP = 2.5, {Perturbation = .00001,
Lower = -9.99999e300, Upper = 9.99999e300, MaxStep = 5,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Maneuver 'Apply TOI' TOI(MoonSat);

GMAT 'Save RAAN' RAAN = MoonSat.RAAN;
GMAT 'Save AOP' AOP = MoonSat.AOP;

```

```

    Propagate 'Prop To Moon' EarthPointMass(MoonSat)
{MoonSat.Earth.Apoapsis, MoonSat.ElapsedDays = 3.24};

    Achieve 'RA = 0' DC1(MoonSat.EarthMoonRot.RA = 0, {Tolerance =
1});
    Achieve 'DEC = 0' DC1(MoonSat.EarthMoonRot.DEC = 0, {Tolerance =
1});

EndTarget;

%% Calculate Refined LOI Burn

GMAT MoonSat = InitSat;
GMAT MoonSat.OrbitColor = 'Yellow';
Toggle MoonInertialView EarthInertialView EarthMoonRotView2 On;

Target 'Fine Lunar Target' DC1 {SolveMode = Solve, ExitMode =
DiscardAndContinue, ShowProgressWindow = true};

    Vary 'Vary RAAN' DC1(MoonSat.RAAN = RAAN, {Perturbation =
.000001, Lower = -9.99999e300, Upper = 9.99999e300, MaxStep =
2, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

    Vary 'Vary AOP' DC1(MoonSat.AOP = AOP, {Perturbation = .000001,
Lower = -9.99999e300, Upper = 9.99999e300, MaxStep = 2,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

    Vary 'Vary TOI' DC1(TOI.Element1 = TOI.Element1, {Perturbation =
.0000001, Lower = -9.99999e300, Upper = 9.99999e300, MaxStep =
.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

    Vary 'Vary TOI' DC1(TOI.Element3 = TOI.Element3, {Perturbation =
.0000001, Lower = -9.99999e300, Upper = 9.99999e300, MaxStep =
.01, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Maneuver 'Apply TOI' TOI(MoonSat);

GMAT 'Save RAAN' RAAN = MoonSat.RAAN;
GMAT 'Save AOP' AOP = MoonSat.AOP;

Propagate 'Prop To Moon' NearEarthProp(MoonSat)
{MoonSat.Luna.Periapsis};

Achieve 'Achieve RadPer' DC1(MoonSat.Luna.RMAG = 3535.8,
{Tolerance = 0.1});

Propagate 'Prop Distance' NearMoonProp(MoonSat)
{MoonSat.Luna.RMAG = 66000}; %Edge of Lunar Sphere of Influence

```

```

        Achieve 'Achieve Return' DC1(MoonSat.Earth.RadPer = 6278,
{Tolerance = 10});

EndTarget;

Propagate 'Prop      3.24      days'     NearEarthProp(MoonSat)
{MoonSat.Earth.RMAG = 6900};

GMAT MoonSat = InitSat;

Maneuver 'Apply TOI' TOI(MoonSat);

GMAT MoonSat.RAAN = RAAN;
GMAT MoonSat.AOP = AOP;
GMAT MoonSat.OrbitColor = 'Cyan';

%% Calculate Refined FRD and Circularization Burns

Optimize NLP {SolveMode = Solve, ExitMode = DiscardAndContinue};

        Vary 'Vary LBurnPoint' NLP(burnLoc = MoonSat.Luna.RMAG,
{Perturbation = .00001, Lower = 20000, Upper = 90000, MaxStep = 20000, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

        Propagate 'Prop      To      Moon'     NearEarthProp(MoonSat)
{MoonSat.Luna.RMAG = burnLoc};

        Vary 'Vary LOI' NLP(LOI.Element1 = LOI.Element1, {Perturbation = .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

        Vary 'Vary LOI' NLP(LOI.Element2 = LOI.Element2, {Perturbation = .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

        Vary 'Vary LOI' NLP(LOI.Element3 = LOI.Element3, {Perturbation = .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

        Maneuver 'Apply LOI' LOI(MoonSat);

        Propagate 'Prop      Closer'     NearMoonProp(MoonSat)
{MoonSat.Luna.Periapsis};

        Vary 'Vary FB' NLP(FinalBurn.Element1 = FinalBurn.Element1,
{Perturbation = .00001, Lower = -9.999999e300, Upper = 9.999999e300, MaxStep = .3, AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

```

```

    Vary 'Vary FB' NLP(FinalBurn.Element2 = FinalBurn.Element2,
{Perturbation = .00001, Lower = -9.999999e300, Upper =
9.999999e300, MaxStep = .3, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});

    Vary 'Vary FB' NLP(FinalBurn.Element3 = FinalBurn.Element3,
{Perturbation = .00001, Lower = -9.999999e300, Upper =
9.999999e300, MaxStep = .3, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});

    Maneuver 'Apply Final Burn' FinalBurn(MoonSat);

    incError = MoonSat.MoonInertial.INC - 90;
    eccError = MoonSat.Luna.ECC - 0;
    radPerError = MoonSat.Luna.RadPer - 1838;

    GMAT DV1 = sqrt(TOI.Element1^2 + TOI.Element2^2 +
TOI.Element3^2)
    GMAT DV2 = sqrt(LOI.Element1^2 + LOI.Element2^2 +
LOI.Element3^2)
    GMAT DV3 = sqrt(FinalBurn.Element1^2 + FinalBurn.Element2^2 +
FinalBurn.Element3^2)
    GMAT DV = DV1 + DV2 + DV3;

    Propagate 'Prop 0.5 Day' NearMoonProp(MoonSat)
{MoonSat.ElapsedDays = 0.5};

    NonlinearConstraint NLP(incError <= 5);
    NonlinearConstraint NLP(incError >= -5);

    NonlinearConstraint NLP(eccError <= 0.05);
    NonlinearConstraint NLP(eccError >= 0);

    NonlinearConstraint NLP(radPerError <= 50);
    NonlinearConstraint NLP(radPerError >= -50);

    Minimize NLP(DV);

EndOptimize;

```

### GMAT Script: LLO Return with Optimization

```

%This script created by Daniel Gochenaur
%Method and initializations based on GMAT sample script:
"Ex_GivenEpochGoToTheMoon.script"

%-----

```

```
%----- Spacecraft
%-----



Create Spacecraft MoonSat;
GMAT MoonSat.DateFormat = UTCGregorian;
GMAT MoonSat.Epoch = '05 Sep 2025 01:00:00.000';
GMAT MoonSat.CoordinateSystem = MoonInertial;
GMAT MoonSat.DisplayStateType = Keplerian;
GMAT MoonSat.SMA = 1838;
GMAT MoonSat.ECC = 0.0;
GMAT MoonSat.INC = 90;
GMAT MoonSat.RAAN = 17.85;
GMAT MoonSat.AOP = 90;
GMAT MoonSat.TA = -10;
GMAT MoonSat.DryMass = 850;
GMAT MoonSat.Cd = 2.2;
GMAT MoonSat.Cr = 1.8;
GMAT MoonSat.DragArea = 15;
GMAT MoonSat.SRPArea = 20;
GMAT MoonSat.SPADDragScaleFactor = 1;
GMAT MoonSat.SPADSRPScaleFactor = 1;
GMAT MoonSat.NAIFId = -10001001;
GMAT MoonSat.NAIFIdReferenceFrame = -9001001;
GMAT MoonSat.OrbitColor = Red;
GMAT MoonSat.TargetColor = Teal;
GMAT MoonSat.OrbitErrorCovariance = [ 1e+70 0 0 0 0 0 ; 0 1e+70 0
0 0 0 ; 0 0 1e+70 0 0 0 ; 0 0 0 1e+70 0 0 ; 0 0 0 0 1e+70 0 ; 0 0
0 0 1e+70 ];
GMAT MoonSat.CdSigma = 1e+70;
GMAT MoonSat.CrSigma = 1e+70;
GMAT MoonSat.Id = 'SatId';
GMAT MoonSat.Attitude = CoordinateSystemFixed;
GMAT MoonSat.SPADSRPInterpolationMethod = Bilinear;
GMAT MoonSat.SPADSRPScaleFactorSigma = 1e+70;
GMAT MoonSat.SPADDragInterpolationMethod = Bilinear;
GMAT MoonSat.SPADDragScaleFactorSigma = 1e+70;
GMAT MoonSat.ModelFile = 'aura.3ds';
GMAT MoonSat.ModelOffsetX = 0;
GMAT MoonSat.ModelOffsetY = 0;
GMAT MoonSat.ModelOffsetZ = 0;
GMAT MoonSat.ModelRotationX = 0;
GMAT MoonSat.ModelRotationY = 0;
GMAT MoonSat.ModelRotationZ = 0;
GMAT MoonSat.ModelScale = 1;
GMAT MoonSat.AttitudeDisplayStateType = 'Quaternion';
GMAT MoonSat.AttitudeRateDisplayStateType = 'AngularVelocity';
GMAT MoonSat.AttitudeCoordinateSystem = MoonInertial;
GMAT MoonSat.EulerAngleSequence = '321';

Create Spacecraft InitSat;
```

```
%-----  
%----- ForceModels  
%-----  
  
Create ForceModel NearMoonProp_ForceModel;  
GMAT NearMoonProp_ForceModel.CentralBody = Luna;  
GMAT NearMoonProp_ForceModel.PointMasses = {Sun, Earth, Luna};  
GMAT NearMoonProp_ForceModel.Drag = None;  
GMAT NearMoonProp_ForceModel.SRP = On;  
GMAT NearMoonProp_ForceModel.RelativisticCorrection = Off;  
GMAT NearMoonProp_ForceModel.ErrorControl = RSSStep;  
GMAT NearMoonProp_ForceModel.SRP.Flux = 1367;  
GMAT NearMoonProp_ForceModel.SRP.SRPMODEL = Spherical;  
GMAT NearMoonProp_ForceModel.SRP.Nominal_Sun = 149597870.691;  
  
Create ForceModel NearEarthProp_ForceModel;  
GMAT NearEarthProp_ForceModel.CentralBody = Earth;  
GMAT NearEarthProp_ForceModel.PointMasses = {Earth, Sun, Luna};  
GMAT NearEarthProp_ForceModel.Drag = None;  
GMAT NearEarthProp_ForceModel.SRP = On;  
GMAT NearEarthProp_ForceModel.RelativisticCorrection = Off;  
GMAT NearEarthProp_ForceModel.ErrorControl = RSSStep;  
GMAT NearEarthProp_ForceModel.SRP.Flux = 1367;  
GMAT NearEarthProp_ForceModel.SRP.SRPMODEL = Spherical;  
GMAT NearEarthProp_ForceModel.SRP.Nominal_Sun = 149597870.691;  
  
Create ForceModel EarthPointMass_ForceModel;  
GMAT EarthPointMass_ForceModel.CentralBody = Earth;  
GMAT EarthPointMass_ForceModel.PointMasses = {Earth};  
GMAT EarthPointMass_ForceModel.Drag = None;  
GMAT EarthPointMass_ForceModel.SRP = Off;  
GMAT EarthPointMass_ForceModel.RelativisticCorrection = Off;  
GMAT EarthPointMass_ForceModel.ErrorControl = RSSStep;  
  
Create ForceModel MoonPointMass_ForceModel;  
GMAT MoonPointMass_ForceModel.CentralBody = Luna;  
GMAT MoonPointMass_ForceModel.PointMasses = {Luna};  
GMAT MoonPointMass_ForceModel.Drag = None;  
GMAT MoonPointMass_ForceModel.SRP = Off;  
GMAT MoonPointMass_ForceModel.RelativisticCorrection = Off;  
GMAT MoonPointMass_ForceModel.ErrorControl = RSSStep;  
  
%-----  
%----- Propagators  
%-----  
  
Create Propagator MoonPointMass;  
GMAT MoonPointMass.FM = MoonPointMass_ForceModel;  
GMAT MoonPointMass.Type = RungeKutta89;
```

```
GMAT MoonPointMass.InitialStepSize = 60;
GMAT MoonPointMass.Accuracy = 9.99999999999999e-12;
GMAT MoonPointMass.MinStep = 0.001;
GMAT MoonPointMass.MaxStep = 86400;
GMAT MoonPointMass.MaxStepAttempts = 50;
GMAT MoonPointMass.StopIfAccuracyIsViolated = true;

Create Propagator NearMoonProp;
GMAT NearMoonProp.FM = NearMoonProp_ForceModel;
GMAT NearMoonProp.Type = RungeKutta89;
GMAT NearMoonProp.InitialStepSize = 60;
GMAT NearMoonProp.Accuracy = 9.99999999999999e-12;
GMAT NearMoonProp.MinStep = 0.001;
GMAT NearMoonProp.MaxStep = 86400;
GMAT NearMoonProp.MaxStepAttempts = 50;
GMAT NearMoonProp.StopIfAccuracyIsViolated = true;

Create Propagator NearEarthProp;
GMAT NearEarthProp.FM = NearEarthProp_ForceModel;
GMAT NearEarthProp.Type = RungeKutta89;
GMAT NearEarthProp.InitialStepSize = 60;
GMAT NearEarthProp.Accuracy = 9.99999999999999e-12;
GMAT NearEarthProp.MinStep = 0.001;
GMAT NearEarthProp.MaxStep = 160000;
GMAT NearEarthProp.MaxStepAttempts = 50;
GMAT NearEarthProp.StopIfAccuracyIsViolated = true;

Create Propagator EarthPointMass;
GMAT EarthPointMass.FM = EarthPointMass_ForceModel;
GMAT EarthPointMass.Type = RungeKutta89;
GMAT EarthPointMass.InitialStepSize = 60;
GMAT EarthPointMass.Accuracy = 9.99999999999999e-12;
GMAT EarthPointMass.MinStep = 0.001;
GMAT EarthPointMass.MaxStep = 2700;
GMAT EarthPointMass.MaxStepAttempts = 50;
GMAT EarthPointMass.StopIfAccuracyIsViolated = true;

%-----
%----- Burns
%-----

Create ImpulsiveBurn EOI;
GMAT EOI.CoordinateSystem = Local;
GMAT EOI.Origin = Luna;
GMAT EOI.Axes = VNB;
GMAT EOI.Element1 = 0.87;
GMAT EOI.Element2 = 0.53;
GMAT EOI.Element3 = 0.18;
GMAT EOI.DecrementMass = false;
GMAT EOI.Isp = 300;
```

```
GMAT EOI.GravitationalAccel = 9.81;

%----- Coordinate Systems
%-----

Create CoordinateSystem EarthMoonRot;
GMAT EarthMoonRot.Origin = Earth;
GMAT EarthMoonRot.Axes = ObjectReferenced;
GMAT EarthMoonRot.XAxis = R;
GMAT EarthMoonRot.ZAxis = N;
GMAT EarthMoonRot.Primary = Earth;
GMAT EarthMoonRot.Secondary = Luna;

Create CoordinateSystem MoonInertial;
GMAT MoonInertial.Origin = Luna;
GMAT MoonInertial.Axes = BodyInertial;

%----- Solvers
%-----



Create DifferentialCorrector DC1;
GMAT DC1.ShowProgress = true;
GMAT DC1.ReportStyle = Normal;
GMAT DC1.ReportFile = 'DifferentialCorrectorDC1.data';
GMAT DC1.MaximumIterations = 80;
GMAT DC1.DerivativeMethod = ForwardDifference;
GMAT DC1.Algorithm = NewtonRaphson;

Create FminconOptimizer NLP;
GMAT NLP.MaximumIterations = 50;

%----- Subscribers
%-----



Create OrbitView EarthMoonRotView;
GMAT EarthMoonRotView.SolverIterations = Current;
GMAT EarthMoonRotView.UpperLeft      = [ 0.05294506949040371
0.003603603603603604 ];
GMAT EarthMoonRotView.Size         = [ 0.5433487756452681
1.243243243243243 ];
GMAT EarthMoonRotView.RelativeZOrder = 23;
GMAT EarthMoonRotView.Maximized = false;
GMAT EarthMoonRotView.Add = {MoonSat, Earth, Luna};
GMAT EarthMoonRotView.CoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView.DrawObject = [ true true true ];
GMAT EarthMoonRotView.DataCollectFrequency = 1;
GMAT EarthMoonRotView.UpdatePlotFrequency = 50;
```

```

GMAT EarthMoonRotView.NumPointsToRedraw = 0;
GMAT EarthMoonRotView.ShowPlot = true;
GMAT EarthMoonRotView.MaxPlotPoints = 20000;
GMAT EarthMoonRotView.ShowLabels = true;
GMAT EarthMoonRotView.ViewPointReference = Earth;
GMAT EarthMoonRotView.ViewPointVector = [ 10000 0 30000 ];
GMAT EarthMoonRotView.ViewDirection = Earth;
GMAT EarthMoonRotView.ViewScaleFactor = 40;
GMAT EarthMoonRotView.ViewUpCoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView.ViewUpAxis = -X;
GMAT EarthMoonRotView.EclipticPlane = Off;
GMAT EarthMoonRotView.XYPlane = Off;
GMAT EarthMoonRotView.WireFrame = Off;
GMAT EarthMoonRotView.Axes = Off;
GMAT EarthMoonRotView.Grid = Off;
GMAT EarthMoonRotView.SunLine = Off;
GMAT EarthMoonRotView.UseInitialView = On;
GMAT EarthMoonRotView.StarCount = 7000;
GMAT EarthMoonRotView.EnableStars = On;
GMAT EarthMoonRotView.EnableConstellations = Off;

Create OrbitView EarthMoonRotView2;
GMAT EarthMoonRotView2.SolverIterations = Current;
GMAT EarthMoonRotView2.UpperLeft = [ 0.1403044341495698
0.003603603603604 ];
GMAT EarthMoonRotView2.Size = [ 0.456651224354732
1.243243243243243 ];
GMAT EarthMoonRotView2.RelativeZOrder = 15;
GMAT EarthMoonRotView2.Maximized = false;
GMAT EarthMoonRotView2.Add = {MoonSat, Earth, Luna};
GMAT EarthMoonRotView2.CoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView2.DrawObject = [ true true true ];
GMAT EarthMoonRotView2.DataCollectFrequency = 1;
GMAT EarthMoonRotView2.UpdatePlotFrequency = 50;
GMAT EarthMoonRotView2.NumPointsToRedraw = 0;
GMAT EarthMoonRotView2.ShowPlot = true;
GMAT EarthMoonRotView2.MaxPlotPoints = 20000;
GMAT EarthMoonRotView2.ShowLabels = true;
GMAT EarthMoonRotView2.ViewPointReference = Earth;
GMAT EarthMoonRotView2.ViewPointVector = [ 10000 0 30000 ];
GMAT EarthMoonRotView2.ViewDirection = Earth;
GMAT EarthMoonRotView2.ViewScaleFactor = 40;
GMAT EarthMoonRotView2.ViewUpCoordinateSystem = EarthMoonRot;
GMAT EarthMoonRotView2.ViewUpAxis = -X;
GMAT EarthMoonRotView2.EclipticPlane = Off;
GMAT EarthMoonRotView2.XYPlane = Off;
GMAT EarthMoonRotView2.WireFrame = Off;
GMAT EarthMoonRotView2.Axes = Off;
GMAT EarthMoonRotView2.Grid = Off;
GMAT EarthMoonRotView2.SunLine = Off;

```

```

GMAT EarthMoonRotView2.UseInitialView = On;
GMAT EarthMoonRotView2.StarCount = 7000;
GMAT EarthMoonRotView2.EnableStars = On;
GMAT EarthMoonRotView2.EnableConstellations = Off;

Create OrbitView MoonInertialView;
GMAT MoonInertialView.SolverIterations = Current;
GMAT MoonInertialView.UpperLeft      = [    0.3170086035737922
0.1153153153153153 ];
GMAT MoonInertialView.Size         = [    1.101257445400397
1.052252252252252 ];
GMAT MoonInertialView.RelativeZOrder = 10;
GMAT MoonInertialView.Maximized = false;
GMAT MoonInertialView.Add = {MoonSat, Luna, Earth};
GMAT MoonInertialView.CoordinateSystem = MoonInertial;
GMAT MoonInertialView.DrawObject = [ true true true ];
GMAT MoonInertialView.DataCollectFrequency = 1;
GMAT MoonInertialView.UpdatePlotFrequency = 50;
GMAT MoonInertialView.NumPointsToRedraw = 150;
GMAT MoonInertialView.ShowPlot = true;
GMAT MoonInertialView.MaxPlotPoints = 20000;
GMAT MoonInertialView.ShowLabels = true;
GMAT MoonInertialView.ViewPointReference = Luna;
GMAT MoonInertialView.ViewPointVector = [ 20000 20000 20000 ];
GMAT MoonInertialView.ViewDirection = Luna;
GMAT MoonInertialView.ViewScaleFactor = 1.5;
GMAT MoonInertialView.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT MoonInertialView.ViewUpAxis = Z;
GMAT MoonInertialView.EclipticPlane = Off;
GMAT MoonInertialView.XYPlane = On;
GMAT MoonInertialView.WireFrame = Off;
GMAT MoonInertialView.Axes = On;
GMAT MoonInertialView.Grid = Off;
GMAT MoonInertialView.SunLine = Off;
GMAT MoonInertialView.UseInitialView = On;
GMAT MoonInertialView.StarCount = 7000;
GMAT MoonInertialView.EnableStars = On;
GMAT MoonInertialView.EnableConstellations = Off;

Create OrbitView EarthInertialView;
GMAT EarthInertialView.SolverIterations = Current;
GMAT EarthInertialView.UpperLeft = [ 0.6042356055592323 0 ];
GMAT EarthInertialView.Size      = [    0.6419589675711449
1.277477477477478 ];
GMAT EarthInertialView.RelativeZOrder = 5;
GMAT EarthInertialView.Maximized = false;
GMAT EarthInertialView.Add = {MoonSat, Earth, Luna};
GMAT EarthInertialView.CoordinateSystem = EarthMJ2000Eq;
GMAT EarthInertialView.DrawObject = [ true true true ];
GMAT EarthInertialView.DataCollectFrequency = 1;

```

```

GMAT EarthInertialView.UpdatePlotFrequency = 50;
GMAT EarthInertialView.NumPointsToRedraw = 0;
GMAT EarthInertialView.ShowPlot = true;
GMAT EarthInertialView.MaxPlotPoints = 20000;
GMAT EarthInertialView.ShowLabels = true;
GMAT EarthInertialView.ViewPointReference = Earth;
GMAT EarthInertialView.ViewPointVector = [ 0 0 30000 ];
GMAT EarthInertialView.ViewDirection = Earth;
GMAT EarthInertialView.ViewScaleFactor = 45;
GMAT EarthInertialView.ViewUpCoordinateSystem = EarthMJ2000Eq;
GMAT EarthInertialView.ViewUpAxis = Z;
GMAT EarthInertialView.EclipticPlane = Off;
GMAT EarthInertialView.XYPlane = On;
GMAT EarthInertialView.WireFrame = Off;
GMAT EarthInertialView.Axes = On;
GMAT EarthInertialView.Grid = Off;
GMAT EarthInertialView.SunLine = Off;
GMAT EarthInertialView.UseInitialView = On;
GMAT EarthInertialView.StarCount = 7000;
GMAT EarthInertialView.EnableStars = On;
GMAT EarthInertialView.EnableConstellations = Off;

Create ReportFile ReportFile1;
GMAT ReportFile1.SolverIterations = Current;
GMAT ReportFile1.UpperLeft = [ 0.02352941176470588 0.2275 ];
GMAT ReportFile1.Size = [ 0.9 0.78875 ];
GMAT ReportFile1.RelativeZOrder = 33;
GMAT ReportFile1.Maximized = false;
GMAT ReportFile1.Filename = 'ReportFile1.txt';
GMAT ReportFile1.Precision = 16;

GMAT ReportFile1.WriteHeader = true;
GMAT ReportFile1.LeftJustify = On;
GMAT ReportFile1.ZeroFill = Off;
GMAT ReportFile1.FixedWidth = true;
GMAT ReportFile1.Delimiter = ' ';
GMAT ReportFile1.ColumnWidth = 23;
GMAT ReportFile1.WriteReport = true;

%----- Arrays, Variables, Strings
%-----

Create Variable Theta RAAN AOP DV DV1 DV2 DV3, bError, incError,
eccError, radPerError, burnLoc, RMAGError, TOF;

GMAT ReportFile1.Add      = {TOF,      Theta,      RAAN,      AOP,
MoonSat.MoonInertial.RAAN,          MoonSat.MoonInertial.AOP,
MoonSat.ElapsedDays,   MoonSat.Earth.RadPer,  MoonSat.Luna.RadPer,
MoonSat.Luna.RMAG,     MoonSat.Earth.RMAG,   MoonSat.MoonInertial.INC,
EOI.Element1,    EOI.Element2,   EOI.Element3,   DV1,     MoonSat.INC,

```

```

MoonSat.MoonInertial.VX,           MoonSat.MoonInertial.VY,
MoonSat.MoonInertial.VZ,           MoonSat.EarthMJ2000Eq.VX,
MoonSat.EarthMJ2000Eq.VY,         MoonSat.EarthMJ2000Eq.VZ,
MoonSat.MoonInertial.VMAG, MoonSat.EarthMJ2000Eq.VMAG};

%----- Mission Sequence
%-----


BeginMissionSequence;

GMAT InitSat = MoonSat;
GMAT MoonSat.OrbitColor = 'Yellow';

Propagate 'Prop      One      Orbit'     NearEarthProp(MoonSat)
{MoonSat.ElapsedDays = 0.081839};

Optimize NLP {SolveMode = Solve, ExitMode = DiscardAndContinue};
Theta = MoonSat.MoonInertial.AOP + MoonSat.TA;
RAAN = MoonSat.MoonInertial.RAAN;
AOP = MoonSat.MoonInertial.AOP;

Vary 'Vary EOI' NLP(EOI.Element1 = EOI.Element1, {Perturbation
= .000001, Lower = -4, Upper = 4, MaxStep = .01,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
Vary 'Vary EOI' NLP(EOI.Element2 = EOI.Element2, {Perturbation
= .000001, Lower = -4, Upper = 4, MaxStep = .01,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});
Vary 'Vary EOI' NLP(EOI.Element3 = EOI.Element3, {Perturbation
= .000001, Lower = -4, Upper = 4, MaxStep = .01,
AdditiveScaleFactor = 0.0, MultiplicativeScaleFactor = 1.0});

Maneuver 'Apply EOI' EOI(MoonSat);

Vary 'Vary TOF' NLP(TOF = 3.25, {Perturbation = .0001, Lower =
-2.0, Upper = 4.5, MaxStep = .3, AdditiveScaleFactor = 0.0,
MultiplicativeScaleFactor = 1.0});

Propagate 'Prop      to      Earth'     NearEarthProp(MoonSat)
{MoonSat.ElapsedDays = TOF};
GMAT MoonSat.OrbitColor = 'Yellow';

GMAT RMAGError = MoonSat.Earth.RMAG - 6478;

NonlinearConstraint NLP(RMAGError <= 10)
NonlinearConstraint NLP(RMAGError >= -10)

GMAT DV1 = sqrt(EOI.Element1^2 + EOI.Element2^2 +
EOI.Element3^2);

```

|                                  |
|----------------------------------|
| Minimize NLP(DV1)<br>EndOptimize |
|----------------------------------|

## MATLAB Script: General Lambert Solver

```

%% General Lambert Computation Script
clear all
close all
clc

%% Known Constants and Parameters
mu = 398600; % Gravitation Parameter. Default is Sun

TOFVect = [3.0:0.5:6.0]*24*3600;
TAVect = [150 160 170 179.9];
numTOF = length(TOFVect);
count2 = 1; %TA Iteration Variable
resultsMatrix = zeros(numTOF);

for (TA = TAVect)

    count = 1;

    for (TOF = TOFVect)

        if TA <= 180
            TRANSFER_NUM = 1; % Transfer Type Number (1 or 2)
        else
            TRANSFER_NUM = 2;
        end

        %% Departure Quantities and Space Triangle Geometry

        % Departure Radii
        r_dep = 6378+600;
        r_arr = 384400;

        if TRANSFER_NUM == 1
            c = sqrt(r_dep^2+r_arr^2-2*r_dep*r_arr*cosd(TA));
        else
            c = sqrt(r_dep^2+r_arr^2-2*r_dep*r_arr*cosd(360-TA));
        end

        % Compute Space Triangle Semi-Perimeter
        s = (r_arr+r_dep+c)/2;

        %% Parabolic TOF and Transfer Type
        a_min = s/2;
        TOF_min = 0;

        % Determine Parabolic TOF
        if TRANSFER_NUM == 1
            TOF_par = 1/3*sqrt(2/mu)*(s^(3/2)-(s-c)^(3/2));
        end
    end
end

```

```

else
    TOF_par = 1/3*sqrt(2/mu)*(s^(3/2)+(s-c)^(3/2));
end

%% Determine Transfer Geometry (ELLIPIC or HYPERBOLIC)
if TOF > TOF_par
    TRANSFER_GEOMETRY = "ELLIPTIC";
    alpha_min = 2*asin(sqrt(s/(2*a_min)));
    beta_min = 2*asin(sqrt((s-c)/(2*a_min)));
    TOF_min = sqrt(a_min^3/mu)*((alpha_min-beta_min)-
(sin(alpha_min)-sin(beta_min)));

    if TOF>TOF_min
        TRANSFER_TYPE = num2str(TRANSFER_NUM)+"B";
    else
        TRANSFER_TYPE = num2str(TRANSFER_NUM)+"A";
    end

else
    TRANSFER_GEOMETRY = "HYPERBOLIC";
    TRANSFER_TYPE = num2str(TRANSFER_NUM)+"H";
end

transType(count) = TRANSFER_TYPE;

%% Select Appropriate alpha/beta fcns
alpha_fcn = @(a) 0;
beta_fcn = @(b) 0;
lambert_fcn = @(a) 0;

switch TRANSFER_TYPE
    case "1A"
        alpha_fcn = @(a) 2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) 2*asin(sqrt((s-c)/(2*a)));
    case "2B"
        alpha_fcn = @(a) 2*pi-2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) -2*asin(sqrt((s-c)/(2*a)));
    case "1B"
        alpha_fcn = @(a) 2*pi - 2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) 2*asin(sqrt((s-c)/(2*a)));
    case "2A"
        alpha_fcn = @(a) 2*asin(sqrt(s/(2*a)));
        beta_fcn = @(a) -2*asin(sqrt((s-c)/(2*a)));
    case "1H"
        alpha_fcn = @(a) 2*asinh(sqrt(s/(2*abs(a))));
        beta_fcn = @(a) 2*asinh(sqrt((s-c)/(2*abs(a))));
    case "2H"
        alpha_fcn = @(a) 2*asinh(sqrt(s/(2*abs(a))));
        beta_fcn = @(a) -2*asinh(sqrt((s-c)/(2*abs(a))));


```

```

    end

    %% Select Appropriate lambert function
    guess = a_min;
    switch TRANSFER_GEOMETRY
        case "ELLIPTIC"
            lambert_fcn      =     @(a)      sqrt(a^3/mu)*((alpha_fcn(a)-
sin(alpha_fcn(a)))-...
                                (beta_fcn(a)-sin(beta_fcn(a))))-TOF;
            guess = a_min;
        case "HYPERBOLIC"
            lambert_fcn          =          @(a)
sqrt(abs(a)^3/mu)*((sinh(alpha_fcn(a))-alpha_fcn(a))-...
                    (sinh(beta_fcn(a))-beta_fcn(a)))-TOF;
            guess = -(r_dep+r_arr)/2;
    end

    %% Solve Lambert
    a = fsolve(lambert_fcn, guess);
    alpha_val = alpha_fcn(a);
    beta_val = beta_fcn(a);
    if abs(lambert_fcn(a)) > 1e-4
        fprintf("ERROR! BAD A");
        5/0;
    end

    %% Compute p
    % Elliptic Case
    if TRANSFER_GEOMETRY == "ELLIPTIC"
        p1           =           4*a*(s-r_dep)*(s-
r_arr)/c^2*(sin((alpha_val+beta_val)/2))^2;
        p2           =           4*a*(s-r_dep)*(s-r_arr)/c^2*(sin((alpha_val-
beta_val)/2))^2;

        if TRANSFER_TYPE == "1A" || TRANSFER_TYPE == "2B"
            p = max(p1, p2);
        else
            p = min(p1,p2);
        end

        % Hyperbolic Case
    else
        p1           =           4*abs(a)*(s-r_dep)*(s-
r_arr)/c^2*(sinh((alpha_val+beta_val)/2))^2;
        p2           =           4*abs(a)*(s-r_dep)*(s-r_arr)/c^2*(sinh((alpha_val-
beta_val)/2))^2;
        if TRANSFER_TYPE == "1H"
            p = max(p1, p2);
        else
            p = min(p1,p2);
    end

```

```

        end
    end

    %% General Orbit Characteristics
    if TRANSFER_GEOMETRY == "ELLIPTIC"
        e = sqrt(1-p/a);
    else
        e = sqrt(1+p/abs(a));
    end

    spec_E = -mu/(2*a);
    v_dep = sqrt(2*(spec_E+mu/r_dep));
    v_arr = sqrt(2*(spec_E+mu/r_arr));

    if (TA == 180)
        theta_star_dep = 0;
        theta_star_arr = 180;
    else
        [theta_star_dep, theta_star_arr] = get_theta_star(p, e,
r_dep, r_arr, TA);
    end

    gamma_dep
sign(theta_star_dep)*acosd(sqrt(mu*p)/(r_dep*v_dep));
    gamma_arr
sign(theta_star_arr)*acosd(sqrt(mu*p)/(r_arr*v_arr));

    rp = a*(1-e);
    ra = a*(1+e);

    %% Delta-Vs Using F and G

    v_0_xyz = sqrt(mu/r_dep)*[0 1 0]';
    v_f_xyz = sqrt(mu/r_arr)*[cosd(TA+90) sind(TA+90) 0]';

    r_dep_xyz = r_dep * [1 0 0]';
    r_arr_xyz = r_arr * [cosd(TA) sind(TA) 0]';

    [f, g] = fg(r_dep, r_arr, TA, p, mu);

    v_dep_xyz = (r_arr_xyz - r_dep_xyz*f)/g;

    [df, dg] = dfg(r_dep_xyz, v_dep_xyz, TA, p, mu);

    v_arr_xyz = df*r_dep_xyz + dg*v_dep_xyz;

    dv1_vnb = rt2vnb(0)*rth2xyz(0, 0, 0)'*(v_dep_xyz-v_0_xyz);
    dv2_vnb = rt2vnb(gamma_arr)*rth2xyz(0, 0, TA)'*(v_f_xyz -
v_arr_xyz);

```

```

dv1 = norm(dv1_vnb);
dv2 = norm(dv2_vnb);
dv_tot = dv1+dv2;

resultsMatrix(count, 1) = TA;
resultsMatrix(count, 2) = TOF/(24*3600);
resultsMatrix(count, 3) = dv_tot;
resultsMatrix(count, 4) = transType(count);

%Spacecraft Transfer Orbit
if (transType(count) == '1A' || transType(count) == '1B' ||
transType(count) == '1H')
    thetaDep = acosd((p/r_dep - 1)/e);
elseif (transType(count) == '2A' || transType(count) == '2B' ||
transType(count) == '2H')
    thetaDep = -acosd((p/r_dep - 1)/e);
end

TASC = linspace(0, TA, 1000);
rSC = p./(1 + e*cosd(TASC + thetaDep));
eHatSC = cosd(TASC); %Translate to eHat coordinate frame
pHatSC = sind(TASC); %Translate to pHat coordinate frame
reSC(count, :) = rSC.*eHatSC;
rpSC(count, :) = rSC.*pHatSC;

count = count + 1;
end

%% Plot The Results

%Lunar Orbit
TALunar = linspace(0, 360, 1000);
rLunar = r_arr./(1 + 0*cosd(TALunar));
eHatLunar = cosd(TALunar);
pHatLunar = sind(TALunar);
reLunar = rLunar.*eHatLunar;
rpLunar = rLunar.*pHatLunar;

%Earth Parking Orbit
TAPark = linspace(0, 360, 1000);
rPark = r_dep./(1 + 0*cosd(TAPark));
eHatPark = cosd(TAPark);
pHatPark = sind(TAPark);
rePark = rPark.*eHatPark;
rpPark = rPark.*pHatPark;

figure(1) %%Begin plotting commands
subplot(2, 2, count2)
MoonPlot = plot(reLunar, rpLunar, '-k');
hold on

```

```

parkPlot = plot(rePark, rpPark, '-k');

for(i = 1:numTOF)
    if transType(i) == '1A'
        SCPlot = plot(reSC(i, :), rpSC(i, :), '-b'); %Blue
    elseif transType(i) == '1B'
        SCPlot = plot(reSC(i, :), rpSC(i, :), '-r'); %Red
    elseif transType(i) == '2A'
        SCPlot = plot(reSC(i, :), rpSC(i, :), '-m'); %Magenta
    elseif transType(i) == '2B'
        SCPlot = plot(reSC(i, :), rpSC(i, :), '-g'); %Green
    elseif transType(i) == '1H' || transType(i) == '2H'
        SCPlot = plot(reSC(i, :), rpSC(i, :), '-c'); %Cyan
    else
        SCPlot = plot(reSC(i, :), rpSC(i, :), '-y'); %Yellow
    end
    set(SCPlot, "lineWidth", 1);
end

hold off

set(parkPlot, "linewidth", 1);
set(MoonPlot, "linewidth", 1);
grid on
grid minor
axis equal

xlabel('$$\hat{X}$$ (km)', 'Interpreter', 'Latex', 'FontSize', 14)
ylabel('$$\hat{Y}$$ (km)', 'Interpreter', 'Latex', 'FontSize', 14)

if(TA >= 179 && TA <= 181)
    str = sprintf(['TA = %0.1d' char(176)], 180);
else
    str = sprintf(['TA = %0.1d' char(176)], TA);
end
sgtitle('Lunar Transfer from 600 km Earth Parking Orbit Using Various Transfer Angles')
title(str)

xlim([-5e5 5e5])
ylim([-5e5 5e5])

figure(2)
subplot(2, 2, count2)
plot(resultsMatrix(:, 2), resultsMatrix(:, 3), 'o')
xlabel('TOF (Days)')
ylabel('\Delta V (km/s)')
grid on
grid minor
sgtitle('Lunar Transfer DV vs TOF Tradeoff')

```

```
title(str)
xlim([0 7])
ylim([3.8 5.0])

count2 = count2 + 1;
end

transTy
```

## XI.Appendix for Orbital Stationkeeping for Propellant Depots

### A. Lunar Orbit Stationkeeping FreeFlyer Code

The lunar and Earth-orbiting propellant depots implement a similarly-structured targeting algorithm consisting of FreeFlyer's built-in targeting commands, so only one script is included here. All of the mission plan files used to generate the plots included here are available on the Project NextStep GitHub page.

```
Spacecraft1.SetCentralBody(Moon);

// View window
vw.CurrentViewpoint.ThreeDView.Source = Moon.ObjectId;
vw.CurrentViewpoint.ThreeDView.Target = Moon.ObjectId;
vw.CurrentViewpoint.ThreeDView.TailReference = Moon.ObjectId;

// Variables
Variable dvTot = 0;
Variable numBurns = 0;
Variable tPrevBurn = 0;
Variable minBurnSep = 13.5;
Array elems = Spacecraft1.GetKeplerianState(7);

// Plot Windows
PlotWindow p1({Spacecraft1.ElapsedTime, Spacecraft1.Height});
p1.PlotTitle.Text = 'LLO Depot Altitude Evolution - Stephen Grabowski';
p1.XAxis.Title.Text = 'Elapsed Days';
p1.YAxis.Title.Text = 'Altitude (km)';

PlotWindow p2({Spacecraft1.ElapsedTime, Spacecraft1.A});
p2.PlotTitle.Text = 'LLO Depot SMA Evolution - Stephen Grabowski';
p2.XAxis.Title.Text = 'Elapsed Days';
p2.YAxis.Title.Text = 'Semi-Major Axis (km)';

PlotWindow p3({Spacecraft1.ElapsedTime, elems[1]});
p3.PlotTitle.Text = 'LLO Depot Eccentricity Evolution - Stephen Grabowski';
p3.XAxis.Title.Text = 'Elapsed Days';
p3.YAxis.Title.Text = 'Eccentricity';
```

```

PlotWindow p4({Spacecraft1.ElapsedTime, elems[2]});  

p4.PlotTitle.Text = 'LLO Depot Inclination Evolution - Stephen Grabowski';  

p4.XAxis.Title.Text = 'Elapsed Days';  

p4.YAxis.Title.Text = 'Inclination (deg)';

PlotWindow p5({Spacecraft1.ElapsedTime, Spacecraft1.Periapsis});  

p5.PlotTitle.Text = 'LLO Depot Periapsis Evolution - Stephen Grabowski';  

p5.XAxis.Title.Text = 'Elapsed Days';  

p5.YAxis.Title.Text = 'Periapsis Distance (km)';

PlotWindow p6({Spacecraft1.ElapsedTime, Spacecraft1.Apoapsis});  

p6.PlotTitle.Text = 'LLO Depot Apoapsis Evolution - Stephen Grabowski';  

p6.XAxis.Title.Text = 'Elapsed Days';  

p6.YAxis.Title.Text = 'Apoapsis Distance (km)';

p1.MaxPoints = 100000;  

p2.MaxPoints = 100000;  

p3.MaxPoints = 100000;  

p4.MaxPoints = 100000;  

p5.MaxPoints = 100000;  

p6.MaxPoints = 100000;

// Mission  

TimeSpan stepEpoch;  

stepEpoch = Spacecraft1.Epoch + TimeSpan.FromDays(365);

While (Spacecraft1.ElapsedTime.ToDays < 365);  

    Step Spacecraft1;  

    elems = Spacecraft1.GetKeplerianState(7);  

    Update vw;

If (Spacecraft1.ElapsedTime.ToDays - tPrevBurn > minBurnSep or numBurns == 0);  

    If (Spacecraft1.E > .02);  

        While (Spacecraft1.Height > 102 or Spacecraft1.Height < 98);  

            Step Spacecraft1;  

            End;  

    Target using DifferentialCorrector1;  

    Iterate Spacecraft1;  

    Vary ImpulsiveBurn1.BurnDirection[0] = 0 + .001;  

    Vary ImpulsiveBurn1.BurnDirection[1] = 0 + .001;  

    Vary ImpulsiveBurn1.BurnDirection[2] = 0 + .001;  

    Maneuver Spacecraft1 using ImpulsiveBurn1;

```

```

Achieve Spacecraft1.E = 0 +/- .001;

If (DifferentialCorrector1.IterationMode == 'Converged');
    numBurns += 1;
    dvTot+=(ImpulsiveBurn1.BurnDirection[0]^2 +
    ImpulsiveBurn1.BurnDirection[1]^2 + ImpulsiveBurn1.BurnDirection[2]^2)^.5;
    tPrevBurn = Spacecraft1.ElapsedTime.ToDays;
End;
End;

If (elems[2] > 93 or elems[2] < 87);
    Spacecraft1.StepToApsis(1);
    Target using DifferentialCorrector1;
    Iterate Spacecraft1;
    Vary ImpulsiveBurn1.BurnDirection[0] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[1] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[2] = 0 + .001;
    Maneuver Spacecraft1 using ImpulsiveBurn1;
        Achieve Spacecraft1.GetKeplerianState(7)[2] = 90 +/- .5;
        If (DifferentialCorrector1.IterationMode == 'Converged');
            numBurns += 1;
            dvTot+=(ImpulsiveBurn1.BurnDirection[0]^2+
            ImpulsiveBurn1.BurnDirection[1]^2
            ImpulsiveBurn1.BurnDirection[2]^2)^.5;
            tPrevBurn = Spacecraft1.ElapsedTime.ToDays;
        End;
    End;
End;
End;
Update p1;
Update p2;
Update p3;
Update p4;
Update p5;
Update p6;
End;
Report dvTot, numBurns;

```

## XII. References and Code for Free Return

[<sup>1</sup>]Miller, David W., “Potential Orbits about the Lagrangian Points”, *Satellite Engineering*, 2003.

[<sup>2</sup>]Farquhar, Robert W., “The Utilization of Halo Orbits In Advanced Lunar Operations”, *NASA Technical Note, TN D-6365*, 1971.

```

clearvars, close all
clc
%%
rMoon = 1738.2;
muMoon = 4902.8005821478;
aMoon = 384400;
rEarth = 6378.1363;
muEarth = 398600.4415;

%% pt. a
alpha1 = 38.2529;
gamPlus1 = 33.7046;
eta = 180-alpha1;
beta2 = 180-eta-gamPlus1;
alpha2 = 180-beta2;
%% pt. b
TA = 173.8;
altEarth = 190;
rp = rEarth + altEarth;
eT = (rp-aMoon)/(aMoon*cosd(TA)-rp);
aT = rp/(1-eT);
ra = aT*(1+eT);
IP = 2*pi*sqrt(aT^3/muEarth);
specEnergy = -muEarth/(2*aT);
E = rad2deg(acos((aT-aMoon)/(aT*eT)));
M = deg2rad(E) - eT*sind(E);
n = (2*pi)/IP;
tMinustP = M/n;
n2 = sqrt(muEarth/(aMoon^3));
phi = rad2deg(deg2rad(TA) - (n2*tMinustP));
%% pt. c
vMinus = sqrt((2*muEarth/aMoon)-(muEarth/aT));
vPlus = vMinus;
rPlus = aMoon;
p = aT*(1-eT^2);
h = sqrt(muEarth*p);
gamMinus = rad2deg(acos(h/(rPlus*vPlus)));
vMoon = sqrt(muEarth/aMoon);

```

```

vInfMoon           = sqrt((vMoon^2)+(vMinus^2)-
(2*vMoon*vMinus*cosd(gamMinus)));
flybyAng = rad2deg(2*acos(((vMinus^2)-(vMoon^2)-(vInfMoon^2))/(-
2*vMoon*vInfMoon)));
eH = 1/sind(flybyAng/2);
specEnergyH = 0.5*(vInfMoon^2);
aH = -muMoon/(vInfMoon^2);
rpPass = aH*(1-eH);
passAlt = rpPass-rMoon;
dvEq           = sqrt((vInfMoon^2)+(vInfMoon^2)-
(2*vInfMoon*vInfMoon*cosd(flybyAng)));
beta1 = (180-2*gamMinus)/2;
alphal = 180-beta1; % alpha negative because s/c appraoching Earth

%% pt. d plotting
% pt. i) earth centered frame
% plot earth orbit
% subplot(2,2,[3,4])
% suptitle('Lunar Free-Return Trajectories');
% figure()
suptitle('Lunar Free Return Trajectories - By: Mohit Pathak');
subplot(2,1,1);
thetaStar = (linspace(0,2*pi, 3600000))';
xEarth = rEarth.*cos(thetaStar);
yEarth = rEarth.*sin(thetaStar);
plot(xEarth,yEarth, 'g', 'LineWidth', 2)
hold on;
grid on
% plot moon orbit
thetaStar = (linspace(0,2*pi, 3600000))';
rTemp2 = aMoon;
xMoon = rTemp2.*cos(thetaStar);
yMoon = rTemp2.*sin(thetaStar);
hold on;
plot(xMoon,yMoon, 'k--', 'LineWidth', 1)
axis equal
hold on;
plot(xMoon(1738000),yMoon(1738000),'k*', 'LineWidth', 2)
grid on
% plot parking orbit around Earth
rTemp3 = rp;
xParking = rTemp3.*cos(thetaStar);
yParking = rTemp3.*sin(thetaStar);
hold on;
plot(xParking,yParking, 'b', 'LineWidth', 2)
hold on;
grid on
axis equal
% plot transfer arcs
% plot out-bound arc

```

```

thetaStarOutbound = (linspace(0, (173.8*pi/180)))';
domega = deg2rad(-12.4);
rTemp3 = p./(1+eT.*cos(thetaStarOutbound));
xtransferOut = rTemp3.*cos(thetaStarOutbound);
ytransferOut = rTemp3.*sin(thetaStarOutbound);
hold on;
% subplot(2,2,1)
% plot(xEarth,yEarth, 'g', 'LineWidth', 2)
% animate orbit
curve = animatedline('LineWidth',2);
% plot(xtransferOut,ytransferOut, 'b-', 'LineWidth', 1)
for i=1:length(xtransferOut)
    hold on
    addpoints(curve,xtransferOut(i), ytransferOut(i));
    head = scatter(xtransferOut(i),
ytransferOut(i),'filled','MarkerFaceColor','b');
    drawnow
    % pause(0.01);
    delete(head);
    if(i == 93)
        hold on;
        plot(xtransferOut(93),ytransferOut(93), 'b-<',
'LineWidth', 1)
    end
    % subplot(2,2,1);
    % hold on;
    % xlim([-90000,90000]); ylim([-60000, 60000]);
    % addpoints(curve,xtransferOut(i), ytransferOut(i));
    % head = scatter(xtransferOut(i),
ytransferOut(i),'filled','MarkerFaceColor','b');
    % drawnow
    % pause(0.01);
    % delete(head);
end
hold on
grid on
axis equal
% plot in-bound arc
thetaStarInbound = (linspace(-(173.8*pi/180),0))';
rTemp3 = p./(1+eT.*cos(thetaStarInbound));
xtransferOut = rTemp3.*cos(thetaStarInbound + domega);
ytransferOut = rTemp3.*sin(thetaStarInbound + domega);
hold on;
curve2 = animatedline('LineWidth',2, 'Color', 'g');
% plot(xtransferOut,ytransferOut, 'g-', 'LineWidth', 1)
hold on;
for i=1:length(xtransferOut)
    addpoints(curve2,xtransferOut(i), ytransferOut(i));
    head = scatter(xtransferOut(i),
ytransferOut(i),'filled','MarkerFaceColor','g');

```

```

drawnow
% pause(0.01);
delete(head);
if(i == 7)
    hold on;
    plot(xtransferOut(7),ytransferOut(7), 'g->', 'LineWidth',
1)
end
hold on;
grid on
% make everything look nice :)
title('Full Trajectory View');
xlabel('Distance x (km)');
ylabel('Distance y (km)');
axis equal
% plot moon
% figure()
subplot(2,1,2)
axis equal
xlim([-10000,7000]); ylim([-7000, 7000]);
title("Moon-Centered View")
hold on
thetaStar = (linspace(0,2*pi))';
rTemp3 = rMoon;
xtransferOut = rTemp3.*cos(thetaStar);
ytransferOut = rTemp3.*sin(thetaStar);
hold on;
plot(xtransferOut,ytransferOut, 'k-', 'LineWidth', 2);
fill(xtransferOut,ytransferOut, 'k-')
hold on;
hold on;
grid on
axis equal
% plot hyperbolic orbit
thetaStar = (linspace(-pi,pi))';
pH = aH*(1-eH^2);
rTemp4 = pH./(1+eH.*cos(thetaStar));
xtransferOut = rTemp4.*cosh(thetaStar);
ytransferOut = rTemp4.*sinh(thetaStar);
xtransferOut = xtransferOut-(2*rpPass);
curve3 = animatedline('LineWidth',2);
hold on;
for i=1:length(xtransferOut)
    addpoints(curve3,xtransferOut(i), ytransferOut(i));
    head = scatter(xtransferOut(i),
ytransferOut(i), 'filled', 'MarkerFaceColor', 'r');
    drawnow
    pause(0.01);
    delete(head);

```

```

end
%plot(xtransferOut-((2*rpPass)),ytransferOut, 'r-', 'LineWidth',
2);
hold on;
grid on;
function lplot(M1, Ms)
r_sun = .1; % scaled diameter of the large body (i.e. sun), not
to scale
r_earth = .05; % scaled diameter of the small body (i.e. earth), not
to scale
points=findLPs(M1, Ms);
MU = Ms / (M1 + Ms); % large body to barycenter in normalized
distance units
figure
hold on
theta = 0:pi/32:2*pi;
%plot(r_sun*sin(theta)-MU, r_sun*cos(theta), 'g')
fill(r_sun*sin(theta)-MU, r_sun*cos(theta), 'g');
%plot(r_earth*sin(theta)+1-MU, r_earth*cos(theta), 'k')
fill(r_earth*sin(theta)+1-MU, r_earth*cos(theta), 'k')
legend('Earth', 'Moon')
plot(sin(theta)-MU, cos(theta), 'b', 'linewidth', 2)
x_axis=-1.5:1.5;
y_axis=-1:1;
plot(x_axis, zeros(1,length(x_axis)), 'k', 'linewidth', 2)
plot(zeros(1,length(y_axis)), y_axis, 'k', 'linewidth', 2)
% plot(points(:,1),points(:,2), 'g*', 'markersize',12, 'linewidth',
2)
plot(points(1,1),points(1,2), 'r*', 'markersize',12, 'linewidth',
2)
plot(points(2,1),points(2,2), 'r*', 'markersize',12, 'linewidth',
2)
plot(points(3,1),points(3,2), 'r*', 'markersize',12, 'linewidth',
2)
plot(points(4,1),points(4,2), 'g*', 'markersize',12, 'linewidth',
2)
plot(points(5,1),points(5,2), 'g*', 'markersize',12, 'linewidth',
2)
legend('Earth', 'Moon', 'line1', 'line2', 'circle1', 'L1', 'L2',
'L3', 'L4', 'L5')
title('Earth-Moon Lagrange Points By: Mohit Pathak')
axis equal
grid on
end
function x=quintic(chi)
% Lagrange's quintic equation, solved by fzero to find L1, L2, L3
global Ma
global Mb
global Mc

```

```
x=(Ma + Mb)*chi^5 + (3*Ma + 2*Mb)*chi^4 + (3*Ma + Mb)*chi^3 - (Mb + 3*Mc)*chi^2 - (2*Mb + 3*Mc)*chi - (Mb + Mc);  
mEarth = 5.9736e24;  
mMoon = 7.347673e22;  
myPlot(mEarth, mMoon)
```

# Propulsion Appendix

## Nomenclature

$A$  = surface area, m<sup>2</sup>

$C_d$  = drag coefficient

$C_p$  = constant pressure heat capacity,  $\frac{J}{kgK}$

$g_e$  = gravitational acceleration at Earth's surface ( $9.81 \frac{m}{s^2}$ )

$I_{sp}$  = specific impulse, seconds

$K$  = thermal conductivity,  $\frac{W}{mK}$

$\lambda_{SHB}$  = inert mass fraction of the Super Heavy Booster not including payload

$\lambda_{Starship}$  = inert mass fraction of Starship not including payload

$LEO$  = low earth orbit

$LLO$  = low lunar orbit

$m$  = mass, kg

$m_f$  = vehicle final mass for a given mission, Mg

$m_{inert}$  = vehicle inert mass, Mg

$m_o$  = vehicle initial mass for given mission, Mg

$m_{payload}$  = mass of payload, Mg

$m_{prop}$  = available propellant mass at start of given mission, Mg

$MR$  = vehicle initial to final mass ratio

$\mu_e$  = gravitational parameter of Earth ( $398600.4 \frac{km^3}{s^2}$ )

$\rho$  = density, kg/m<sup>3</sup>

$Q$  = heat flux, Watts

$Q_{vap}$  = vaporization energy of a fluid,  $\frac{J}{kg}$

$r_e$  = radius of Earth, km

$T_{boil}$  = boiling point of a liquid, K

$t_{diffusiv}$  = saturation time of the thermal diffusivity layer, seconds

$T_{\text{subcool}}$  = the temperature below the boiling point of a liquid, K

$\Delta V$  = delta-v in  $\frac{km}{s}$

$z$  = altitude of orbit, km

## I. Launch Vehicles

### A. Launch Vehicles Considered - Preliminary Analysis

The process of researching and choosing the appropriate launch vehicles was a rigorous one. Faced with a challenge such as starting a colony on the Moon, we researched a wide range of possible launch vehicles that could be used to accomplish the mission. To accomplish the goal, we investigated vehicles from all categories: light, medium, heavy, and super-heavy lift launch vehicles. The purpose of each vehicle is dependent on the nature of the launch, for example if a lot of cargo, such as concrete, equipment etc. is needed, a super-heavy would be optimal, whereas if a small satellite was required, a light launch vehicle could achieve the mission while not taking too much from the overall budget.

#### *1. Light-Lift Launch Vehicles*

The main motivator for using light-lift launch vehicles, such as converted ICBMs, is convenience and cost. Converted ICBMs, such as the Minotaur V, Dnepr and Delta II, are small and accommodating in terms of payload, and cheaper to launch compared to larger vehicles. These missiles can launch satellites to TLI (Trans-Lunar Injection) during the initial launching phase as well as auxiliary cases. For example, if a satellite malfunctions and needs replacement on short notice, a converted ICBM is more suitable as opposed to a Falcon Heavy. Table 1 provides a quantitative comparison of the considered light-lift launch vehicles.

| Vehicle                         | Payload Mass - TLI<br>(Mg) | Fairing Volume (m <sup>3</sup> ) | Cost (million USD) |
|---------------------------------|----------------------------|----------------------------------|--------------------|
| Minotaur V <sup>[1,2,3]</sup>   | 0.447                      | 23.45                            | ~30                |
| Dnepr (R) <sup>[4,5,6]</sup>    | 0.75                       | 40.29                            | 15-30              |
| Delta II - 7925H (R)<br>[7,8,9] | 1.519                      | 65.38                            | 51                 |

Table 1 – Table of light launch vehicles considered.

R – retired

For this mission, we consider a limited number of light launch vehicles. We consider the ULA Delta II, the Northrop Grumman Minotaur V and the ISC Kosmotras Dnepr. Each with its own advantages and drawbacks. For example, while the Delta II has the largest payload mass-volume parameter, it is costlier to launch than the Dnepr, which can carry a slightly lighter mass but for ~20 million USD less to the same orbit. On the other hand, Minotaur V is still in active use compared to the Dnepr and Delta II, which have been retired. Therefore, if we want to reinstate the vehicles, we require access to the respective documentation so we can recreate them with high fidelity. Moreover, the Dnepr's design is not of U.S. origin, which can complicate matters if the nations are not in agreement with each other.

However, following continued refinements in terms of mass and volume from the satellites team, these launch vehicles are now obsolete. The satellites weigh more than the provided payload

mass capabilities of the missiles. Nonetheless, these launch vehicles can still be deployed as impromptu cargo-delivering systems, since all of these light vehicles are capable of reaching TLI and, consequently, the Moon.

## *2. Medium-Lift Launch Vehicles*

Some medium-lift launch vehicles are considered for this mission. Medium-lift launch vehicles can launch satellites into their designated orbits. They can also bring cargo to the Moon while costing less than heavier vehicles. However, the medium-lift vehicles are limited due to their design and intended purpose. In most cases, medium launch vehicles are not performant enough to reach the Moon with any considerable payload. In other cases, the design was not made with the intent of reaching the Moon, which is what the purpose of our current mission is. For this mission, we consider the following medium-lift launch vehicles: Atlas V from ULA, Soyuz-2 from PRSC (Progress Rocket Space Centre) and the Falcon 9 with the Block 5 configuration from SpaceX. Out of the aforementioned vehicles, the Falcon 9 Block 5 is currently the most performant launch vehicle capable of leaving Earth's orbit in terms of payload mass, with cost per ton payload at \$2.72 million for the expendable configuration and cheaper for reusable configuration. The alternatives, namely the Soyuz and Atlas V, are not able to match the Falcon 9 Block 5 parameters. The Soyuz's design is not intended to leave Earth's atmosphere, therefore making it unsuitable for the current mission. The Atlas V, while having the capability of leaving Earth's atmosphere, is less performant than the Falcon 9 Block 5. Atlas V costs at least twice as much to launch as a Falcon 9 Block 5 while carrying 2 tons less. Moreover, Falcon 9 is the most suitable choice for our medium-lift launch vehicle due to its capability of undertaking manned missions via the Dragon Capsule. The Dragon Capsule can launch seven people at once and has a good track record, with

25 total launches out of which 23 have successfully delivered astronauts to the ISS. As a result, we choose to include the Falcon 9 Block 5 in our list of employable vehicles for this mission. Table 2 provides a quantitative comparison of the medium-lift launch vehicles considered.

| Vehicle                        | Payload mass - LEO (Mg) | Fairing Volume (m <sup>3</sup> ) | Cost (million USD) |
|--------------------------------|-------------------------|----------------------------------|--------------------|
| Soyuz-2 [10,11,12]             | 4.85                    | 151.64                           | 35-48.5            |
| Atlas V [13,14]                | 18.85                   | 535.91                           | >109               |
| Falcon 9 Block 5<br>[15,16,17] | 22.8                    | 295.19                           | 62                 |

Table 2 - Table listing Medium-Lift Launch Vehicles considered

### 3. Heavy-Lift Launch Vehicles

The nature of heavy-lift launch vehicles make them desirable for the current mission. A generously sized payload fairing, capability to reach TLI and moderate to high launch costs make them good candidates to undertake the mission of delivering cargo to the Moon. Out of the currently active heavy-lift launch vehicles, we consider the Ariane 5 ES from ESA, Long March 5 from CALT and the Delta IV Heavy from ULA. Besides the currently employed vehicles, we also consider some in-development vehicles. These are: Ariane 6 from ESA, Vulcan Centaur Heavy from ULA and New Glenn from Blue Origin.

The parameters of the vehicles are listed in table 3. However, some of the developed and in-development vehicles do not have their parameters published due to lack of testing, company secrets or other factors. These vehicles are the Long March, New Glenn and Vulcan Centaur Heavy. Therefore, it is hard to measure their capability in accomplishing the current mission. Other vehicles, such as Ariane 5 ES and Delta IV heavy, are still used and have successful launch histories, proving their capabilities and effectiveness in achieving their missions, making them desirable. However, cost of payload-ton per million USD and volume of supported payload become the primary concerns once we explore these vehicles more in-depth. For example, the Ariane 5 ES has a conservative \$10 million per ton payload cost, whereas the Delta IV Heavy will cost \$16 million per ton. Deciding on a heavy-lift launch vehicle is further complicated if we consider the Falcon Heavy, a vehicle from the super-heavy-lift category. The Falcon Heavy eclipses every heavy launch vehicle we consider, both in payload mass capabilities and cost per ton, which is further analyzed in the next section. Therefore, since the Falcon Heavy outperforms every single heavy-lift launch vehicle in terms of payload while being cheaper to launch, we do not include any heavy lift launch vehicle in our list of employable rockets.

| Vehicle                 | Fairing Volume          |                   |                    |
|-------------------------|-------------------------|-------------------|--------------------|
|                         | Payload Mass - LEO (Mg) | (m <sup>3</sup> ) | Cost (million USD) |
| Ariane 5 ES [18,19,20]  | 19.3                    | 389.34            | 165-220            |
| Long March 5/5B [21,22] | 25                      | 435.36            | TBD                |

|                                    |       |         |     |
|------------------------------------|-------|---------|-----|
| Ariane 64 (est.)                   | 20    | ~389.34 | 82  |
| [23,24,25,26]                      |       |         |     |
| New Glenn (est.)                   | 45    | 338.95  | TBD |
| [27,28]                            |       |         |     |
| Vulcan Centaur                     | 27.9  | TBD     | 100 |
| Heavy (est.) <sup>[29,30,31]</sup> |       |         |     |
| Delta IV Heavy                     | 22.98 | 402.84  | 350 |
| [32,33]                            |       |         |     |

Table 3 – Table listing Heavy-Lift Launch Vehicles considered.

TBD – to be determined; est. - estimated

#### 4. Super-Heavy-Lift Launch Vehicles

Super-heavy-launch lift vehicles will be the primary vehicles used to kickstart the Moon colony at pre- and post-2023. These behemoths can carry tens, in some cases hundreds, of tons of payload to LEO (Low-Earth Orbit) and beyond. Some of the vehicles we consider from the super-heavy-lift launch vehicle bracket are the SLS from NASA, the Starship and Falcon Heavy from SpaceX. However, there is a big range of capabilities among these vehicles. The SLS, despite promising capabilities, is far from being finalized. As a result, we are left with the Falcon Heavy and the Starship from SpaceX.

In terms of performance, Starship is projected to be a truly next-generation launch vehicle capable of going beyond Earth and capable of settling people on Mars. Moreover, it's primary

attractiveness lies in its reusability, cutting costs of re-launching the same system and saving materials that would otherwise be used to manufacture new vehicles. However, it is still being developed and projected to launch in 2023. Which is why the Falcon Heavy is projected to be the most used launch vehicle prior to 2023. The Falcon Heavy is the most competitive launch vehicle out of all of the previously mentioned ones, delivering a staggering 63.8 Mg for \$150 million (expendable) or 57 Mg for \$90 million (reusable) to LEO and beyond. Post 2023, Falcon Heavy can still be used to deliver smaller cargo more often, especially after the colony becomes self-sustainable and delivering large payloads becomes less of a necessity. Table 4 provides a quantitative comparison of the super-heavy-lift launch vehicles considered.

| Vehicle          | Payload Mass - LEO (Mg) | Fairing Volume (m <sup>3</sup> ) | Cost (million USD) |
|------------------|-------------------------|----------------------------------|--------------------|
| Falcon Heavy     | 63.8                    | 295.19                           | 150                |
| [34]             |                         |                                  |                    |
| Starship (est.)  | 100+                    | TBD                              | 873 (adjusted)     |
| [35, 36, 37, 38] |                         |                                  |                    |
| SLS (est.) [39]  | 130                     | TBD                              | TBD                |
|                  |                         |                                  |                    |

Table 4 – Table listing Super Heavy-Lift Launch Vehicle.

TBD - to be determined; est. - estimated

### *5. Conclusions*

In conclusion, we choose the following launch vehicles for the current mission: Falcon 9, Falcon Heavy and Starship from SpaceX. A preliminary analysis was conducted in this section to support our choices, with a focus on payload mass, fairing volume and cost per launch. We conclude that the selected vehicles are the optimal choices for the current mission based on the parameters chosen.

## B. Launch Vehicles Considered - In Depth Analysis

### 1. Falcon 9



**Fig B1: Falcon 9 with Cargo Fairing[40]**

### Introduction

The requirements of Project NextStep necessitate a launch vehicle that can not only transport large payload masses to lunar orbit and the surface, but also cost effective enough to support the number of launches without going over budget. The main constraint is payload capacity, as the  $\Delta V$  required for low lunar orbit is substantial which immediately eliminates some of the currently used vehicles. Then looking at the cost of these vehicles, the Falcon 9 stands out as a potential selection. Falcon 9 first flew in June 2010, with the most recent version making its

maiden flight in December of 2015[41], making it one of the newest and most technologically advanced vehicles on the market with a 111/113[40] success rate. Combining payload mass, low cost, and reliability make the Falcon 9 an ideal launch vehicle to consider.

### **Capabilities and Design**

Unlike any other launch vehicle not produced by SpaceX, the Falcon 9 has the capability to land its core stage after use which is the main driver of the low cost. This feature is also relatively successful, with 72[40] boosters landing upright and recovered. This allows for fewer vehicles to be produced, something attractive in a project with a ten year lifespan. Reusability is a relatively new capability in commercial rocketry, and a dramatic cost reduction is seen from it with a tradeoff in some payload mass loss.

Classified as a medium launch vehicle, the Falcon 9 was not considered as the main vehicle for landing large payloads on the Moon. Instead, these medium sized vehicles would have the requirement to place communication and observation satellites into a specified orbit around the Moon based on the location of the colony, along with other potential smaller payloads later in the mission where the cost of a large vehicle is not necessitated. That being said, Falcon 9 still has the largest payload mass of any medium launch vehicle, capable of boosting up to 22.8 Mg[42] to LEO. Table B1 below summarizes performance characteristics for the Falcon 9 reusable and expendable vehicles

| <b>Parameter</b> | <b>Falcon 9 Reusable</b> | <b>Falcon 9 Expendable</b> |
|------------------|--------------------------|----------------------------|
|------------------|--------------------------|----------------------------|

|                                  |                      |                      |
|----------------------------------|----------------------|----------------------|
| Flight Cost                      | \$50 Million USD[42] | \$62 Million USD[42] |
| Payload to LEO (Mg)              | 12.1*                | 22.8                 |
| Payload to LLO (Mg)              | 3.21*                | 5.15*                |
| Fairing Volume (m <sup>3</sup> ) | 295.2[43]            | 295.2[43]            |
| Cost per Mg to LLO (M\$/Mg)      | 15.57*               | 12.03*               |
| Liftoff Thrust (kN)              | 7607[40]             | 7607[40]             |

**Table B1: Falcon 9 Reusable and Expendable Performance Parameters**

SpaceX publishes some of these values for public knowledge, but their vehicles do not operate in the same flight regime where Project NextStep requires some of the smaller payloads flown by medium launch vehicles to be placed in specific orbits around the moon. This led to some values being calculated (Noted by an asterisk) for feasibility of the Falcon 9 to meet these requirements.

In the context of Project NextStep, payload masses to LLO as well as their costs per Mg were required so proper constraints could be put on the systems/vehicles the other teams designed. Using published SpaceX numbers, these payloads could be backsolved using known values and those calculated by other teams. The initial published payload chosen was that to GTO (Geostationary Transfer Orbit), as this was the value published for both the reusable and expendable version. For the reusable the payload mass to GTO is 5.5 Mg [42] while for the

expendable is 8.30 Mg [42]. The  $\Delta V$  of a Hohmann Transfer from the desired parking orbit of 600 km to this orbit was calculated, and then the payload to 600 km orbit was solved from this. As SpaceX only publishes values to LEO of an unknown altitude, the actual payload to our orbit had to be adjusted. From here,  $\Delta V$  values from the Mission Design team for TLI (Trans-Lunar Injection), correction burns, and circularization were applied to the payload masses at 600 km, leading to the final mass possible after reaching LLO. This yielded an approximate payload of 3.21 Mg to LLO for the Falcon 9 reusable variant and ~~5.15~~ Mg for the expendable variant.

As a check to this values, SpaceX published values for the Falcon 9 reusable version to TMI (Trans-Martian Injection) of 4.02 Mg [42], so using the characteristic energy (C3) equations for  $\Delta V$ , the payload to the desired orbit of 600 km was again back solved using  $\Delta V$  and final payload. Again, using Mission Design  $\Delta V$  values, the payload mass to LLO was calculated. Using this method, the payload to LLO was determined to be approximately 4.35 Mg, which is quite close to the previous value.

Another notable value is the cost and cost per Mg to LLO, especially when comparing with other launch vehicles. As noted in Appendix A by Rodion Callistru, a comparable launch vehicle when it comes to payload mass is the Delta IV heavy with 22.98 Mg to LEO, almost identical to the expendable Falcon 9 variant. But when it comes to cost, the Delta IV is \$350 million USD (Again, noted in Appendix) to the Falcon 9 being only \$62 million. Clearly when it comes to cost per Mg, the Falcon 9 is in a class of its own for medium launch vehicles with merely 17.7% of the launch cost of a comparable vehicle.

## Conclusion

The Falcon 9 is outclassed by few available launch vehicles today in terms of payload mass, but it is the superior medium launch vehicle in this category. Along with this, cost advantages

over all of its competitors coupled with a proven launch success rate contributed to our decision to choose Falcon 9 as the medium launch vehicle for Project NextStep. The application for a vehicle of this size in our scope may be limited, but when needed the Falcon 9 is the superior cheap alternative.

## 2. Falcon Heavy



**Fig. C1 Falcon Heavy [44]**

### Introduction

Another launch vehicle we considered for the purpose of transporting heavier payloads was Space Exploration Technologies Corporation's" Falcon Heavy launch platform. The Falcon Heavy is an extremely capable and versatile launch platform that can be used for a wide variety of payload and missions. The heavy platform, derived from the earlier Falcon 9, features a modified and enhanced Falcon 9 center stage with the addition of dual strap on side boosters which augment the

platform's capabilities. Development on the Heavy came shortly after SpaceX began successfully flying its predecessor, the Falcon 9. SpaceX first announced the Falcon Heavy publicly in 2011 and developed the rocket through early 2018[45]. The Falcon Heavy first launched in February of 2018, when it famously carried a Tesla Roadster into an interplanetary orbit. Following this maiden flight, Falcon Heavy flew two commercial missions in 2019, and SpaceX has plans for future launches, including two in July and October of 2021 for the United States Space Force[46].

### **Capabilities and Design**

One of the most compelling features of the Heavy is its reusability. Unlike many traditional launch platforms, sections of the Falcon Heavy have the ability to safely land back on Earth through the reignition of its Merlin engines. All three first stage cores have the ability to land on either sea or land based platforms, and these abilities have been successfully demonstrated in launches of the Heavy. This unique reusability helps to dramatically lower the price of using the launch platform, an attractive feature for a budgeted program.

In terms of capabilities the Falcon Heavy is unrivaled by any other launch system currently in private or government use. Claiming the title of "*The world's most powerful rocket*", the Heavy is capable of boosting nearly 64MG into LEO[47] and up to 15MG into LLO. The calculation of this payload capacity can be found in subsequent sections of the report. This impressive payload capacity makes it an attractive choice for any mission requiring a significant mass and volume of payload be delivered into orbit. In addition to this consideration, significant cost savings could be found using the Falcon Heavy as opposed to other launch systems, due to the need for fewer launches to deliver an equivalent payload mass to a desired orbit. It is also important to note that while the Falcon Heavy has a high payload capacity, the payload fairing used is identical to the Falcon 9, therefore similar volume constraints exist for Falcon 9 and Falcon Heavy payloads.

The powerhouse driving the Falcon Heavy is SpaceX's Merlin 1D liquid rocket engine. Powered by RP-1 and Liquid Oxygen, this engine uses a fuel rich gas-generator cycle to power its turbopumps and features a regenerative cooling jacket in the nozzle in order to maintain nozzle structural integrity[48]. The Merlin 1D possesses a robust thrust vectoring capability, allowing for precise trajectory adjustments during operation. Additionally, the engine has been designed to be sea recoverable, meaning it is salvageable if its entire booster falls intact into the ocean. The Falcon Heavy also uses a Merlin 1D vacuum engine, optimized to operate outside of the Earth's atmosphere, on its second stage.

Shown in Table C1 below are some important design parameters of both the Merlin 1D and the Merlin 1D vacuum engines.

|                             | <b>Max Thrust<br/>(kN)</b> | <b>Max Throttle<br/>Down</b> | <b>ISP (s)</b>      | <b>Expansion<br/>Ratio</b> |
|-----------------------------|----------------------------|------------------------------|---------------------|----------------------------|
| <b>Merlin 1D</b>            | 850[48]                    | 40%[48]                      | 282 (SL)[48]        | 16[48]                     |
| <b>Merlin 1D<br/>Vacuum</b> | 981[48]                    | 39%[48]                      | 365<br>(Vacuum)[48] | 165[48]                    |

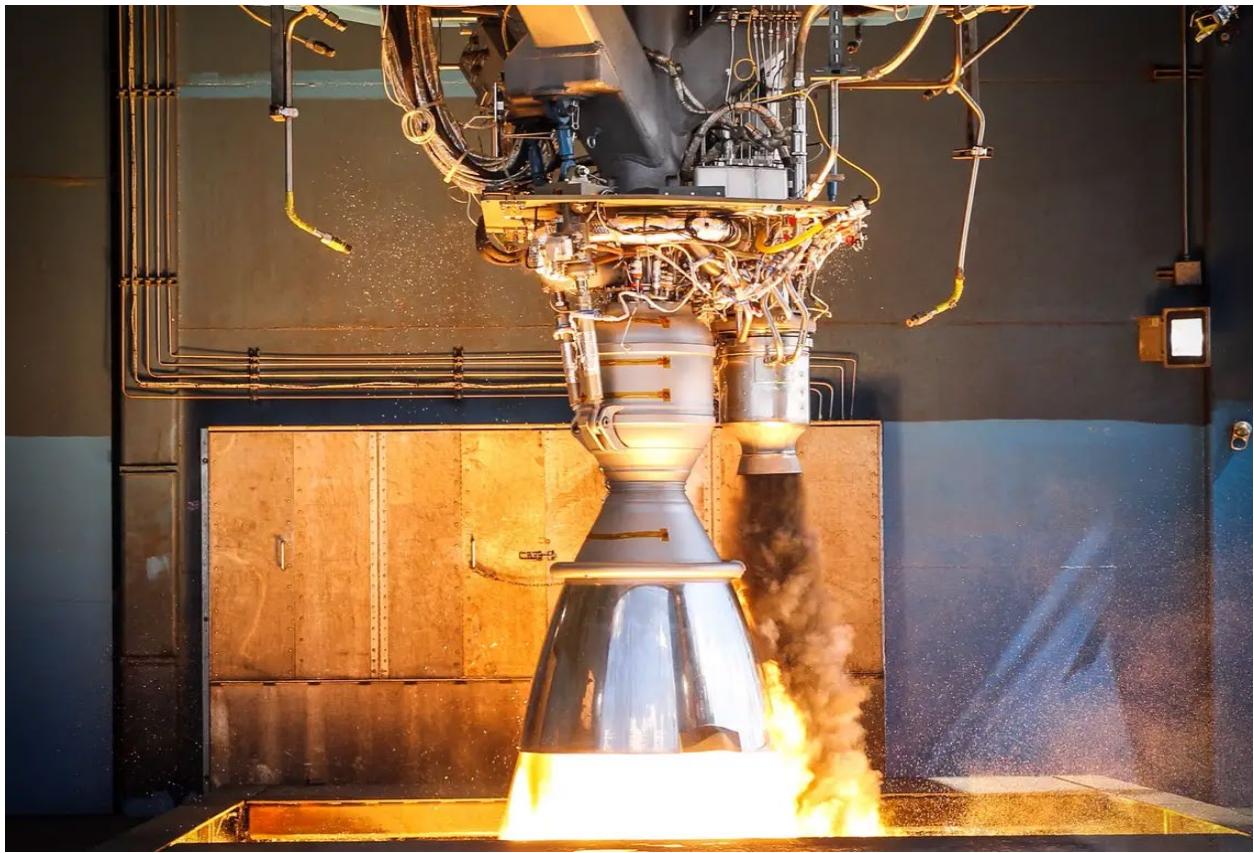
**Table. C1 Engine Comparison**

The Merlin engine has been designed for reusability, making it one of the more cost effective engines on the market. Each engine is carefully cleaned and inspected after each use to

ensure peak performance on each mission. Even though it has sent payloads past LEO, the concerns with this engine are with its restart after the long coast to the moon.

To put a payload in lunar orbit, the Merlin vacuum must first fire in LEO to place the payload on a proper trajectory, then must reignite to circularize the orbit around the moon. This trip takes approximately three days, in the cold vacuum of space. SpaceX has not tested this case for the engine, so we truly do not know how the Merlin will operate in these conditions. The major concern comes with the engine cycle and fuel selection. Using a fuel-rich gas generator with RP-1 and LOX propellants, it does not burn as “cleanly” as other fuels such as liquid hydrogen or liquid methane. This generates excess combustion products other than carbon dioxide and water vapor, which can leave soot deposits on engine components.

Inside the turbine this problem is most concerning, where it is driven by the fuel-rich combustion products from the gas generator. This incomplete combustion guarantees some sooting inside the turbine, and the blades inside are some of the most delicate components of the entire engine. The figure below illustrates the dark coloring of the gas generator exhaust.



**Fig. C2: Merlin Engine Test with Gas Generator Exhaust [50]**

The gas generator is located on the right of the image next to the main engine bell, with the dark colored gases exiting above the main plume. This dark coloration indicates products associated with incomplete combustion, and hydrocarbon fuels such as RP-1 will leave soot like deposits behind after firing. After the TLI burn, these deposits will sit in the engine at extremely cold temperatures for an extended period of time which could degrade all components of the engine. The high operating temperature and pressure of the turbine already place high stress on the rotating machinery, and leaving them exposed to potentially corrosive exhaust in extreme conditions on the trip to the moon could be mission threatening.

## General Overview

Table C2 below shows the summarized important design and performance characteristics for the Falcon Heavy platform.

| <b>Launch Vehicle</b>                  | <b>Falcon Heavy</b>             |
|----------------------------------------|---------------------------------|
| Expendable Cost                        | \$150 Million USD               |
| Reusable Cost                          | \$90 Million USD                |
| Liftoff Thrust (kN)                    | 22,819[47]                      |
| Liftoff Mass (MG)                      | 1421[49]                        |
| Tank Pressurization Method             | Helium                          |
| Payload to LEO (MG)                    | 64[47]                          |
| Reusable Payload to LLO (MG)           | 4.5*                            |
| Expendable Payload to LLO (MG)         | 14.7*                           |
| Fairing Volume (m <sup>3</sup> )       | 295.2                           |
| Fairing Dimensions (m)                 | 5.2 diameter by 13.9 length[49] |
| Expendable Cost per MG to LLO (M\$/MG) | 15.12                           |
| Reusable Cost per MG to LLO (M\$/MG)   | 20.25                           |

**Table. C2 Falcon Heavy Parameters**

These values for LLO payload are crucial to mission planning, as for the first three years of the project the reusable Falcon Heavy is the launch vehicle with the highest payload capability that is commercially available. It will be doing a majority of the payload transport in the initial

phases, so these masses were necessary to see first, if the Falcon Heavy had the capability to actually land anything on the moon in this mission phase, and second, what types of payloads could be transported to LLO. With these, other teams could make mass and volume adjustments to crucial early mission systems, as well as schedule launches based on the maximum payload at the given mission phase.

### **Conclusion**

In summary, the Falcon Heavy is an extremely versatile and powerful launch vehicle. Its capabilities are unmatched by any other operating launch system in terms of its competitive price point, reusability, and heavy payload capacities. This launch platform has advantages over alternatives given its implementation of modern technologies and relatively new design when compared to other heavy launch platforms. Additionally, the use of proven technologies in the form of the modified Falcon 9 platform helped to make this platform a strong candidate as a viable launch platform for our use in our mission. All of these factors contributed to our eventual decision to use the Falcon Heavy launch platform for portions of our mission. We will further develop and define the Falcon Heavy's role within our program in subsequent sections of this report.

### 3. Starship



**Fig. D1 Starship [51]**

### Introduction

Starship is a super heavy launch platform currently being developed and tested by Space Exploration Technologies. Building off of earlier lift vehicles such as the Falcon 9 and Falcon Heavy, Starship aims to become the most powerful operational rocket in existence, rivaling even the mighty Saturn V in terms of sheer size and power. This two stage rocket consists of a lower heavy booster stage, and a large upper stage dubbed “Starship”. Standing at over 394ft tall and capable of sending nearly 100 metric tons into LEO, Starship is poised to complete development within the next couple of years and begin regular commercial service[52].

### Capabilities and Design

The design of Starship is somewhat unique when compared to more traditional launch platforms. Like other SpaceX launch platforms, Starship will have the ability to land via the use of engine relight. This is true for both the Super Heavy Booster and the upper stage of the Starship launch

platform, although the manner in which each will land is unique. The Super Heavy Booster will slow down via a controlled vertical descent, using engine relights and stabilizer fins. The Booster will then touch back down on its original launch platform, and be “caught” by its drag fins via a crane apparatus mounted to the launch tower[53]. The upper stage descent profile will vary, and depends upon the environment in which it is present in. When landing in the atmosphere, the vehicle will first enter an angled “bellyflop” in order to create a drag force and slow the vehicle. When the craft comes close to the surface, its engines will relight, and use gimbaling to orient the vehicle in a 90 degree fashion, before coming to a stop on the ground. This unique initial descent profile sets Starship apart from other SpaceX launch vehicles such as the Falcon 9 and Falcon Heavy.

The launch capabilities of Starship will be unrivaled to any other operational launch vehicle in terms of raw lifting capacity and versatility. For a purely cargo mission, Starship will be able to launch 100MG into LEO, or an estimated 90.7MG into LLO. Additionally, Starship will have the capability to be configured for transport of up to 100 passengers if deemed necessary[52].

### General Overview

Table D3 below shows the summarized important design and performance characteristics for the Starship platform.

| Launch Vehicle      | Starship         |
|---------------------|------------------|
| Cost                | \$873.39 Million |
| Mass (MG)           | 5000[52]         |
| Liftoff Thrust (kN) | 76000[52]        |

|                                  |                             |
|----------------------------------|-----------------------------|
| Tank Pressurization Method       | Helium                      |
| Payload to LEO (MG)              | 100[52]                     |
| Payload to LLO (MG)              | 90.7                        |
| Fairing Volume (m <sup>3</sup> ) | 1100[52]                    |
| Fairing Dimensions (m)           | 9 diameter by 18 length[52] |
| Cost per MG to LLO (M\$/MG)      | 9.63                        |

**Table. D3 Starship Parameters****Conclusion**

In conclusion, Starship is the most powerful and versatile launch platform that can be realistically considered for use in our project. The launch platform is at an advantage to every other currently in service rocket given its enormous payload capacity and landing capabilities, with the assumption of course that Starship will be commercially operational by the year 2023. Overall, the large advantages Starship holds over any comparable heavy launch system led to its selection as our primary launch vehicle in the post 2023 phase of our mission.

#### 4. Space Launch System (SLS)



**Fig. E1. A size comparison between SLS and Space-X's Falcon heavy [54]**

### Introduction

The Space Launch System is a heavy lift launch vehicle that NASA has had in development since its conception in 2011 for the Artemis Program. The Artemis Program aims to land the next man and first woman on the Moon. The Space Launch System is designed to produce more thrust than Saturn 5, the only launch vehicle to transport humans to the lunar surface. Upon completion, this launch platform will have both crew and cargo transportation capabilities to lunar orbit and the Moon. The Space Launch System has not yet been tested on a full scale due to unexpected design and technical challenges but it's maiden flight is scheduled in 2024 for the Block 1 design. Continuous improvements in the design and operation of SLS are promised through the Block 1B and Block 2 of the Space Launch System into the late 2020s.

## Implementation

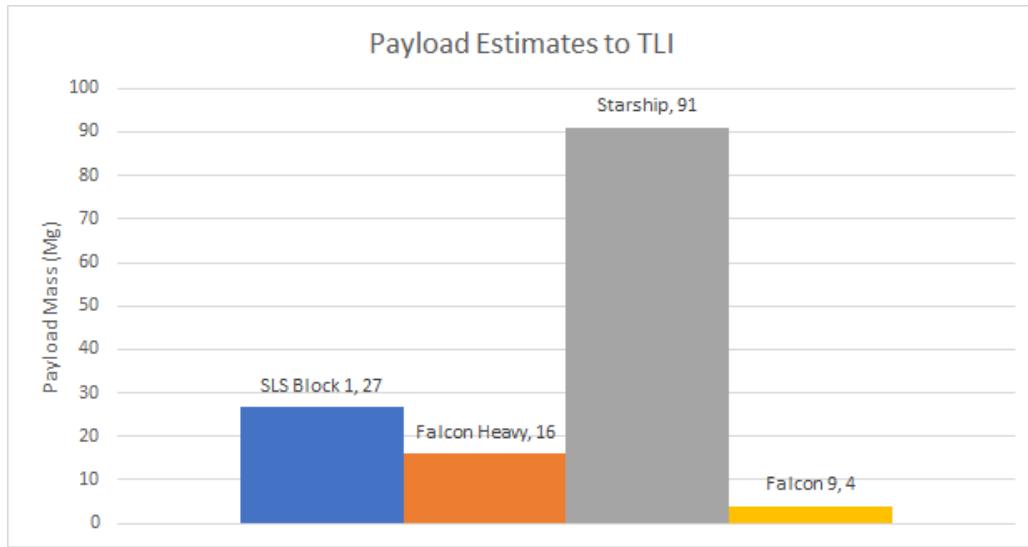
The Space Launch System is dependent on a number of other contracts and technologies: Lockheed Martin's Orion Capsule, Lunar Gateway, and the Human Landing System contract. Orion is used to take astronauts to the Lunar Gateway, and is now ready to be used as a crew capsule on the Space Launch System. After completing a trans-lunar injection burn, Orion will separate from the second stage of SLS and dock with the Lunar Gateway [55]. The Gateway is still under development as both stationkeeping needs and life support functions are required. Using the Power and Propulsion Element (PPE) designed by Maxar technologies, the gateway will be able to maintain itself in low lunar orbit [56]. Another crucial piece of Gateway is the Habitation and Logistics outpost (HALO) that will provide shelter for astronauts before landing on the Moon [56]. It will also have the capability for multiple vehicles to dock, and for our purposes: bring another 2-3 Moon colonists. The Lunar Gateway will be launched aboard a Falcon Heavy in 2024 in preparation for the Artemis missions [56].

The Human Landing System (HLS) contract is the last piece of the puzzle and has Space-X, Blue Origin, and Dynetics in competition to produce the most promising lunar lander. The current leader is Blue Origin with its lander Blue Moon, and is expected to be available for human use in 2024. It will ferry up to six astronauts from the lunar Gateway to the Moon's surface [56]. Space-X approached the contract from a different perspective. They are currently working on Starship as their entry for the HLS contract and their goal is to propulsively land the Starship upper stage on the Moon or martian surface. Dynetics is taking a straightforward approach and is also designing a lander, similar to Blue Moon, but have kept the project under wraps after winning the contract from NASA.

## Capabilities and Variations

### SLS Block 1

Block 1 will be the first iteration of the Space Launch System, available for use in the later half of 2024. The core stage of block one is powered by four RS-25 engines made by Aerojet Rocketdyne and previously used in the Space Shuttle Program. These engines have been modified and outfitted to function on SLS and have caused several delays in the overall project timeline. An RS-25 uses liquid hydrogen and liquid oxygen propellants, which is an outlier in our other examined vehicles; most use Rocket Propellant 1 (RP-1) as fuel with the exception of Starship which uses liquid methane. Also attached to the core stage, are 2 solid rocket boosters propelled by PBAN fuel and generating 16,000 kN of thrust each, making up 80% of the overall thrust at sea level [57]. To clarify, PBAN is Polybutadiene acrylonitrile, a very common propellant used in solid rocket boosters and formally used to carry the Space Shuttle into Low Earth Orbit [57]. After stage separation, the Interim Cryogenic Propulsion Stage carries Orion to Gateway. Block 1 has two variations: Block 1 Crew and Block 1 Cargo. Both can take 27 Mg to trans-lunar injection but a crewed mission comes with 14.6 cubic meters of payload space versus 229 cubic meters for a cargo launch [58]. Figure E1 shows how Block 1 stacks up against the leading options for launch vehicles.



**Fig. E2. SLS block 1 comparison to Falcon 9, Falcon Heavy and Starship in terms of mass to lunar injection**

### SLS Block 1B and Block 2

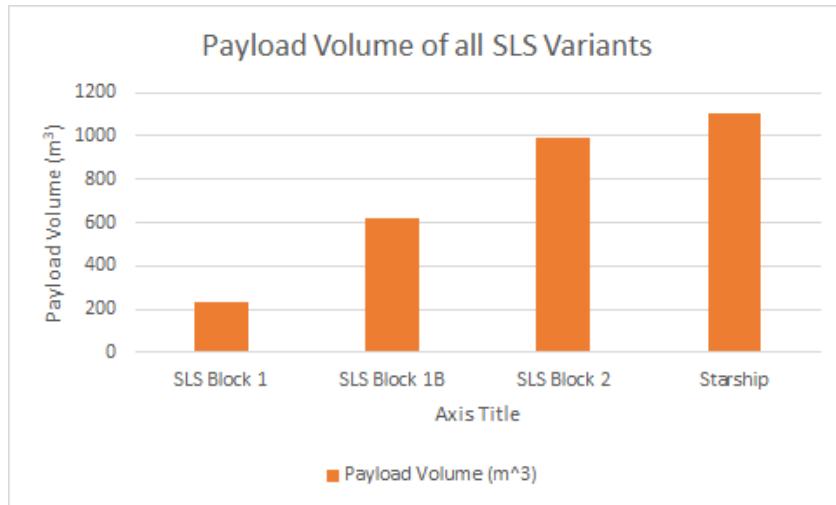
NASA has promised that the Space Launch System Program will improve over time and that a new version of SLS will become available; namely Block 1B and Block 2 variants. The main difference between them and Block 1, is that the Exploration Upper Stage (EUS) is used over the Interim Cryogenic Propulsion Stage (ICPS) that was employed on Block 1. The EUS will use 4 upgraded RL-10 engines producing 409 kN of thrust each, quadruple that of the ICPS [58]. With the increased performance, much larger payloads to trans-lunar injection are possible, rivaling the capability of Starship. All SLS variants are compared in Table E1 . The single launch cost of SLS has fluctuated over past years and has not been published by NASA. Current estimates price an SLS launch around 2 billion dollars after finishing development [59].

| Cargo Vehicle                    | SLS Block 1 | SLS Block 1B | SLS Block 2 |
|----------------------------------|-------------|--------------|-------------|
| Thrust (kN)                      | 39,000      | 40,000       | 42,250      |
| Fairing Volume (m <sup>3</sup> ) | 229         | 621          | 988         |
| Payload Mass to TLI (MG)         | 27          | 38           | 46          |
| Payload to LEO (MG)              | 95          | 110          | 130         |
| Expected Launch Time             | 2024        | 2026+        | 2028+       |
| Cost per MG to TLI<br>(M\$/MG)   | 74.1        | 52.6         | 43.47       |

**Table E1. Performance values of SLS variants [58]**

### Conclusion

No launch vehicle is currently in operation to transport humans to the Moon. If we are able to wait until 2024 for the first SLS crewed launch, it would be a very reliable option. NASA has developed many capable launch vehicles in the past and we have no reason to believe that SLS will be any different. The continuous improvement over the next decade in the SLS launch platform is also very attractive. No other vehicle besides Space-X's Starship has a larger fairing volume of 988 cubic meters or a payload to TLI of 46 Mg [58].



**Fig. E3. Comparison of Payload volumes of all SLS variants against Starship**

However, the price tag that comes with SLS is nothing short of crippling. The 9 billion of dollars sunk into development thus far makes it a one off option at best [60]. A single launch would spend 20% of our one year budget, \$2B out of \$10B, and not enough cargo or colonists can be brought to justify using the Space Launch System, at least with the Space Launch System Block 1. The high cost also comes from the lack of reusability which other vehicle options have. SLS also relies on many other technologies such as HLS, Gateway and Orion and any setbacks in these products would also delay SLS further. While it is highly capable and reliable, we decided to stray away from SLS due it's high cost and possibility of delay. However, including SLS in the consideration process does offer some redundancy in the case that Starship or another one of our vehicles fails. It also could be used in the case that Starship is delayed significantly and no astronauts have been delivered to the Moon.

### C. Estimation of $\Delta V$ Splits Between First and Second Stages: Falcon 9 and Falcon Heavy

Falcon 9 and Falcon Heavy were two of the very first launch vehicles considered in the early stages of the design process. Space-X streams each of their launches and broadcasts live telemetry data to the screen for velocity and altitude. Using the numbers from Starlink missions, Transporter missions and Arab-Sat missions, we roughly determine how much  $\Delta V$  is imparted by the first stage just before stage separation occurs. These early estimates are used to inform preliminary propellant mass estimates and examine feasibility with given  $\Delta V$  estimates from the Mission Design team. An analysis of Falcon 9 using this method can be seen below in Table F1.

| Falcon 9      |                 |        |                                    |                                  |                           |
|---------------|-----------------|--------|------------------------------------|----------------------------------|---------------------------|
| Mission       | Payload<br>(MG) | Orbit  | First Stage<br>$\Delta V$ (km/sec) | 2nd Stage<br>$\Delta V$ (km/sec) | Total $\Delta V$ (km/sec) |
| Transporter 1 | 5               | 500 km | 1.97<br>25% of total $\Delta V$    | 5.97<br>75% of total $\Delta V$  | 7.94                      |
| Starlink 8    | 15.4            | 218 km | 2.17<br>28% of total $\Delta V$    | 5.34<br>72% of total $\Delta V$  | 7.54                      |
| Starlink 16   | 15.6            | 216 km | 2.19<br>29% of total $\Delta V$    | 5.3<br>71% of total $\Delta V$   | 7.49                      |

**Table F1. Early  $\Delta V$  split estimates for Falcon 9**

Using the data in Table F1, it would be fair to assume that the first stage of Falcon 9 delivers roughly 28% of the overall mission  $\Delta V$ . However, this statement would not be fully accurate because the payload size for Statlink 8 and Starlink 16 are much larger than what is possible to

bring on a Falcon 9 to the Moon. Those missions had abnormally large payloads, upwards of 15 Megagrams. Based on our calculations and Space-X, the Falcon 9 is able to deliver a 4 megagram payload to the Moon or Mars using a trans-lunar injection burn. Using the same analysis for Falcon heavy yields slightly different results as shown in Table F2.

| Falcon Heavy |         |        |                   |                   |          |
|--------------|---------|--------|-------------------|-------------------|----------|
| Mission      | Payload | Orbit  | First Stage       | 2nd Stage         | Total ΔV |
|              | (MG)    |        | ΔV (km/sec)       | ΔV (km/sec)       | (km/sec) |
| ArabSat 6-A  | 6       | 165km  | 2.97              | 4.45              | 7.42     |
|              |         |        | 40% of total ΔV   | 60% of total ΔV   |          |
| STP-2        | 3.7     | 315 km | 3.1               | 4.34              | 7.44     |
|              |         |        | 41.7% of total ΔV | 58.3% of total ΔV |          |

**Table F2. Falcon heavy ΔV splits based on past launches**

In this case, the ΔV splits are exactly opposite from what was seen in Falcon 9. Falcon Heavy is capable of taking 63 Mg payloads to Low Earth Orbit and both the Arab-Sat 6A mission and STP-2, very small payloads were transported [61]. This data also suggests that more ΔV is imparted from the first stage of Falcon Heavy. Intuitively this makes sense, as Falcon Heavy uses two Falcon 9's in addition to the core stage or essentially, three Falcon 9's.

## D. Starship Propellant Estimation

The same  $\Delta V$  split data is especially hard to come by for Starship, as the Super Heavy Booster is still in development and there is lack of performance data for Starship as a whole. Early estimations to investigate starships propellant usage was done using assuming that 40% of the total LEO  $\Delta V$  would be imparted by Super Heavy, based on historical evidence and early guesses. A Matlab Script is also generated to observe propellant usage in getting to an earth parking orbit using Starship called upper\_stage\_masses.m:

```
% Constants and assumptions
deltav = 8650; % m/sec, constant from mission design
min = 122182; % kg, inert mass
mpay = 5000; %kg, payload mass estimate from human factors
rho_lox = 1141; %kg/m^3, density of liquid oxygen
rho_ch4 = 422.6; %kg/m^3, density of liquid methane
r = 3.52; % mixture ratio
g = 9.81; % m/s^2
isp = 370; % sec estimate

% propellant mass guess
lambda = .9; % propellant mass fraction
% propellant mass calculation using prop mass fraction
mp = lambda * min / (1-lambda);

% initialize index, loop on delta v split
i = 1;
for split = 30:0.5:100
    deltav_new = deltav * split/100;
    leftover_propellant(i) = (mp + mpay + min)/exp(deltav_new/g/isp) - mpay - min;
    propellant_burned(i) = mp - leftover_propellant(i); % kg
    i = i+1;
end

split_vector = 30:0.5:100;
% assuming the first stage imparts 40% of the total delta v
index = find(split_vector == 40);

figure
plot(split_vector,propellant_burned)
hold on

% payload mass vector
mp_vector = linspace(mp,mp,length(split_vector));
```

```
plot(split_vector,mp_vector, "r")
x = linspace(40,40,100);
y = linspace(propellant_burned(index),mp);
plot(x,y,:k")
title("Starship Propellant burned Getting to Earth Parking Orbit")
xlabel("%\delta V Imparted by Second Stage")
ylabel("Mass of Propellant (kg)")
legend("Propellant Burned", "Total Available Propellant","Location","Southeast")

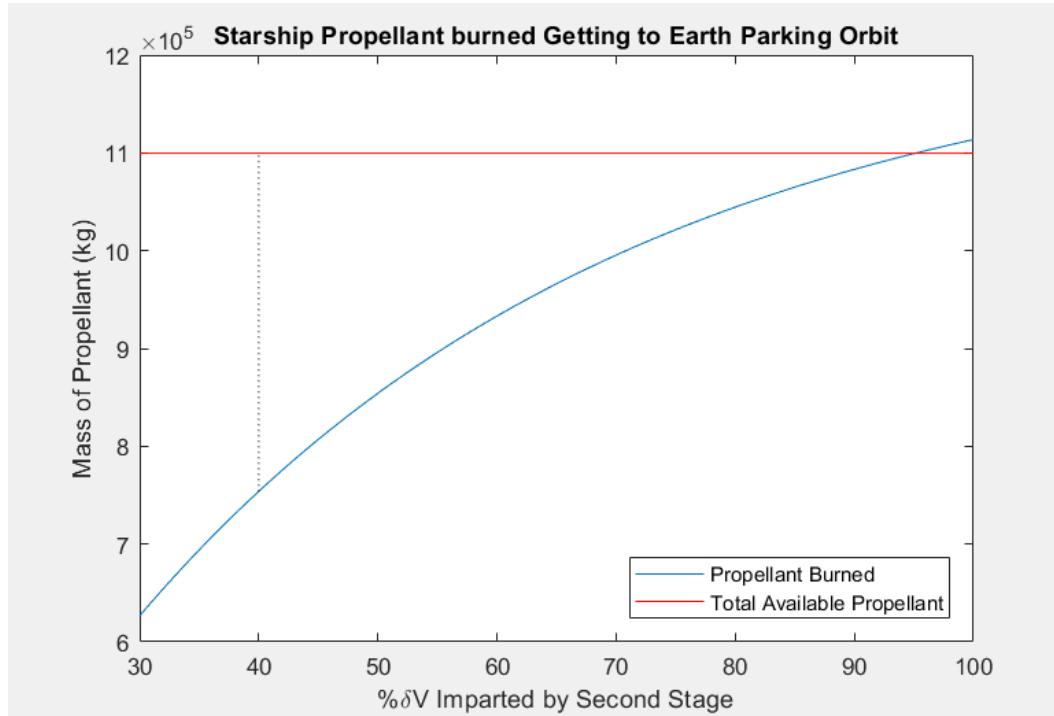
m_ch4 = mp / (1+r); % mass of methane using mixture ratio
m_lox = r*mp / (1+r); % mass of Liquid Oxygen using mixture ratio

% computing rough fuel costs for propellants
% in dollars
cost_ch4 = 1.35 * m_ch4;
cost_lox = .16 * m_lox;
total_prop_cost = cost_ch4 + cost_lox;

% TLI delta V
deltaV_moon = 4000; % m/sec
m_fuel_left = mp - propellant_burned(index);

% mass in lunar parking orbit
mfinal = (min + mpay + m_fuel_left) / exp(deltaV_moon/(g*isp));
% calculate propellant needed to get to TLI with no prop left
mfinal = min + mpay;
mprop_TLI = mfinal*(exp(deltaV_moon/(g*isp)) - 1)
```

**upper\_stage\_masses.m Results and Explanations:**

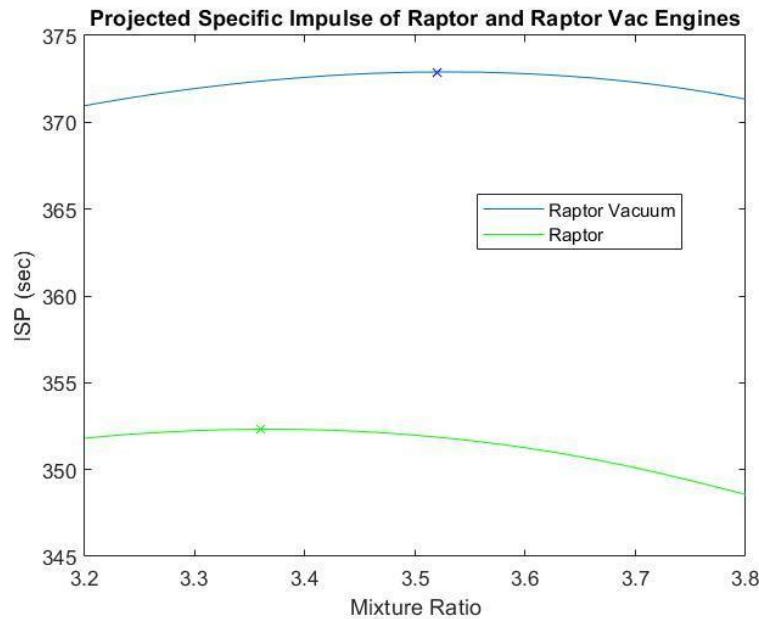


**Fig. G1. Propellant required for Starship to get to LEO when the  $\Delta V$  varies from 30%–100%**

The significance of Fig. G1, is that when 40% of the  $\Delta V$  is imparted by Super Heavy, shown by the black vertical line, Starship's upper stage will have 345,000 kg of fuel leftover. The script also calculates the minimum amount of fuel required for Starship to reach the Moon through a trans lunar injection burn which comes out to 255,670 kg of fuel. This means that Starship will be left with only 8% of it's total fuel capacity for a low lunar orbit circularization burn and a propulsive landing on the moon's surface. These burns would not be possible for the amount of  $\Delta V$  required and this was when the team decided to investigate the idea of Earth and lunar fuel depots to ensure Starship was sufficiently fueled.

In calculating these propellant numbers, a number of assumptions were made. The specific impulse of the Raptor and Raptor Vacuum engines were estimated using NASA's Chemical Equilibrium Analysis (CEA) and an average of the two was used in the code. With Starship still

in development, we cannot estimate all of Raptor or Starships performance parameters easily. One of the given numbers for specific impulse from a quote from Elon Musk was 380 seconds, which for most engines is very high performance. We decide to check this number for accuracy, as it can very max payload and rocket performance significantly. Specific impulse depends on several factors. The first is the type of fuel and oxidizer used; typical LOX and RP1 engines have specific impulse in the 320-340 second range, while Starship uses LOX and Methane. The next factor is the expansion ratio of the engine, or the exit area of the nozzle. With different expansion ratios, Raptor and Raptor Vacuum engines have different specific impulse values. Altitude also plays a role here, as the pressure thrust at low altitudes is negative and the pressure thrust is maximized in the vacuum of space. The last factor is the mixture ratio, which is closely affected by the propellant types. All of these factors were input into the NASA CEA Matlab wrapper to determine optimal mixture ratios and corresponding specific impulse values.



**Fig G2. Optimal Mixture Ratios And Corresponding Maximum Specific Impulses for both Raptor and Raptor Vacuum Engines**

A propellant mass fraction was also assumed to be  $\lambda = 0.9$ . This was based on historical evidence of other launch vehicles such as the Saturn 5 and Atlas IV [62]. This assumption was later used to compare calculated propellant masses to Space-X's published propellant masses. The result was within 1% of Space-X's published numbers, implying that the propellant mass fraction guess was sufficient. Another important finding of this script is the total costs to fuel a Starship:

| Liquid Methane                   | Liquid Oxygen     |
|----------------------------------|-------------------|
| 243.3 Mg required                | 856.4 Mg required |
| \$1.35/kg [63]                   | \$0.16/kg [63]    |
| Total Propellant Cost: \$465,000 |                   |

**Table G1. Starship fuel costs**

## E. Concerns of Falcon 9 or Falcon Heavy Upper Stage Re-ignition in Deep Space

### 1. Zero-g Propellant Slosh

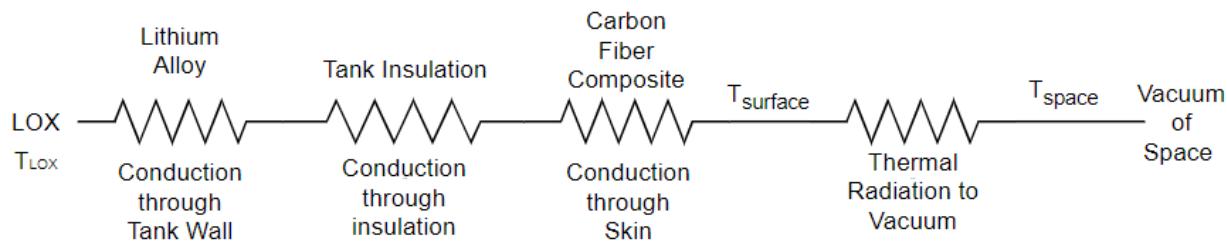
During the firing of a rocket engine, propellant in fluid form is continuously sucked out of its tank and through a turbopump that routes the propellant to the injector for combustion. As long as the engine is firing, the fluid is suctioned out without risk. However, when the engine and turbopumps are turned off, the fluid is no longer being pulled out of its tank in the same way. In space, with zero-g, these liquid propellants will float around in the tank and they no longer sit at the bottom where the suction point is. Trying to restart the engine risks damaging the turbopumps, as they are designed to pump liquids to high pressure and can break and when pumping gas. If a mixture of liquid and gas is pumped, combustion is “sputtery” and the structural integrity of the vehicle is put at risk. The Saturn V rocket made use of small ullage motors, which provide a very small jolt of acceleration in space, forcing the liquid to the bottom of the tank. We believe SpaceX approaches this problem in a similar way, as the Merlin 1-D Vacuum engine is currently capable of in space re-ignition, although this is not commonly advertised.

### 2. In Space Boil-off of Cryogenic Propellants

A drawback of using cryogenic propellants is the difficulty of storing them without heating up. If liquid oxygen is allowed to heat up in its containment vessel, thousands of kilograms of the propellant are lost from boil-off. On earth, this problem is easily avoided. Fuels such as liquid oxygen or liquid hydrogen are chilled on the ground before being pumped into the rocket for launch. In space, the sun is able to heat spacecraft to very high temperatures, around 400 K [64]. The vacuum in space means that heat is unable to escape through convection or conduction due to the lack of particles. Therefore, outside the spacecraft, radiation is the dominant mode of heat transfer. In short, the heating from the sun in a vacuum causes the boil-off of cryogenic propellants

in space. This is especially relevant to us as the upper stage makes it's four day journey to the Moon.

To analyze this problem, a thermal resistance network is created to estimate a rate of heat transfer across the network. The thermal resistances are all in series, due to the lack of particles in the vacuum of space. From the liquid oxygen (LOX), conduction through tank walls is followed by conduction through tank insulation, and then conduction through the carbon fiber of the rocket's skin. After that is radiative heat transfer from the vacuum.



**Fig. E1 Thermal resistance network of the Falcon 9 and Falcon Heavy upper stage**

Assumptions made:

- Heat transfer occurs in 1-D, the radial direction. This is an underlying assumption of all thermal resistance models.
- The surface temperature of the spacecraft is at an average temperature of  $(398K+2.7K)/2 = 200.3K$  to account for a rotating spacecraft that is heated evenly [64].

The following thermal and geometric values were used in the calculation of rate of heat transfer:

| <b>Property</b>                                              | <b>Value</b> |
|--------------------------------------------------------------|--------------|
| <b>Temperature of LOX</b>                                    | 106K         |
| <b>Temperature of skin</b>                                   | 394K         |
| <b>Tank Thickness</b>                                        | .4 inches    |
| <b>Thermal Conductivity of Tank (Lithium Alloy)</b>          | 36.2 W/mk    |
| <b>Skin Thickness</b>                                        | 3/16 inch    |
| <b>Thermal Conductivity of skin (Carbon Fiber Composite)</b> | 6 W/mk       |
| <b>Heat of Vaporization of LOX</b>                           | 214 kJ/kg    |
| <b>Thickness of insulation</b>                               | .4 inches    |
| <b>Thermal Conductivity of Insulation</b>                    | 15 W/mk      |

**Table E1. Thermal properties and geometric values of thermal resistance network.**

The Matlab script inspaceBoiloff.m was created to implement the thermal resistance network and analyze boil-off:

```
% Falcon 9 tank volumes upper stage
Vlox = 65.9; % m^3 of liquid oxygen can be held

upper_stage_prop_mass = 92000; % mass in kg from source
density = 1141; % kg/m^3

% add extra space for ullage, boiloff, etc
Vlox = Vlox * 1.1/30; % tank is partially full, does not affect end percentage
propellant_left = Vlox*1141; % calculates propellant mass from volume and density
T_LOX = 106; % kelvin, boiling point of LOX at 4 bar pressure
T_skin = (2.7+398)/2; % kelvin, average skin temp value
```

% thermal conductivities

k\_LA = 34; % W/mK

k\_CC = 6; % W/mK

k\_insulation = 15; % W/mk

% boltzmanns constant and emissivity for radiation

sigma = 5.67e-8;

e = .85;

% geometric properties of tank

diameter = 3.5; %tank diameter, meters

r = diameter/2; % radius of tank, in meters

height = Vlox / pi\*(diameter/2)^2; % tank height after adjusting for being partially full

% surface area not including endcaps

surface\_areal = height \* pi \* diameter;

% thickness

t\_tank = .1016; % meters

t\_skin = 0.047625; % meters

% Insulation layers same as NASA ZBOT tanks

insulation\_layers = 34;

% thermal resistances

```

R1 = t_tank / (surface_area1 * k_LA); % thermal resistance 1, tank
R2 = t_skin / (surface_area1 * k_CC); % thermal resistance 2, rocket skin
R3 = t_tank * insulation_layers / (surface_area1*k_insulation); % thermal resistance 3, insulation
Rrad = 1 / (sigma*e*surface_area1 * (T_skin^2+3^2) * (T_skin+3)); % thermal radiation
resistance

% total thermal resistance
R_total = R1+R2+R3+Rrad;

Qdot = (T_skin - T_LOX) / (R_total); % watts, overall rate of heat transfer
Hvap = 214000; % J/kg, heat of vaporization

time = 4 * 24 * 60 * 60; % time of journey in seconds
joules = Qdot*time; % heat transfer in joules
kg_boiloff = joules/Hvap; % kg of liquid nitrogen boiled off

percent_boiloff = kg_boiloff / propellant_left * 100; % percentage of nitrogen boiled off from the
start of 4 day journey

```

The script, inspaceBoiloff.m, reports that in the current configuration, 51% of the liquid oxygen started with is lost to boil-off on the 4 day trip to the moon. To increase the fidelity of the model further, a Computational Fluid Dynamics simulation is required.

There are many solutions to limit this phenomenon and NASA has done some investigation in a program they call Zero Boil-Off Tanks (ZBOT). NASA observed boil-off on

the ISS and tried to negate it using insulation and cryocoolers. There are also more experimental ideas such as a sun shield to block the sun's rays from the spacecraft therefore limiting surface temperature. An even simpler solution is to rotate the spacecraft, not allowing the sun to heat one side excessively and keeping the overall surface temperature low. One of these solutions should be implemented if a re-ignition of the Falcon upper stage is needed after a trans lunar injection burn.

## II. Launch Mission Tool

The following section describes the purpose, motivation, and development of a MATLAB script, named the Launch Mission Tool, created by members of the Launch Vehicle team. We develop the tool to allow users to input various parameters relating to a mission payload. The tool then finds the ideal launch vehicle and associated parameters of the launch for the intended payload.

### *1. Motivation and Purpose*

Prior to our development of the Launch Mission tool, teams worked on the launch schedule for the Moon colony by assuming the launch vehicles necessary for their payloads. To aid in formulating the launch schedule, we come up with a means of helping teams decide the ideal launch vehicle for their payload, depending on their payload size, destination, and when they would be launching said payload.

Ultimately, the purpose of the Launch Mission tool is to offer a simple and user-friendly way to find which launch vehicle is best for a certain payload. With a known payload mass, destination, and the approximate time during the mission, a user runs the tool and easily knows the best rocket, cost, and other launch parameters. We hope to provide a helping hand into launch scheduling for the Moon colony mission and also set the amount of mission budget that would go toward launches each year by using the tool.

### *2. Version 1*

In the initial version of the Launch Mission tool, we decide to base the entire function algorithm on single user-specified input values for a payload intended to go to the Moon, whether that be lunar orbit or lunar surface. We present a final function output as a short report

outlining the ideal launch to lunar destinations, including the ideal launch vehicle that should be used for the inputted payload.

### *Inputs*

The initial user-specified inputs include:

- Time of mission, indicating when during the 10-year mission timeline your mission would be taking place
- Destination, where in the lunar vicinity the payload would be reaching
- Payload mass, the mass of the payload in kilograms

The MATLAB function has guidance text which outlines specifications of the inputs for the user. The inputted time of mission is initially a single string, either ‘pre 2023’ or ‘post 2023’. We decided to split the entire mission into these two intervals to designate when Starship would be a viable launch vehicle option for the payload in question. As mentioned in previous sections, we assume the Starship launch vehicle is unavailable for usage until 2023 based on claims made by Elon Musk [65]. The tool then presents only Falcon 9 Expendable, Falcon Heavy Reusable, or Falcon Heavy Expendable as possible launch vehicles with a user-specified input of ‘pre-2023’ and the additional option of Starship with an input of ‘post-2023’.

Using the tool, the options for the user-specified input for payload destination are either ‘lunar orbit’ or ‘lunar surface’. All previously mentioned launch vehicles can be used for a ‘lunar orbit’ mission during their respective time intervals as outlined above, with the entire rocket payload capabilities, outlined in preceding launch vehicle sections, available for the effective payload. For a ‘lunar surface’ mission prior to 2023, Falcon 9 Expendable and Falcon Heavy Expendable serve as the two viable options before Starship availability. From Space.com [66], the Falcon 9 rocket is set to launch a small private Moon lander called Nova-C to the lunar surface in mid to late 2021.

With the estimated number for effective payload of the lander at 100 kg and the total lander size of about 2000 kg, we assume initially that the Falcon 9 is able to send an effective payload of 2100 kg to lunar orbit and 100 kg to the surface using the Nova-C lander. The Sky Crane at the current point in tool development is assumed as a viable Moon lander, paired with a Falcon Heavy Expendable Launch, for payloads going to the lunar surface. Based on an analysis from Propulsion team member Joey Heying, the effective Sky Crane payload is constrained to 1025 kg to the lunar surface. Post-2023 launches inputted into the tool include Starship as an added option for larger payloads exceeding Nova-C and Sky Crane limitations. Once available, the Starship vehicle serves as the primary means of landing mission payloads on the surface. Values for Starship payload capabilities reflected in the tool can be found in preceding sections.

Lastly, the user specifies a payload mass as the final input to the tool. The initial tool uses mass-based logic to choose a launch vehicle. We set the payload mass ranges within the code to decide which launch vehicle is best for the given payload, based on each launch vehicle's capabilities. Typically, smaller payloads use the Falcon 9, mid-range payloads use the Falcon Heavy, and the largest use Starship, once the vehicle is available.

### *Outputs*

As the Launch Vehicle team, our goal consists of presenting the launch mission of each user-specified payload input in a clear and easily understandable manner.

Following an enthusiastic greeting, “Here is your mission!”, the initial Launch Mission function outputs include:

- Launch Vehicle: the name of the ideal rocket for launching the inputted payload
- Cost: total cost of using launch vehicle for payload launch in millions of dollars

- Volume constraints: the dimensions of the fairing or total fairing volume (if known) in m<sup>3</sup>
- Thrust: the thrust of the launch vehicle in kilonewtons
- Lander: the lander used for the mission if ‘lunar surface’ destination is specified

If the mission is not possible at the inputted time of mission and payload destination, the function presents a message to the user saying, “Launch of the specified parameters is not possible at this mission time”. The code outputs the message if, for example, the payload mass exceeds the limitations of Falcon Heavy Expendable prior to 2023 when Starship is not available.

### *3. Version 2*

The code was updated by Trevor Glascock after the initial implementation in order to optimize and improve upon the initial code architecture. There were several changes and improvements made to the code in the update. First, the cost of Starship was adjusted to an estimate of \$2-\$10 million in order to reflect on Elon Musk’s comments on the estimated launch cost for production models of Starship. This number was, however, later updated to a cost in the hundreds of millions in later versions of the code. This was due to a more detailed cost analysis done by our team, which revealed that Musk’s assumptions were likely unrealistic given current demand for spaceflight. The primary modification done to the code was adding an array functionality. Previously, the code could only take singular string inputs and subsequently produce singular outputs at a time. We added array functionality in order to allow for analysis of any number of inputs at a given time. This was done so teams would be able to produce outputs for multiple launch scenarios while significantly simplifying the process and reducing the amount of time taken to run the script. Lastly, a caveat was added to the code which warned the user that selecting any

vehicle post 2023 other than Starship would be economically inefficient. However, given the updated Starship cost estimate, this revision to the code became redundant in later versions.

#### *4. Version 3*

Version 1.3 was again updated with array functionality, allowing users to input a vector of payload masses and a string of destinations and mission times. A looping mechanism was added to the tool, where the script first checks if the mission is possible, then stores the optimal launch vehicle with proper costs and constraints. These are all then outputted to the command window for the end user.

To allow this new addition to work, some fundamental changes had to be made regarding the user inputs and program outputs. Initially, the tool was written using a character array to store the variables, where MATLAB treats each individual character in a word as value in the array. When looping through multiple inputs, this will not work with a char array, as the words are all different lengths so the program treats them as different sized arrays. This will break the loop when concatenating multiple values. This issue was solved simply by converting all of the char inputs to a string. The difference is that MATLAB now recognizes each word or phrase inside the quotation marks as a single array value. This allows the vectors of inputs and outputs to stay the same length regardless of the letter amount, allowing the loop to run properly.

With this updated functionality, users can determine their launch vehicles as before but now can also test different mission times and payloads together to optimize the cost of their launch. For example, if there is a system that is scheduled to launch in the first few years, but not mission critical, the user can test both before and after Starship introduction for this specific payload. The individual outputs can then be compared for a cost optimized solution, and repeated as many times as necessary.

### 5. Final Version Updates

To improve the functionality and accuracy of the Launch Mission tool, we add numerous updates to conclude with the cheapest and most ideal launch for given payload parameters. We apply updates to the user-specified inputs, outputs, logic, launch vehicle capabilities and parameters, and outputs to the user from the tool.

#### *Inputs*

The final version of the Launch Mission function retains the previous user-specified inputs of destination and time of mission, while introducing three new inputs to refine user-friendliness and accuracy:

- Mission Title: title of each mission for the output launch report
- Payload Mass: the mass of the payload in megagrams rather than kilograms
- Payload Volume (optional): the volume of the payload in cubic meters

We also add another option for a user-specified destination of low earth orbit, inputted as ‘LEO’, specifically for the low earth orbit fuel depot launches. We update the function help text to include these changes for the user in order to run the function correctly. If the user specifies the inputs as arrays, the inclusion of the mission title improves readability for each outputted launch report.

With the inclusion of the optional fifth input, the user is able to specify the volume of their payload if known. We include additional logic statements to take the parameter into account and ensure the volume in addition to the mass of the payload can be accommodated by the chosen launch vehicle. If the user does not specify the payload volume, the function assigns the variable to an array of zeros. The user must then keep in mind the volume constraints for the particular chosen launch vehicle given in the launch report output.

Within the final version of the tool, we also update the Starship costs per launch based on Power and Thermal team member Noah Stockwell's cost analysis.

By default, MATLAB chooses the first true conditional statement to continue a script or function. We organize the logic of the function such that the first launch vehicle able to accommodate the given payload parameters is the cheapest one available. Thus, the final launch budget outputted to the user is ideal.

### *Outputs*

The outputs of the final tool are the same as Version 1 aside from two additions:

- Remaining Payload Mass: the remaining amount of effective payload mass for the launch in megagrams
- Remaining Payload Volume: the remaining amount of effective payload volume for the launch in cubic meters

Outputting the remaining payload mass and volume available for a specific launch allows the user to observe if adding or combining payloads to a launch is possible. An additional Launch Mission MATLAB script allows for the combining of different payloads to achieve the cheapest launch budget possible for the mission. The script first requires users to specify their inputs and runs the Launch Mission function. From the initial function call, the respective launch reports are outputted to the command window, each of which include the remaining payload mass and volume available for that launch. The script then presents the following prompt to the user and requires an appropriate response entered into the command window in order to move on:

- Would you like to add payloads to any launch? Say "yes" or "no".

If the user inputs “no” into the command window, the script terminates.

If the user inputs “yes” into the command window, the script gives the following prompts:

- Which launch would you like to add to? Give a single array element integer, e.g. 1 or 5
- Payload mass to add in Mg as a scalar
- Is the payload volume known? Say "yes" or "no".

If the user inputs “yes”, the script gives the following prompt before continuing:

- Payload volume to add in m<sup>3</sup> as scalar

If the user inputs “no”, the script moves on to the succeeding prompts.

- New mission title for the launch, e.g. "New Mission"
- Do you want to delete a mission?

If the user inputs “no”, the script re-runs the function and preceding prompts. If the user inputs “yes”, the script gives the following prompt:

- Which launch? Give a single array element integer, e.g. 1 or 5

Then the script re-runs until the user adds no more payloads to a launch.

The final launch output also contains any extra notes regarding the launch that the user should consider, if for example additional costs or launches are required to send their payload to the specified destination. Ultimately, the final version of the Launch Mission tool provides a user-friendly and accurate means of choosing a launch vehicle for a given set of launch parameters.

## 6. Launch Vehicle Function Version 1

```
function LaunchMission(timeofmission,destination,payloadmasskg)

%LAUNCHMISSION the Launch Mission function serves as a tool to allow an

%inputted payload mass, with given time of mission (pre-2023 or 2023),

%and given destination (orbit or lunar surface)

%INPUTS:

% 1. timeofmission: string, period of mission for launch, 'pre 2023' or

% 'post 2023' assuming Starship is available in 2023

% 2. destination: string, destination of payload 'lunar orbit' or

% 'lunar surface'

% 3. payloadmasskg: scalar, mass of payload in [kg]

%OUTPUTS Printed:

% 1. launchvehicle: string, ideal launch vehicle for given parameters

% 2. cost: string, launch cost (approximate) for given parameters in

% [millions of $]

% 3. volumeconstraints: string, volume constraints for given launch vehicle

% 4. thrust: scalar, first stage thrust value in [kN]

% 5. lander: lander vehicle to lunar surface (if 'lunar surface'

% destination)

%

% Launch Vehicles:

% Pre-2023

% 1. Falcon 9 Expendable

% 2. Falcon Heavy Reusable

% 3. Falcon Heavy Expendable
```

```
% 3. Starship (post-2023)

%
% Landers (if 'lunar surface'):

% 1. Nova-C (on F9 capable of 100 kg)

% 2. Sky Crane

% 3. Starship (post-2023)

%
% Assumptions:

% -plan A (Starship available starting in 2023 (destination = 'post 2023')

% -rough calculations of kg to LLO for Falcon 9 and FH (Hohmann after GTO)

% -Nova-C private lander available for small surface payloads before 2023

% -Starship is actually ~$200 million per launch and is assumed to be

% refueled in LEO, able to send 100 tons to LLO

lander = 0;

launchvehicle = 0;

cost = 0;

volumeconstraints = 0;

if strcmp('pre 2023',timeofmission) %Starship not available

    if strcmp('lunar surface',destination)

        if payloadmasskg > 0 && payloadmasskg <= 100

            launchvehicle = 'Falcon 9';

            cost = '%62 million';

            volumeconstraints = '< 9 m^3';

        end

    end

end
```

```
lander = 'Nova-C';

thrust = 7607; %kN

elseif payloadmasskg > 100 && payloadmasskg <= 1025

    launchvehicle = 'Falcon Heavy Expendable';

    cost = '$150 millions';

    lander = 'Sky Crane';

    volumeconstraints = 'fairing size 13 m long by 5.2 m in diameter';

    thrust = 22800; %kN

end

elseif strcmp('lunar orbit',destination)

if payloadmasskg > 0 && payloadmasskg <= 2100

    launchvehicle = 'Falcon 9';

    cost = '$62 million';

    volumeconstraints = 'fairing size 13.9 m long by 5.2 m in diameter';

    thrust = 7607; %kN

elseif payloadmasskg > 2100 && payloadmasskg <= 5000

    launchvehicle = 'Falcon Heavy Reusable';

    cost = '$90 million';

    volumeconstraints = 'fairing size 13 m long by 5.2 m in diameter';

    thrust = 22800; %kN

elseif payloadmasskg > 5000 && payloadmasskg <= 15300

    launchvehicle = 'Falcon Heavy Expendable';

    cost = '$150 million';

    volumeconstraints = 'fairing size 13 m long by 5.2 m in diameter';

    thrust = 22800; %kN
```

```
end

endif strcmp('post 2023',timeofmission)

if strcmp('lunar surface',destination)

if payloadmasskg > 0 && payloadmasskg <= 100

launchvehicle = 'Falcon 9';

cost = '$62 million';

volumeconstraints = '< 9 m^3';

lander = 'Nova-C';

thrust = 7607; %kN

elseif payloadmasskg > 100 && payloadmasskg <= 1025

launchvehicle = 'Falcon Heavy Expendable';

cost = '$150 million';

lander = 'Sky Crane';

volumeconstraints = 'fairing size 13 m long by 5.2 m in diameter';

thrust = 22800; %kN

elseif payloadmasskg > 1025 && payloadmasskg <= 90719

launchvehicle = 'Starship';

cost = '$2-10 million';

volumeconstraints = '< 1100 m^3';

lander = 'Starship Lunar Lander';

thrust = 65000; %kN

end

elseif strcmp('lunar orbit',destination)

if payloadmasskg > 0 && payloadmasskg <= 2100
```

```
launchvehicle = 'Falcon 9';
cost = '$62 million';
volumeconstraints = 'fairing size 13.9 m long by 5.2 m in diameter';
thrust = 7607; %kN

elseif payloadmasskg > 2100 && payloadmasskg <= 5000
    launchvehicle = 'Falcon Heavy Reusable';
    cost = '$90 million';
    volumeconstraints = 'fairing size 13 m long by 5.2 m in diameter';
    thrust = 22800; %kN

elseif payloadmasskg > 5000 && payloadmasskg <= 15300
    launchvehicle = 'Falcon Heavy Expendable';
    cost = '$150 million'; %millions of dollars
    volumeconstraints = 'fairing size 13 m long by 5.2 m in diameter';
    thrust = 22800; %kN

elseif payloadmasskg > 15300 && payloadmasskg <= 90719
    launchvehicle = 'Starship';
    cost = '$2-10 million'; %millions of dollars
    volumeconstraints = '< 1100 m^3';
    thrust = 65000; %kN

end
end
end

if launchvehicle == 0
    fprintf("\nLaunch of the specified parameters is not possible at this mission time\n\n");

```

```

else

    fprintf("\nHere is your mission!");

    fprintf("\nLaunch Vehicle: %s", launchvehicle);

    fprintf("\nThrust: %.2f kN", thrust);

    fprintf("\nCost: %s", cost);

    fprintf("\nVolume Constraints: %s", volumeconstraints);

    if lander == 0

        fprintf("\nNo lander specified\n\n");

    else

        fprintf("\nLander: %s\n\n", lander);

    end

end

```

## 7. Launch Vehicle Function Version 4 (Final Version)

### Function

LaunchMissionV4(missiontitle,timeofmission,destination,payloadmassMg,payloadvolm3)

%LAUNCHMISSION the Launch Mission function serves as a tool to allow an

%inputted payload mass, with given time of mission (pre-2023 or 2023),

%and given destination (LEO, lunar orbit, or lunar surface) as arrays or

%single values/strings. The tool outputs

%INPUTS:

% 1. missiontitle\*: string, titles of each payload/mission

% 2. timeofmission\*: string, period of mission for launch, 'pre 2023' or

% 'post 2023' assuming Starship is available in 2023

% 3. destination\*: string, destination of payload 'LEO', 'lunar orbit', or

```
% 'lunar surface'

% 4. payloadmassMg*: scalar, mass of payload in [Mg]

% 5. (optional) payloadvolm3*: scalar, volume of payload in [m^3] if known

% *all inputs must be in vector form (1xi array), all with i elements. Each

% element in the i position of each of the 3 vectors should coorespond, and

% represents one launch with a given time period, destination, and payload

% mass.

%OUTPUTS Printed as Launch Report:

% 1. Launch Vehicle: string, ideal launch vehicle for given parameters

% 2. Cost: scalar, launch cost (approximate) for given parameters in

% [millions of $]

% 3. Volume Constraints: string, volume constraints for given launch vehicle

% 4. Thrust: scalar, launch thrust value in [kN]

% 5. Lander: string, lander vehicle to lunar surface (if 'lunar surface'

% destination)

% 6. Remaining payload mass: remaining amount of effective payload mass

% [Mg]

% 7. Remaining payload mass: remaining amount of effective payload mass

% [Mg]

%

% Launch Vehicles:

% Pre-2023

% 1. Falcon 9 Expendable

% 2. Falcon Heavy Reusable

% 3. Falcon Heavy Expendable
```

```
% 3. Starship (post-2023)
```

```
%
```

```
% Landers (if 'lunar surface'):
```

```
% 1. Modified Apollo Lunar Lander
```

```
% 2. Starship (post-2023)
```

```
% Assumptions:
```

```
% -plan A (Starship available starting in 2023 (destination = 'post 2023')
```

```
% -rough calculations of kg to LLO for Falcon 9 and FH (Hohmann after GTO)
```

```
% -Starship is $873.2 million per launch (returned) and is assumed to be
```

```
% refueled in LEO, able to send 90.7 Mg to LLO, $2.5 million per day
```

```
% opportunity cost if not returned ASAP
```

```
if nargin == 3
```

```
    payloadvolm3 = zeros(1,length(payloadmassMg));
```

```
end
```

```
launchvehicle = strings(1,length(payloadmassMg));
```

```
LVcost = zeros(1,length(payloadmassMg));
```

```
totalcost = zeros(1,length(payloadmassMg));
```

```
volumeconstraints = strings(1,length(payloadmassMg));
```

```
thrust = strings(1,length(payloadmassMg)); %kN
```

```
lander = strings(1,length(payloadmassMg));
```

```
rempaymass = zeros(1,length(payloadmassMg));
```

```
rempayvol = zeros(1,length(payloadmassMg));
```

```
%max LV masses in Mg
```

```

maxF9payloadmass_LEO = 22.8;
maxFHRpayloadmass_LEO = 50;
maxFHpayloadmass_LEO = 63.8;
maxFHpayloadmass_land = 5.346;
maxF9payloadmass_LLO = 5;
maxFHRpayloadmass_LLO = 5;
maxFHpayloadmass_LLO = 15.3;
maxStarshipmass_LEO = 100;
maxStarshipmass_LLO = 90.7;

%max LV volumes in Mg
maxF9payloadvol = 240.61;
maxFHpayloadvol = 295.2;
maxFHpayloadvol_land = 7;
maxStarshipvol = 1100;

for i=1:length(payloadmassMg)
    if strcmp("pre 2023",timeofmission(i)) %Starship not available
        if strcmp("LEO",destination(i))
            if (payloadmassMg(i) <= maxF9payloadmass_LEO) && (payloadvolm3(i) <=
maxF9payloadvol)
                launchvehicle(i) = "Falcon 9 Expendable";
                LVcost(i) = 62; %millions of dollars
                totalcost(i) = LVcost(i);
                volumeconstraints(i) = "<= 240.61 m^3"; %in case payload volume not given
        end
    end
end

```

```

thrust(i) = 7607; %kN

rempaymass(i) = maxF9payloadmass_LEO - payloadmassMg(i);

rempayvol(i) = maxF9payloadvol - payloadvolm3(i);

elseif (payloadmassMg(i) <= maxFHRpayloadmass_LEO) && (payloadvolm3(i) <=
maxFHpayloadvol)

launchvehicle(i) = "Falcon Heavy Reusable";

LVcost(i) = 90; %millions of dollars

totalcost(i) = LVcost(i);

volumeconstraints(i) = "<= 295.2 m^3";

thrust(i) = 22800; %kN

rempaymass(i) = maxFHRpayloadmass_LEO - payloadmassMg(i);

rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);

elseif (payloadmassMg(i) <= maxFHpayloadmass_LEO) && (payloadvolm3(i) <=
maxFHpayloadvol)

launchvehicle(i) = "Falcon Heavy Expendable";

LVcost(i) = 150; %millions of dollars

totalcost(i) = LVcost(i);

volumeconstraints(i) = "<= 295.2 m^3";

thrust(i) = 22800; %kN

rempaymass(i) = maxFHpayloadmass_LEO - payloadmassMg(i);

rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);

end

elseif strcmp("lunar surface",destination(i))

if (payloadmassMg(i) <= maxFHpayloadmass_land) && (payloadvolm3(i) <=
maxFHpayloadvol_land)

```

```
launchvehicle(i) = "Falcon Heavy Expendable";
LVcost(i) = 150; %millions of dollars
lander(i) = "Apollo Cargo Variant, $210 million";
totalcost(i) = 150+210; %millions of dollars
volumeconstraints(i) = "< 7 m^3";
thrust(i) = 22800; %kN
rempaymass(i) = maxFHpayloadmass_land - payloadmassMg(i);
rempayvol(i) = maxFHpayloadvol_land - payloadvolm3(i);
end

elseif strcmp("lunar orbit", destination(i))
if (payloadmassMg(i) <= maxF9payloadmass_LLO) && (payloadvolm3(i) <=
maxF9payloadvol)
    launchvehicle(i) = "Falcon 9 Expendable";
    LVcost(i) = 62; %millions of dollars
    totalcost(i) = LVcost(i);
    volumeconstraints(i) = "< 240.61 m^3";
    thrust(i) = 7607; %kN
    rempaymass(i) = maxF9payloadmass_LLO - payloadmassMg(i);
    rempayvol(i) = maxF9payloadvol - payloadvolm3(i);
elseif (payloadmassMg(i) <= maxFHRpayloadmass_LLO) && (payloadvolm3(i) <=
maxFHpayloadvol)
    launchvehicle(i) = "Falcon Heavy Reusable";
    LVcost(i) = 90; %millions of dollars
    totalcost(i) = LVcost(i);
```

```

volumeconstraints(i) = "< 295.2 m^3";
thrust(i) = 22800; %kN
rempaymass(i) = maxFHRpayloadmass_LLO - payloadmassMg(i);
rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);
elseif (payloadmassMg(i) <= maxFHpayloadmass_LLO) && (payloadvolm3(i) <=
maxFHpayloadvol)
    launchvehicle(i) = "Falcon Heavy Expendable";
    LVcost(i) = 150; %millions of dollars
    totalcost(i) = LVcost(i);
    volumeconstraints(i) = "< 295.2 m^3";
    thrust(i) = 22800; %kN
    rempaymass(i) = maxFHpayloadmass_LLO - payloadmassMg(i);
    rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);
end
end
elseif strcmp("post 2023",timeofmission(i))
if strcmp("LEO",destination(i))
if (payloadmassMg(i) <= maxF9payloadmass_LEO) && (payloadvolm3(i) <=
maxF9payloadvol)
    launchvehicle(i) = "Falcon 9 Expendable";
    LVcost(i) = 62; %millions of dollars
    totalcost(i) = LVcost(i);
    volumeconstraints(i) = "<= 240.61 m^3"; %in case payload volume not given
    thrust(i) = 7607; %kN
    rempaymass(i) = maxF9payloadmass_LEO - payloadmassMg(i);
end
end

```

```

rempayvol(i) = maxF9payloadvol - payloadvolm3(i);

elseif (payloadmassMg(i) <= maxFHRpayloadmass_LEO) && (payloadvolm3(i) <=
maxFHpayloadvol)

launchvehicle(i) = "Falcon Heavy Reusable";

LVcost(i) = 90; %millions of dollars

totalcost(i) = LVcost(i);

volumeconstraints(i) = "<= 295.2 m^3";

thrust(i) = 22800; %kN

rempaymass(i) = maxFHRpayloadmass_LEO - payloadmassMg(i);

rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);

elseif (payloadmassMg(i) <= maxStarshipmass_LEO) && (payloadvolm3(i) <=
maxStarshipvol)

launchvehicle(i) = "Starship";

LVcost(i) = 116.8; %millions of dollars

totalcost(i) = LVcost(i);

volumeconstraints(i) = "< 1100 m^3";

thrust(i) = 65000; %kN

rempaymass(i) = maxStarshipmass_LEO - payloadmassMg(i);

rempayvol(i) = maxStarshipvol - payloadvolm3(i);

elseif (payloadmassMg(i) <= maxFHpayloadmass_LEO) && (payloadvolm3(i) <=
maxFHpayloadvol)

launchvehicle(i) = "Falcon Heavy Expendable";

LVcost(i) = 150; %millions of dollars

totalcost(i) = LVcost(i);

```

```
volumeconstraints(i) = "<= 295.2 m^3";  
thrust(i) = 22800; %kN  
  
rempaymass(i) = maxFHpayloadmass_LEO - payloadmassMg(i);  
rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);  
end  
  
elseif strcmp("lunar surface",destination(i))  
if (payloadmassMg(i) <= maxFHpayloadmass_land) && (payloadvolm3(i) <= maxFHpayloadvol_land)  
launchvehicle(i) = "Falcon Heavy Expendable";  
LVcost(i) = 150; %millions of dollars  
totalcost(i) = 150+210;  
lander(i) = "Apollo Cargo Variant, $210 million";  
volumeconstraints(i) = "<= 7 m^3";  
thrust(i) = 22800; %kN  
rempaymass(i) = maxFHpayloadmass_land - payloadmassMg(i);  
rempayvol(i) = maxFHpayloadvol_land - payloadvolm3(i);  
elseif (payloadmassMg(i) <= maxStarshipmass_LLO) && (payloadvolm3(i) <= maxStarshipvol)  
launchvehicle(i) = "Starship";  
LVcost(i) = 873.39; %millions of dollars  
totalcost(i) = LVcost(i);  
volumeconstraints(i) = "< 1100 m^3";  
lander(i) = "Starship Lunar Lander, with 6 LEO tanker launches";  
thrust(i) = 65000; %kN  
rempaymass(i) = maxStarshipmass_LLO - payloadmassMg(i);
```

```
rempayvol(i) = maxStarshipvol - payloadvolm3(i);

end

elseif strcmp("lunar orbit",destination(i))

    if (payloadmassMg(i) <= maxF9payloadmass_LLO) && (payloadvolm3(i) <=
maxF9payloadvol)

        launchvehicle(i) = "Falcon 9 Expendable";

        LVcost(i) = 62; %millions of dollars

        totalcost(i) = LVcost(i);

        volumeconstraints(i) = "< 240.61 m^3";

        thrust(i) = 7607; %kN

        rempaymass(i) = maxF9payloadmass_LLO - payloadmassMg(i);

        rempayvol(i) = maxF9payloadvol - payloadvolm3(i);

    elseif (payloadmassMg(i) <= maxFHRpayloadmass_LLO) && (payloadvolm3(i) <=
maxFHPayloadvol)

        launchvehicle(i) = "Falcon Heavy Reusable";
```

```
LVcost(i) = 90; %millions of dollars  
totalcost(i) = LVcost(i);  
volumeconstraints(i) = "< 295.2 m^3";  
thrust(i) = 22800; %kN  
rempaymass(i) = maxFHRpayloadmass_LLO - payloadmassMg(i);  
rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);  
elseif (payloadmassMg(i) <= maxFHpayloadmass_LLO) && (payloadvolm3(i) <= maxFHpayloadvol)  
    launchvehicle(i) = "Falcon Heavy Expendable";  
    LVcost(i) = 150; %millions of dollars  
    totalcost(i) = LVcost(i);  
    volumeconstraints(i) = "< 295.2 m^3";  
    thrust(i) = 22800; %kN  
    rempaymass(i) = maxFHpayloadmass_LLO - payloadmassMg(i);  
    rempayvol(i) = maxFHpayloadvol - payloadvolm3(i);  
elseif (payloadmassMg(i) <= maxStarshipmass_LLO) && (payloadvolm3(i) <= maxStarshipvol)  
    launchvehicle(i) = "Starship";  
    LVcost(i) = 873.39; %millions of dollars  
    volumeconstraints(i) = "< 1100 m^3";  
    thrust(i) = 65000; %kN  
    rempaymass(i) = maxStarshipmass_LLO - payloadmassMg(i);  
    rempayvol(i) = maxStarshipvol - payloadvolm3(i);  
end
```

```
end  
end  
end  
  
for j = 1:length(launchvehicle)  
    fprintf("\n<strong>%s</strong>\n", missontitle(j));  
    if (launchvehicle(j) == "" && destination(j) == "lunar surface")  
        fprintf("\nCannot land payload on surface before 2023\n")  
    elseif (launchvehicle(j) == "")  
        fprintf("\nLaunch of the specified parameters is not possible at this mission time\n\n");  
    else  
        fprintf("\nHere is your mission!");  
        fprintf("\nLaunch Vehicle: %s", launchvehicle(j));  
        if (launchvehicle(j) == "Starship" && destination(j) ~= "LEO")  
            fprintf("\nStarship launch includes 6 prior LEO tanker launches.");  
        end  
        if (launchvehicle(j) == "Starship" && destination(j) == "lunar surface")  
            fprintf("\nStarship launch to lunar surface includes additional fueling by lunar depot.")  
            fprintf("\nStarship Opportunity cost: $2.5 million/day if not returned ASAP.");  
        end  
        fprintf("\nThrust: %.2f kN", thrust(j));  
        fprintf("\nCost: $%.2f million", totalcost(j));  
        fprintf("\nVolume Constraints: %s", volumeconstraints(j));  
        if lander(j) == ""  
            fprintf("\nNo lander specified\n");
```

```
else
    fprintf("\nLander: %s\n", lander(j));
    end
    fprintf("\nRemaining Effective Payload Mass: %.2f Mg", rempaymass(j));
    fprintf("\nRemaining Effective Payload Volume: %.2f m^3\n", rempayvol(j))
    end
end

launchbudget = sum(totalcost);

fprintf("\n<strong>Total launch budget</strong> for your missions: $%.2f million\n\n",
       launchbudget);

end
```

### 8. Launch Vehicle Script (Final Version)

```
% Launch Mission Tool Script

%INPUTS - change these values

% can be single values/strings or 1xi vectors

missiontitle = "Three Habitat Modules"; %title of your mission

timeofmission = "post 2023"; %time of mission you are launching e.g. "pre 2023"

destination = "lunar surface" ; %destination of your payload e.g. "LEO", "lunar surface"

payloadmassMg = 27.735; %payload mass in Mg

payloadvolm3 = 21.6*3; %payload volume in m^3

%DO NOT TOUCH REST OF CODE

LaunchMissionV4(missiontitle,timeofmission,destination,payloadmassMg,payloadvolm3)

i = 1;

while (i)

    prompt = "Would you like to add payloads to any launch? Say ""yes"" or ""no"".\n";

    x = input(prompt);

    if strcmp("yes",x) || strcmp("Yes",x)

        ques = "Which launch would you like to add to? Give a single array element integer, e.g. 1 or

5\n";

        launchnum = input(ques);

        addpaymass = input("Payload mass to add in Mg as scalar\n");

        volq = input("Is the payload volume known? Say ""yes"" or ""no"".\n");

        if strcmp("yes",volq) || strcmp("Yes",volq)

            addpayvol = input("Payload volume to add in m^3 as scalar\n");
```

```
else
    addpayvol = 0;
end

newmisstitle = input("New mission title for this launch, e.g. ""New Mission""\n");
missiontitle(launchnum) = newmisstitle;

payloadmassMg(launchnum) = payloadmassMg(launchnum)+addpaymass;
payloadvolm3(launchnum) = payloadvolm3(launchnum)+addpayvol;

deleteq = input("Do you want to delete a mission?\n");
if strcmp("yes",deleteq) || strcmp("Yes",deleteq)

    deletenum = input("Which launch? Give a single array element integer, e.g. 1 or 5\n");
    missiontitle(deletenum) = [];
    timeofmission(deletenum) = [];
    destination(deletenum) = [];
    payloadmassMg(deletenum) = [];
    payloadvolm3(deletenum) = [];
end

LaunchMissionV4(missiontitle,timeofmission,destination,payloadmassMg,payloadvolm3);

else
    i = 0;
end
end
```

### III. Upper Stage Engine Capability Comparison For Use on a Space Tug

Payload capacity to low lunar orbit (LLO) is limited to 15 tonnes before Starship becomes available in 2023. Using a dedicated low earth orbit (LEO) to LLO transport, or space tug, is one potential method for increasing the payload capacity. The tug and the payload would be launched separately and then rendezvous in LEO before burning to reach LLO. Several characteristics are worth considering: weight, volume, longevity and cost. The tug must be able to be lifted to LEO by, at the largest, a Falcon Heavy, thus it must weigh less than 63.8 Mg. The tug ideally should be able to fit in the Falcon Heavy's payload fairing. The longevity of the tug is related to the total burn time a given engine can withstand without requiring maintenance or replacement. This is mainly relevant for a "reusable" tug which would be refueled at both LEO and LLO. The final factor is cost—the tug should not be excessively expensive to remain competitive with other options.

Five engines were chosen for comparison. Each engine spans its own "thrust regime" or has another important factor which sets it apart from the others. The five engines chosen are the AJ10, RL-10B-2, Merlin 1D Vacuum, Raptor Vacuum, and NERVA (The Nuclear Engine for Rocket Vehicle Application). Some other notable engines considered include Blue Origin's BE-3U and Rocket Lab's Rutherford. These two engines provided some promise, but limited availability of information on their operation made a reasonable analysis impossible.

Aerojet Rocketdyne's AJ10 is a pressure-fed, hypergolic rocket engine, most notable for its use one the Space Shuttle's Orbital Maneuvering System and the Apollo service module. The engine produces 40 kN of thrust and has a specific impulse of 320s [68]. It burns aerozine 50 and

nitrogen tetroxide and has a mixture ratio of 1.9 [69]. The cost of the engine is not widely publicized, so a cost of \$1 million per engine was assumed for the analysis.

The next engine, another Aerojet Rocketdyne product, the RL-10B-2, is an expander cycle bipropellant rocket engine. The RL-10 burns liquid hydrogen and liquid oxygen and has a mixture ratio of 5.88 [68]. It has a specific impulse of 460s and produces 110 kN of thrust [68]. Each RL-10 costs \$17 million [70]

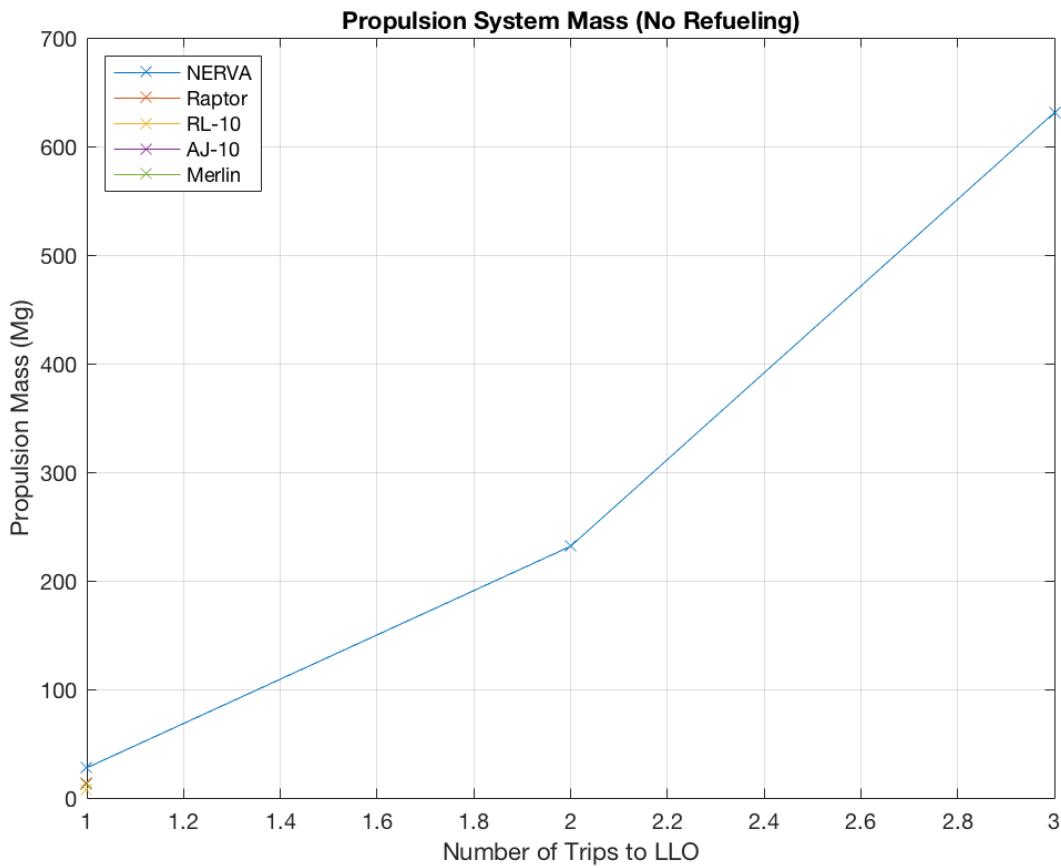
SpaceX's Merlin 1D Vacuum is a bipropellant gas-generator cycle rocket engine. It burns liquid oxygen and RP-1 with a mixture ratio of 2.34 [27]. The Merlin produces 981 kN of thrust and has a specific impulse of 350s [72]. The engine cost roughly \$1 million.

SpaceX's Raptor Vacuum engine is a bipropellant full-flow staged combustion rocket engine. The Raptor is capable of producing 2200 kN of thrust with a specific impulse of 380s [73]. The Raptor burns liquid methane and liquid oxygen with a mixture ratio of 3.55 [73]. The engine cost \$1 million to manufacture [73].

The final engine is an experimental engine developed in the 1960s and early 1970s. NERVA is a nuclear thermal rocket engine; it produces through the heat generated by nuclear fission. The program reached a technology readiness level (TRL) of six before it was canceled in 1973. Since then, due to closure of facilities and the retirement of many of the project's engineers, NASA has downgraded NERVA to a TRL of 3 [75]. According to the planned specifications for a flight engine, NERVA will produce roughly 333 kN of thrust and have a specific impulse of 825s. The engine only uses liquid hydrogen. One massive drawback of a NERVA engine is its weight; the engine is projected to weigh 15 Mg. Cost is another consideration, the program would likely need to be restarted and much of the research and manufacturing techniques relearned. Thus a

reasonable cost estimate for developing and constructing the first engine is about \$660 million, afterwards it is assumed the cost of each additional engine is roughly 5% of the development cost.

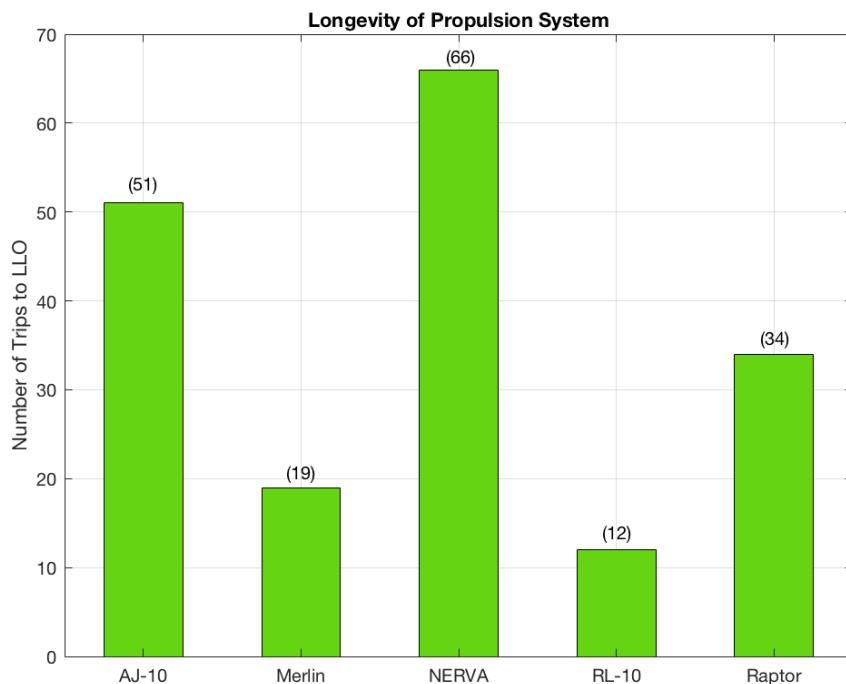
Now that we have an idea of the capabilities of the engines available, we can start the analysis. For the analysis, the distance for LEO to LLO is assumed to be 4 km/s. Each engine is assumed to be able to relight after an extended period in space and negligible boiloff of the propellants occur. The first analysis performed studies the endurance capabilities of each engine. Figure J.0.1 displays the number of trips to LLO each engine could take and the total mass of the stage without refueling hauling a five tonne payload.



**Fig J.0.1, The number of trips a given engine could perform (x-axis) and the stage mass required to do so (y-axis).**

Only NERVA is able to make more than one trip to LLO without burning longer than its expected lifespan. It should be noted “Trip to LLO” refers to reaching LLO from LEO, returning to LEO, and then back to LEO for two trips to be taken. The results show it is largely impractical to use a space tug with refueling at LEO and LLO each trip.

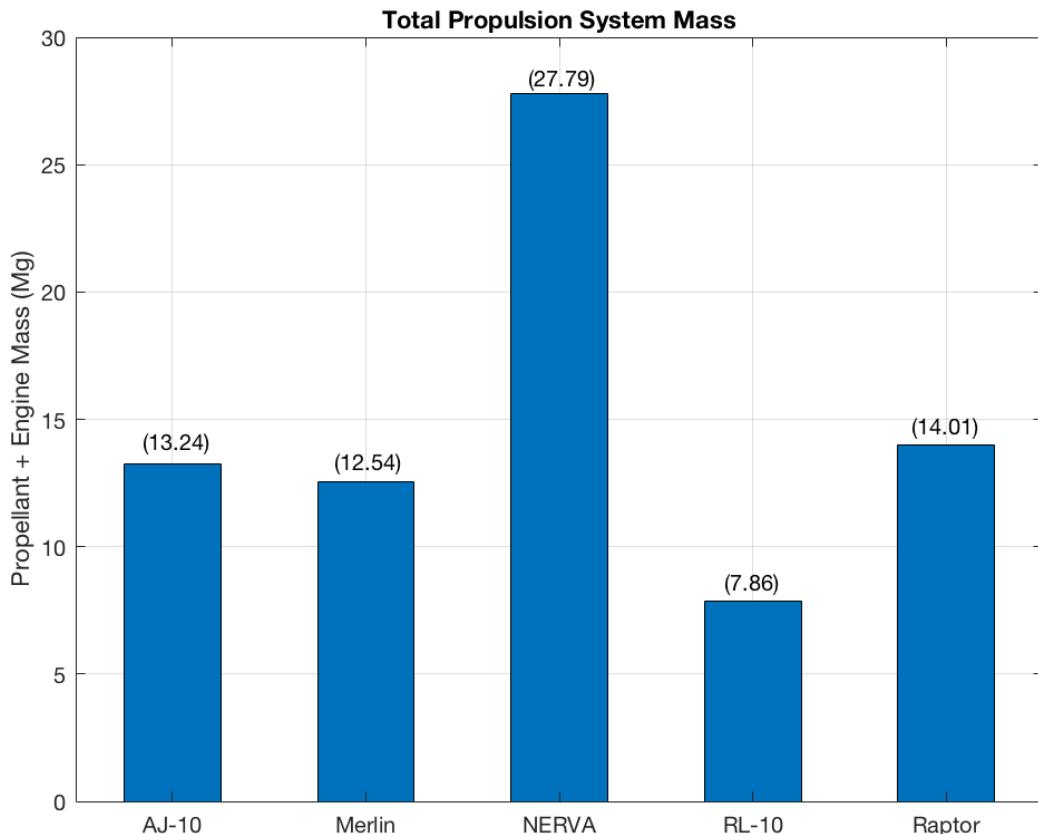
The data becomes more interesting when we assume the tug is refueled by a LEO and LLO. Figure J.0.2 shows the maximum number of trips a tug could make carrying a 5 tonne payload from LEO to LLO.



**Fig. J.0.2 Total number of trips with 5 Mg payload to LLO with LEO and LLO refueling.**

NERVA again stands out as a clear winner, followed by the AJ10. Refueling NERVA is also simpler than some of the other engines. NERVA is a monopropellant rocket which only burns liquid hydrogen. The major drawbacks of NERVA become more apparent with the next step in the

analysis. The mass of the fully fueled tug, neglecting tank and payload mass, is displayed in Figure J.0.3.



**Fig. J.0.3 Propulsion system mass given a 5 Mg payload and LEO and LLO refueling.**

A single NERVA engine, weighing 15 Mg, weighs more than the total mass of any of the other competing stage configurations.

Now that more information regarding the capabilities of each engine has been calculated based on the benchmark payload of 5 Mg, it is time to determine optimal stage sizing and configuration for different sized payloads for a single use tug. We need to determine the optimal number of engines and model of engine which produces a stage capable of moving the most amount of mass,

and takes up the least amount of volume and cost and is still able to be lifted by a Falcon Heavy to LEO.

A script was created which calculates the configuration with the least amount of total mass for a given payload mass and delta v. The mass flowrate of the engine is calculated given its thrust ( $F$ ) and specific impulse ( $I_{sp}$ ).

$$\dot{m} = \frac{F}{I_{sp}g} \quad (1)$$

Where g is the gravitational acceleration of Earth.  $g=9.81 \text{ m/s}^2$ . The Tsiolkovsky rocket equation is used to calculate the mass ratio of the stage ( $MR$ ).

$$MR = e^{\frac{\Delta v}{I_{sp}g}} \quad (2)$$

Once the mas ratio has been calculated the required propellant mass can be obtained. From equation (2) the propellant volume can be calculated using the mixture ratio and the density of the propellants. From there it is assumed the tank structural volume is 1% of the total tank volume. The tank weight is calculated and the script is iterated until the total mass stops changing significantly. The price of each stage is calculated by taking the cost of the engines and adding it to the cost of the propellants plus an additional \$1 million for other materials. 1.25x the sum of the individual price components is taken as the cost for the entire stage. A table of important output values is shown below for a given payload of 25 Mg and a delta v of 4 km/s.

**Table J.0.1 Sample Script Outputs for 20 Mg and 4 km/s**

| Engine | Total Stage Mass (kg) | # of Engines | Propellant Volume (m <sup>3</sup> ) | Cost (M\$) |
|--------|-----------------------|--------------|-------------------------------------|------------|
| AJ10   | 64,754                | 1            | 54.106                              | 3.29       |
| Merlin | 56,654                | 1            | 47.982                              | 2.53       |
| RL-10  | 36,388                | 1            | 82.955                              | 22.53      |
| Raptor | 52,491                | 1            | 57.298                              | 2.52       |
| NERVA  | 40,570                | 1            | 284.11                              | 828.79     |

The script also outputs inert mass, thrust to weight ratio, and total burn time. The sample output demonstrates the calculated costs and benefits of each engine system. The RL-10 represents the lowest mass engine, but the second highest volume. The RL-10 uses liquid hydrogen, a very low density propellant. The raptor provides the lowest cost engine. The table makes NERVA's largest drawback clear. A single stage would cost nearly 37x more expensive than the next most expensive option. Ultimately it was decided the added complexity of using a space tug was not worth the benefit of increased payload capacity to LLO. Starship is capable of delivering a significant amount of payload without the need for a dedicated third stage.

### 1. Space Tug Sizing Code

```
function [results] = enginecom(m_pl,deltav)

%% AAE 45000 -- Engine Comparison

%% Description

% Program: enginecom.m

% Author: Josh Romanowski (jromanow@purdue.edu), MAR 2021

% Description: Compares several engines for use on upper stage given deltav

% and payload mass.

%

% INPUTS:

% m_pl - payload mass [kg]

% deltav - expected delta v for stage [m/s]

%

% OUTPUTS:

% results - a table of engine values
```

```
%  
  
% NOTE: The program errors out if one of the engines being compared needs a  
% cluster of more than 100 engines. Try lowering the payload mass or the  
% delav. Alternatively increase the total number of engines by increasing  
% the range of 'k' at the start of the main 'for' loop.  
  
%% Constant Parameters  
g=9.81; % [m/s^2]  
tol = 0.1;  
  
rho_LCH4 = 440; % [kg/m^3]  
rho_LOX = 1250; % [kg/m^3]  
rho_LH2 = 90; % [kg/m^3]  
rho_NTO = 1440; % [kg/m^3]  
rho_Aero50 = 903; % [kg/m^3]  
rho_RP1 = 1020; % [kg/m^3]  
  
% Source: NIST  
  
c_LOX = 0.16; % [$/kg]  
c_LH2 = 3.66; % [$/kg]  
  
% Source: https://www.quora.com/How-much-does-NASA-pay-per-kg-for-hydrogen-and-oxygen-in-rocket-fuel  
c_LCH4 = 1.35; % [$/kg]
```

```
% Source:
```

```
https://www.thespacereview.com/article/2893/1#:~:text=Liquid%20methane%20costs%20about%20%241.35,which%20fuel%20will%20be%20cheaper.
```

```
c_NTO = 6.00; % [$/kg]
```

```
% Source: http://www.astronautix.com/n/n2o4.html
```

```
c_Aero50 = 17.00; % [$/kg]
```

```
% Source: http://www.astronautix.com/h/hydrazine.html
```

```
c_RP1 = 1.2; % [$/kg]
```

```
rho_tank = 1560; % [kg/m^3]
```

```
% Source: https://www.americanelements.com/lithium-aluminum-alloy-87871-87-2
```

```
tank_vol = 0.01;
```

```
cost_est = 1.25;
```

```
%% Single Use Engine(engine cluster)
```

```
for k = 1:100
```

```
% AJ-10
```

---

```
mInrt_AJ10 = 100; % [kg]
```

```
Isp_AJ10 = 320; % [s]
```

```
F_AJ10 = 40e3; % [N]
```

```
tOp_AJ10 = 15*60*60; % [s]
```

```
r_AJ10 = 1.9;
```

```

c_AJ10 = 1e6; %couldnt find anything taking a guess since its pressure fed it should be kinda
low

% Source: https://www.alternatewars.com/BBOW/Space_Engines/Aerojet_Engines.htm

% Source: https://en.wikipedia.org/wiki/AJ10

mInrt_AJ10_T(k) = 0;

mInrt_AJ10_i = 0;

while abs(mInrt_AJ10_T(k) + k*mInrt_AJ10 - mInrt_AJ10_i) >= tol

mdot_AJ10(k) = k*F_AJ10/(Isp_AJ10*g);

MR_AJ10(k) = exp(deltav/(g*Isp_AJ10));

mProp_AJ10(k) = MR_AJ10(k)*(k*mInrt_AJ10+mInrt_AJ10_T(k)+m_pl) - k*mInrt_AJ10
- mInrt_AJ10_T(k) - m_pl;

mTOT_AJ10(k) = k*mInrt_AJ10 + mInrt_AJ10_T(k) + mProp_AJ10(k);

tBurn_AJ10(k) = mProp_AJ10(k)/mdot_AJ10(k);

Vprop_AJ10(k) = r_AJ10*mProp_AJ10(k)/(1+r_AJ10)/rho_NTO +
mProp_AJ10(k)/(1+r_AJ10)/rho_Aero50;

mInrt_AJ10_T(k) = Vprop_AJ10(k)*tank_vol*rho_tank;

mInrt_AJ10_i = mInrt_AJ10_T(k) + k*mInrt_AJ10;

cost_AJ10(k) = r_AJ10*mProp_AJ10(k)/(1+r_AJ10)*c_NTO +
mProp_AJ10(k)/(1+r_AJ10)*c_Aero50 + k*c_AJ10 + 1e6;

end

```

```
% Merlin
=====
mInrt_Merlin = 470; % [kg]
Isp_Merlin = 350; % [s]
F_Merlin = 981e3; % [N]
tOp_Merlin = 15*60; % [s]
r_Merlin = 2.34;
c_Merlin = 1e6; % [$]
% Source: https://www.nextbigfuture.com/2019/05/spacex-raptor-engine-will-be-best-on-cost-
and-nearly-best-on-
isp.html#:~:text=SpaceX%20Merlin%20engine%20costs%20less,in%20later%20simplified%20R
apton%20versions.

% Source: https://www.spacex.com/media/falcon_users_guide_042020.pdf

mInrt_Merlin_T(k) = 0;
mInrt_Merlin_i = 0;

while abs(mInrt_Merlin_T(k) + k*mInrt_Merlin - mInrt_Merlin_i) >= tol
    mdot_Merlin(k) = k*F_Merlin/(Isp_Merlin*g);
    MR_Merlin(k) = exp(deltav/(g*Isp_Merlin));
    mProp_Merlin(k) = MR_Merlin(k)*(k*mInrt_Merlin+mInrt_Merlin_T(k)+m_pl) -
        k*mInrt_Merlin - mInrt_Merlin_T(k) - m_pl;
    mTOT_Merlin(k) = k*mInrt_Merlin + mInrt_Merlin_T(k) + mProp_Merlin(k);
    tBurn_Merlin(k) = mProp_Merlin(k)/mdot_Merlin(k);
```

```

Vprop_Merlin(k) = r_Merlin*mProp_Merlin(k)/(1+r_Merlin)/rho_LOX +
mProp_Merlin(k)/(1+r_Merlin)/rho_RP1;

mInrt_Merlin_T(k) = Vprop_Merlin(k)*tank_vol*rho_tank;
mInrt_Merlin_i = mInrt_Merlin_T(k) + k*mInrt_Merlin;

cost_Merlin(k) = r_Merlin*mProp_Merlin(k)/(1+r_Merlin)*c_LOX +
mProp_Merlin(k)/(1+r_Merlin)*c_RP1 + k*c_Merlin + 1e6;

end

```

% RL-10

---

```

=====
mInrt_RL10 = 300; % [kg]
Isp_RL10 = 460; % [s]
F_RL10 = 110e3; % [N]
tOp_RL10 = 4000; % [s]
r_RL10 = 5.88;
c_RL10 = 17e6; % [$]

% Source: https://yarchive.net/space/rocket/rl10.html
% Soruce: https://arstechnica.com/science/2017/12/nasa-is-trying-to-make-the-space-launch-system-rocket-more-affordable/#:~:text=But%20these%20engines%2C%20manufactured%20by,maiden%20Exploration%20Upper%20Stage%20vehicle\).

```

mInrt\_RL10\_T(k) = 0;

mInrt\_RL10\_i = 0;

```

while abs(mInrt_RL10_T(k) + k*mInrt_RL10 - mInrt_RL10_i) >= tol

mdot_RL10(k) = k*F_RL10/(Isp_RL10*g);

MR_RL10(k) = exp(deltav/(g*Isp_RL10));

mProp_RL10(k) = MR_RL10(k)*(k*mInrt_RL10+mInrt_RL10_T(k)+m_pl) -

k*mInrt_RL10 - mInrt_RL10_T(k) - m_pl;

mTOT_RL10(k) = k*mInrt_RL10 + mInrt_RL10_T(k) + mProp_RL10(k);

tBurn_RL10(k) = mProp_RL10(k)/mdot_RL10(k);

Vprop_RL10(k) = r_RL10*mProp_RL10(k)/(1+r_RL10)/rho_LOX +

mProp_RL10(k)/(1+r_RL10)/rho_LH2;

mInrt_RL10_T(k) = Vprop_RL10(k)*tank_vol*rho_tank;

mInrt_RL10_i = k*mInrt_RL10+mInrt_RL10_T(k);

cost_RL10(k) = r_RL10*mProp_RL10(k)/(1+r_RL10)*c_LOX +

mProp_RL10(k)/(1+r_RL10)*c_LH2 + k*c_RL10 + 1e6;

end

=====

% Raptor

mInrt_Raptor = 1500; % [kg]

Isp_Raptor = 380; % [s]

F_Raptor = 2200e3; % [N]

tOp_Raptor = 15*60; % [s]

r_Raptor = 3.55;

```

```
c_Raptor = 1e6; % [$]
```

% Source: <https://arstechnica.com/science/2020/05/nasa-will-pay-a-staggering-146-million-for-each-sls-rocket-engine/#:~:text=Speaking%20of%20engines%2C%20SpaceX%20is,to%20build%20a%20Raptor%20engine.>

% Source: <https://www.teslarati.com/spacex-raptor-engine-crushes-russian-record/#:~:text=For%20Starship%2C%20SpaceX%20needs%20%E2%80%93%20at,for%20efficient%20orbital%20Starship%20flights>

% Source: <https://twitter.com/elonmusk/status/1183866120240955392>

```
mInrt_Raptor_T(k) = 0;
```

```
mInrt_Raptor_i = 0;
```

```
while abs(mInrt_Raptor_T(k) + k*mInrt_Raptor - mInrt_Raptor_i) >= tol
    mdot_Raptor(k) = k*F_Raptor/(Isp_Raptor*g);
    MR_Raptor(k) = exp(deltav/(g*Isp_Raptor));
    mProp_Raptor(k) = MR_Raptor(k)*(k*mInrt_Raptor+mInrt_Raptor_T(k)+m_pl) -
    k*mInrt_Raptor - mInrt_Raptor_T(k) - m_pl;
    mTOT_Raptor(k) = k*mInrt_Raptor + mInrt_Raptor_T(k) + mProp_Raptor(k);
    tBurn_Raptor(k) = mProp_Raptor(k)/mdot_Raptor(k);
```

```
Vprop_Raptor(k) = r_Raptor*mProp_Raptor(k)/(1+r_Raptor)/rho_LOX +
mProp_Raptor(k)/(1+r_Raptor)/rho_LCH4;
mInrt_Raptor_T(k) = Vprop_Raptor(k)*tank_vol*rho_tank;
mInrt_Raptor_i = k*mInrt_Raptor + mInrt_Raptor_T(k);
```

```

cost_Raptor(k) = r_Raptor*mProp_Raptor(k)/(1+r_Raptor)*c_LOX +
mProp_Raptor(k)/(1+r_Raptor)*c_LCH4 + k*c_Raptor + 1e6;
end

% NERVA
=====

mInrt_NERVA = 15000; % [kg]
Isp_NERVA = 825; % [s]
F_NERVA = 333616.62; % [N]
nStarts_NERVA = 1000;%60
tOp_NERVA = 600*60; % [s]
c_NERVA = 662e6; % Source: https://www.researchgate.net/figure/ESTIMATED-COSTS-TO-REBUILD-A-NERVA-ENGINE\_tbl1\_234397993
% Source: https://fas.org/nuke/space/nerva-spec.pdf

mInrt_NERVA_T(k) = 0;
mInrt_NERVA_i = 0;

while abs(mInrt_NERVA_T(k) + k*mInrt_NERVA - mInrt_NERVA_i) >= tol
    mdot_NERVA(k) = k*F_NERVA/(Isp_NERVA*g);
    MR_NERVA(k) = exp(deltav/(g*Isp_NERVA));
    mProp_NERVA(k) = MR_NERVA(k)*(k*mInrt_NERVA+mInrt_NERVA_T(k)+m_pl) -
    k*mInrt_NERVA - mInrt_NERVA_T(k) - m_pl;
end

```

```

mTOT_NERVA(k) = k*mInrt_NERVA + mInrt_NERVA_T(k) + mProp_NERVA(k);
tBurn_NERVA(k) = mProp_NERVA(k)/mdot_NERVA(k);

Vprop_NERVA(k) = mProp_NERVA(k)/rho_LH2;
mInrt_NERVA_T(k) = Vprop_NERVA(k)*tank_vol*rho_tank;
mInrt_NERVA_i = k*mInrt_NERVA + mInrt_NERVA_T(k);

cost_NERVA(k) = mProp_NERVA(k)*c_LCH4 + (k-1)*.05*c_NERVA + c_NERVA +
1e6;
end

end

%% Results and Such

mTOT_AJ10_Opt = min(mTOT_AJ10(find(tBurn_AJ10 <= tOp_AJ10)));
k_AJ10_Opt = find(mTOT_AJ10_Opt == mTOT_AJ10);
Vprop_AJ10_Opt = Vprop_AJ10(k_AJ10_Opt);
mInrt_AJ10_Opt = k_AJ10_Opt*mInrt_AJ10 + mInrt_AJ10_T(k_AJ10_Opt);
mProp_AJ10_Opt = mProp_AJ10(k_AJ10_Opt);
tBrun_AJ10_Opt = tBurn_AJ10(k_AJ10_Opt);
TWR_AJ10 = k_AJ10_Opt*F_AJ10/(g*(mTOT_AJ10_Opt+m_pl));
cost_AJ10_opt = cost_AJ10(k_AJ10_Opt);

mTOT_Merlin_Opt = min(mTOT_Merlin(find(tBurn_Merlin <= tOp_Merlin)));
k_Merlin_Opt = find(mTOT_Merlin_Opt == mTOT_Merlin);

```

```

Vprop_Merlin_Opt = Vprop_Merlin(k_Merlin_Opt);

mInrt_Merlin_Opt = k_Merlin_Opt*mInrt_Merlin + mInrt_Merlin_T(k_Merlin_Opt);

mProp_Merlin_Opt = mProp_Merlin(k_Merlin_Opt);

tBrun_Merlin_Opt = tBurn_Merlin(k_Merlin_Opt);

TWR_Merlin = k_Merlin_Opt*F_Merlin/(g*(mTOT_Merlin_Opt+m_pl));

cost_Merlin_opt = cost_Merlin(k_Merlin_Opt);

mTOT_RL10_Opt = min(mTOT_RL10(find(tBurn_RL10 <= tOp_RL10)));

k_RL10_Opt = find(mTOT_RL10_Opt == mTOT_RL10);

Vprop_RL10_Opt = Vprop_RL10(k_RL10_Opt);

mInrt_RL10_Opt = k_RL10_Opt*mInrt_RL10 + mInrt_RL10_T(k_RL10_Opt);

mProp_RL10_Opt = mProp_RL10(k_RL10_Opt);

tBrun_RL10_Opt = tBurn_RL10(k_RL10_Opt);

TWR_RL10 = k_RL10_Opt*F_RL10/(g*(mTOT_RL10_Opt+m_pl));

cost_RL10_opt = cost_RL10(k_RL10_Opt);

mTOT_Raptor_Opt = min(mTOT_Raptor(find(tBurn_Raptor <= tOp_Raptor)));

k_Raptor_Opt = find(mTOT_Raptor_Opt == mTOT_Raptor);

Vprop_Raptor_Opt = Vprop_Raptor(k_Raptor_Opt);

mInrt_Raptor_Opt = k_Raptor_Opt*mInrt_Raptor + mInrt_Raptor_T(k_Raptor_Opt);

mProp_Raptor_Opt = mProp_Raptor(k_Raptor_Opt);

tBrun_Raptor_Opt = tBurn_Raptor(k_Raptor_Opt);

TWR_Raptor = k_Raptor_Opt*F_Raptor/(g*(mTOT_Raptor_Opt+m_pl));

cost_Raptor_opt = cost_Raptor(k_Raptor_Opt);

```

```

mTOT_NERVA_Opt = min(mTOT_NERVA(find(tBurn_NERVA <= tOp_NERVA)));
k_NERVA_Opt = find(mTOT_NERVA_Opt == mTOT_NERVA);
Vprop_NERVA_Opt = Vprop_NERVA(k_NERVA_Opt);
mInrt_NERVA_Opt = k_NERVA_Opt*mInrt_NERVA + mInrt_NERVA_T(k_NERVA_Opt);
mProp_NERVA_Opt = mProp_NERVA(k_NERVA_Opt);
tBrun_NERVA_Opt = tBurn_NERVA(k_NERVA_Opt);
TWR_NERVA = k_NERVA_Opt*F_NERVA/(g*(mTOT_NERVA_Opt+m_pl));
cost_NERVA_opt = cost_NERVA(k_NERVA_Opt);

mTOT_Opts = [mTOT_AJ10_Opt; mTOT_Merlin_Opt; mTOT_RL10_Opt; mTOT_Raptor_Opt;
mTOT_NERVA_Opt];
k_Opt = [k_AJ10_Opt; k_Merlin_Opt; k_RL10_Opt; k_Raptor_Opt; k_NERVA_Opt];
Vprop_Opt = [Vprop_AJ10_Opt; Vprop_Merlin_Opt; Vprop_RL10_Opt; Vprop_Raptor_Opt;
Vprop_NERVA_Opt];
mInrt_Opt = [mInrt_AJ10_Opt; mInrt_Merlin_Opt; mInrt_RL10_Opt; mInrt_Raptor_Opt;
mInrt_NERVA_Opt];
mProp_Opt = [mProp_AJ10_Opt; mProp_Merlin_Opt; mProp_RL10_Opt; mProp_Raptor_Opt;
mProp_NERVA_Opt];
tBurn_Opt = [tBrun_AJ10_Opt; tBrun_Merlin_Opt; tBrun_RL10_Opt; tBrun_Raptor_Opt;
tBrun_NERVA_Opt];
TWR = [TWR_AJ10; TWR_Merlin; TWR_RL10; TWR_Raptor; TWR_NERVA];
cost = cost_est*[cost_AJ10_opt; cost_Merlin_opt; cost_RL10_opt; cost_Raptor_opt;
cost_NERVA_opt

results = table(mTOT_Opts, k_Opt, Vprop_Opt, mInrt_Opt, mProp_Opt, tBurn_Opt, TWR, cost);

```

```
results.Properties.VariableNames = {'Total Stage Mass', 'Number of Engines', 'Propellant  
Volume', 'Inert Mass', 'Propellant Mass', 'Burn Time', 'Thrust to Weight Ratio','Cost Estimate'};  
results.Properties.VariableUnits = {'kg', '-', 'm^3', 'kg', 'kg', 's', '-', '$'};  
results.Properties.RowNames = {'AJ-10', 'Merlin', 'RL-10', 'Raptor', 'NERVA'};  
end
```

## References

- [1]. Kyle, E. (2020, July 15). Space Launch Report: Minotaur 4 Data Sheet. Minotaur 4 Data Sheet. <https://spacelaunchreport.com/minotaur4.html>.
- [2]. NASA. (2010). Minotaur V. NASA. Minotaur V.
- [3]. Schoneman, S., Amorosi , L., Cheke, D., & Chadwick, M. Minotaur V Space Launch Vehicle for Small, Cost-Effective Moon Exploration Missions. 21st Annual AIAA/USU Conference on Small Satellites.
- [4]. Krebs, G. D. Dnepr. Gunter's Space Page. [https://space.skyrocket.de/doc\\_lau/dnepr.htm](https://space.skyrocket.de/doc_lau/dnepr.htm).
- [5]. Kyle, E. (2015, March 25). Space Launch report: Dnepr Data Sheet. Dnepr Data Sheet. <https://spacelaunchreport.com/dnepr.html>.
- [6]. РИА. (2012, August 30). "Днепр" является самым выгодным средством запуска микроспутников. РИА Новости. <https://ria.ru/20120830/733725013.html>.
- [7]. Blau, P. Delta II 7920H-10. SPACEFLIGHT101. <https://web.archive.org/web/20140714122955/http://www.spaceflight101.com/delta-ii-7920h-10.html>.
- [8]. Kyle, E. (2018, October 20). Space Launch Report: Delta II Data Sheet. Delta II Data Sheet. <https://spacelaunchreport.com/delta2.html>.
- [9]. ULA. (2007). Delta II Payload Planners Guide. ULA Launch. <https://www.ulalaunch.com/docs/default-source/rockets/deltaiidayloadplannersguide2007.pdf>.
- [10]. ArianeSpace. (2012, March). Soyuz User's Manual. ArianeSpace. <https://www.arianespace.com/wp-content/uploads/2015/09/Soyuz-Users-Manual-March-2012.pdf>

[11]. Kyle, E. (2021, March 25). Space Launch Report: Soyuz Data Sheet. Soyuz Data Sheet.

<https://spacelaunchreport.com/soyuz.html>.

[12].Russian Aviation. (2018, October 3). Russian launch service provider reveals cost of Soyuz-2.1 rocket launch - News - Russian Aviation - RUAVIATION.COM. Russian Aviation.

<https://www.ruaviation.com/news/2018/10/3/12074/?h>

[13]. ULA. (2019, November 19). Rocket Rundown: A Fleet Overview. ULA.

<https://www.ulalaunch.com/docs/default-source/rockets/atlas-v-and-delta-iv-technical-summary.pdf>

[14].Wade, M. Atlas V. <http://www.astronautix.com/a/atlasv.html>

[15].Kyle, E. (2021, March 24). Space Launch Report: SpaceX Falcon 9 v1.2 Data Sheet.

SpaceX Falcon 9 v1.2 Data Sheet. <https://spacelaunchreport.com/falcon9ft.html>.

[16].SpaceX. Falcon 9: Capabilities & Services. SpaceX.

<https://www.spacex.com/media/Capabilities&Services.pdf>.

[17].SpaceX. Falcon 9. <https://www.spacex.com/vehicles/falcon-9/>.

[18].ESA. Ariane 5 ES. European Space Agency.

[https://web.archive.org/web/20140903072324/http://www.esa.int/Our\\_Activities/Launchers/Launch\\_vehicles/Ariane\\_5\\_ES](https://web.archive.org/web/20140903072324/http://www.esa.int/Our_Activities/Launchers/Launch_vehicles/Ariane_5_ES)

[19].Kyle, E. (2020, August 15). Space Launch Report: Ariane 5 Data Sheet. Ariane 5 Data Sheet. <https://spacelaunchreport.com/ariane5.html>.

[20].Wade, M. Ariane 5ES. <http://www.astronautix.com/a/ariane5es.html>

[21].Kyle, E. (2021, March 13). Space Launch Report: CZ-5 Data Sheet. CZ-5 Data Sheet. <https://spacelaunchreport.com/cz5.html>.

[22].Wade, M. Chang Zheng 5. <http://www.astronautix.com/c/changzheng5.html>

[23]. ArianeSpace. (2019, September). Ariane 6 Technical Overview. ArianeSpace.

[https://www.arianespace.com/wp-content/uploads/2020/06/Arianespace\\_Brochure\\_Ariane6\\_Sept2019.pdf](https://www.arianespace.com/wp-content/uploads/2020/06/Arianespace_Brochure_Ariane6_Sept2019.pdf)

[24]. Avio. (2018, January 1). Successful First Test Firing for the P120C Solid Rocket Motor for Ariane 6 and Vega-C: Avio. Avio.com. <https://www.avio.com/press-release/successful-test-firing-p120c-solid-rocket-motor>

[25]. ESA. Ariane 6. ESA.

[https://www.esa.int/Enabling\\_Support/Space\\_Transportation/Launch\\_vehicles/Ariane\\_6](https://www.esa.int/Enabling_Support/Space_Transportation/Launch_vehicles/Ariane_6)

[26]. Kyle, E. (2018, July 28). Space Launch Report: Ariane 6 Data Sheet. Ariane 6 Data Sheet. <https://spacelaunchreport.com/ariane6.html>.

[27]. Blue Origin. New Glenn. Blue Origin. <https://www.blueorigin.com/new-glenn/>

[28]. Kyle, E. (2020, August 29). Space Launch Report: New Glenn Data Sheet. New Glenn Data Sheet. <https://spacelaunchreport.com/newglenn.html>.

[29]. Clark, S. (2015, April 22). ULA needs commercial customers to close Vulcan rocket business case. Spaceflight Now <https://spaceflightnow.com/2015/04/22/ula-needs-commercial-business-to-close-vulcan-rocket-business-case/>

[30]. Kyle, E. (2020, September 6). Space Launch Report: Vulcan Data Sheet. Vulcan Data Sheet. <https://spacelaunchreport.com/vulcan.html>.

[31]. Peller, M. (2015, October 8). ULA Vulcan Centaur Overview. ULA. <https://web.archive.org/web/20160412062627/http://www.ispcs.com/content/files/Mark%20Peller.pdf>

[32]. Kyle, E. (2020, December 11). Space Launch Report: Delta IV Data Sheet. Delta IV Data Sheet. <https://spacelaunchreport.com/delta4.html>.

- [33]. ULA. (2013, June). Delta IV Users Guide. ULA Launch.  
[https://web.archive.org/web/20140710005717/http://www.ulalaunch.com/uploads/docs/Launch\\_Vehicles/Delta\\_IV\\_Users\\_Guide\\_June\\_2013.pdf](https://web.archive.org/web/20140710005717/http://www.ulalaunch.com/uploads/docs/Launch_Vehicles/Delta_IV_Users_Guide_June_2013.pdf).
- [34]. Kyle, E. (2019, June 25). Space Launch Report: SpaceX Falcon Heavy Data Sheet. SpaceX Falcon Heavy Data Sheet. <https://spacelaunchreport.com/falconH.html>.
- [35]. Kyle, E. (2021, March 3). Space Launch Report: SpaceX Super Heavy/Starship Data Sheet. SpaceX Super Heavy/Starship Data Sheet. <https://spacelaunchreport.com/bfr.html>.
- [36]. SpaceX. Starship. SpaceX. <https://www.spacex.com/vehicles/starship/>
- [37]. Spacexcmsadmin. (2019, September 28). Starship.  
<https://web.archive.org/web/20190930163150/https://www.spacex.com/starship>
- [38]. Wall, M. (2019, November 6). SpaceX's Starship May Fly for Just \$2 Million Per Mission, Elon Musk Says. Space.com. <https://www.space.com/spacex-starship-flight-passenger-cost-elon-musk.html>
- [39]. Kyle, E. (2020, May 2). Space Launch Report: SLS Data Sheet. SLS Data Sheet.  
<https://spacelaunchreport.com/sls0.html>.
- [40]. SpaceX, “Falcon 9,” *SpaceX* URL: <https://www.spacex.com/vehicles/falcon-9/> [Retrieved 4 April 2021]
- [41]. SpaceX, “Mission,” *SpaceX* URL: <https://www.spacex.com/mission/> [Retrieved 20 March 2021]
- [42]. SpaceX, “Capabilities & Services,” *SpaceX* URL:  
<https://www.spacex.com/media/Capabilities&Services.pdf> [Retrieved 20 March 2021]
- [43]. SpaceX, “Falcon User's Guide,” *SpaceX* URL:  
[https://www.spacex.com/media/Falcon\\_Users\\_Guide\\_082020.pdf](https://www.spacex.com/media/Falcon_Users_Guide_082020.pdf).

[Retrieved 5 April 2021]

[44]. SpaceX. (2018, January 13). Falcon Heavy on Launch Pad [Photograph]. Theverge.

<https://www.theverge.com/2018/1/3/16844472/spacex-falcon-heavy-rocket-upright-nasa-launch-pad-lc-39a>

[45]. Howell, E. H. (2018, February 22). Facts About SpaceX's Falcon Heavy Rocket.

Space.Com. <https://www.space.com/39779-falcon-heavy-facts.html>

[46]. Clark, S. C. (2021, February 15). SpaceX planning launch of two Falcon Heavy missions in summer and fall – Spaceflight Now. Spaceflight Now.

<https://spaceflightnow.com/2021/02/15/spacex-planning-launch-of-two-falcon-heavy-missions-in-summer-and-fall/>

[47]. SpaceX. (n.d.). Falcon Heavy. Retrieved April 5, 2021, from

<https://www.spacex.com/vehicles/falcon-heavy/>

[48]. SpaceX Merlin. (n.d.). Wikiwand. Retrieved April 5, 2021, from

[https://www.wikiwand.com/en/SpaceX\\_Merlin](https://www.wikiwand.com/en/SpaceX_Merlin)

[49]. SpaceX. (2020). Falcon User's Guide (1st ed., Vol. 1). SpaceX.

[https://www.spacex.com/media/falcon\\_users\\_guide\\_042020.pdf](https://www.spacex.com/media/falcon_users_guide_042020.pdf)

[50]. SpaceX. (2017, November 9). Merlin Engine [Photograph]. Businessinsider.

<https://www.businessinsider.com/spacex-merlin-block-5-engine-test-failure-2017-11>

[51]. Etherington, D. E. (2019, December 30). TechCrunch is now a part of Verizon Media.

Techcrunch. <https://techcrunch.com/2019/12/30/elon-musk-details-spacex-progress-on-latest-starship-spacecraft-build-and-flight-timelines/>

[52]. SpaceX. (n.d.). Starship. Retrieved April 5, 2021, from

<https://www.spacex.com/vehicles/starship/>

[53]. Wall, M. (2021, January 5). SpaceX targets bold new “catch” strategy for landing Super Heavy rockets. Space.Com. <https://www.space.com/spacex-starship-super-heavy-landing-plan>

[54]. Burghardt, T. “NASA

Moving Ahead with Return to the Moon, with or without SLS,” NASA SpaceFlight. URL: <https://www.nasaspacesflight.com/2019/05/nasa-return-moon-with-without-sls/> [Retrieved April 1. 2021]

[55]. NASA. “Orion Spacecraft,” NASA. URL:

<https://www.nasa.gov/exploration/systems/orion/index.html> [Retrieved April 1. 2021]

[56] NASA. “Gateway,” NASA. URL: <https://www.nasa.gov/gateway> [Retrieved April 1. 2021]

[57] NASA. “Space Launch System Solid Rocket Booster,” NASA. URL:

[https://www.nasa.gov/sites/default/files/atoms/files/sls\\_solid\\_rocket\\_booster\\_fs\\_03122018.pdf](https://www.nasa.gov/sites/default/files/atoms/files/sls_solid_rocket_booster_fs_03122018.pdf)

[Retrieved April 1. 2021]

[58] NASA. (2020). “Space Launch System (SLS) Overview,” NASA. URL:

<https://www.nasa.gov/exploration/systems/sls/overview.html> [Retrieved April 1. 2021]

[59] Office of Management and Budget. “Commerce, Justice, Science, and Related Agencies Appropriations Act, 2020,” Executive Office of the President. URL:

<https://www.whitehouse.gov/wp-content/uploads/2019/10/shelby-mega-approps-10-23-19.pdf>

[Retrieved April 1. 2021]

[60] Foust, J. “NASA Increases Cost Estimate for SLS Development,” SpaceNews. URL:

<https://spacenews.com/nasa-increases-cost-estimate-for-sls-development/#:~:text=A%20U.S.%20Government%20Accountability%20Office,billion%2C%20both%20as%20of%20January> [Retrieved April 1. 2021]

[61] Space-X. (2021). “Falcon Heavy,” Space-X. URL:

<https://www.spacex.com/vehicles/falcon-heavy/>

[Retrieved April 1, 2021]

[62] Heister, S., Anderson, W., Pourpoint, T., Cassady, R., (2019). (2019). Rocket Propulsion, Pg. 53.

[63] The Space Review. “Increasing the Profit Ratio,” SpaceNews. URL:

<https://www.thespacereview.com/article/2893/1#:~:text=Liquid%20methane%20costs%20about%20%241.35,which%20fuel%20will%20be%20cheaper.> [Retrieved March 8, 202 ]

[64] Forbes. (2017). “Why it’s So Cold in Outer Space,” Forbes.

URL: <https://www.forbes.com/sites/quora/2017/02/23/why-its-so-cold-in-outer-space/?sh=16653af06f4b> [Retrieved April 1, 2021]

[65]. Wall, M. “SpaceX’s Starship May Fly for Just \$2 Million Per Mission, Elon Musk Says,” *Space.com*. URL: <https://www.space.com/spacex-starship-flight-passenger-cost-elon-musk.html> [Retrieved April 1, 2021]

[66]. Wall, M. “SpaceX Falcon 9 Rocket Will Launch Private Moon Lander in 2021,” *Space.com*. URL: <https://www.space.com/intuitive-machines-moon-lander-spacex-2021.html> [Retrieved April 1, 2021]

[67] *National Institute of Standards and Technology*, URL: <https://www.nist.gov/> [Retrieved 5 April 2021]

[68] “Aerojet General Space Engines,” *Space Engine Encyclopedia*, URL:  
[https://www.alternatewars.com/BBOW/Space\\_Engines/Aerojet\\_Engines.htm](https://www.alternatewars.com/BBOW/Space_Engines/Aerojet_Engines.htm) [Retrieved 5 April 2021]

[69] “AJ10-118,” *Encyclopedia Astronautica*, URL:

<https://web.archive.org/web/20080706190928/http://astronautix.com/engines/aj10118.htm>

[Retrieved 5 April 2021]

[70] Berger, E., “NASA is trying to make the Space Launch System rocket more affordable,”

*arsTechnica*, URL: [https://arstechnica.com/science/2017/12/nasa-is-trying-to-make-the-space-launch-system-rocket-more-](https://arstechnica.com/science/2017/12/nasa-is-trying-to-make-the-space-launch-system-rocket-more-affordable/#:~:text=But%20these%20engines%2C%20manufactured%20by,maiden%20Explorat)

affordable/#:~:text=But%20these%20engines%2C%20manufactured%20by,maiden%20Exploration%20Upper%20Stage%20vehicle) [Retrieved 5 April 2021]

[71] Berger, E., “NASA will pay a staggering \$146 million for each SLS rocket engine,”

*arsTechnica*, URL: [https://arstechnica.com/science/2020/05/nasa-will-pay-a-staggering-146-million-for-each-sls-rocket-](https://arstechnica.com/science/2020/05/nasa-will-pay-a-staggering-146-million-for-each-sls-rocket-engine/#:~:text=Speaking%20of%20engines%2C%20SpaceX%20is,to%20build%20a%20Raptor)

engine/#:~:text=Speaking%20of%20engines%2C%20SpaceX%20is,to%20build%20a%20Raptor%20engine [Retrieved 5 April 2021]

[72] “Falcon User’s Guide,” *SpaceX*,

[https://www.spacex.com/media/falcon\\_users\\_guide\\_042020.pdf](https://www.spacex.com/media/falcon_users_guide_042020.pdf) [Retrieved 5 April 2021]

[73] Wang, B., “SpaceX Raptor Engine Will Be Best on Cost and Nearly Best on ISP,”

*NextBIGFuture*, URL: <https://www.nextbigfuture.com/2019/05/spacex-raptor-engine-will-be-best-on-cost-and-nearly-best-on-isps.html#:~:text=SpaceX%20Merlin%20engine%20costs%20less,in%20later%20simplified%20Raptor%20versions> [Retrieved 5 April 2021]

[74] “NERVA Rocket Engine Detail Specification,” Federation of American Scientists, URL:

<https://fas.org/nuke/space/nerva-spec.pdf> [Retrieved 5 April 2021]

[75] Howe, S. D., "ESTIMATED COSTS TO REBUILD A NERVA ENGINE," ResearchGate, URL: [https://www.researchgate.net/figure/ESTIMATED-COSTS-TO-REBUILD-A-NERVA-ENGINE\\_tbl1\\_234397993](https://www.researchgate.net/figure/ESTIMATED-COSTS-TO-REBUILD-A-NERVA-ENGINE_tbl1_234397993) [Retrieved 5 April 2021]

[76]. Blue Origin. BE-3. Blue Origin. <https://www.blueorigin.com/engines/be-3>

[77]. Blue Origin. BE-4. Blue Origin. <https://www.blueorigin.com/engines/be-4>

[78]. Brugge. (2008). Delta-II, Fairings.

[http://www.b14643.de/Spacerockets\\_2/United\\_States\\_5/Delta\\_II/Fairings/Delta\\_II.htm](http://www.b14643.de/Spacerockets_2/United_States_5/Delta_II/Fairings/Delta_II.htm).

[79]. Blue Origin. BE-3. Blue Origin. <https://www.blueorigin.com/engines/be-3>\

[80]. Blue Origin. BE-4. Blue Origin. <https://www.blueorigin.com/engines/be-4>

[81]. Brugge. (2008). Delta-II, Fairings.

[http://www.b14643.de/Spacerockets\\_2/United\\_States\\_5/Delta\\_II/Fairings/Delta\\_II.htm](http://www.b14643.de/Spacerockets_2/United_States_5/Delta_II/Fairings/Delta_II.htm).

[82]. Gunter Dirk Krebs. Delta-4 (Delta-IV). Gunter's Space Page.

[https://space.skyrocket.de/doc\\_lau/delta-4.htm](https://space.skyrocket.de/doc_lau/delta-4.htm).

[83]. J.A. Dallas, S. Raval, J.P. Alvarez Gaitan, S. Saydam, A.G. Dempster, "The environmental impact of emissions from space launches: A comprehensive review", Journal of Cleaner Production, Volume 255, 2020, 120209, ISSN 0959-6526,

<https://doi.org/10.1016/j.jclepro.2020.120209>.

(<https://www.sciencedirect.com/science/article/pii/S0959652620302560>)

[84]. Kosmotras. Космические головные части РН "Днепр". Kosmotras.

[http://kosmotras.ru/upload/kgc\\_fevral\\_2014.pdf](http://kosmotras.ru/upload/kgc_fevral_2014.pdf).

[85]. Musk, E. (2019, September 26). Mk1 ship is around 200 tons dry & 1400 tons wet, but aiming for 120 by Mk4 or Mk5. Total stack mass with max payload is 5000 tons. Twitter.

<https://twitter.com/elonmusk/status/1177066483375058944>.

- [86]. Northrop Grumman. (2018, September). GEM 63/ GEM 63XL. Northrop Grumman.  
[https://web.archive.org/web/20180918143456/http://www.northropgrumman.com/Capabilities/GEM/Documents/GEM\\_63\\_GEM\\_63XL.pdf](https://web.archive.org/web/20180918143456/http://www.northropgrumman.com/Capabilities/GEM/Documents/GEM_63_GEM_63XL.pdf)
- [87]. Smith, R. (2018, June 2). Europe Complains: SpaceX Rocket Prices Are Too Cheap to Beat. The Motley Fool. <https://www.fool.com/investing/2018/06/02/europe-complains-spacex-rocket-prices-are-too-cheap.aspx>
- [88]. Wade, M. Vinci. <http://www.astronautix.com/v/vinci.html>
- [89]. Wade, M. Vulcain 2. <http://www.astronautix.com/v/vulcain2.html>
- [90]. Waldron, G. (2019, October 9). Arianespace aims high in Asia-Pacific. Flight Global.  
<https://www.flightglobal.com/arianespace-aims-high-in-asia-pacific/120757.article>

## I. Landing Vehicles Appendix

### A. Ice Calculator

```

function ice_calc(fuel_req, ox_req)
%fuel_req mass of fuel required(kg), ox_req mass of ox required(kg)

%% Starship Propellant Mass: 1200 tons (78% O2, 22% CH4)-> 8.4913e+05 kg LOX,
2.3950e+05 kg CH4 (https://www.spacex.com/vehicles/starship/, mixture ratio 3.55

%https://www.hou.usra.edu/meetings/lpsc2019/pdf/2628.pdf CO2 ice on the
%moon

% Ice on the moon https://nssdc.gsfc.nasa.gov/planetary/ice/ice\_moon.html

density_ice_h20 = 917;           %kg/m^3      https://www.usgs.gov/special-topic/water-science-school/science/water-density?qt-science\_center\_objects=0#

density_ice_CO2 = 1560;          %kg/m^3      https://sciencing.com/density-co2-7324875.html

ch4_mw = 16.04E-3;             %kg/mol

h2_mw = 2E-3;                 %kg/mol

h2o_mw = 18E-3;               %kg/mol

o2_mw = 32E-3;                %kg/mol

%% Methane to Carbon Dioxide

fuel_req_mole = fuel_req / ch4_mw;    %mole CH4 required

mole_CO2_req = fuel_req_mole;        %mole CO2

mass_CO2_req = mole_CO2_req * 44.01E-3; %kg CO2

volume_CO2_req = mass_CO2_req / density_ice_CO2; %Volume of frozen CO2 required

mole_H2_req = mole_CO2_req * 4;

mass_H2_req = mole_H2_req * h2_mw;

mole_H2O_req = mole_H2_req;

mass_H2O_req = mole_H2O_req * h2o_mw;

mole_O2_req = mole_H2_req / 2;

mass_O2_req = mole_O2_req * o2_mw;

volume_H2O_req = mass_H2O_req / density_ice_h20;

if mass_O2_req < ox_req

```

```
add_mass_O2 = ox_req - mass_O2_req;  
add_mole_O2 = add_mass_O2 / h2o_mw;  
add_mole_H2O = add_mole_O2*0.5;  
add_mass_H2O = add_mole_H2O * h2o_mw;  
mass_H2O_req = mass_H2O_req + add_mass_H2O;  
volume_H2O_req = mass_H2O_req / density_ice_h20;  
end  
fprintf('-----\n')  
fprintf('Water Ice Required: %.2f kg | %.02f m^3\n',mass_H2O_req, volume_H2O_req);  
fprintf('CO2 Ice Required: %.2f kg | %.02f m^3\n',mass_CO2_req, volume_CO2_req);  
End
```

## B. Apollo Cargo Sizing

```
[1] Apollo Cargo Sizing: Apollo_Cargo_Size
```

```
clear
```

```
clc
```

```
%% Initial Constants
```

```
g = 9.81; %m/s^2
```

```
dv_NASA = 2.500; %km/s
```

```
isp = 311; %s
```

```
mu = .04908*10^5; %km^3/s
```

```
r = 937.9; %km
```

```
z = 110; %km
```

```
th = -87; %degrees
```

```
m_lander = 10334; %kg
```

```
m_p = 8200; %kg
```

```
m_i = m_lander - m_p;
```

```
mpl_NASA = 2445 + 2376; %kg
```

```
mp_NASA = 8200; %kg
```

```
%% Landing Trajectory
```

```
vc = sqrt(mu/(r+z));
```

```
vsurf = (2*pi*r*cosd(th))/(24*60*60);
```

```
dv = vc - vsurf;
```

```
%% Mass Analysis
```

```
lam = m_p/(m_lander);
```

```
mpl = linspace(1000,8000,1000);

MR = (mpl + m_p + m_i)./(mpl + m_i);

MRr = exp(dv*1000/(g*isp));

m_pr = mpl.* (MRr-1)./(MRr-(MRr-1)/lam); %Required Prop Mass

m_tot_r = m_pr+m_i+mpl; %Total Mass

m_tot_NASA = mpl_NASA+mp_NASA+m_lander;
```

%% Constants For Graphs:

```
prop_tank = 8200; %kg, propellant capacity of the Apollo Lander

fh_moon = 15000; %kg, Falcon Heavy Payload to the moon
```

%% Graphs

```
figure()

plot(mpl, m_tot_r)

grid on

xlabel('Payload Mass, kg')

ylabel('Lander Mass, kg')

title('Total Lander Mass for Given Payload Mass')

yline(fh_moon,'r');

legend('Lander Mass', 'Falcon Heavy Lunar Payload')

figure()

plot(mpl, m_pr)

grid on

yline(prop_tank,'r');

xlabel('Payload Mass, kg')
```

```
ylabel('Propellant Mass, kg')  
title('Required Propellant Mass for Given Payload Mass')  
legend('Propellant Mass', 'Lunar Module Fuel Capacity')
```

## References

- [1] Apollo 17 (AS-512) Available: <https://airandspace.si.edu/explore-and-learn/topics/apollo/apollo-program/landing-missions/apollo17.cfm>.
- [2] “Artemis,” NASA Available: <https://www.nasa.gov/specials/artemis/>.
- [3] “About the Spacecraft,” *About the Spacecraft* | National Air and Space Museum Available: <https://airandspace.si.edu/exhibitions/apollo-to-the-moon/online/apollo-11/about-the-spacecraft.cfm>.
- [4] “Entry, Descent and Landing (EDL),” NASA Available: <https://mars.nasa.gov/mars2020/timeline/landing/entry-descent-landing/>.
- [5] “Cruise Timeline,” NASA Available: <https://mars.nasa.gov/msl/timeline/cruise/>.
- [6] Cohen, D., “The Sky Crane Solution,” NASA Available: <https://appel.nasa.gov/2012/07/31/the-sky-crane-solution/>.
- [7] Mahoney, E., “NASA Selects Blue Origin, Dynetics, SpaceX for Artemis Human Landers,” NASA Available: <https://www.nasa.gov/feature/nasa-selects-blue-origin-dynetics-spacex-for-artemis-human-landers>
- [8] “HLS National Team,” *Blue Origin* Available: <https://www.blueorigin.com/blue-moon/national-team>.
- [9] Foust, J., “Blue Origin team delivers lunar lander mockup to NASA,” *SpaceNews* Available: <https://spacenews.com/blue-origin-team-delivers-lunar-lander-mockup-to-nasa/>.
- [10] “Dynetics To Develop Nasas Artemis Human Lunar Landing System,” *Dynetics* Available: <https://www.dynetics.com/newsroom/news/2020/dynetics-to-develop-nasas-artemis-human-lunar-landing-system>.

[11] Clark, S., “Companies release new details on human-rated lunar lander concepts,” *Spaceflight Now* Available: <https://spaceflightnow.com/2020/04/30/companies-release-new-details-on-human-rated-lunar-lander-concepts/>

[12] Wall, M., “Jeff Bezos' Blue Origin won't launch its 1st New Glenn rocket until late 2022,” Space.com Available: <https://www.space.com/blue-origin-new-glenng-rocket-first-launch-2022>.

[13] Henry, C., “Blue Origin enlarges New Glenn's payload fairing, preparing to debut upgraded New Shepard,” *SpaceNews* Available: <https://spacenews.com/blue-origin-enlarges-new-glenng-payload-fairing-preparing-to-debut-upgraded-new-shepard/>.

[14] Clark, S., “ULA needs commercial customers to close Vulcan rocket business case,” *Spaceflight Now* Available: <https://spaceflightnow.com/2015/04/22/ula-needs-commercial-business-to-close-vulcan-rocket-business-case/>.

[15] “Blue Origin's BE-7 Engine Testing Further Demonstrates Capability to Land on the Moon,” *Blue Origin* Available: <https://www.blueorigin.com/news/be7-engine-testing>.

[16] “Rocket Rundown: A Fleet Overview,” *ulalaunch.com* Available: <https://www.ulalaunch.com/docs/default-source/rockets/atlas-v-and-delta-iv-technical-summary>.

[17] “Blue Moon,” *Blue Origin* Available: <https://www.blueorigin.com/blue-moo>

[18] Wattles, J., "NASA says moon rocket could cost as much as \$1.6 billion per launch," *CNN* Available: <https://www.cnn.com/2019/12/09/tech/nasa-sls-price-cost-artemis-moon-rocket-scn/index.html>

[19] "Space Launch System Fact Sheet," *NASAfacts* Available: [https://www.lpi.usra.edu/lunar/constellation/SLS\\_FactSheet\\_long](https://www.lpi.usra.edu/lunar/constellation/SLS_FactSheet_long)

[20] Mann, A., "SpaceX now dominates rocket flight, bringing big benefits-and risks-to NASA," *Science* Available: <https://www.sciencemag.org/news/2020/05/spacex-now-dominates-rocket-flight-bringing-big-benefits-and-risks-nasa>

[21] "SpaceX Raptor ,," *Spaceflight101* Available: <https://spaceflight101.com/spx/spacex-raptor/>

[22] "Starship," *SpaceX* Available: <https://www.spacex.com/vehicles/starship/>

[23] Cherne, J. M. (n.d.). *Mechanical Design of the Lunar Module Descent Engine* [PDF]. Redondo Beach: TRW Systems. Pg. 1-3,8,10

[24] NASA. (1966, June 25). APOLLO LUNAR LANDING MISSION SYMPOSIUM. Retrieved April 05, 2021, from [https://www.hq.nasa.gov/alsj/LunarLandingMIssionSymposium1966\\_1978075303.pdf](https://www.hq.nasa.gov/alsj/LunarLandingMIssionSymposium1966_1978075303.pdf) pg.49

[25] Williams, D., Dr. (2016, May 16). The Apollo Lunar Roving Vehicle. Retrieved April 05, 2021, from [https://nssdc.gsfc.nasa.gov/planetary/lunar/apollo\\_lrv.html#:~:text=The%20Lunar%20Roving%20Vehicle%20had,a%20wheelbase%20of%202.3%20meters](https://nssdc.gsfc.nasa.gov/planetary/lunar/apollo_lrv.html#:~:text=The%20Lunar%20Roving%20Vehicle%20had,a%20wheelbase%20of%202.3%20meters). pg.1

[26] Grumman. (1970). *Lunar Module Derivatives for Future Space Missions* [PDF]. NASA. pg. 2,

[27] Foust, J. (2020, May 04). NASA evaluation sees SpaceX lunar lander as innovative but risky.

Retrieved April 06, 2021, from <https://spacenews.com/nasa-evaluation-sees-spacex-lunar-lander-as-innovative-but-risky/>

[28] Blue moon. (n.d.). Retrieved April 05, 2021, from <https://www.blueorigin.com/blue-moon/>

[29] NASA. (n.d.). Artemis. Retrieved April 05, 2021, from <https://www.nasa.gov/specials/artemis/>

[30] Foust, J. (2020, August 20). Blue origin team delivers lunar LANDER mockup to NASA.

Retrieved April 05, 2021, from <https://spacenews.com/blue-origin-team-delivers-lunar-lander-mockup-to-nasa/>

[31] SpaceX. (n.d.). Starship. Retrieved April 05, 2021, from <https://www.spacex.com/vehicles/starship/>

[32] Ivankov, Artem. Luna 9. <https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1966-006A>. Accessed Apr.5,2021

[33] Gwinn, K. W., “Structural Analyses of the JPL Mars Pathfinder Impacts,” *Structural Analyses of the JPL Mars Pathfinder Impacts (Technical Report)* | OSTI.GOV Available: <https://www.osti.gov/servlets/purl/5898>.

# Propellant Depot Appendix

## I. Starship and Superheavy Performance

As discussed in the report, both Starship and the Super Heavy Booster must be landed propulsively. To determine their available performance, we first find how much propellant is needed to land each vehicle. To determine the velocity change requirements for these landings, we calculate the terminal velocity of each vehicle.

To determine the terminal velocity that must be cancelled, we assume that both Starship and the SHB have drag characteristics of simple solids. The SHB was approximated as a cylinder in axial flow with a  $C_d$  of 0.81 [1] and a frontal area of  $63.62\text{m}^2$ . This was found using the diameter of the SHB of 9m [2]. Additionally, the SHB has flat fins that stabilize it during landings. Based on scale relative to Falcon 9, we estimate these grid fins to have a surface area of roughly  $50\text{m}^2$ . We model the grid fins as flat plates with a  $C_d$  of 1.28 [3]. For Starship, we model it as a cylinder in crossflow, with a  $C_d$  of 0.74 [4] and a cross sectional area of  $450\text{m}^2$ . The cross sectional area is based off of Starships width and height of 9m and 50m [2], respectively.

Using the drag equation, we can determine the velocity at which the drag force equals the weight of each vehicle. With a velocity, the ideal rocket equation can be used to determine the propellant requirements to provide that much velocity change. As the weight of the vehicle depends on the propellant needed to land the vehicle, and the propellant required depends on the terminal velocity of the vehicle, we have a circular dependency. The equations 1 are solved iteratively.

$$m_1 g_e = \frac{\rho A v^2 C_d}{2} \quad \Delta v = g_e \cdot I_{SP} \cdot \ln\left(\frac{m_o}{m_f}\right) \quad \rho = 1.225 \text{ kg/m}^3 \quad (1)$$

From this, we determine that 6,362kg of propellant is needed to land Starship, and 27,302kg of propellant is needed to land the SHB.

While the SHB tumbles back to Earth after it separates, Starship exits orbit first, and then falls back to Earth. This requires propellant to do. We use a Hohmann transfer to lower the periapsis of Starship's orbit from 500km to 100km, where Starship is subject to atmospheric drag, slowing it to a ballistic trajectory back to the Earth's surface. The velocity change for this maneuver described by equation 2.

$$\Delta v = \sqrt{\frac{\mu_e}{r_e + z_2}} \cdot \left(1 - \sqrt{\frac{2(z_1 + r_e)}{2r_e + z_1 + z_2}}\right) \text{ where } z_2 > z_1 \quad (2)$$

The deorbit maneuver requires Starship to reduce its velocity by 141.7 m/s if done impulsively. At 500km, the time it takes Starship to achieve this velocity change is less than 10 seconds or 0.18% of the orbital period so we assume the maneuver to be impulsive. We then input this velocity into the ideal rocket equation to determine the mass of propellant required for this maneuver as shown in equation 3.

$$m_{prop} = m_f \left( e^{\frac{\Delta v}{g_e I_{SP}}} - 1 \right) \quad (3)$$

We know that when the ascent part of the flight is done, Starship will be carrying the payload, the propellant required for a de-orbiting burn, and the propellant needed for landing. This is a final mass that can be used in the ideal rocket equation to determine the amount of  $\Delta v$

that Starship can provide. To determine the amount of  $\Delta v$  available accurately, the changes in Raptor engine performance with altitude are considered. When Starship separates from the SHB, its vacuum optimized Raptor engines aren't operating at maximum efficiency due to overexpansion of its exhaust gasses. As such, the  $I_{SP}$  is reduced while in the atmosphere. To avoid a full flight path and altitude simulation, we estimate the average  $I_{SP}$  for Starship to be 365 seconds. With this information, the ideal rocket equation shows that Starship can provide 6,237m/s of  $\Delta v$ .

Similarly, we know that the SHB will separate once it only has enough fuel left to land, and we therefore know how much fuel it burns and how much  $\Delta v$  it can provide. We assume the average  $I_{SP}$  for the SHB to be 353 seconds as we assume the SHB Raptor engines are optimized for post liftoff conditions. With this information, the ideal rocket equation shows that the SHB can provide 3,681m/s of  $\Delta v$ . Bringing the total system  $\Delta v$  to 9,918m/s.

## II. Decision on Lunar Station

When we originally began designing we had considered having a station in lunar orbit. This station would operate in a way that was similar to a gateway and act as a hub for the astronauts to stop at before they made their ascent to the lunar colony. The team never got into specifics about the design, instead working on the gateway design as we weighed our options on whether or not we should design one for our mission. When described by NASA [5] gateway is split into two main components. These are the power and propulsion element and the Habitation and Logistics Outpost (HALO). The Power and propulsion element is an electric propulsion spacecraft. This would provide power, altitude control, orbital transfer capabilities and high-rate communications. The Habitation and Logistics Outpost is the crew cabin. It has all the basic life support needed to allow the astronauts to stay there while they prepare for their mission. This craft has command and control as well as energy storage, power distribution, thermal control, environment control, life support systems, communication and tracking, and finally data handling.

We examined the advantages and disadvantages of having such a craft. The main advantages that we found were that it allowed for a longer mission duration, could allow for the use of mixed fleet launch vehicles, would supply an established human landing system, give access to the entire lunar surface, allow for constant communication and offer a sense of sustainability. The disadvantages were that it added extra cost, and extra risk from a systems standpoint. It was also not necessary due to the ability to land directly on the moon, and offered few real advantages to our system before we had created the colony and even after that it might not be worth it due to other design choices, mainly the fact that we will not be manufacturing

fuel on the moon. Below is a greater breakdown of what we found in our research that made us ultimately decide not to have a station in lunar orbit.

One of the largest advantages was that the lunar station could allow for a longer mission duration. The Gateway station is meant to receive deliveries of logistics modules. These carry extra supplies to the station that can increase the mission time estimate. By some estimates they can even double the mission time that NASA is preparing for with one delivery [6] . Taking it from 30 days for NASA, to 60 days. This is a huge advantage for their purposes but since we are planning on having a lunar colony that is equipped to have people live there for much longer times this would not be very helpful for us in the long run. Though there may be slight upside during the construction of the lunar colony it would not be enough to justify the station.

Another advantage was the ability to use mixed fleet launches [7]. This is due to the fact that getting to the station is simpler than getting to the lunar surface. This could allow for collaboration from different companies with different launch vehicles. Allowing for the possibility that one day they could send up supplies in a way that is similar to how the ISS receives its shipments, a rotation of different companies and countries completing the deliveries. This would also allow for groups that normally did not get access to the moon to participate in the colony. Though this is a nice notion it is not really a big priority for us. Also due to our decision to use the Starship line once it is available for pretty much all of our launches this ability was not going to do us any long term favors.

Another advantage is that we would have an established and reusable landing system that would be docked in the station [7] . This would mean that we would not need to send a landing

system with every manned launch meant to take astronauts to the lunar surface. The station would also have the ability to refuel and refurbish the lander as needed. This overall would have been helpful for our proposed system.

The ability to have access to the entire lunar surface was another attractive quality of the Lunar station [7]. With the use of its PPE to maneuver between orbits, the gateway will be able to give its astronauts the ability to land anywhere. This would be helpful if there was interest in creating multiple structures in different areas of the moon, allowing the astronauts to be able to land multiple places with an ease that would not be allowed when having to calculate from the initial launch from earth. Instead they could be received into the station at its regular orbit and then change orbits to allow them to descend to their ideal location.

The communication system planned to be included on Gateway is also a possible plus. This is due to its large link budget, the value of which seems not to be publicly available, and the ability for constant communication [7]. This is due to the home orbit of the gateway that is used which allows the gateway to always reside on the correct side of the moon allowing for constant connection. This is a great feature that we would obviously want to have if we were going to have a gateway since we would want the people there to be able to have good communication with those on earth. However, this is not a reason in itself to have a lunar station, especially because the communication team had already begun creating a robust system of their own using smaller satellites and showed little interest in the Gateways extra abilities.

An overarching advantage was just Gateway's ability to add some sustainability to the lunar station system. This was partially due to making things like delivering equipment simpler

and easier to accomplish. Other than that having a reusable structure that can be built up modularly to fit any new additional needs is very helpful and can promise some security for the future.

An argument against gateway is that its true worth is tied to whether there is a functioning base making propellant on the moon [8]. Saying that it is not worth it to have one until you require a place to store it in orbit. Then having this station store that propellant and the reusable lander to taxi between the propellant manufacturing plant would be a big plus. Also the ability of the place to then use this fuel to refuel then would be great since it would be able to act as a lunar propellant depot. This however does not apply to our colony due to the fact that we are not producing any fuel on the moon. Therefore we will never be able to put this advantage to use.

A great disadvantage was cost. Below in Table 1 we can see the cost estimation for Gateway. This shows the breakdown for Gateways 3.8 billion dollars. We can ignore the space suit cost since that will already be necessary before gateway. But the rest of that is the record of the cost that will be added on. This is a large amount of money and the advantages listed above do not really justify it.

**Table 1. Cost Estimation for Gateway [9]**

| <b>Section of Project</b>   | <b>Estimated cost (in Millions)</b> |
|-----------------------------|-------------------------------------|
| PPE                         | 550.6                               |
| HALO                        | 999.3                               |
| Logistics Element           | 264.6                               |
| Program Mission Execution   | 768.5                               |
| Space Suits                 | 863.6                               |
| Mission Directorate Support | 93.2                                |
| Total                       | 3,592.40                            |

Another problem that people have pointed out about adding the stations is that from a systems standpoint it adds risk [8] . When trying to create a low risk system it is important to attempt to streamline everything as much as you can, creating a system that has a small amount of simple steps. By adding a gateway to the system you complicate it, adding a lot of unnecessary steps, each step bringing more risk of failure.

Overall after examining how the advantages and disadvantages related to the goals we have as a team we decided against having it. This was due to the fact that the advantages were rather specific to NASAs plans and either did not apply to our goals at all or were marginal at best. The disadvantages on the other hand were rather general and therefore still applied.

### III. Propellant Mass Bookkeeping Along the Mission

To determine whether vehicle refueling is needed and at which points in the mission we would require refueling, we must create an analysis for bookkeeping the propellant mass available in Starship's tanks throughout the mission. We perform this analysis by stepping through the entire mission in 'legs,' based on the  $\Delta V$  required for each mission leg. The breakdown of the mission and the required  $\Delta V$  for both crewed and cargo missions are shown in Table 2.

**Table 2  $\Delta V$  Requirements for Crewed and Uncrewed Missions**

| Mission Leg                 | $\Delta V$ Required (km/s) |          |
|-----------------------------|----------------------------|----------|
|                             | Crewed                     | Uncrewed |
| Earth Ascent to 600 km LEO  | 9.5                        | 9.5      |
| TLI                         | 3.1468                     | 3.1809   |
| FRD                         | 0.19963                    | -        |
| LOI                         | 0.95023                    | -        |
| Lunar Circularization       | -                          | 0.8928   |
| 100 km LLO to Lunar Surface | 2.5                        | 2.5      |
| Lunar Surface to 100 km LLO | 2.2                        | 2.2      |
| 100 km LLO to Earth Reentry | 1.1388                     | 1.1388   |

Using the ideal rocket equation, we relate the vehicle's initial and final masses to the  $\Delta V$  required for that leg. Knowing that the tanks will still carry leftover propellant after each leg, we can also calculate the mass of propellant used and leftover afterwards. When using this method to step through the mission, the  $m_{prop, leftover}$  variable becomes the  $m_{prop}$  for the following leg.

$$MR = \frac{m_o}{m_f} = e^{\frac{\Delta V}{g_e I_{sp}}} \quad (4)$$

$$m_o = m_{inert} + m_{payload} + m_{prop} \quad m_f = m_{inert} + m_{payload} + m_{prop, leftover} \quad (5-6)$$

$$m_{prop, leftover} = m_{prop} - m_{prop, used} \quad (7)$$

We assume the inert mass of the Starship upper stage is 120 Mg, as this is SpaceX's target design weight [10]. Considering SpaceX's published "100+ tons" of payload capacity in their Starship user guide, the payload mass capacity used for the analysis is a maximum of 100 Mg to be conservative [11]. Carrying out these calculations for the entire mission, we determine that even when the Starship upper stage carries zero payload, there is not enough propellant to accomplish the lunar orbit insertion burn. This is true for both crewed and uncrewed mission types. We conclude that a propellant depot located in LEO for vehicle refueling is necessary to send the Starship upper stage to the lunar surface.

Another important portion of the mission is returning crew from the Moon back to Earth. As a test case, we consider the situation where a Starship with zero payload launches from Earth, fully refuels at a 600 km LEO propellant depot, and continues on the rest of the mission. Using the propellant bookkeeping analysis discussed earlier, we find that there is not enough propellant to complete the final burn from LLO to earth reentry. In addition, the propellant deficit occurs during earlier mission legs when payload is added to the vehicle. To complete the entire mission using the Starship upper stage, we must incorporate LEO and LLO propellant depots into the overall mission architecture.

Initially, we had considered an alternate launch system in-case Starship would be unavailable for the early years of the mission. We decided to look into the fuel requirements associated with using the Falcon 9 and Falcon Heavy as the main launch vehicles. The Falcon Heavy uses a Merlin 1D with an Isp of 311s for its first stage. It uses a Merlin 1Dvac with an Isp of 348s. The Falcon 9 uses a Merlin 1C with an Isp of 304s for its first stage and Merlin 1Cvac

with an Isp of 342s for its second stage [12,13,14]. Using the above mentioned equation and assuming a two-staged flight, we get the following fuel requirements:

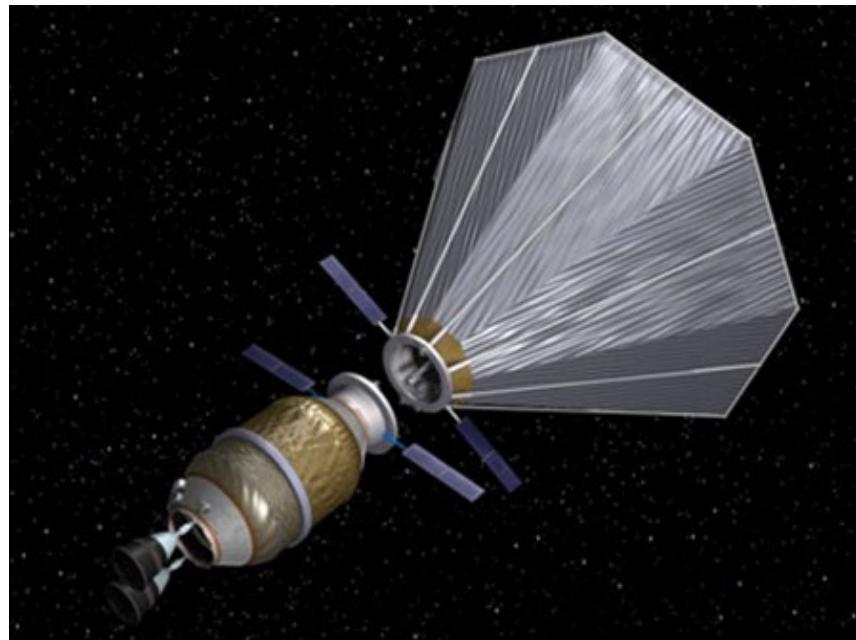
**Table 3: Fuel Requirements for Alternate Launch Vehicles**

| <b>Alternate Launch Vehicles</b> |                              |
|----------------------------------|------------------------------|
| <b>Fuel Requirements</b>         |                              |
| <b>Vehicle</b>                   | <b>Total Propellant (Mg)</b> |
| Falcon Heavy                     | 433.669                      |
| Falcon 9 Expendable              | 163.215                      |
| Falcon 9 Reusable                | 120.263                      |

#### IV. Thermal and Power Systems

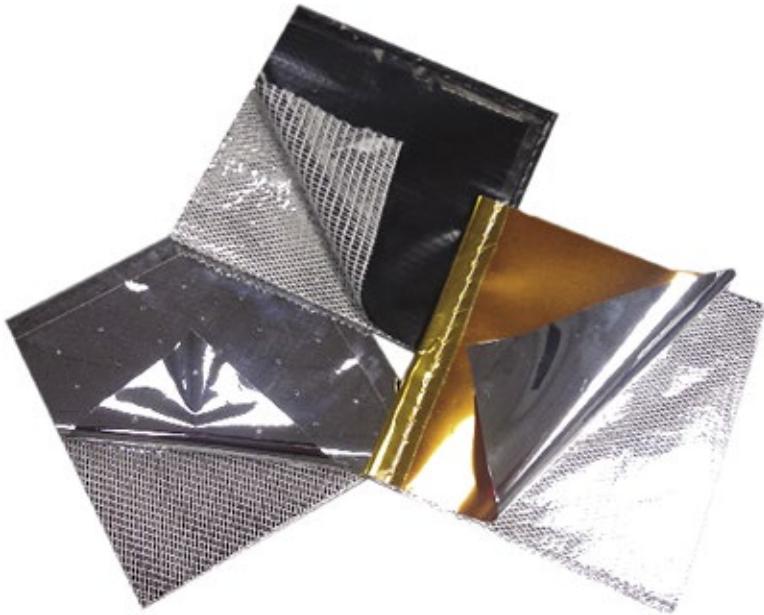
A typical propellant depot will contain liquid oxygen and liquid methane, which respectively are maintained about 90K and 111K. To keep the propellant cold and reduce wastage through boil off, the propellant depot can use four methods [15, 16]:

- Multi-layered super-insulation around the tank to reduce heat conduction.
- Structural isolation of the tank, like a thermos bottle, to reduce points where heat can be conducted through the tank. While in space, the tank can use vacuum as “free-insulation”.
- A multi-layered sunshield to reduce radiated heat from the Sun.
- An active cryocooler powered by solar panels to keep the propellants cold and recondense boil-off to prevent loss of propellant.



**Figure 1: Cryogenic propellant depot concept art, ULA [15]**

Looking into the various passive cooling systems, the most used is the multi-layered super-insulation (MLIs). Essentially, MLIs are 40-60 layers of thin film that prevent heat loss from the surface it is wrapped around. For this mission, we decided to use MLI consisting of layers made from double aluminized mylar and polyethylene terephthalate films from Meyer Tool & MFG. Thickness and mass values are discussed in the main report. The MLIs cost  $\$2.15/m^2$ . The total surface area of tanks required for LEO and LLO propellant depot is  $1.07E+05m^2$ . This brings the total cost for passive insulation to \$229,422.



**Figure 2: Multi-layered super insulation [17]**

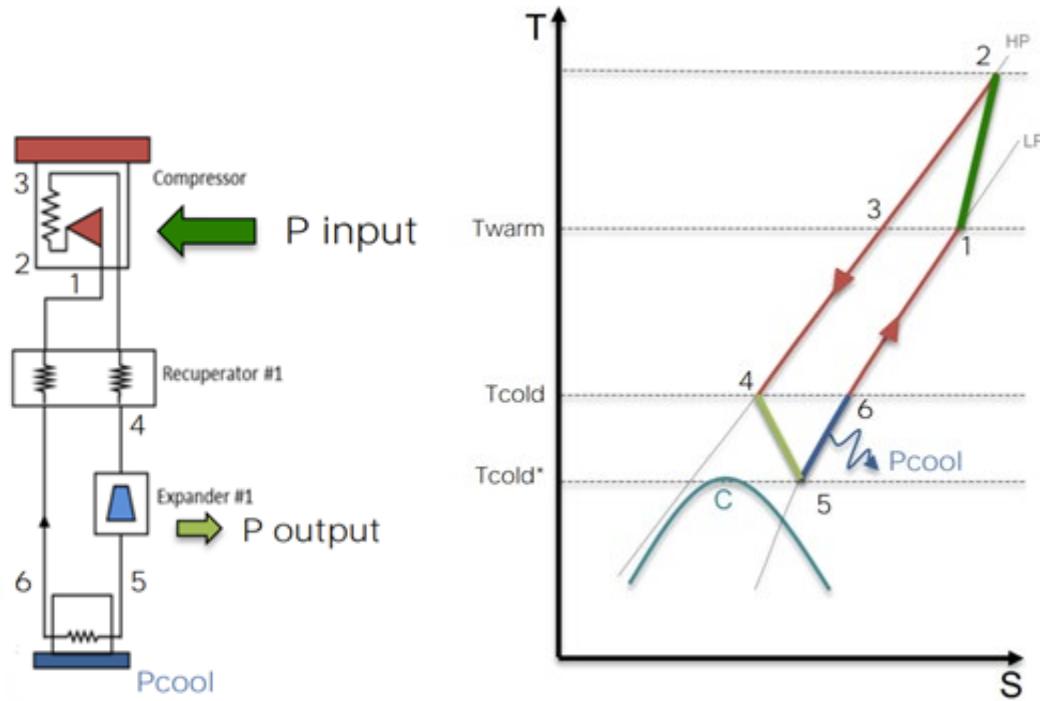
Taking into consideration the mass and size restrictions related to the launch and construction of the depot, we decided to only use one type of passive cooling system. Adding a multi-layered sunshield and components for structural insulation would improve the heat rejection capabilities of the propellant depot but the extra mass and cost to the mission outweighs the benefits.

To further reduce the boil-off and keep the propellants cold, we are using an active cooling system to accompany the MLIs. These active cooling systems have been researched by NASA since 2007. They investigate reducing power/mass while optimizing the cooling effect. The goal is to develop high efficiency Cryogenic Boil-Off Reduction Systems (CBRS) and Zero Boil-Off (ZBO).

There are several methods that are employed to create an environment suitable for maintain cryogenic temperatures, namely:

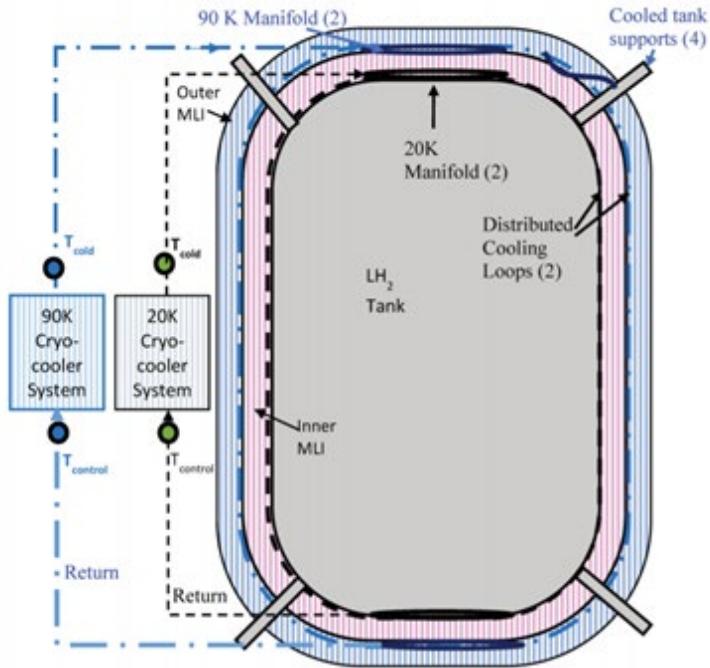
- Joule-Thomson Recuperative Cycle (low thermodynamic efficiency)
- Brayton Cycle (low vibration cycle)
- Stirling Regenerative Cycle (high complexity cycle)
- Pulse Tube Cycle (high complexity cycle)
- Gifford-McMahon Cycle (high complexity cycle)

From the various cycles mentioned above, the Brayton cycle provides high efficiency, compact sizing and reliability [18]. A turbo Brayton cycle cooler uses turbomachinery for its turbines and compressors. The regular compressor is replaced with a direct drive centrifugal compressor and high-speed synchronous motors on active magnetic bearings. An expander is added to the cycle to recover power and reduce power consumption by the motor. The working fluid that is recommended for a reverse Brayton cycle is a monoatomic gas with low specific heat ratio. Some good working fluids are Helium, Neon or Helium/Neon mixture.



**Figure 3: Reverse Brayton Cycle [18]**

Some other benefits that lead us to choose a reverse Brayton cycle system are the almost vibration free operation below 1kHz and long lifetime without repair or replacement. Due to the vibration free operation, the propellant depot does not require vibration canceling electronic systems or damping structures. The CBRS and ZBO systems that were tested by NASA Ames Research lab use a reverse turbo-Brayton cycle with a two-stage cooling system [19]. This active cooling system has an operating temperature of 90K, which is low enough to maintain both LOX and liquid methane. The schematic below shows the two-stage cooling circuit to achieve zero boil-off and optimized power consumption.

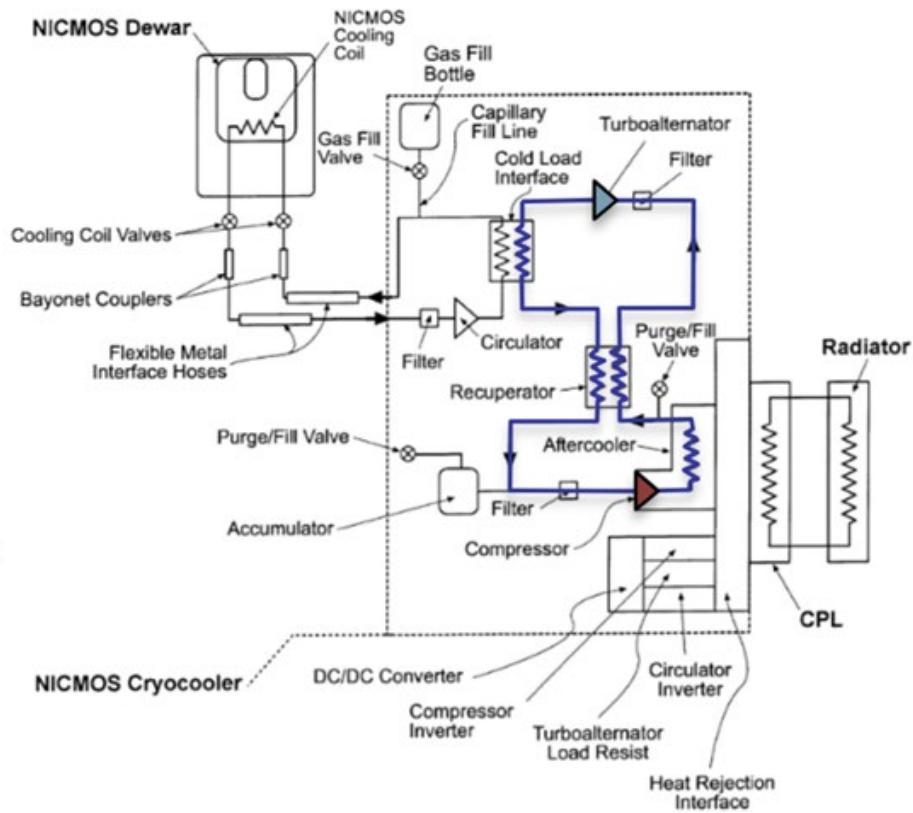


**Figure 4: Two stage cooling system at 20K and 90K [19]**

We decided to choose the Creare High Efficiency Cryocooler as the active cooling system for the propellant tanks. This Creare system uses the same basic principles of the NASA Ames experimental two stage cryocooler. Creare cryocooler operates at 150W of refrigeration at 90K. The specific power is 8W/W of refrigeration and the system has a specific mass of 0.4 kg/W of refrigeration. The total specifications of the Creare cryocooler are seen below:

**Table 4: System specifications for Creare Cryocooler [20]**

| Mass(kg)                                         | Power (kW) | Cost                |
|--------------------------------------------------|------------|---------------------|
| <b>LEO (2 LOX tanks 2 LCH<sub>4</sub> tanks)</b> |            |                     |
| 4.52E+05                                         | 4.8        | \$ 3,000,000.00     |
| <b>LLO (1 LOX tank 1 LCH<sub>4</sub> tank)</b>   |            |                     |
| 1.96E+05                                         | 2.4        | \$ 1,500,000.00     |
| <b>Total</b>                                     | 6.48E+05   | 7.2 \$ 4,500,000.00 |

**Figure 5: Creare NICMOS Cryocooler [18]**

The above schematic shows the layout for the Creare cryocooler that will be used on the propellant depots. We plan to add one active cooling system per tank i.e., one cooler for

LOX tank and one cooler for the liquid methane tank. These active cooling systems will be powered by solar panels as discussed in the main report section.

In summary, the propellant depot will be using a combination of MLIs and the Creare Cryocooler to maintain the cryogenic propellants at the required temperature conditions. In total, there are  $1.07E+5 m^2$  of MLIs wrapped around the propellant tanks at  $0.08 kg/m^2$ . The active cooling systems for the 6 tanks have a combined mass of  $6.48E+5$  kg. The table below summarises all the main costs associated with the propellant depots:

**Table 5: Total Costs for thermal systems**

| Item                     | Cost/unit (\$) | Number of units | Total cost              |
|--------------------------|----------------|-----------------|-------------------------|
| <b>Thermal Systems</b>   |                |                 |                         |
| <b>Creare Cryocooler</b> | 750,000        | 6               | \$ 4,500,000.00         |
| <b>MLI (per sq m)</b>    | 2.15           | $1.07E+05$      | \$ 229,422.20           |
| <b>Solar Panels</b>      | -              | 4               | \$ 68,061,000.00        |
| <b>Total</b>             |                |                 | <b>\$ 72,790,422.20</b> |

## V. Boiloff

To determine the rate at which propellant is lost, we quantify the amount of heat entering the tank. According to NASA [21], an insulated tank can expect an absolute maximum heat flux of  $0.4\text{W/m}^2$  from radiative sources and  $0.2\text{W/m}^2$  from conductive sources while in space near Earth.

For the low lunar orbit (LLO) and low Earth orbit (LEO) depots, the tanks are cylindrical with hemispherical end caps. Each depot has one tank for oxygen and one tank for methane. The surface area is then multiplied by the heat flux total of  $0.6\text{W/m}^2$ .

| Tank Heat Fluxes |                               |                   |
|------------------|-------------------------------|-------------------|
| Tank             | Surface Area ( $\text{m}^2$ ) | Heat Flux (Watts) |
| LEO Oxygen       | 371.20                        | 222.72            |
| LEO Methane      | 262.83                        | 157.70            |
| LLO Oxygen       | 279.27                        | 167.56            |
| LEO Methane      | 202.46                        | 121.48            |

Table 6: As the LEO depot tanks are much larger than the LLO tanks, they are subjected to higher heat fluxes.

The propellant loaded into the depot is cooled below its boiling point to prevent instant boiling inside of the tank. This provides a saturation time before the propellants begin to vaporize as the propellant warms to its boiling point first. According to NASA [21], without any mixing of the propellant or any disturbances to the tank, the heat transferred into the propellant

only propagates a distance set by the thermal conductivity of the liquid. This distance is the thermodiffusive distance and points radially inward towards the tank center axis. Before all of the fluid heats in the tank heats to the boiling point, the fluid in the thermodiffusive distance reaches the boiling point first. The time span required for this to occur is set by equation 8.

$$t_{diffusive} = \frac{c_p \rho K (T_{boil} - T_{subcool})^2}{Q^2} \quad (8)$$

We note that the saturation time depends on the level of subcooling. Thus, we subcool the liquid to postpone the start of boiloff. The depot takes multiple trips to fully load, and each time it is loaded, the incoming propellant is cooler than the fluid in the tank as the tank has time to warm between loadings. We use the subcooled fluid being loaded to keep the bulk temperature and the temperature in the thermodiffusive zone below the boiling point. We load the depot one day before the thermodiffusive saturation time with propellant subcooled to a greater degree of 4 K to maximize the time extension before boiling happens. The new bulk temperature is determined with equation 9. The subscripts denote the properties of the fluid currently stored and of the fluid being added from the refuelling craft.

$$T = \frac{m_{stored}T_{stored} + m_{added}T_{added}}{m_{stored} + m_{added}} \quad (9)$$

From which a new thermodiffusive saturation time can be computed. Repeating this over the 6 refuels that are required to fill the LEO depot, we find that the thermodiffusive saturation time doesn't change by a significant amount, meaning refuels can be scheduled at regular intervals. Oxygen is the limiting factor, being the first propellant to reach saturation temperature

in the thermodiffusive region, so we refuel the LEO depot every 5.28 days so that propellant can be stored for 31.68 days (from first propellant transfer into depot tanks) before boiloff begins.

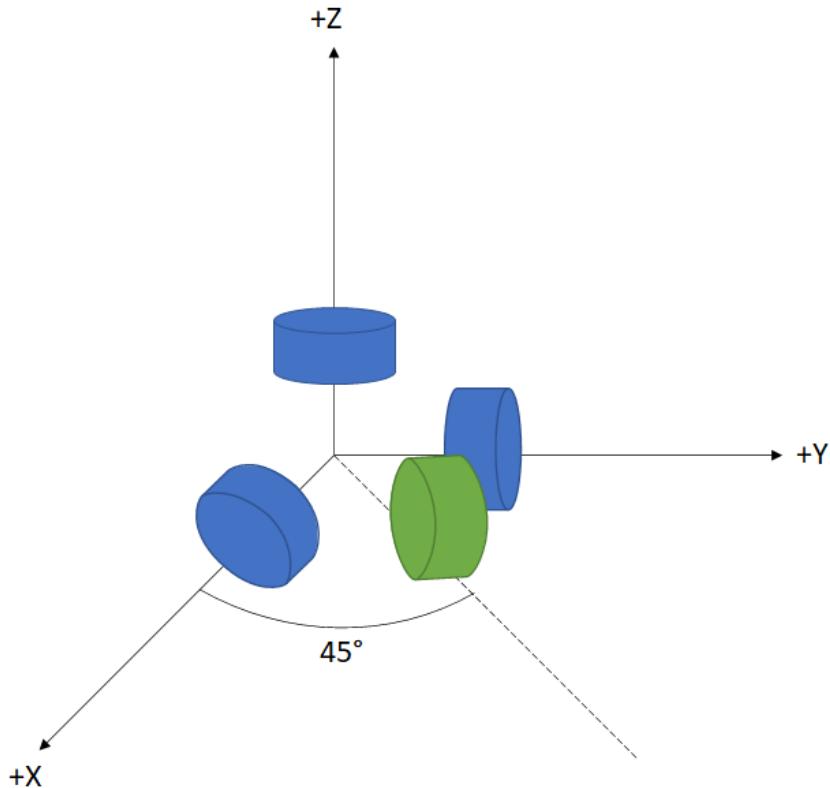
## VI. Attitude Dynamics/Control

An attitude control system is an integral piece of the propellant depot system. The propellant depot whether in orbit around Earth or the Moon will be experiencing disturbance torques causing the depot to spin. Without a system to counteract these disturbances and at least decrease the rotation of the propellant depot, there is no way a spacecraft would be able to rendezvous to receive propellant.

A system comprised of a reaction wheel assembly and thrusters much like the systems used on the FISTO and CHELL satellites was investigated, but to no avail. The requirements of the reaction wheel assemblies is far beyond the capabilities of any RWAs in production today and would thus not be feasible for this project.

A detailed analysis is explained below:

A four-wheel reaction wheel assembly (RWA) was investigated as the actuation method for the attitude determination and control system for both the LEO and LLO propellant depots, due to reaction wheels' high pointing accuracy, relatively low price point, and contractor convenience. Three wheels will be placed along the individual satellite's three body axes and the fourth will be placed  $45^\circ$  between the x and y body axes to create redundancy, as pictured below.



**Fig. 6 Redundant reaction wheel assembly configuration for propellant depots.**

The reaction wheel assembly for each of the propellant depots has a required torque, momentum storage, and power necessary for successful attitude control. These requirements are determined from the external and environmental disturbances imposed on the satellite which is highly dependent on the orbiting body, the desired orbit, and satellite configuration, and thus are different for the LEO and LLO propellant depots.

The environmental disturbances considered in this analysis were different for the two propellant depots. The LEO propellant depot needs to take into account torques due to gravity gradient, solar radiation pressure, and Earth's magnetic field (this analysis assumes the altitude of the orbit is high enough to neglect torques due to Earth's atmosphere). While the LLO

propellant depot can be simplified due to orbit placement around the Moon (disturbances due to magnetic fields and atmosphere can be neglected) and thus the only torques included are gravity gradient torques and solar radiation pressure torque. A detailed analysis can be seen in the Propellant Depot Code Appendix code E for the LEO propellant depot and in Appendix VI.B.1 for the LLO propellant depot. The maximum torques are included in Table 7:

|                      | Maximum Gravity Gradient Torque Due to Earth | Maximum Gravity Gradient Torque Due to Moon | Solar Radiation Pressure Torque | Magnetic Field Torque |
|----------------------|----------------------------------------------|---------------------------------------------|---------------------------------|-----------------------|
| LEO propellant depot | $1.6945 \times 10^3 \text{ Nm}$              | $7.9633 \times 10^{-8} \text{ Nm}$          | $0.0187 \text{ Nm}$             | $0.0469 \text{ Nm}$   |
| LLO propellant depot | $2.3680 \times 10^{-6} \text{ Nm}$           | $1.6531 \times 10^3 \text{ Nm}$             | $0.0150 \text{ Nm}$             | $0 \text{ Nm}$        |

**Table 7 Maximum environmental torques on both the LEO and LLO propellant depots.**

To combine and convert the torques in Table 7 to the required torque produced by the RWA for each propellant depot, only the largest gravity gradient torque is considered (the gravity gradient torque due to the Earth for the LEO propellant depot and the gravity gradient due to the Moon for the LLO propellant depot) and summed with the solar radiation pressure torque and magnetic field torque if there is one. This combined torque is then scaled by 1.25 to account for any other disturbances on the propellant depot and additional torque needed for orientation maneuvers and can now be considered the required torque of the reaction wheel assembly.

The required momentum storage and power of the reaction wheels can be determined from the required torque and orbital characteristics and can be seen in the Propellant Depot Code

Appendix as Code E and Appendix VI.B.1. The determined required torque, momentum storage, and power for each reaction wheel assembly are listed in Table 8:

|                      | Required Torque | Required Momentum Storage | Required Power |
|----------------------|-----------------|---------------------------|----------------|
| LEO propellant depot | 2118.20 Nm      | 54760.0 Nm/s              | 2118.96 kW     |
| LLO propellant depot | 2066.44 Nm      | 32767.8 Nm/s              | 2067.04 kW     |

**Table 8 Requirements torque, momentum storage, and power for the LEO and LLO propellant depots.**

Reaction wheel assemblies meeting these specifications could potentially be commissioned from a global leader in RWA design such as Honeywell or Collins Aerospace. Realistically, these RWAs would not be something that should be considered for this project. Current RWAs in production and practice do not have torques greater than about 10 Newton-meters. To believe we could have a company manufacture a RWA twenty times more powerful than any ever made is not a good assumption to make even in the scope of this project. It is also important to note the immense power draw of this system on both the LEO and LLO propellant depots. This large required power would cause an increase in the size of the solar panels resulting in a larger propellant depot and updates to the moments of inertia and surface area in the calculations likely causing the environmental torques to be greater and thus make the requirements of this system even more out of reach.

From this analysis it may be important to consider decreasing the size of the propellant depots or increasing the orbit altitudes in order to have a system that can be controlled for necessary spacecraft rendezvous.

## VII. Attitude Control System (Lorin)

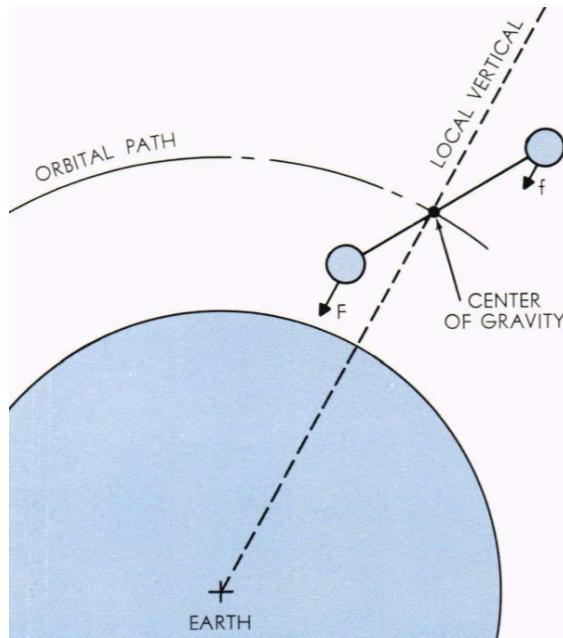
In addition to the attitude control aspect provided by the reaction wheel assemblies, a separate system is required for each of the propellant depots to periodically desaturate the momentum wheels. For the propellant depot in low lunar orbit, we assume this can be accomplished with hydrazine thrusters, as is used for the lunar satellites. No further analysis was done on this option due to time limitations toward the end of the semester. However, we consider a few options for the depot in low-Earth orbit.

For this propellant depot, we have the benefit of the Earth's magnetic field. Many small spacecraft use the Earth's magnetic field for attitude corrections or momentum dumps via magnetic torquers or 'magnetotorquers'. These devices consist of a conductive coil which generates a magnetic field when powered. This magnetic field then interacts with the Earth's magnetic field, which introduces an external torque on the spacecraft. This torque can be used to unload the reaction wheel assemblies. There are certain advantages to using magnetotorquers over other attitude control systems. Magnetotorquers are sustainable as long as enough power can be generated to run them. Where thrusters only have a limited propellant supply, magnetotorquers can theoretically work indefinitely. They are also generally very reliable due to the lack of moving parts [22]. For these reasons, magnetotorquers are often a good choice for small spacecraft. However, preliminary calculations showed that for a vehicle the size of the proposed propellant depots, this system is not at all feasible. The magnetotorquers would have to be unrealistically large, and use very high current and thus power. The mass and power requirements for a system of this scale are not worth the savings in propellant.

Another possibility for stabilization that would eliminate the need for reaction wheel assemblies is gravity gradient stabilization. This is a passive form of spacecraft stabilization that uses the gradient of gravity about a massive body like the Earth to maintain a relatively fixed attitude. From Newton's Universal Law of Gravitation, we know that the attractive force between two bodies is given by the equation below:

$$|F| = \frac{Gm_1m_2}{r^2} \quad (10)$$

In this equation, F is the force (N), G is the Universal Gravitational Constant ( $N \cdot m^2/kg^2$ ),  $m_1$  and  $m_2$  are masses (kg), and r is the magnitude of the radius between the two masses (m). We see that the force is inversely proportional to the square of the radius. In other words, a mass at a farther distance from a planet experiences a smaller gravitational force than the same mass would at a closer distance to the planet. Gravity gradient stabilization uses this fact by introducing a second mass attached via a long rod or tether that is closer to the Earth than the main satellite. Thus, any small disturbance from a vertical orientation will cause a stabilizing torque that causes the system to correct itself.



**Figure 7: A dumbbell satellite experiences a stabilizing moment due to the gravity gradient when its attitude is perturbed [23].**

In figure 7, we see that the lower mass of the dumbbell satellite experiences a larger gravitational force than the upper mass due to its closer proximity to the Earth. This larger force introduces a moment in the counterclockwise direction which is larger than the moment from the upper mass in the clockwise direction. This net moment causes the spacecraft to move back toward the local vertical orientation.

Since we want the propellant depot to remain in an orientation which is fixed relative to the orbit frame, this passive form of stability is worth consideration. A Matlab script was started to perform calculations regarding the feasibility of this passive system. This code may be a useful starting point for future projects, but was not finished due to the lack of time between when the problem was identified and when the final report was due. Once again, due to the very high mass and inertia properties of the depots, it is likely that the mass required for this system to work would

have been unreasonably large, or that the distance between the two masses might also be absurdly large.

For a spacecraft of this size and mass, we assume that the final system for attitude control is a series of very large control moment gyroscopes like those used on the International Space Station, paired with a series of chemical thrusters for momentum desaturation.

### VIII. Costs

**Table 9: Total Costs of the System [24, 25]**

| Item                                           | Cost/unit (\$) | Number of units | Total cost           |
|------------------------------------------------|----------------|-----------------|----------------------|
| <b>Thermal Systems</b>                         |                |                 |                      |
| <b>Creare Cryocooler</b>                       | 750,000        | 6               | \$4,500,000          |
| <b>MLI (per sq m)</b>                          | 2.15           | 1.07E+05        | \$229,422            |
| <b>Solar Panels</b>                            | -              | 4               | \$68,061,000         |
| <b>Propellant LLO</b>                          |                |                 |                      |
| <b>LOX (per kg)</b>                            | 20.7           | 8.81E+05        | \$18,234,423         |
| <b>Liquid Methane (per kg)</b>                 | 1.35           | 2.48E+05        | \$334,989            |
| <b>Propellant LLO</b>                          |                |                 |                      |
| <b>LOX (per kg)</b>                            | 20.7           | 3.83E+05        | \$7,930,998          |
| <b>Liquid Methane (per kg)</b>                 | 1.35           | 1.08E+05        | \$145,706            |
| <b>Station keeping (for both LEO and LLO):</b> |                |                 |                      |
| <b>Hydrazine (per kg)</b>                      | 38.55          | 131907.672      | \$5,085,041          |
| <b>Hydrazine thruster</b>                      | 50,000         | 7               | \$350,000            |
| <b>Propellant Tank</b>                         |                |                 |                      |
| <b>Ti-6Al-4V (per kg) (for tanks)</b>          | 45             | 7023.11         | \$316,040            |
| <b>Total</b>                                   |                |                 | <b>\$105,187,618</b> |

### IX. Lunar Mining Background and Need

Similar to how exploration of our planet required mankind to adapt and learn to use local resources depending on the continent, region, and climate, so too will we learn to find, extract, and use local resources to continue our expansion and exploration of space. This is driven by the strong contrast between the relatively small amount of materials that can be launched from Earth, and the enormous volume and diversity of resources available in space. The commercialization of space resources represents the future of resource extraction industries. As capabilities in space continue

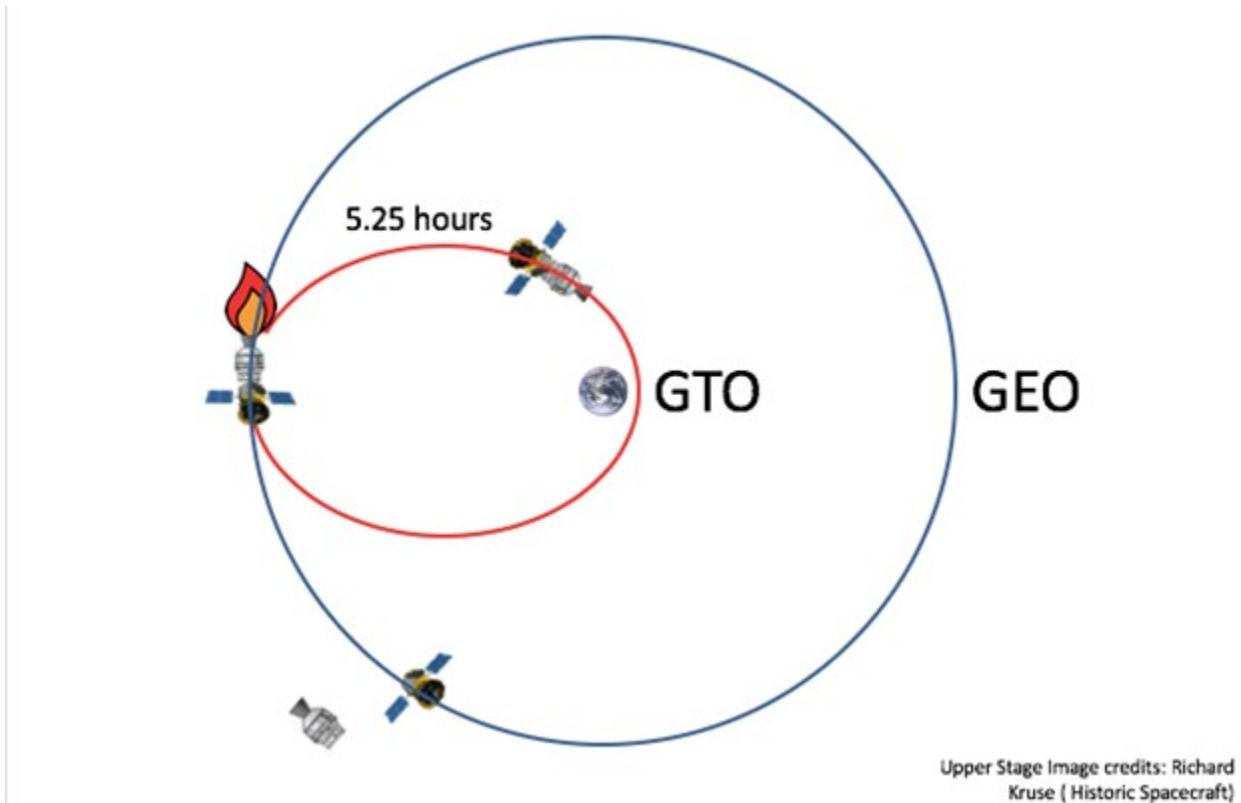
to grow with the advancement of technology, strategic planning and prioritization of resource exploration, it is imperative to guide policy and commercial development of space-based natural resources. In-Situ Resource Utilization (ISRU) represents the near future of the space industry, enabling the efficient use of resources both on Earth and in space, as well as continued expansion and development of human presence outside of our planet. Technologies developed and refined for ISRU will continue to deliver additional benefits to Earth-bound industries, as demonstrated by the ubiquity of modern technologies first developed for space exploration programs. Thus, ISRU is an important area for investment and rapid development in the near future. The most pressing need for resources in space is that of fuel; transporting cargo and humans in space requires a vast amount of propellant and launching the full mass of propellant needed for long-term space missions from the Earth's surface places severe limitations on missions of all kinds. Thus, developing an architecture for prospecting, mining, processing, storing, and transporting fuel products in space is the first critical step to creating a sustainable space development strategy. Between the abundance of resources available, relative proximity to the Earth, and decades of scientific study, the Moon presents an ideal objective for early-stage ISRU activities, providing a testing ground for the development of new methods and technologies as well as a platform for continued expansion to other planets and Near-Earth Objects (NEOs).

### *Possible Fuel Sources*

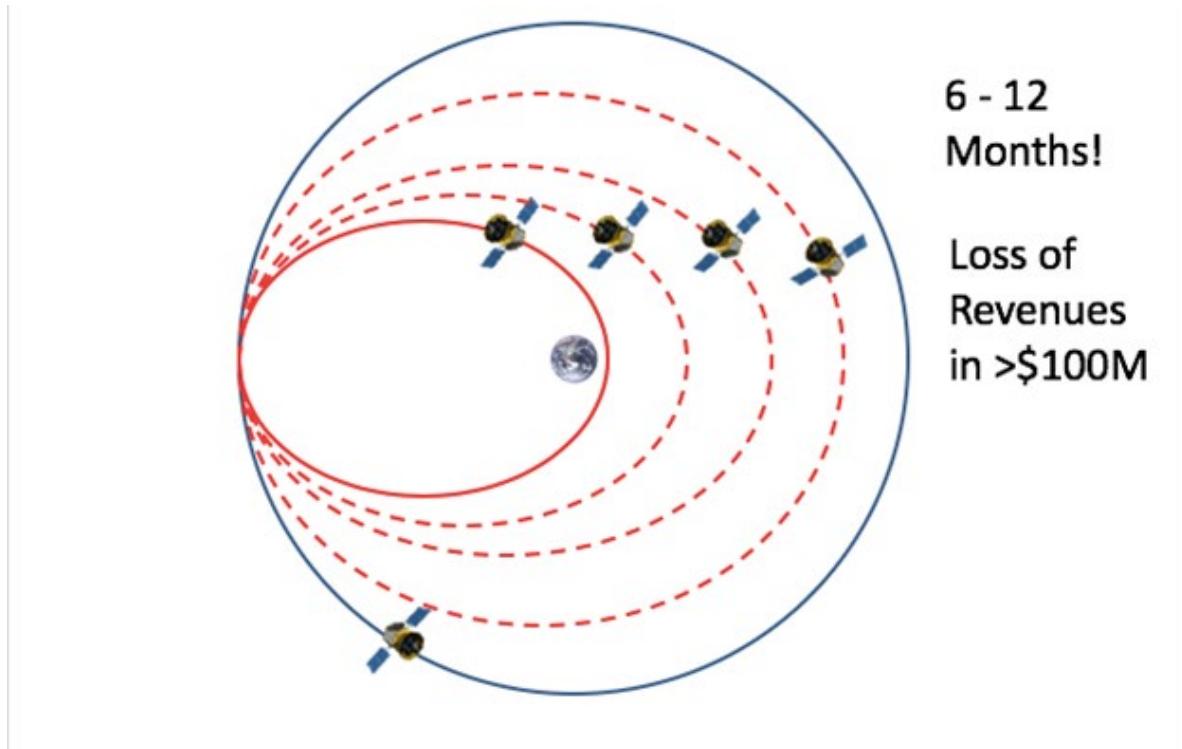
#### *1. Water*

Why? If you split water into hydrogen and oxygen, and then liquefy those constituents, you have rocket fuel. If you can stop at the moon's orbit or a lunar base to refuel, you no longer

need to bring all your propellant with you as you take off, making your spacecraft significantly lighter and cheaper to launch. That's important because Earth's atmosphere and gravitational pull necessitate use of tons of fuel per second when rockets launch. Creating a sustainable source of fuel in space could reduce the costs and hazards associated with heavy liftoffs. One NASA [26] estimate suggests there might be 600 million metric tons of lunar ice to harvest, and other higher-end estimates say one billion metric tons is a possibility [27]. The lunar water would be launched off the Moon and delivered to the propellant depot in Lower Earth Orbit (LEO). [28] This propellant depot will use solar energy to turn the water into rocket fuel. Then, space tugs can refill their tanks so they can repeatedly boost spacecraft from Geosynchronous Transfer Orbit (GTO) (where the launch rocket throws them) into Geosynchronous Orbit (GEO) where they can begin operating. This can save money when putting new telecommunications satellites (or other satellites) into their final orbits. In the old days, every time a satellite was launched, the rocket had to include an upper stage that would boost the satellite from GTO to GEO. The upper stage was used only one time and then thrown away to become space junk. It was very expensive to carry a heavy upper stage, with all of its fuel, and throw it away every time, but there's a better way. Nowadays, instead of including an upper stage with every satellite, we use a very lightweight electric thruster instead. These thrusters are highly efficient and do not cost much to launch with the satellite, but they are very slow. Instead of taking just one day to boost the satellite into its final orbit, an electric thruster takes 6 to 12 months. During that time, the owner of the satellite loses something like \$100M in revenues, all the while they are still paying insurance, finance costs, and operational costs for the satellite. So that \$100M is a real loss, but it is still cheaper than launching and throwing away an upper stage every time.



[28] Fig 8. Original Method to Boost Comsats



[28] Fig 9. Improved Method to Boost Comsats

### Challenges with mining water

There are plenty of obstacles. The cold temperatures and radiation could endanger humans and degrade sensitive equipment. It is not ideal to have a large crew of human beings running these kinds of operations day in and day out, but it's equally unclear how much can be delegated to autonomous systems. Lunar soil itself—coarse and jagged, and prone to sticking to everything—could wreck machinery and pose safety issues to workers in spacesuits. Although we have shown the feasibility of refueling satellites in orbit, doing the same thing for large spacecraft on the moon or in lunar orbit will create its own set of challenges thanks to microgravity and regolith, the layer of loose material covering the lunar bedrock. Even assuming these obstacles are

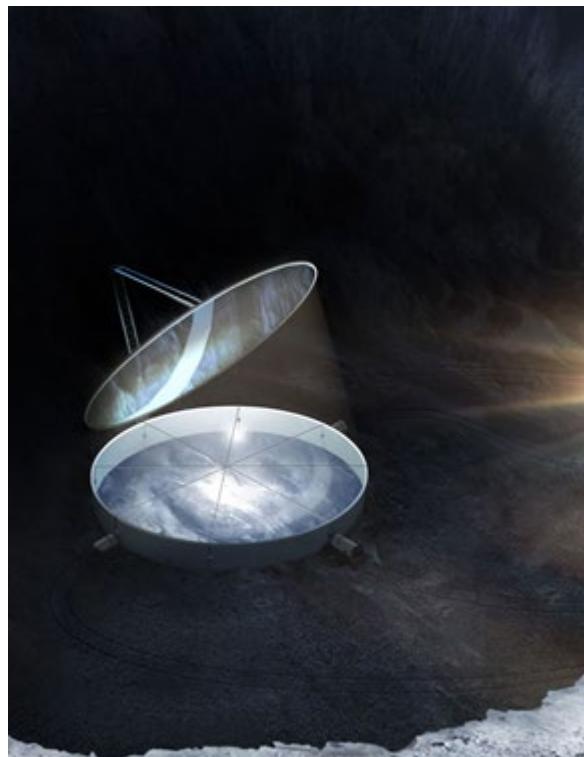
surmountable, how easy would it be to extract water once we were there? First, lunar water is not that easy to access. Water on the moon is in the form of tiny ice grains mixed into the soil, mostly located in the permanently shadowed regions within craters near the poles. [29] Here, temperatures of 40 K (-233.15 °C) keep the water ice stable and undisturbed. The grains are heavily mixed with complex organics and metals. In 2009, NASA's LCROSS mission shot a rocket into the moon to fling plumes of moon rock into the air [30]. An analysis of this airborne material found it was only 5.6% water by weight. [31] That data, which at 10 years old is still the most recent direct analysis of lunar soil water content we have, suggests even if water ice can be separated from the lunar soil, it's still very impure and would require aggressive purification to rid it of contaminants that would ruin any fuel made from it.

| Compound                      | Concentration Percentage relative to H <sub>2</sub> O |
|-------------------------------|-------------------------------------------------------|
| H <sub>2</sub> O              | 100 %                                                 |
| H <sub>2</sub> S              | 16.75 %                                               |
| NH <sub>3</sub>               | 6.03 %                                                |
| SO <sub>2</sub>               | 3.19 %                                                |
| C <sub>2</sub> H <sub>4</sub> | 3.12 %                                                |
| CO <sub>2</sub>               | 2.17 %                                                |
| CH <sub>3</sub> OH            | 1.55 %                                                |
| CH <sub>4</sub>               | 0.65 %                                                |
| OH                            | 0.03 %                                                |

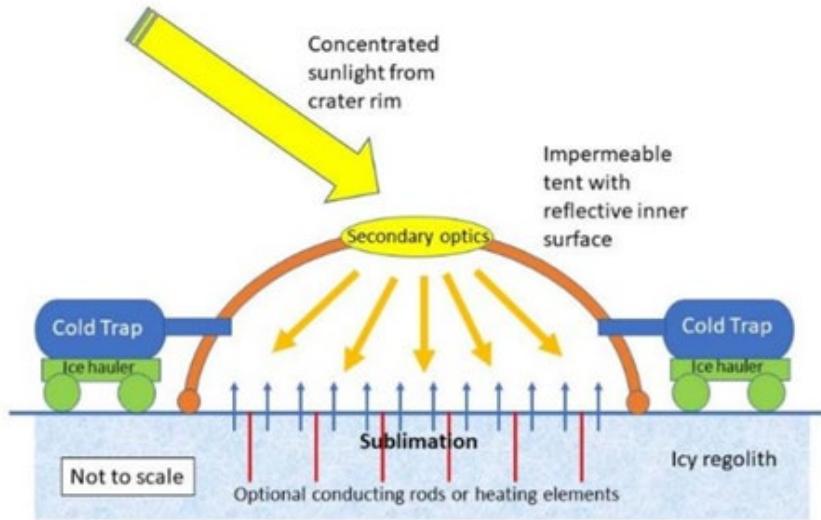
**Table 10. Lunar Water Contaminant Concentration**

Aside from the contaminant challenge, exploration within the Permanently Shadowed Regions (PSR) must be undertaken with all the technical challenges that it would bring. one proposed method for processing lunar water ice. Large towers with concave mirrors on the top would be erected and installed around the crater edges to reflect sunlight down into

permanently shadowed regions. This energy would heat the lunar soil up to 220 K (-53.15 °C), warm enough to get the water ice to sublimate into vapor. A tent cover over the soil (transparent, so the redirected sunlight could hit the surface) would trap and capture this water vapor, which would be moved into large aluminum units where it would freeze back into ice. Haulers (maybe robotic, or maybe driven by astronauts) would drive the ice out to a facility where it could be purified.



**Fig 10. Courtesy of George Sowers. Mirrors use sunlight to heat water ice in lunar soil.**



**Fig 11. Thermal Mining Concept [32]**

### Thermal Mining Sizing

A thermal mining sizing solution for water extraction of 2,450 MT per year (producing 1,640 MT of propellant per year) is shown in Table 11. Thermal mining can be scaled to meet any water extraction goal, larger or smaller by scaling the tent or adding more tents, cold traps, and other thermal mining system elements. In this scenario, the dwell time, move time, and downtime, were fixed for both scenarios. Extractable ice per surface area is estimated to be 25 kg/m<sup>2</sup> and 250 kg/m<sup>2</sup> for 4wt% and 30wt% regions respectively. The solution includes tents of 32 m and 10 m diameter again, for 4wt% and 30wt% regions respectively. A single tent would need to be placed 128 times per year. Additional tents would provide additional margin. The number of ice haulers depends on the density of ice in the cold trap, volume of cold trap, ice-

hauler traverse speed, distance between ice field and processing station, and transfer time at the processing station.

| Requirement                               | Value (4%wt) | Value (30%wt) |
|-------------------------------------------|--------------|---------------|
| Area mined ( $\text{m}^2 / \text{year}$ ) | 100,000      | 10,000        |
| Yield per $\text{m}^2$ (kg)               | 25           | 250           |
| Ice sublimated per $\text{m}^2$ (kg)      | 27           | 280           |
| Dwell time (hour)                         | 48           |               |
| Move time (hour)                          | 12           |               |
| Number of moves per year                  | 128          |               |
| Power (kW)                                | 580          | 370           |
| Tent diameter ( $\text{m}^2$ )            | 32           | 10            |
| Tent plan area ( $\text{m}^2$ )           | 780          | 76            |
| Exit area to cold traps ( $\text{m}^2$ )  | 2            |               |
| Leak area ( $\text{m}^2$ )                | 0.2          |               |

Table 11. Thermal Mining Design for 4wt% and 30wt% Regions

## Processing

After water is extracted from the Permanently Shadowed Regions (PSR) of the Moon, it is processed to purify and electrolyze the water into hydrogen and oxygen. Paragon Space Development Corporation (Paragon) [33] and its partner Giner, Inc. (Giner) are developing the ISRU-derived water purification and Hydrogen Oxygen Production (IHOP) subsystem through a recently awarded NASA Next Space Technologies for Exploration Partnerships. Paragon's Ionomer-membrane Water Processing (IWP) technology is optimized to perform primary water purification for this ISRU application. The purified water receives final polishing and is then electrolyzed using a Giner static feed water electrolyzer to produce H<sub>2</sub> and O<sub>2</sub> propellant.

### 1. *Methane*

Methane (CH<sub>4</sub>) has also been proposed as a fuel for lunar use. Carbon is present in the lunar regolith in concentrations several times that of hydrogen, and heating the regolith to extract volatiles would result in some methane being produced, along with carbon monoxide and dioxide (which could be converted to methane by reacting with hydrogen). Burning methane with oxygen would give a specific impulse of around 300 seconds, requiring more fuel than a hydrogen-oxygen rocket. However, methane is only about 25% hydrogen by weight, and using methane as fuel results in about a 50% reduction in the amount of hydrogen needed for a given launch mass. Methane would be plentiful once volatiles are extracted from lunar regolith. Indeed, it might be more plentiful than oxygen, in that case it might be very efficient to use CH<sub>4</sub> as reaction mass for solar thermal rockets. It would be a somewhat lower specific impulse than hydrogen, but much easier to store. Below is a table showing the major constituent volatiles in lunar regolith.[34]

| <b>Volatile</b> | <b>Concentration (ppm)</b> |
|-----------------|----------------------------|
| Hydrogen        | 0.1-211                    |
| Helium          | 1-63                       |
| Nitrogen        | 1-153                      |
| Carbon          | 10-280                     |

|        |         |
|--------|---------|
| Sulfur | 20-1330 |
|--------|---------|

Table 12

The exact process of producing methane is using a Sabatier reactor. The Sabatier reactor reacts CO<sub>2</sub> and H<sub>2</sub> to produce CH<sub>4</sub>. Their total contribution to the mass and power requirements of the plant is basically irrelevant compared to the total plant mass. In a real system, conditioning of the gases may require a small amount of additional hardware. The performance or economic value of these differences can be viewed considering the amount of propellant produced as a function of the system mass. Whereas the hydrogen-oxygen plant can reproduce its own mass in over 5 months, the methane-oxygen plant can reproduce its own mass in about two and a half months, when considering a small plant of 1000 kg of oxygen production per month.[35] Figure 5. shows a schematic of the plant to produce Methane and Oxygen from Lunar regolith.

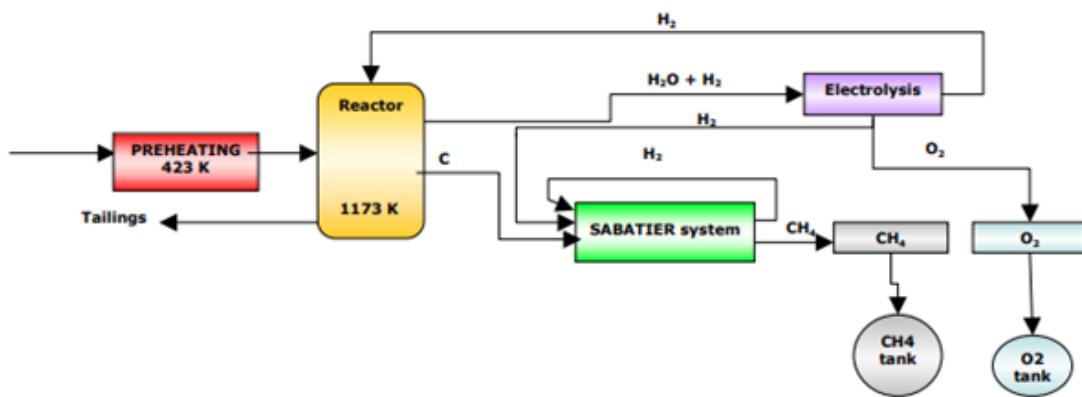


Fig 12. Credit to Dr. B.Ruiz. [35]

| Element                   | Mass (kg) |
|---------------------------|-----------|
| Excavation                | 13,737    |
| Hauler                    | 11,327    |
| Feed Bin System           | 1,200     |
| Reactor                   | 14,120    |
| Sabatier + Heat Exchanger | 170       |
| Electrolyzer              | 1,945     |
| CH4 Liquifier             | 327       |
| O2 Liquifier              | 1,111     |
| Power                     | 12,896    |
| O2 Storage                | 352       |
| CH4 Storage               | 276       |
| Total Mass                | 57460     |

**Table 13**

The cost estimation of the process is based on today's plants with a power consumption of up to 5 MW. The cost is estimated to be between 220 and 467 USD/kW. Given the expected scale effects due to the enormous project size, the most optimistic value of 220 USD/kW is used for further calculations and the resulting investment is \$4677 million. The costs for depreciation and interests are 157 million USD/a. Due to a lack of experience, especially over long periods of time, there is currently little usable data available. However, various plant builders estimate a range of 1–3% of investment costs. A safety margin of 5% is additionally used and sum up to a total of \$233 million, required for the operation of the plants.[36]

## X. Code

### A. Propellant Bookkeeping and Refueling Optimization:

```
% OPTIMIZED: this version will give you the minimum refuelling prop masses
% you need at each depot to complete the mission of going to the moon and
% back. It accounts for avoiding running out of fuel once
% getting to the depot. It does this by setting the leftover prop at the
% depot = 2% of whatever prop you needed to get there for that recent leg of
% the mission. You can change this percentage in lines 87, 144, 181
% inputs: mission dV's and payloads
% outputs: prop mass reqd for each leg of mission, prop mass reqd for refueling

% mission dV's, (km/s)
% crewed
dV_esurf_to_600km = 9.5;
dV_vous_depot = 10/1000;      % rendezvous with depot
dV_tli = 3.0468+0.1;
dV_frd = 0.09963+0.1;        % free return departure
dV_loi = 0.850233+0.1+dV_vous_depot;      % lunar orbit insertion
dV_100km_to_lsurf = 2.5;
dV_lsurf_to_100km = 2.2+dV_vous_depot;
dV_llo_to_reentry = 1.0388+0.1;
% mp_star_eland = mass of propellant needed to land starship after reentry

% % uncrewed (cargo)
% dV_esurf_to_600km = 9.5;
% dV_vous_depot = 10/1000;      % rendezvous with depot
% dV_tli = 3.0809+0.1;
% dV_frd = 0;        % free return departure
% dV_loi = 0.7928+0.1+dV_vous_depot;      % lunar orbit insertion
% dV_100km_to_lsurf = 2.5;
% dV_lsurf_to_100km = 2.2+dV_vous_depot;
% dV_llo_to_reentry = 1.0388+0.1;

% Starship values
mp_max = 1200*1000;      % max prop mass of starship, kg
m_pl1 = 0*1000;      % payload sending to moon (starship max), kg
m_pl1_tons = m_pl1/1000;      % payload sending to moon, tons (for trevors code to work)

% return payload
n_crew = 0;
m_crew = n_crew*63.5;      % mass crew in kg, 19 people with an average weight of 140 lbs
m_suits = n_crew*130;      % mass of spacesuits, kg
m_items = n_crew*1.5;      % mass of personal items (3.3 lbs for iss), kg
m_pl2 = m_crew + m_suits + m_items; % return payload, kg
m_pl2_tons = m_pl2/1000;
```

```
% 1. earth surface --> leo depot (~600 km)
% dV superheavy can contribute while still saving prop for landing (trevors code):
[dV_shb, dV_star_reuse, dV_star_expend, dVtotal_reuse, dVtotal_expend, mp_star_eland,
mp_star_eland_tot] = delta_v_payload(m_pl1_tons, m_pl2_tons);

if dVtotal_reuse < (dV_esurf_to_600km + dV_vous_depot)
    fprintf('\n\nsorry, you cant make it to LEO being reusable/reqd dV, ignore everything
after\n\n');
end

if dVtotal_expend < (dV_esurf_to_600km + dV_vous_depot)
    fprintf('\n\nsorry, you cant make it to LEO being expendable and with that payload/reqd dV,
ignore everything after\n\n');
end

[mp_used1, mp_leftover1] = esurf_to_depot(m_pl1, dV_esurf_to_600km, dV_shb,
dV_vous_depot);
if (mp_used1 > mp_max)
    fprintf('uh oh, you ran out of fuel on leg 1\n');
end

% 2. refuel once at leo depot
m_refuel1 = [1000:1:(mp_max - mp_leftover1)];
mp_leftover2 = mp_leftover1 + m_refuel1;
n = length(m_refuel1);
j = 1;

for i = 1:n

% 3. TLI
[mp_used3, mp_leftover3] = calc_prop_mass_func(dV_tli, m_pl1, mp_leftover2(i));
if (mp_used3 > mp_leftover2(i))
    continue

    fprintf('uh oh, you ran out of fuel on leg 3\n');
    fprintf('you needed %.3f kg more fuel\n', abs(mp_leftover3));
end

% 4. FRD
[mp_used4, mp_leftover4] = calc_prop_mass_func(dV_frd, m_pl1, mp_leftover3);
if (mp_used4 > mp_leftover3)
    continue

    fprintf('uh oh, you ran out of fuel on leg 4\n');
    fprintf('you needed %.3f kg more fuel\n', abs(mp_leftover4));
end

% 5. LOI
[mp_used5, mp_leftover5] = calc_prop_mass_func(dV_loi, m_pl1, mp_leftover4);
if (mp_used5 > mp_leftover4)
```

```

continue
fprintf('uh oh, you ran out of fuel on leg 5\n');
fprintf('you needed %.3f kg more fuel\n', abs(mp_leftover5));
end

if (mp_leftover5 >= 0.02*mp_used5)
    m_refuel1_new(j) = m_refuel1(i);
    mp_used3_new(j) = mp_used3; mp_leftover3_new(j) = mp_leftover3;
    mp_used4_new(j) = mp_used4; mp_leftover4_new(j) = mp_leftover4;
    mp_used5_new(j) = mp_used5; mp_leftover5_new(j) = mp_leftover5;
    j = j+1;

end
end
[val, ind] = min(m_refuel1_new);
m_refuel1 = val;
mp_used3 = mp_used3_new(ind); mp_leftover3 = mp_leftover3_new(ind);
mp_used4 = mp_used4_new(ind); mp_leftover4 = mp_leftover4_new(ind);
mp_used5 = mp_used5_new(ind); mp_leftover5 = mp_leftover5_new(ind);

% 6. refuel in LLO
m_refuel2 = [1000:1:(mp_max - mp_leftover5)];
mp_leftover6 = mp_leftover5 + m_refuel2;

n = length(m_refuel2);
j = 1;
for i = 1:n
% 7. lunar descent, lunar 100 km --> lunar surface
[mp_used7, mp_leftover7] = calc_prop_mass_func(dV_100km_to_lsurf, m_pl1,
mp_leftover6(i));
if (mp_used7 > mp_leftover6(i))
    continue
    fprintf('uh oh, you ran out of fuel on leg 7\n');
    fprintf('you needed %.3f kg more fuel\n', abs(mp_leftover7));
end

% 8. lunar surface --> lunar 100 km

[mp_used8, mp_leftover8] = calc_prop_mass_func(dV_lsurf_to_100km, m_pl2, mp_leftover7);
if (mp_used8 > mp_leftover7)
    continue
    fprintf('uh oh, you ran out of fuel on leg 8\n');
    fprintf('you needed %.3f kg more fuel\n', abs(mp_leftover8));
end

if (mp_leftover8 >= 0.02*mp_used8)
    m_refuel2_new(j) = m_refuel2(i);
    mp_used7_new(j) = mp_used7; mp_leftover7_new(j) = mp_leftover7;
end

```

```

mp_used8_new(j) = mp_used8; mp_leftover8_new(j) = mp_leftover8;
j = j+1;

end
end

[val, ind] = min(m_refuel2_new);
m_refuel2 = val;
mp_used7 = mp_used7_new(ind); mp_leftover7 = mp_leftover7_new(ind);
mp_used8 = mp_used8_new(ind); mp_leftover8 = mp_leftover8_new(ind);

%9. refuel in LLO
m_refuel3 = [1000:1:(mp_max - mp_leftover8)];
mp_leftover9 = mp_leftover8 + m_refuel3;
% mp_leftover6 = mp_max;

n = length(m_refuel3);
j = 1;

for i = 1:n
% 10. lunar 100 km --> earth reentry
[mp_used10, mp_leftover10] = calc_prop_mass_func(dV_ll0_to_reentry, m_pl2,
mp_leftover9(i));
if (mp_used10 > mp_leftover9(i))
    continue
    fprintf('uh oh, you ran out of fuel on leg 10\n');
    fprintf('you needed %.3f kg more fuel\n', abs(mp_leftover10));
end

% if (mp_leftover10 >= 0.02*mp_used10 && mp_leftover10 >= mp_star_eland)
if (mp_leftover10 >= 1.02*mp_star_eland)
    m_refuel3_new(j) = m_refuel3(i);
    mp_used10_new(j) = mp_used10; mp_leftover10_new(j) = mp_leftover10;
    j = j+1;

end
end

[val, ind] = min(m_refuel3_new);
m_refuel3 = val;
mp_used10 = mp_used10_new(ind); mp_leftover10 = mp_leftover10_new(ind);

mp_refuel_total = m_refuel1 + m_refuel2 + m_refuel3

function[mp_used, mp_leftover] = esurf_to_depot(mpl, mission_dV, SH_dV, dV_vous_depot)
ge = 9.81;           % earth gravity, m/s^2

% Starship values

```

```
% mpl = 100*1000;      % payload mass, kg
minert = 120000;      % inert mass, kg
mp_max = 1200*1000;   % max prop mass starship can hold, kg
isp_vac = 380;         % raptor vac isp, s

% dV starship needs to contribute to meet up w depot:
dV_star = mission_dV - SH_dV + dV_vous_depot % km/s

MR = exp(dV_star*1000/(ge*isp_vac));
mp_used = (-1/MR)*(minert + mpl + mp_max - MR*(minert + mpl + mp_max)); %kg
mp_leftover = mp_max - mp_used;

function[mp_used, mp_new_leftover] = calc_prop_mass_func(mission_dV, mpl,
mp_old_leftover)
ge = 9.81;           % earth gravity, m/s^2

% Starship values
% mpl = 100*1000;      % payload mass, kg
minert = 120000;      % inert mass, kg
mp_max = 1200*1000;   % max prop mass starship can hold, kg
isp_vac = 380;         % raptor vac isp, s

MR = exp(mission_dV*1000/(ge*isp_vac));
mp_used = (-1/MR)*(minert + mpl + mp_old_leftover - MR*(minert + mpl + mp_old_leftover));
%kg
mp_new_leftover = mp_old_leftover - mp_used;

% where:
% MR = m0/mf
% m0 = minert + mpl + m_old_leftover_fuel
% mf = minert + mpl + (m_old_leftover_fuel - m_fuel_used)
```

## B. Stationkeeping Sizing Calculations

```
g = 9.81;  % m/s2
Isp = 230; % s, isp for hydrazine
rho_h = 1021; % kg/m^3, density of hydrazine
yrs = 8.5;
%yrs = 4.5; % yrs of operation

% uncomment below for leo vals:
% for leo
dV = 20; % m/s, per year
depot_prop_mass = 0.75*1065.121*1000; % mass of propellant stored at depot, kg
depot_ptank_mass = 26706.1926740865;      % mass of prop tanks for stored prop, kg
depot_ptank_mass = 5718.66523170042;
coolers_mass = (4*60);
%coolers_mass = 0.4*4.8*1000;      % mass of cryo coolers, kg
solarp_mass = 120.53*2;            % solar panels mass, kg
```

```

sk_tanks_mass = 0.25*depot_prop_mass;           % stationkeeping tanks mass guess, kg
acs_mass = 0;
r = 2.2;  % m, radius of sk tank (2.2 for falcon heavy)

% % uncomment below for lunar vals:
% % for lunar
% dV = 130;  % m/s, per year
% depot_prop_mass = 0.75*463.274*1000;  % mass of propellant stored at depot, kg
% depot_ptank_mass = 2934.56950340985;    % mass of prop tanks for stored prop, kg
% coolers_mass = 60*2;
% coolers_mass = 0.4*2.4*1000;    % mass of cryo coolers, kg
% solarp_mass = 60.265*2;        % solar panels mass, kg
% sk_tanks_mass = 0.25*depot_prop_mass;      % stationkeeping tanks mass guess, kg
% acs_mass = 0;
% r = 2.7;  % m, radius of sk tank (2.6 for starship)

diff = 100;
while diff > 0.01
mdry = depot_prop_mass + depot_ptank_mass + coolers_mass + solarp_mass + sk_tanks_mass +
acs_mass;

MR = exp(dV/(g*Isp));
mp_sk = mdry*(MR-1)*yrs;    % mass of sk prop for x yrs

V_sk = mp_sk/rho_h;          % volume of sk prop, m^3
% V_sk = 1.05*V_sk;          % accounting for 5% ullage
V_he = 0.25*V_sk;
m_he = V_he*0.1786;          % mass of helium
V_sk = 1.25*V_sk;            % accountning for pressurant separated by a diaphragm

% using geometry of cyl w capped ends:
h_cyl = (V_sk - (4/3*pi*r^3))/(pi*r^2);  % height of cyl, m
h_tot = h_cyl + 2*r;                  % total height including ends, m
SA = (4*pi*r^2) + (2*pi*r*h_cyl);     % surface area, m^2

% calculating tank thickness, material: Ti-6Al-4V (Grade 5), STA
p_max = 24.5*0.1; % max p, Mpa
% from :
http://www.matweb.com/search/datasheet\_print.aspx?matguid=b350a789eda946c6b86a3e4d3c577b39
Ftu = 1170; % MPa, ultmate tensile strength
Fy = 1100; % Mpa, yield tensile strength
rho_tank = 4430; % kg/m^3

fac_safety = 1.25; % unmanned
max_stress = min((Fy/1.1), (Ftu/fac_safety)); % from heister's textbook
thick = p_max*r/max_stress; % tank thickness, m

V_cyl_new = pi*((r+thick)^2)*(h_cyl + thick);

```

```
V_sph_new = (4/3)*pi*((r+thick)^3);
V_tank = V_cyl_new + V_sph_new - V_sk; % volume that the tank takes up
m_sk_tank = rho_tank*V_tank; % mass of tank, kg
h_incl_tank = (h_cyl+thick) + 2*(r+ thick); % height of tank including thickness of tank, m
m_sk_sys = m_sk_tank + mp_sk + m_he; % mass of entire stationkeeping sys, kg

r_sphere = ((3*V_sk/(4*pi))^(1/3));
% iterating until sk prop tank mass settles out to a value
diff = abs(m_sk_tank - sk_tanks_mass);
sk_tanks_mass = m_sk_tank;
end

m_sk_sys
h_incl_tank
```

### C. Code for calculating launch numbers for fueling depot

```
%Bradley Bulczak
%Depot fueling flight logistics
%This MATLAB script determines the number of launches are required in order
%To provide both of the propellant depots with the necessary amount of fuel

clc
clear

%% Initialization
%Values derived from other group member calculations

dV_esurf_to_600km = 9.5;
dV_vous_depot = 10/1000;      % rendezvous with depot
dV_tli = 3.1809;
dV_loi = 0.8928;            % lunar orbit insertion
dV_100km_to_lsurf = 2.5;
dV_lsurf_to_100km = 2.2;
dV_llo_to_leo = 4.2124;
m_inert = 120*1000;
mpl = 0;
mp_max = 1200000;
isp_vacuum = 380;
isp_landing = 335;

m_fuel_required_leo = 1065.121 * 1000; %kg
max_leo_capacity = 1128.9*1000; %kg

m_fuel_required_lunar = 463.274*1000; %kg

two_way_min = 378.983*1000; %kg

current_fuel_lunar = 0;
current_fuel_leo = 0;

%% Calculations

%Calculates the amount of fuel leftover at the leo and lunar depots
[dV_shb, dV_star_reuse, dV_star_expend, dVtotal_reuse, dVtotal_expend, mpLanding] =
delta_v_payload(mpl);
[mp_used1, mp_leftover1] = esurf_to_depot(mpl, dV_esurf_to_600km, dV_shb,
dV_vous_depot);

mp_leftover1 = mp_leftover1 - mpLanding;

dV_leo_to_llo = dV_tli + dV_loi;
[mp_used2, mp_leftover2] = calc_prop_mass_func(dV_leo_to_llo, mpl, mp_max);
```

```
%Adjusts for starship returning to earth returning to earth
[mp_used, mp_leftover2] = calc_prop_mass_func(1.1, mpl, mp_leftover2);
mp_leftover2 = mp_leftover2 - mpLanding;

leo_flight = 0;
lunar_flight = 0;
i = 1;
j = 1;

%Refueling both depots
while current_fuel_leo < max_leo_capacity && current_fuel_lunar < m_fuel_required_leo
    %Ensures both depots are fully refueled
        current_fuel_leo = current_fuel_leo + mp_leftover1; %Adds leftover fuel from launch to leo
        if current_fuel_leo > max_leo_capacity %Checks to make sure fuel at leo isn't over capacity
            current_fuel_leo = max_leo_capacity;
        end
        leo_flight = leo_flight + 1; %Adds a flight to the number of flights to leo depot
        leo_flights(i) = leo_flight; %Puts flight number in a vector
        fuel_leo(i) = current_fuel_leo; %Puts current fuel at leo in a vector
        i = i + 1; %
        if current_fuel_leo > mp_used1
            if current_fuel_lunar < m_fuel_required_lunar
                current_fuel_lunar = current_fuel_lunar + mp_leftover2;
                lunar_flight = lunar_flight + 1; %counts the lunar flights
                lunar_flights(j) = lunar_flight; %Puts lunar flights in a vector
                fuel_lunar(j) = current_fuel_lunar; %creates a vector of the lunar fuel
                current_fuel_leo = current_fuel_leo - mp_used1; %Adjusts the new fuel at leo
                j = j + 1;
                leo_flight = leo_flight - 1; %makes sure lunar launch isn't counted as leo launch
            end
        end
    end

%% Plotting
minimum_leo = ones(1,length(fuel_leo))*m_fuel_required_leo/1000;
max_leo = ones(1,length(fuel_leo))*max_leo_capacity/1000;
two_way = ones(1,length(fuel_leo))*two_way_min/1000;

%Plots the fuel at the leo depot against the number of launches to the leo
%depot
figure(1)
plot(leo_flights,fuel_leo/1000)
hold on
grid on
plot(leo_flights,max_leo)
plot(leo_flights, minimum_leo)
plot(leo_flights,two_way)
```

```
xticks(0:length(fuel_leo))
title('Fuel at LEO')
xlabel('Launches to leo depot')
ylabel('Fuel at lunar depot (Mg)')
legend('Fuel at leo','Max leo capacity','Minimum for one way trip','Minimum for two way trip')
legend('Location','southeast')
hold off

minimum_lunar = ones(1,length(fuel_lunar))*m_fuel_required_lunar/1000;

%Plots the fuel at the lunar station against the number of flights
figure(2)
plot(lunar_flights,fuel_lunar/1000)
hold on
grid on
plot(lunar_flights,minimum_lunar)
xticks(0:length(fuel_lunar))
title('Fuel at lunar depot')
xlabel('Flight to lunar depot #')
ylabel('Fuel at leo depot (Mg)')
legend('Fuel at llo','Minimum fuel required')
legend('Location','southeast')
hold off

final_fuel_leo = fuel_leo(end)/1000;
final_fuel_lunar = fuel_lunar(end)/1000;
```

## D. Solar Array Sizing

```
%% EOL Initialization
EOL_LEO = 4800; % End of Life power requirement for LEO depot
EOL_LLO = 2400; % End of Life power requirement for LLO depot

%% BOL Calculation
BOL_LEO = EOL_LEO/((1-0.005)^14); % Beginning of Life power requirement for
LEO depot
BOL_LLO = EOL_LLO/((1-0.005)^14); % Beginning of Life power requirement for
LLO depot

%% Length and Width Calculation
total_area_LEO = BOL_LEO/300; % Total solar array area for LEO depot
total_area_LLO = BOL_LLO/300; % Total solar array area for LLO depot

per_array_LEO = total_area_LEO/2; % Area of each solar array on LEO depot
per_array_LLO = total_area_LLO/2; % Area of each solar array on LLO depot

width_LEO = sqrt(per_array_LEO/3); % Width of each solar array on LEO depot
width_LLO = sqrt(per_array_LLO/3); % Width of each solar array on LLO depot

length_LEO = width_LEO*3; % Length of each solar array on LEO depot
length_LLO = width_LLO*3; % Length of each solar array on LLO depot

%% Output
fprintf("\nThe BOL power requirement for the LEO propellant depot is %f W",
BOL_LEO)
fprintf("\nThe BOL power requirement for the LLO propellant depot is %f W",
BOL_LLO)
fprintf("\nThe solar arrays for the LEO propellant depot are %f x %f meters each",
width_LEO, length_LEO)
fprintf("\nThe solar arrays for the LLO propellant depot are %f x %f meters each\n\n",
width_LLO, length_LLO)
```

Code for ACS system in lunar orbit included in satellites portion.

### E. Code for ACS system in LEO orbit: (Jaxon)

```
%%%%%%%%
% ADCS_LEO_sizing.m
%
% Environmental Disturbances and Attitude Determination and Control System
% sizing - this program calculates the maximum disturbance torques around
% the Earth in LEO. The program uses this max torque to determine RWA
% sizing.
% Also determines the force needed for magnetic torquers
%
% AAE450: Project Next Step
%%%%%%%%%%%%% PROGRAMMERS %%%%%%%%%%%%%%
% Jaxon Connolly -- Controls
%%%%%%%%%%%%%
%
% INPUTS
% satellite - prop_depot
% r_a - spacecraft apoapsis
% r_p - spacecraft periapsis
% I_xx, I_yy, I_zz - spacecraft moments of inertia
% theta - maximum angle of deviation from z-axis
% q - reflectance factor
% i - angle of incidence from sun
% A_s - spacecraft surface area
% P - orbital period
% theta_slew - slew angle
% t_slew - time in which slew maneuvers must be performed
% L - distance from center of gravity to thrusters
% t_burn - burn time of thruster
% N_wheels - number of reaction wheels being used
% days - mission duration
%
% OUTPUTS
% T_gEarth - gravity gradient from the Earth
% T_gMoon - gravity gradient from the moon
% T_sp - torque from solar pressure radiation
% T_mag - torque from Earth's magnetic field
% T_max - maximum projected environmental torque
% T_mag - torque from Earth's magnetic field
% T_req - required torque for reaction wheel disturbance mediation
% h_RW - momentum storage in reaction wheel
% P_RW - reaction wheel power
% F_MD - force required by thrusters for momentum dumping
%
% PLOTS
```

```

% none
%
% FLAGS
% none
%
% ADDITIONAL DEVELOPMENT NOTES:
% ---Equations for environmental disturbances and reaction wheel sizing
% come from ValiSpace How To's
%%%%%%%%%%%%%%%
clear all;close all;clc;

% Environmental Constants
mu_earth = 398600.4418; %[km^3/s^2] Earth's gravitational parameter
mu_moon = 4902.8695;   %[km^3/s^2] Moon's gravitational parameter
r_earth_moon = 384400;  %[km] distance between Earth and Moon
J_s = 1367;             %[W/m^2] solar constant
c = 3e8;                %[m/s] speed of light
g_earth = 9.81;          %[m/s] acceleration due to earth's gravity
r_earth = 6378;          %[km] radius of Earth

% Inputs
satellite = "prop-depot";

if satellite == "prop-depot"
    r_a = 600;      %[km] LEO propellant depot apoapsis
    r_p = 600;      %[km] LEO propellant depot periapsis
    I_xx = 1.301e7; %[kg*m^2] LEO propellant depot moment of inertia x-axis
    I_yy = 4.192e7; %[kg*m^2] LEO propellant depot moment of inertia y-axis
    I_zz = 4.362e7; %[kg*m^2] LEO propellant depot moment of inertia z-axis
    A_s = 160.3085; %[m^2] LEO propellant depot surface area
    P = 146.264;    %[s] LEO propellant depot orbital period
    theta_slew = 0;  %[rad] LEO propellant depot slew angle
    t_slew = 1;      %[s] LEO propellant depot time for slew maneuver
    L = 50;          %[m] distance from center of gravity to thrusters
    theta = 0.01;    %[rad] maximum angle of deviation from z-axis
end

q = 0.6;      % reflectance factor
inc = 0;       %[rad] angle of incidence from sun
D = 1;         %[A*m^2] residual dipole of the satellite
mag_mom = 7.96e15; %[tesla*m^2] magnetic moment of Earth
t_burn = 5;    %[s] time of burn for thrusters
N_wheels = 4;  % number of reaction wheels
days = 15*365; %[day] mission duration

% Initializing
orbital_rad_earth = [r_a, r_p];
orbital_rad_moon = [r_earth_moon - r_a, r_earth_moon - r_p, r_earth_moon + r_a, r_earth_moon + r_p];

```

```
% Gravity Gradient Torque
for i = 1 : length(orbital_rad_earth)
    T_gEarth_vec(i) = 3 * mu_earth * abs(I_zz - I_yy) * sin(2 * theta) / (2 * orbital_rad_earth(i) ^
3); %[Nm]
end
T_gEarth_zy = max(T_gEarth_vec);
for i = 1 : length(orbital_rad_earth)
    T_gEarth_vec(i) = 3 * mu_earth * abs(I_zz - I_xx) * sin(2 * theta) / (2 * orbital_rad_earth(i) ^
3); %[Nm]
end
T_gEarth_zx = max(T_gEarth_vec);
T_gEarth = max(T_gEarth_zy, T_gEarth_zx); % max gravity gradient due to Earth
for i = 1 : length(orbital_rad_moon)
    T_gMoon_vec(i) = 3 * mu_moon * abs(I_zz - I_yy) * sin(2 * theta) / (2 * orbital_rad_moon(i) ^
3); %[Nm]
end
T_gMoon_zy = max(T_gMoon_vec);
for i = 1 : length(orbital_rad_moon)
    T_gMoon_vec(i) = 3 * mu_moon * abs(I_zz - I_xx) * sin(2 * theta) / (2 * orbital_rad_moon(i) ^
3); %[Nm]
end
T_gMoon_zx = max(T_gMoon_vec);
T_gMoon = max(T_gMoon_zy, T_gMoon_zx); % max gravity gradient due to Moon

% Solar Radiation Pressure Torque
solar_force = J_s * A_s * cos(inc) * (1 + q) / c; %[N]
diff_cps_cg = 0.1 * A_s; % difference of center of solar pressure and center of gravity
T_sp = abs(solar_force) * diff_cps_cg; %[Nm]

% Magnetic Field Torque
T_mag = 2 * D * mag_mom / ((r_p + r_earth) * 100)^3;

% Required Torque
T_g_max = max(T_gEarth, T_gMoon);
T_max = T_sp + T_g_max + T_mag;
T_req = 1.25 * T_max; %[Nm]

% Required Momentum Storage
h_RW = T_req * P * 0.707 / 4; %[Nm/s]

% Required Power
P_RW = 1000 * T_req + 4.51 * h_RW ^ 0.47; %[W]

% Momentum Dump Requirements
% Required Force
F_MD = h_RW / L / t_burn; %[N]
% Total Impulse
I_t = t_burn * N_wheels * days;
```

## F. Gravity Gradient Stabilization Analysis Code

```
%%%%%%%%%%%%%%%
%%%%%
% PropDepotACS.m
%
% This script is intended to perform basic calculations to assess the
% feasibility of passive gravity gradient stabilization for attitude
% control of the propellant depot in low-Earth orbit
%
% AAE450: Project Next Step
%%%%%%%%%%%%%%%
% PROGRAMMERS %%%%%%
% Lorin Nugent -- Controls
%%%%%%%%%%%%%%%
%
% INPUTS
% mdepot - mass of the propellant depot (kg)
% h - altitude of orbit (km)
% th - perturbation angle (degrees)
% mdb - dumbbell mass vector (kg)
%
% OUTPUTS
% none
%
% PLOTS
% Distance required to ensure stability for a vector of input dumbbell
% masses
%
% FLAGS
% none
%
% ADDITIONAL DEVELOPMENT NOTES:
% For this code, masses are assumed to be point masses and all other
% attitude disturbances are ignored. To increase the fidelity of the model,
% consider including the gravity torque on the body itself.
%
% CAUTION:
% This code is unfinished and has not yet produced the expected results. It
% is possible that the point mass assumption made is too unrealistic for
% this type of calculation.
%%%%%%%%%%%%%%%
%
% Inputs %
mdepot = 966292; % INPUT - mass of fully-fueled propellant depot (kg)
h = 600; % INPUT - altitude of orbit (km)

%
% Constants %
mu = 3.986e5; % gravitational parameter of the Earth (km^3/s^2)
```

```

R_earth = 6371; % radius of the orbit (km)

% Gravity-Gradient Stability Calculations
R = R_earth + h; % orbit radius (km)

th = 5; % INPUT - perterbation angle (deg)
L = 10; % distance between dumbbell mass and depot (m)
mdb = 100:100:5000; % INPUT dumbbell mass vector
L_out = zeros(1,length(mdb)); % distance required for the given mass

for i = 1:length(mdb)
    mass = mdb(i);
    dist = L;
    Mdepot = 1;Mdb = 0; % ICs to start the loop

    % with a fixed mdb, iterate through L to find minimum distance for stability
    while Mdepot > Mdb
        x_L_cm = mdepot/(mdepot + mass); % dimensionless location of cm relative to dumbbell
        mass
        Ldb = x_L_cm*dist; % lever arm of dumbbell (db) mass (m)
        Ldepot = (1 - x_L_cm)*dist; % lever arm of depot mass (m)
        Rdb = R - Ldb*cosd(th)/1000; % radius from center of Earth to db mass (km)
        Rdepot = R + Ldepot*cosd(th)/1000; % radius from center of Earth to depot mass (km)
        Fdb = mu*mass/Rdb^2; % grav force on dumbbell mass (kN)
        Fdepot = mu*mdepot/Rdepot^2; % grav force on depot mass (kN)

        Mdb = Ldb*sind(th)*Fdb; % moment about cm generated by db mass (kN*m)
        Mdepot = Ldepot*sind(th)*Fdepot; % moment about cm generated by depot mass
        (kN*m)

        if Mdepot > Mdb
            dist = dist+1;
        end
    end

    L_out(i) = dist;
end

% Plot Results
plot(mdb,L_out)
xlabel('Dumbbell Mass (kg)')
ylabel('Distance (m)')
title('Distance between Masses Required for Stability')
grid on

```

## G. Tank Sizing Code

```
%Uncomment the mass you want to look at
m = 463.274*1000 *1.06; % for Lunar from tons to lunar

%Pick amount of tanks you are going to use and Max radius
rOx = 2.7;
tankNumberOx = 1;
tankNumberM = 1;

%SPLITING UP MASS
mOx = 3.55* m / 4.55;
mM = m/4.55;

%TOTAL VOLUME NEEDED
vOxt = mOx/1141
vMt = mM/465

%IF YOU HAVE TO SPLIT INTO MULTIPLE TANKS
vOx = vOxt/tankNumberOx;
vM = vMt/tankNumberM;

%FIGURE OUT THE TANK DIMENSIONS

%HEIGHT MAX IS 6.5 + 2.2 = 8.7 FOR FALCON HEAVY AND 18 FOR STARSHIP
%FIND HEIGHT OF CYLINDER AND THEN TOTAL HEIGHT WITH CAPPED ENDS
hOxC = (vOx - (4/3*pi()*rOx^3))/ (pi()*rOx^2);
hOx = hOxC + 2*rOx
rM = rOx;
hMC = (vM - (4/3*pi()*rM^3))/ (pi()*rM^2);
hM = hMC + 2*rM

%SURFACE AREA FOR EACH
AOx = (4*pi()*rOx^2) + (2*pi()*rOx*hOxC);
AM = (4*pi()*rM^2) + (2*pi()*rM*hMC);

%% THICKNESS SIZING LLO
%[LOX,CH4]
% PRESSURE OX
R= 8.3145; %J/mK
M_Gas = [0.05 * mOx .05*mM]; %kg, 16% of total fuel now gas
Mw = [16 16.04]; %g/mol
n = [M_Gas(1) * 100 / Mw(1) M_Gas(2) * 100 / Mw(2)]; % Moles of Ox
Tb = [54.36,111.55]; %Boiling point of LOX,CH4
Vgas = (4/3)*pi()*rOx^3; %m^3
Pgas = [n(1) * R * Tb(1) / Vgas n(2) * R * Tb(1) / Vgas]; %PA
```

```
%Material
```

$F_y = 608E6$ ; %PA, aluminum yield tensile strength

$F_tu = 700E6$ ; %PA, ultimate tensile strength

$f_s = 1.25$ ; %Factor of safety

$\text{Stress\_allowable} = \min([F_y/1.1 F_tu/f_s])$ ; %maximum allowed working stress level

$t_c = [P_{\text{gas}}(1) * r_{\text{Ox}} / \text{Stress\_allowable} P_{\text{gas}}(2) * r_{\text{Ox}} / \text{Stress\_allowable}]$

%  $E_{\text{al}} = 88.5E9$ ; % Pa, youngs modulus

%  $\text{Crit\_Stress} = -E_{\text{al}} * (9 * (t_c/r_{\text{Ox}})^{1.6} + 0.16 * (t_c/h_{\text{Ox}})^{1.3})$ ;

% Ox Cylindrical Tank dimensions

$\rho_{\text{AL2195}} = 2685$ ; %kg/m<sup>3</sup>

$r_{\text{in}} = r_{\text{Ox}}$ ;

$r_{\text{out}} = t_c(1) + r_{\text{in}}$ ;

$V_{\text{OxAL}} = \pi * h_{\text{OxC}} * (r_{\text{out}}^2 - r_{\text{in}}^2)$ ;

$m_{\text{OxAL}} = \rho_{\text{AL2195}} * V_{\text{OxAL}}$ ;

$V_{\text{OxCaps}} = (4/3) * \pi * (r_{\text{out}}^3 - r_{\text{in}}^3)$ ;

$m_{\text{OxCaps}} = \rho_{\text{AL2195}} * V_{\text{OxCaps}}$ ;

$M_{\text{OxTank}} = m_{\text{OxAL}} + m_{\text{OxCaps}}$

$A_{\text{ExternalOx}} = (4 * \pi * r_{\text{out}}^2) + (2 * \pi * r_{\text{out}} * h_{\text{OxC}})$

$r_{\text{outOx}} = r_{\text{out}}$

% CH4 Cylindrical Tank dimensions

$\rho_{\text{AL2195}} = 2685$ ; %kg/m<sup>3</sup>

$r_{\text{in}} = r_{\text{Ox}}$ ;

$r_{\text{out}} = t_c(2) + r_{\text{in}}$ ;

$V_{\text{CH4AL}} = \pi * h_{\text{MC}} * (r_{\text{out}}^2 - r_{\text{in}}^2)$ ;

$m_{\text{CH4AL}} = \rho_{\text{AL2195}} * V_{\text{CH4AL}}$ ;

$V_{\text{CH4Caps}} = (4/3) * \pi * (r_{\text{out}}^3 - r_{\text{in}}^3)$ ;

$m_{\text{CH4Caps}} = \rho_{\text{AL2195}} * V_{\text{CH4Caps}}$ ;

$M_{\text{CH4Tank}} = m_{\text{CH4AL}} + m_{\text{CH4Caps}}$

$A_{\text{ExternalM}} = (4 * \pi * r_{\text{out}}^2) + (2 * \pi * r_{\text{out}} * h_{\text{MC}})$

%MLI Calculations

$\text{thicknessLayer} = 2.286 * 10^{-5}$ ;

$\text{layers} = 60$ ;

$\text{MassPerArea} = .08$ ;

$\text{thicknessMLI} = \text{thicknessLayer} * \text{layers}$ ;

%Area of outside of MLI per Tank

$r_{\text{MLIOx}} = r_{\text{outOx}} + \text{thicknessMLI}$ ;

$r_{\text{MLIM}} = r_{\text{out}} + \text{thicknessMLI}$ ;

$A_{\text{MLIM}} = (4 * \pi * r_{\text{MLIM}}^2) + (2 * \pi * r_{\text{MLIM}} * h_{\text{MC}})$

$A_{\text{MLIOx}} = (4 * \pi * r_{\text{MLIOx}}^2) + (2 * \pi * r_{\text{MLIOx}} * h_{\text{OxC}})$

```
%Total Areas used for cost estimate that adds for each layer
TotalSurfaceAreaMLIM = 0;
TotalSurfaceAreaMLIOx = 0;

for i = 1:60
    rMLIOx = routOx + thicknessLayer * i;
    rMLIM = rout + thicknessLayer *i ;
    AreaMLIM = (4*pi()*rMLIM^2) + (2*pi()*rMLIM*hMC);
    AreaMLIOx = (4*pi()*rMLIOx^2) + (2*pi()*rMLIOx*hOxC);
    TotalSurfaceAreaMLIM = TotalSurfaceAreaMLIM + AreaMLIM;
    TotalSurfaceAreaMLIOx = TotalSurfaceAreaMLIOx + AreaMLIOx;
end

TotalSurfaceAreaSystem = TotalSurfaceAreaMLIOx * tankNumberOx +TotalSurfaceAreaMLIM
* tankNumberM

%Mass of the MLI per Each tank
MassMliOxPerTank = TotalSurfaceAreaMLIOx * MassPerArea
MassMliCH4PerTank = TotalSurfaceAreaMLIM* MassPerArea

%Total Mass of Tank and MLI
MassOfEachTankWithMliOx = MassMliOxPerTank+ MassOXTank
MassofEachTankWithMliCH4 = MassMliCH4PerTank + MassCH4Tank

%Total Mass of System
totalMass = MassOfEachTankWithMliOx * tankNumberOx + MassofEachTankWithMliCH4 *
tankNumberM
```

## H. Wrapper for finding available delta v for Starship (Trevor Pfeil)

Delta\_v\_payload.m

```
%% Starship launch platform delta v estimator
% Author: Trevor Pfeil

%%Definitions
payload = 100; %metric tons
isp_landing = 353; %isp used for sea level propulsive landing calculations in seconds
isp_starship_ascent = 365; %average isp of starship during its ascent phase in seconds
isp_SHB_ascent = 353; %average isp of super heavy booster during its ascent phase in seconds
isp_vacuum = 380; %isp of starship during its deorbiting phase in seconds
inert_mass_starship = 120; %not including payload as inert mass. In metric tons
propellant_mass_starship = 1200; %in metric tons
inert_mass_SHB = 280; %not including payload as inert mass. In metric tons
propellant_mass_SHB = 3300; %in metric tons

%% Vehicle variable calculations
inert_mass_fraction_SHB = inert_mass_SHB/(inert_mass_SHB + propellant_mass_SHB); %inert mass fraction for 1st stage
inert_mass_fraction_starship = inert_mass_starship/(inert_mass_starship + propellant_mass_starship); %inert mass fraction for 2nd stage
payload_stage_1 = payload + inert_mass_starship + propellant_mass_starship; %effective payload that the 1st stage must carry in metric tons
GLOW = payload + inert_mass_starship + propellant_mass_starship + inert_mass_SHB + propellant_mass_SHB; %Gross liftoff weight in metric tons

%% Available delta V for each configuration.
dv_1 = SHB_delta_v(inert_mass_fraction_SHB, payload_stage_1, GLOW, isp_landing, isp_SHB_ascent); %delta v from 1st stage
dv_2 = Starship_delta_v(inert_mass_fraction_starship, payload, inert_mass_starship + propellant_mass_starship, isp_landing, isp_starship_ascent, isp_vacuum); %delta v from 2nd stage
dv_2_reuse = dv_2(1);
dv_2_expend = dv_2(2);
dv_total_reuse = dv_1 + dv_2_reuse; %delta V available if we land SHB and land the starship upper stage.
dv_total_expend = dv_1 + dv_2_expend; %delta V available if we land SHB but expend the starship upper stage.
```

## I. Determining the delta v available from the Super Heavy Booster)

SHB\_delta\_v.m

```
function[delta_v_ideal] = SHB_delta_v(lambda, payload_mass, GLOW, isp_landing,
isp_SHB_ascent)
%%Inputs
% lambda = inert mass fraction of the Super Heavy booster only: calculated
% without considering payload as inert mass
% payload_mass = the payload that the first stage is carrying in metric
% tons
% GLOW = gross liftoff weight of entire vehicle (1st stage, 2nd stage, and
% the payload of the entire vehicle) in metric tons
% isp_landing = isp of engines at sea level in seconds
% isp_SHB_ascent = average isp of super heavy booster during its
% ascent phase in seconds
% Author: Trevor Pfeil

%% Converting to kg
GLOW = GLOW * 1000;
payload_mass = payload_mass * 1000;
%%Determining the delta_v available for only reusable configurations
SHB_mass = (GLOW-payload_mass); %mass of SHB at liftoff in kg
SHB_final = SHB_mass * lambda + SHB_landing(lambda, SHB_mass, isp_landing); %mass of
SHB at stage separation in kg
delta_v_ideal = 9.81*isp_SHB_ascent*log(GLOW/(SHB_final+payload_mass));
```

## J. Determining the propellant and delta v requirements to propulsively land the Super Heavy Booster

SHB\_landing.m

```
function[mp] = SHB_landing(lambda,SHB_mass, isp_landing)
% Author: Trevor Pfeil
%% Definitions
CDA = 64.8+52.17; %summed frontal areas * local drag coefficients
a_dens = 1.225; %air density in kg/m^3
g = 9.81; %m/s^2
isp = isp_landing; %sec
raptor_thrust = 2200000; %N
engine_count = 3; %number of engines used in descent
dv_safety_factor = 0.1; % extra percent delta v desired for safety margins
inert_mass = SHB_mass*lambda;%kg

%% Propellant needed to land propulsively
m_dot_raptor = raptor_thrust/isp/g; %kg/s
```

```
m_dot_engines = m_dot_raptor * engine_count; %kg/s
thrust = raptor_thrust * engine_count; %N
mp_guess = 80000;%kg
err = 999;

while(abs(err)>0.1)
mt = mp_guess + inert_mass; %kg
v_term1 = sqrt((2*g*(mt))/(a_dens*CDA)); %delta v needed based on mp_guess
v_term2 = g*isp*log((mt)/inert_mass) - g*(mt-mt*exp(-m_dot_engines*v_term1 /
thrust))/m_dot_engines; %delta v available based on mp_guess
err = v_term1-v_term2;
if v_term2>v_term1
    mp_guess = mp_guess - 10; %decrease propellant mass
else
    mp_guess = mp_guess + 1; %increase propellant mass
end
end
mp = mp_guess * (1+dv_safety_factor);
```

## K. Determining the delta v available from Starship

Starship\_delta\_v.m

```
function[outputs] = Starship_delta_v(lambda, payload, starship_mass, isp_landing,
isp_starship_ascent, isp_vacuum)
% Author: Trevor Pfeil

%% Converting to kg
payload = payload * 1000;
starship_mass = starship_mass * 1000;
%%Determining the delta_v available for each configuration
starship_final = (starship_mass * lambda) + Starship_landing(lambda, starship_mass,
isp_landing, isp_vacuum) + payload; %kg

delta_v_starship_reuse = 9.81 * isp_starship_ascent * log(starship_mass / starship_final); %m/s
delta_v_starship_expend = 9.81 * isp_starship_ascent * log(starship_mass / (payload +
starship_mass*lambda)); %m/s
outputs = [delta_v_starship_reuse, delta_v_starship_expend];
```

## L. Determining the propellant and delta v requirements to propulsively land Starship

Starship\_landing.m

```
function[mp] = Starship_landing(lambda, starship_mass, isp_landing, isp_vacuum)
% Author: Trevor Pfeil

%% Definitions
CDA = 334.9044; %summed frontal areas * local drag coefficients
a_dens = 1.225; %air density in kg/m^3
g = 9.81; %m/s^2
isp = isp_landing; %sec
isp_vac = isp_vacuum; %sec
raptor_thrust = 2200000; %N
engine_count = 3; %number of engines used in descent
dv_safety_factor = 1; % extra percent delta v desired for safety margins
inert_mass = lambda * starship_mass;%kg

%%Propellant to land propulsively
m_dot_raptor = raptor_thrust/isp/g; %kg/s
m_dot_engines = m_dot_raptor * engine_count; %kg/s
thrust = raptor_thrust * engine_count; %N
mp_guess = 10000;%kg
err = 999;
while(abs(err)>0.1)
mt = mp_guess + inert_mass; %kg
v_term1 = sqrt((2*g*(mt))/(a_dens*CDA)); %delta v needed based on mp_guess
```

```

v_term2 = g*isp*log((mt)/inert_mass) - g*(mt-mt*exp(-m_dot_engines*v_term1 /
thrust))/m_dot_engines; %delta v available based on mp_guess

err = v_term1-v_term2;
if v_term2>v_term1
    mp_guess = mp_guess - 10; %decrease propellant mass
else
    mp_guess = mp_guess + 1; %increase propellant mass
end
end
mp1 = mp_guess * (1+dv_safety_factor);
%%Propellant to deorbit starship
target = 100; %km target altitude for effective aerobraking
current = 600; %km current orbital altitude
dv = sqrt(398600.4415/(6378.136+current)) * (1-
sqrt(2*(target+6378.136)/(2*6378.136+target+current))) *1000; %m/s delta v needed to reach
target periapsis
mp2 = (exp(dv / 9.81 /isp_vac) * (inert_mass + mp1)) - inert_mass - mp1;%mass of propellant to
de-orbit starship
mp = mp1 + mp2;

```

## M. Estimating boiloff and tracking propellant mass in propellant depots (Trevor Pfeil)

cryos.m

```

%Author: Trevor Pfeil
close all
clear
%% setup
q_vap_lox = 214000; %Vaporization energy of oxygen J/kg
q_vap_ch4 = 136360; %Vaporization energy of methane J/kg
cp_lox = 920; %Heat capacity of liquid oxygen J/kgK
cp_ch4 = 2087; %Heat capacity of liquid methane J/kg
t_lox = 90.19; %lox boiling temp at 1 atm in K
t_ch4 = 111.6; %ch4 boiling temp at 1 atm in K
subcool_lox = 1;% temp below boiling lox in K
subcool_ch4 = 1;% temp below boiling ch4 in K
rho_ch4 = 438.9; %density of liquid ch4 kg/m3
rho_lox = 1141; %density of lox kg/m3
thermal_conductivity_lox = 0.026; %Thermal conductivity of liquid oxygen w/mK
thermal_conductivity_ch4 = 0.1197; %Thermal conductivity of liquid methane w/mK
prop_storage = 1100; %total amount of propellant stored in Mg or the amount required for a leg
of the trip
mix_lox = 0.78; %mixture ratio of lox
mix_ch4 = 1-mix_lox; %mixture ratio of ch4
ullage = 0.06; %size of ullage
heat_load = 0.4; %heat load in w/m^2 from solar radiation

```

```

heat_load_extra = 0.2;%heat load in w/m^2 from conduction in the depot
active_cooling = 300; %active cooling refrigeration capacity for methane in Watts
active_cooling_lox = 300; %active cooling refrigeration capacity for oxygen in Watts
launch_schedule = [1,8,14,16,20,25]; %days on which tankers will fly to the LEO depot
launch_schedule = [launch_schedule,0];%the launch schedule should always start on day 1.
propellant_payload_capacity = 193.593669; %in Mg
if(size(launch_schedule)<prop_storage/propellant_payload_capacity)
    fprintf("\n Insufficient launches given to fill LEO depot to required amount\n Add another day
to the launch schedule\n")
end
propellant_payload_capacity = propellant_payload_capacity*1000; %converting propellant to kg

%% tank sizing
tank_count_lox = 2;
tank_count_ch4 = 2;
radius = 2.2;%meters
height_lox = 26;%meters
height_ch4 = 26;%meters
sa_cap_lox = 4*pi*radius^2; %m^2
sa_cap_ch4 = 4*pi*radius^2; %m^2
sa_wall_lox = (height_lox - 2*radius)*pi*2*radius; %m^2
sa_wall_ch4 = (height_ch4 - 2*radius)*pi*2*radius; %m^2
sa_lox = sa_cap_lox+sa_wall_lox; %surface area of 1 lox tank
sa_ch4 = sa_cap_ch4+sa_wall_ch4; %surface area of 1 ch4 tank
tank_sa_lox = sa_lox*tank_count_lox; %surface area of all lox tanks
tank_sa_ch4 = sa_ch4*tank_count_ch4; %surface area of all ch4 tanks
heat_flux_lox = tank_sa_lox*(heat_load); %w
heat_flux_ch4 = tank_sa_ch4*(heat_load); %w

%% no active cooling
evap_rate_lox = heat_flux_lox / q_vap_lox *60*60*24; %kg/day
evap_rate_ch4 = heat_flux_ch4 / q_vap_lox*60*60*24; %kg/day
point_evap_rate_lox = tank_sa_lox * heat_load_extra / (q_vap_lox+cp_lox*(subcool_lox))
*60*60*24; %kg/day
point_evap_rate_ch4 = tank_sa_ch4 * heat_load_extra / (q_vap_ch4+cp_ch4*(subcool_ch4))
*60*60*24; %kg/day
total_evap_rate_lox = evap_rate_lox + point_evap_rate_lox;
total_evap_rate_ch4 = evap_rate_ch4 + point_evap_rate_ch4;

%% active cooling
active_evap_rate_lox = (heat_flux_lox-active_cooling_lox) / q_vap_lox *60*60*24; %kg/day
active_evap_rate_ch4 = (heat_flux_ch4-active_cooling) / q_vap_lox*60*60*24; %kg/day

active_point_evap_rate_lox = tank_sa_lox * heat_load_extra /
(q_vap_lox+cp_lox*(subcool_lox)) *60*60*24; %kg/day
active_point_evap_rate_ch4 = tank_sa_ch4 * heat_load_extra /
(q_vap_ch4+cp_ch4*(subcool_ch4)) *60*60*24; %kg/day

active_total_evap_rate_lox = active_evap_rate_lox + active_point_evap_rate_lox;%kg/day

```

```

active_total_evap_rate_ch4 = active_evap_rate_ch4 + active_point_evap_rate_ch4;%kg/day

%% tracking propellant over time with flights to refuel
daily_heat_flux_lox_passive = tank_sa_lox*(heat_load+heat_load_extra)*60*60*24; %J/day
daily_heat_flux_ch4_passive = tank_sa_ch4*(heat_load+heat_load_extra)*60*60*24; %J/day
daily_heat_flux_lox_active = (tank_sa_lox*(heat_load+heat_load_extra)-
active_cooling_lox)*60*60*24; %J/day
daily_heat_flux_ch4_active = (tank_sa_ch4*(heat_load+heat_load_extra)-
active_cooling)*60*60*24; %J/day

temp_lox_in = t_lox-subcool_lox; %initial temp in kelvin when transferred
temp_ch4_in = t_ch4-subcool_lox; %initial temp in kelvin when transferred

stored_lox_mass_passive = 0;
stored_ch4_mass_passive = 0;
stored_lox_mass_active = 0;
stored_ch4_mass_active = 0;
temp_lox_passive = 0;
temp_ch4_passive = 0;
temp_lox_active = 0;
temp_ch4_active = 0;

for n = 1:length(launch_schedule)-1

    temp_lox_passive = (temp_lox_in*(propellant_payload_capacity*mix_lox) +
    temp_lox_passive*stored_lox_mass_passive)/((propellant_payload_capacity*mix_lox)+stored_lox_mass_passive);
    temp_ch4_passive = (temp_ch4_in*(propellant_payload_capacity*mix_ch4) +
    temp_ch4_passive*stored_ch4_mass_passive)/((propellant_payload_capacity*mix_ch4)+stored_ch4_mass_passive);
    temp_lox_active = (temp_lox_in*(propellant_payload_capacity*mix_lox) +
    temp_lox_active*stored_lox_mass_active)/((propellant_payload_capacity*mix_lox)+stored_lox_mass_active);
    temp_ch4_active = (temp_ch4_in*(propellant_payload_capacity*mix_ch4) +
    temp_ch4_active*stored_ch4_mass_active)/((propellant_payload_capacity*mix_ch4)+stored_ch4_mass_active);

    stored_lox_mass_passive = stored_lox_mass_passive+propellant_payload_capacity * mix_lox;
    %adding delivered propellant to total
    stored_ch4_mass_passive = stored_ch4_mass_passive+propellant_payload_capacity * mix_ch4;
    stored_lox_mass_active = stored_lox_mass_active+propellant_payload_capacity * mix_lox;
    %adding delivered propellant to total
    stored_ch4_mass_active = stored_ch4_mass_active+propellant_payload_capacity * mix_ch4;

    temp_lox_passive = daily_heat_flux_lox_passive*(launch_schedule(n+1)-launch_schedule(n))
    /cp_lox / stored_lox_mass_passive +temp_lox_passive; %finding how much temp rise there is
    until the next delivery

```

```

temp_ch4_passive = daily_heat_flux_ch4_passive*(launch_schedule(n+1)-
launch_schedule(n)) /cp_ch4 / stored_ch4_mass_passive +temp_ch4_passive;
temp_lox_active = daily_heat_flux_lox_active*(launch_schedule(n+1)-launch_schedule(n))
/cp_lox / stored_lox_mass_active +temp_lox_active; %finding how much temp rise there is until
the next delivery
temp_ch4_active = daily_heat_flux_ch4_active*(launch_schedule(n+1)-launch_schedule(n))
/cp_ch4 / stored_ch4_mass_active +temp_ch4_active;
%check if temp is above boiling, if it is find out how much prop boiled
%off
if(temp_lox_passive>t_lox)
    q_excess = stored_lox_mass_passive*cp_lox*(temp_lox_passive-t_lox);
    lox_passive_boiloff = q_excess/q_vap_lox;
    stored_lox_mass_passive = stored_lox_mass_passive - lox_passive_boiloff;
    temp_lox_passive = t_lox; %boiloff keeps liquid at boiling point
end
if(temp_ch4_passive>t_ch4)
    q_excess = stored_ch4_mass_passive*cp_ch4*(temp_ch4_passive-t_ch4);
    ch4_passive_boiloff = q_excess/q_vap_ch4;
    stored_ch4_mass_passive = stored_ch4_mass_passive - ch4_passive_boiloff;
    temp_ch4_passive = t_ch4; %boiloff keeps liquid at boiling point
end
if(temp_lox_active>t_lox)
    q_excess = stored_lox_mass_active*cp_lox*(temp_lox_active-t_lox);
    lox_active_boiloff = q_excess/q_vap_lox;
    stored_lox_mass_active = stored_lox_mass_active - lox_active_boiloff;
    temp_lox_active = t_lox; %boiloff keeps liquid at boiling point
end
if(temp_ch4_active>t_ch4)
    q_excess = stored_ch4_mass_active*cp_ch4*(temp_ch4_active-t_ch4);
    ch4_active_boiloff = q_excess/q_vap_ch4;
    stored_ch4_mass_active = stored_ch4_mass_active - ch4_active_boiloff;
    temp_ch4_active = t_ch4; %boiloff keeps liquid at boiling point
end

lox_mass_passive(n) = stored_lox_mass_passive;
ch4_mass_passive(n) = stored_ch4_mass_passive;
lox_mass_active(n) = stored_lox_mass_active;
ch4_mass_active(n) = stored_ch4_mass_active;
launch_plot(n) = launch_schedule(n+1);
mass_lox_passive = lox_mass_passive(n);
mass_lox_active = lox_mass_active(n);
mass_ch4_passive = ch4_mass_passive(n);
mass_ch4_active = ch4_mass_active(n);
end

figure
hold on
plot(launch_schedule(1:n), lox_mass_passive,'o')
plot(launch_schedule(1:n), lox_mass_active,'x')

```

```

plot(launch_schedule(1:n), ch4_mass_passive,'+')
plot(launch_schedule(1:n), ch4_mass_active,'*')
xlabel('Mass of Propellant (Kg)')
ylabel('Days')
title('Propellant Mass During Loading')
legend('LOx No Cooling','LOx Active Cooling','CH4 No Cooling','CH4 Active
Cooling','Location','best')

%% tracking propellant mass over time (no refuelling)
if(temp_lox_active < t_lox)
saturation_time_lox_active = rho_lox*thermal_conductivity_lox * cp_lox*(temp_lox_active-
t_lox)^2 /heat_load^2/60/60/24; %days
else
    saturation_time_lox_active =0;
end
if(temp_lox_passive < t_lox)
saturation_time_lox_passive = rho_lox*thermal_conductivity_lox * cp_lox*(temp_lox_passive-
t_lox)^2 /heat_load^2/60/60/24; %days
else
    saturation_time_lox_passive = 0;
end
if(temp_ch4_active<t_ch4)
saturation_time_ch4_active = rho_ch4*thermal_conductivity_ch4 * cp_ch4*(temp_ch4_active-
t_ch4)^2 /heat_load^2/60/60/24; %days
else
    saturation_time_ch4_active = 0;
end
if(temp_ch4_passive<t_ch4)
saturation_time_ch4_passive = rho_ch4*thermal_conductivity_ch4 *
cp_ch4*(temp_ch4_passive-t_ch4)^2 /heat_load^2/60/60/24; %days
else
    saturation_time_ch4_passive = 0;
end

time = [1:1:500];
for n = time
    if(n<saturation_time_lox_passive)
        mass_lox_tracked_passive(n) = mass_lox_passive;
        mass_lox_tracked_active(n) = mass_lox_active;
    else
        mass_lox_tracked_passive(n) = mass_lox_tracked_passive(n-1) - total_evap_rate_lox;
        mass_lox_tracked_active(n) = mass_lox_tracked_active(n-1) - active_total_evap_rate_lox;
    end
    if(n<saturation_time_ch4_passive)
        mass_ch4_tracked_passive(n) = mass_ch4_passive;
        mass_ch4_tracked_active(n) = mass_ch4_active;
    else
        mass_ch4_tracked_passive(n) = mass_ch4_tracked_passive(n-1) - total_evap_rate_ch4;
        mass_ch4_tracked_active(n) = mass_ch4_tracked_active(n-1) - active_total_evap_rate_ch4;
    end
end

```

```

end
end
required_ch4 = 1093*1000*mix_ch4;
required_lox = 1093*1000*mix_lox;
n=1;
while(mass_ch4_tracked_active(n)>required_ch4 && n<length(mass_ch4_tracked_active))
    max_storage_length_ch4_active = n;
    n = n+1;
end
if(n==1)
    max_storage_length_ch4_active = 0;
end
n=1;
while(mass_ch4_tracked_passive(n)>required_ch4 && n<length(mass_ch4_tracked_passive))
    max_storage_length_ch4_passive = n;
    n = n+1;
end
if(n==1)
    max_storage_length_ch4_passive = 0;
end
n=1;
while(mass_lox_tracked_active(n)>required_lox && n<length(mass_lox_tracked_active))
    max_storage_length_lox_active = n;
    n = n+1;
end
if(n==1)
    max_storage_length_lox_active = 0;
end
n=1;
while(mass_lox_tracked_passive(n)>required_lox && n<length(mass_lox_tracked_passive))
    max_storage_length_lox_passive = n;
    n = n+1;
end
if(n==1)
    max_storage_length_lox_passive = 0;
end
max_storage_length_passive = min(max_storage_length_lox_passive,
max_storage_length_ch4_passive);
max_storage_length_active = min(max_storage_length_lox_active,
max_storage_length_ch4_active);
fprintf('\nMaximum storage length with no cooling = %f days',max_storage_length_passive)
fprintf('\nMaximum storage length with active cooling = %f days',max_storage_length_active)
figure
hold on
plot(time, mass_lox_tracked_passive/required_lox,'g-','LineWidth',2)
plot(time, mass_lox_tracked_active/required_lox,'g--','LineWidth',2)
plot(time, mass_ch4_tracked_passive/required_ch4,'r:','LineWidth',2)
plot(time, mass_ch4_tracked_active/required_ch4,'r-','LineWidth',2)
plot(max_storage_length_passive, 1,'bx','LineWidth',2)

```

```
plot(max_storage_length_active, 1,'bx','LineWidth',2)
legend('LOx No Cooling','LOx Active Cooling','CH4 No Cooling','CH4 Active
Cooling','Location','best')
title('Percentage of Required Propellant Remaining')
```

## N. Settling Burn Mass Requirement

```

Ullage_burn.m

clear
clc
%% Initialization
m_pump = 182351.03; %kg
m_LEO = 1132882.753; %kg
m_LEO_empty = 38777; %kg
m_LLO = 577588.903; %kg
m_LLO_empty = 102301.88; %kg
g = 9.81; %ms^-2
p_rate = 1200/60; %kg/s,
p_time_depot = m_pump / p_rate; %s, from starship to depot
p_time_ship = 1088622 / p_rate; %s, from depot to starship

%% Engine Characteristics, taken from 'Rocket Propulsion'
isp_methalox = 379; %s
isp_hydrazine = 234; %s

%% Required Accelerations
a_sc = .0001*g; %Short Coast <20 mins
a_ic = 8*10^(-5)*g; %Intermediate Coast 20mins to 2 hrs
a_lc = 2*10^(-5)*g; %Long Coast up to 17 hrs

%% Thrust, Mass calcs for full tanker and depots comparing coast missions
% Thrusts
F_sc_LEO = m_LEO*a_sc; %N
F_ic_LEO = m_LEO*a_ic; %N
F_lc_LEO = m_LEO*a_lc; %N
F_sc_LLO = m_LLO*a_sc; %N
F_ic_LLO = m_LLO*a_ic; %N
F_lc_LLO = m_LLO*a_lc; %N

% Mass Flow, kg/s
m_sc_LEO_m = F_sc_LEO / (isp_methalox * g);
m_sc_LEO_h = F_sc_LEO / (isp_hydrazine * g);
m_ic_LEO_m = F_ic_LEO / (isp_methalox * g);
m_ic_LEO_h = F_ic_LEO / (isp_hydrazine * g);
m_lc_LEO_m = F_lc_LEO / (isp_methalox * g);
m_lc_LEO_h = F_lc_LEO / (isp_hydrazine * g);
m_sc_LLO_m = F_sc_LLO / (isp_methalox * g);
m_sc_LLO_h = F_sc_LLO / (isp_hydrazine * g);
m_ic_LLO_m = F_ic_LLO / (isp_methalox * g);
m_ic_LLO_h = F_ic_LLO / (isp_hydrazine * g);
m_lc_LLO_m = F_lc_LLO / (isp_methalox * g);

```

```

m_lc_LLO_h = F_lc_LLO / (isp_hydrazine * g);

% Mass Calculations
mass_LEO_depot_m = p_time_depot * m_ic_LEO_m; %kg, methalox
mass_LEO_depot_h = p_time_depot * m_ic_LEO_h; %kg, hydrazine
mass_LLO_depot_m = p_time_depot * m_ic_LLO_m; %kg, methalox
mass_LLO_depot_h = p_time_depot * m_ic_LLO_h; %kg, hydrazine
mass_LEO_ship_m = p_time_ship * m_lc_LEO_m; %kg, methalox
mass_LEO_ship_h = p_time_ship * m_lc_LEO_h; %kg, hydrazine
mass_LLO_ship_m = p_time_ship * m_lc_LLO_m; %kg, methalox
mass_LLO_ship_h = p_time_ship * m_lc_LLO_h; %kg, hydrazine

%%% Depot Masses (Found in "depot min mass requirements by mission")
% Case 1, Earth to Lunar Surface, Uncrewed, No Return
% This is just a LEO refill
m_refuel_1 = 984.493136 * 1000; %kg, LEO
p_time_1 = m_refuel_1 / p_rate;
% Case 2, Earth to Lunar Surface, Crewed, No Return
% This is just a LEO refill
m_refuel_2 = 1065.121136 * 1000; %kg, LEO
p_time_2 = m_refuel_2 / p_rate;
% Case 3, Earth to Lunar surface and back to earth, crewed both ways
% stopping at both lunar and leo depots, 100 tons payload (to moon), 19 crew return
m_refuel_LEO_3 = 378.983 * 1000; %kg
m_refuel_LLO_in = 411.834 * 1000; %kg
m_refuel_LLO_out = 51.44 * 1000; %kg
p_time_LEO_3 = m_refuel_LEO_3 / p_rate;
p_time_LLO_in = m_refuel_LLO_in / p_rate;
p_time_LLO_out = m_refuel_LLO_out / p_rate;

%%% Mass Calcs for mission requirements from Prop Depot
p_times = [p_time_1 p_time_2 p_time_LEO_3 p_time_LLO_in p_time_LLO_out];
a = zeros(1,length(p_times));
%mass, kg
m_LEO_arr= [m_refuel_1 m_refuel_2 m_refuel_LEO_3];
m_LLO_arr = [m_refuel_LLO_in m_refuel_LLO_out];
m = [(m_LEO_arr + m_LEO_empty) (m_LLO_arr + m_LLO_empty)];
%Find Required Acceleration
t_sc = 20 * 60;
t_ic = 2 * 60 * 60;
t_lc = 17 * 60 * 60;
for i = 1:length(p_times)
    if p_times(i) < t_sc
        a(i) = a_sc;
    elseif p_times(i) < t_ic
        a(i) = a_ic;
    else
        a(i) = a_lc;
    end
end

```

```

end
%Thrust, N
F_req = m.* a;
%Mass Flow, kg/s
m_dot = F_req / (isp_hydrazine * g);
%Mass, kg
mass = m_dot .* p_times;
m_case_1 = mass(1);
m_case_2 = mass(2);
m_case_3_LEO = mass(3);
m_case_3_LLO = mass(4)+mass(5);
m_case_3 = m_case_3_LEO+m_case_3_LLO;

%% Mass per trip (refuels)
m_LEO_ref = linspace(m_LEO_empty,m_LEO,6);
m_LLO_ref = linspace(m_LLO_empty,m_LLO,3);
%Required Thrusts, N
% These will all be intermediate coast missions
F_LEO_ref = m_LEO_ref * a_ic;
F_LLO_ref = m_LLO_ref * a_ic;
mdot_LEO_ref = F_LEO_ref / (isp_hydrazine * g);
mdot_LLO_ref = F_LLO_ref / (isp_hydrazine * g);
%Times to fill depots, s
t_LEO_ref = mdot_LEO_ref * p_time_depot;
t_LLO_ref = mdot_LLO_ref * p_time_depot;
%Masses for refuel missions, kg
m_LEO_ref = mdot_LEO_ref .* t_LEO_ref;
m_LLO_ref = mdot_LLO_ref .* t_LLO_ref;
%Sum of masses for a full depot refuel
m_LEO_ref_total = sum(m_LEO_ref);
m_LLO_ref_total = sum(m_LLO_ref);
%Sum of masses for Case 3 (LEO wont be fully fueled)
m_LEO_ref_total_3 = m_LEO_ref_total - (m_LEO_ref(5) + m_LEO_ref(6));

%% Full mission masses:
%Case 1
m_tot_LEO_1 = m_LEO_ref_total + m_case_1;
%Case 2
m_tot_LEO_2 = m_LEO_ref_total + m_case_2;
%Case 3
m_tot_3 = 4*m_LEO_ref_total_3 + m_LLO_ref_total + m_case_3;
LLO_mass = m_LLO_ref_total + m_case_3_LLO;
LEO_mass_3 = m_tot_3 - LLO_mass;

```

**References:**

- [1] N.A., “Starship Users Guide,” Space Exploration Technologies Corp, March 2020.
- [2] NASA, Main Shuttle External Tank.
- [3] “Aluminum Alloy,” MakeItFrom.com Available: <https://www.makeitfrom.com/material-properties/2195-2195-T8-Aluminum>.
- [4] Pietrobon, S. S., “Analysis of Propellant Tank Masses,” Small World Communications, Payneham South, SA , 2009 (unpublished).
- [5] Prasad, N. E., Gokhale, A. A., and Rao, P. R., “Mechanical behaviour of aluminium-lithium alloys,” Sadhana, vol. 28, 2003, pp. 209–246.
- [6] Street, D., “A Scalable Orbital Propellant Depot Design” Available: <http://www.ssdl.gatech.edu/sites/default/files/ssdl-files/papers/mastersProjects/StreetD-8900.pdf>.
- [7] “Falcon 9 Launch Vehicle, Payload User’s Guide,” SpaceX:<https://spaceflightnow.com/falcon9/001/f9guide.pdf>
- [8] G. Madhusudhan Reddy, Amol A. Gokhale, Chapter 9 - Welding Aspects of Aluminum-Lithium Alloys, Aluminum-lithium Alloys, Butterworth-Heinemann, 2014, Pages 259-302, ISBN 9780124016989
- [9] Dunbar, Brian. “Keeping Rocket Engine Fuel Lines Bubble Free in Space.” NASA, NASA, [www.nasa.gov/mission\\_pages/station/research/news/CCF.html#:~:text=When%20a%20spacecraft%20tank%20is,the%20center%20of%20the%20tank](http://www.nasa.gov/mission_pages/station/research/news/CCF.html#:~:text=When%20a%20spacecraft%20tank%20is,the%20center%20of%20the%20tank).
- [10] Heister, Stephen D., Anderson, William E., Pourpoint, Timothée L., Cassady, R. Joseph. Rocket Propulsion (Cambridge Aerospace Series) (p. 301). Cambridge University Press. Kindle Edition.
- [11] Julia Kozlowski's “Launch Mission Tool”

- [12] Vyas, Kashyap. “Cold Welding: Joining Metals Without Heat.” Interesting Engineering, Interesting Engineering, 25 Jan. 2021, [interestingengineering.com/cold-welding-joining-metals-without-heat](https://interestingengineering.com/cold-welding-joining-metals-without-heat)0created
- [13] Collins, Danielle. How to Size a Rack and Pinion Drive, Linear Motion Tips , 2 Jan. 2019, [www.linearmotiontips.com/how-to-size-a-rack-and-pinion-drive/](https://www.linearmotiontips.com/how-to-size-a-rack-and-pinion-drive/).
- [14] Mann, A., “SpaceX now dominates rocket flight, bringing big benefits-and risks-to NASA,” Science Available: <https://www.sciencemag.org/news/2020/05/spacex-now-dominates-rocket-flight-bringing-big-benefits-and-risks-nasa>
- [15] Mission Design “Target Launch Dates”
- [16] Noah Stockwell’s “Starship Cost Estimates”
- [17] Zagarola, M., “A High Efficiency Cryocooler for In-Space Cryogenic Propellant Storage,” SBIR Database, retrieved 12 March 2021.
- [18] Kerslake, T., Gustafson, E., “On-Orbit Performance Degradation of the International Space Station P6 Photovoltaic Arrays,” Glenn Research Center, Cleveland, Ohio, Jul. 2003.
- [19] Brown, R.L., and Stein, S.E., “Boiling Point Data ,” NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Eds. P.J. Linstrom and W.G. Mallard
- [20] Anderson, E. Z., Chintalapati, S., and Kirk, D. R., “Modeling of Ullage Collapse Within Rocket Propellant Tanks at Reduced Gravity,” *Journal of Spacecraft and Rockets*, vol. 51, 2014, pp. 1377–1389.
- [21] Kutter, B., Kutter, B., Sakla, S., Wall, J., Saks, G., Duffey, J., and Hopkins, J., “Settled Cryogenic Propellant Transfer,” *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2006.

[22] Dressler, G., Joseph, G., Behrens, H., Asato, D., and Carlson, R., “Compton Gamma Ray Observatory - Lessons learned in propulsion,” 37th Joint Propulsion Conference and Exhibit, 2001.

[23] Keller, J. W., Petro, N. E., and Vondrak, R. R., “The Lunar Reconnaissance Orbiter Mission – Six years of science and exploration at the Moon,” Icarus, vol. 273, 2016, pp. 2–24.

[24] “Hydrazine Propellant Tanks,” Hydrazine Propellant Tanks for Satellites and Spacecraft Available: <https://www.space-propulsion.com/spacecraft-propulsion/hydrazine-tanks/index.html#177>.

[25] MT Aerospace, Spacecraft Propellant Tanks Available: <https://www.mt-aerospace.de/tanks-produkte.html>.

[26] Titanium Ti-6Al-4V (Grade 5), STA Available: <http://www.matweb.com/search/DataSheet.aspx?MatGUID=b350a789eda946c6b86a3e4d3c577b39&ckck=1>.

[27] Theresa Setter’s “calc\_prop\_mass” function

[28] Trevor Pfeil’s “starship\_landing” function

[29] “Multi-Layer Insulation,” Meyer Tool & MFG, Oak Lawn, IL

[30] Plachta, D. W., Feller, J. R., Guzik, M. C., “In-Space Cryogenic Propellant Storage Applications for a 20 W at 20 K Cryocooler”, NASA Glenn Research Center, Cleveland, OH, NASA Ames Research Center, Moffett Field, CA

[31] The Aerospace Corporation. (1971, September). Final Report Orbiting Propellant Depot Safety Volume II: Technical Discussion. <https://ntrs.nasa.gov/api/citations/19730004172/downloads/19730004172.pdf>

[32] Kornuta, D., Abbud-Madrid, A., Atkinson, J., Barr, J., Barnhard, G., Bienhoff, D., Blair, B., Clark, V., Cyrus, J., DeWitt, B., Dreyer, C., Finger, B., Goff, J., Ho, K., Kelsey, L., Keravala, J.,

Kutter, B., Metzger, P., Montgomery, L., ... Zhu, G. (2019). Commercial lunar propellant architecture: A collaborative study of lunar propellant production. *REACH*, 13, [100026].  
<https://doi.org/10.1016/j.reach.2019.100026>

[33] <https://www.paragonsdc.com/>

[34] Jordan JL. Lunar Hydrogen and Other Volatiles. Huntsville: AIAA Space Programs and Technologies Conference; 1992. 7 p.

[35] <http://www.spaceagepub.com/pdfs/Begona.pdf>

[36]

[https://www.frontiersin.org/articles/10.3389/fenrg.2018.00005/full#:~:text=The%20cost%20estimation%20of%20the,kW%20\(Zapf%2C%202017\).](https://www.frontiersin.org/articles/10.3389/fenrg.2018.00005/full#:~:text=The%20cost%20estimation%20of%20the,kW%20(Zapf%2C%202017).)

[37] Anvari, A., Farhani, F., Niaki, K., “Comparative Study on Space Qualified Paints Used for Thermal Control of a Small Satellite,” Iranian Research Organization for Science and Technology, Department of Mechanical Engineering, Jan. 2009.

[38] Anderson, E. Z., Chintalapati, S., and Kirk, D. R., “Modeling of Ullage Collapse Within Rocket Propellant Tanks at Reduced Gravity,” *Journal of Spacecraft and Rockets*, vol. 51, 2014, pp. 1377–1389.

[39] Kutter, B., Kutter, B., Sakla, S., Wall, J., Saks, G., Duffey, J., and Hopkins, J., “Settled Cryogenic Propellant Transfer,” *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2006.

## **Communications**

## I. Data Rate Example Calculation

In this Appendix we provide an example of the method used in this design for choosing data rates for various links. All data rates chosen in this report were chosen by this method.

For illustration purposes, we will use the example of email being downlinked from the communications satellites to Earth.

First, it is necessary to determine the size of the data that has to be downlinked. The maximum email size according to Gmail and Outlook is 25 MB, or 160 Mb. Next, the sending rate must be determined. We determined a 5 second email downlink time would be optimal, so that meant 160 Mb had to be sent in 5 seconds:  $\frac{160 \text{ Mb}}{5 \text{ sec}} = 32 \text{ Mbps}$ .

As the reader can see, this process for data rate determination is quite simple. For other types of data, the same exact process was used, although perhaps with some modifications based on data type. For example, for telemetry data from satellites and transit vehicles, where the size and number of bits per sample is not readily defined as it is for email, we had to choose sample rate and number of bits per sample to achieve certain quantization error goals. For this process, we will use the example of transit vehicle telemetry to illustrate.

We determined that receiving 50 samples every 5 seconds for telemetry data would be best. This is not fast enough to be called real time but is very close. We also determined that in order to achieve a 0.01% quantization error, 12 bits were needed per sample. By multiplying 50 samples/second and 12 bits/sample, we obtained 120 bps for transit vehicle telemetry.

## II. Detailed Link Budget Calculation

As shown throughout this report, the link budget was used heavily to calculate a number of antenna specifications and properties for the multiple pathways of the communication network. A brief explanation of the algorithm was introduced earlier, but this section seeks to provide a detailed look at input parameters involved in the calculation and how we arrived at specific values. This section also reproduces a number of link budgets that were presented previously in the report, but with a number of additional parameters that were omitted earlier for being extraneous.

Recall that the link budget process was primarily adapted from Chapter 13.3.6 of *Space Mission Analysis and Design, 3<sup>rd</sup> Edition* (“SMAD”) [11] which outlines the necessary requirements for a downlink link budget. The SMAD process required the following as input parameters: frequency, transmit power, transmit line loss, transmit antenna diameter, transmit antenna pointing offset, propagation path length, receive antenna diameter, receive antenna pointing error, data rate, and the bit error rate. Four of these parameters were already discussed in detail earlier: transmit power, transmit antenna diameter, receive antenna diameter, and data rate – as they have the greatest effect on the system as a whole. Here we will describe the parameters that have not been discussed.

### 1. Frequency

One of the most obvious parameters for radio communication is the frequency being used to transmit and receive. Different choices of frequency bands are optimal for different purposes, and use of different bands has different implications depending on the band. For example, higher frequency bands enable more bandwidth than lower ones do. Additionally, it is important to choose bands for different purposes within a mission such that the probability of interference is very low.

For these and a variety of other reasons, choice of frequency is extremely important and must be carefully considered. In the table below, the frequency bands used in this mission are summarized.

**Table B.1 Frequencies of Communication Links**

| <b>Communication</b> |             | <b>Frequency</b> |  |
|----------------------|-------------|------------------|--|
| <b>Link</b>          | <b>Band</b> | <b>Frequency</b> |  |
| WSE-to-NEN           | S           | 2.2 GHz          |  |
| CHELL-to-WSS         | X           | 8.2 GHz          |  |
| CHELL-to-NEN         | Ka          | 27.5 GHz         |  |
| Rover-to-CHELL       | S           | 2.2 GHz          |  |
| FISTO-to-CHELL       | S           | 2.2 GHz          |  |

## *2. Transmit Line Loss, Transmit Pointing Offset, and Receive Antenna Pointing Error*

In accordance with the algorithm outlined by SMAD, the transmitter line loss was assumed to be  $-1\text{ dB}$  for all cases. Moreover, for the two pointing offsets of the receive and transmit antennas, it was assumed that these values would be low enough as to not cause significant error. For the five pathways, on a case-by-case basis, the pointing errors were either assumed to be  $0.1^\circ$  or to be half of the half-power-beamwidth of the antenna in question – whichever was greater.

## *3. Propagation Path Length*

The propagation path length is essentially the distance that the signal needs to travel from the transmitter to the receiving antenna. This path length then directly affects the space loss term which

is a greater loss for greater distances. Thus, for longer distances, naturally, a greater transmitter power is required to effectively carry the signal to such a distance.

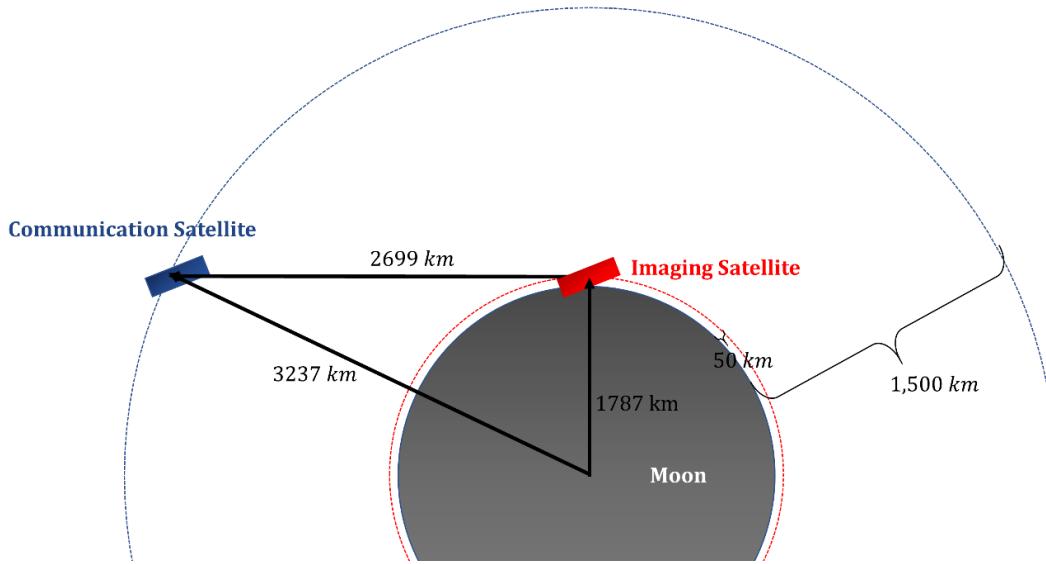
As mentioned in earlier sections, the CHELL communication satellites orbit above the lunar surface at an altitude of 1,500 km. Thus, for the CHELL-to-WSS and Rover-to-CHELL communication paths, the path length is then assumed to be 1,500 km. For the CHELL-to-WSS connection, this assumes that the satellite is directly above the WSS station and ignores any changes with elevation angle. Moreover, for the rovers, this also assumes that the rovers are directly below the satellites at the lunar base. For both cases, even if the satellite is not directly above the station or if the rovers are far away from the lunar base – the effect of additional space loss is not a significant loss for the overall link budget. Also, since there is no atmospheric attenuation effects, any additional loss from passing through the lunar “atmosphere” is negligible as well.

Next, for the two paths that connect the Moon to the NEN, we have used the average distance between the Earth and the Moon – and has again, neglected any changes in path length due to elevation angles or non-direct-boresight pointing. Since the distance from the Earth to the Moon is on a larger magnitude to any changes in distance due to the aforementioned reasons, a constant path length assumption is valid. For the WSE-to-NEN connection, this average path length is used: 382,500 *km*. For the CHELL-to-NEN connection, the altitude of the satellites above the Moon is subtracted to obtain a total distance of 381,000 *km*.

Lastly, for the path between FISTO and CHELL, a reasonable approximation had to be made for a maximum distance between the two satellites at any given time. Since the satellite should be designed to have continual crosslinking ability, the other parameters needed to be designed such

that it could operate at the distance when the two satellites are furthest apart. Shown below in Fig.

B.1 is a diagram that outlines the calculation of this value.



**Fig. B.1 Crosslink Propagation Path Calculation**

Note that the calculation above also assumes the distances between orbital planes in this calculation is negligible for simplicity. Then, knowing the altitudes of both satellites, the distance between the satellites can be found. In this diagram, the furthest distance the satellites would be apart is as shown on the diagram. If the shown communication satellite would be below the horizontal line connecting the two, by the nature of the constellation, there would be another communication satellite in range. Thus, by simply computing,

$$S = \sqrt{r_{comm}^2 - r_{imag}^2}$$

the propagation path length,  $S$ , can be found. Hence, comprising the above, the following Table B.2 displays a list of all the propagation path lengths involved:

**Table B.2 Path Lengths of Communication Links**

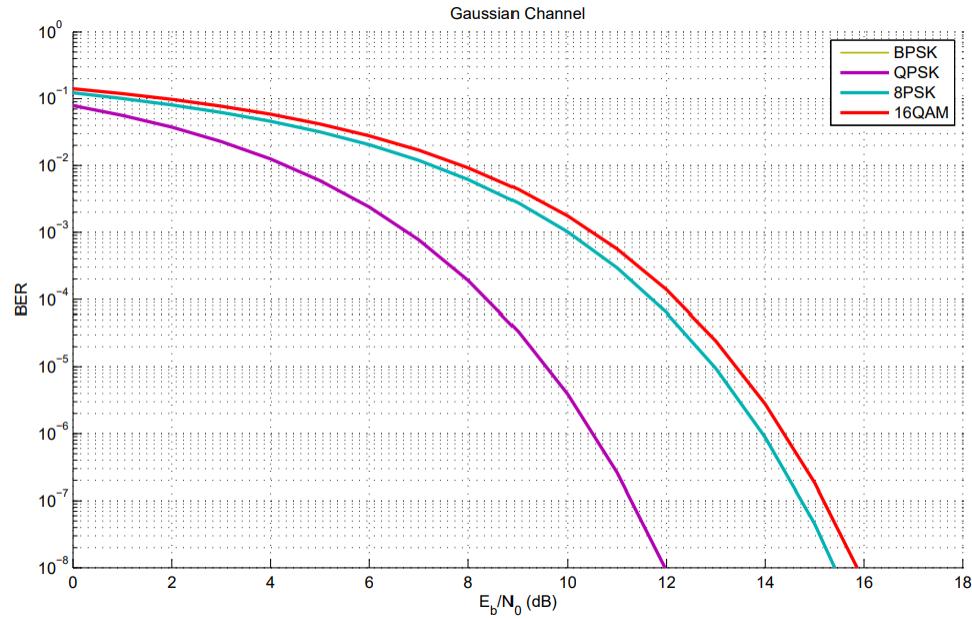
| Communication Path    | Propagation Path Length |
|-----------------------|-------------------------|
| <b>WSE-to-NEN</b>     | 382,500 km              |
| <b>CHELL-to-WSS</b>   | 1,500 km                |
| <b>CHELL-to-NEN</b>   | 381,000 km              |
| <b>Rover-to-CHELL</b> | 1,500 km                |
| <b>FISTO-to-CHELL</b> | 2,699 km                |

#### 4. Modulation Type and Bit Error Rate

The BER is a measure of the quality of information that is downlinked from the transmitter. In particular, this parameter influences the calculation of the Required  $E_b/N_0$  (energy-per-bit to noise spectral density ratio). All of which in turn directly affects the tolerances of the link margin. Thus, here, we will assume a modulation scheme and an appropriate BER in accordance. For the modulation type, residual carrier binary phase-shift keying (BPSK) using a squarewave subcarrier is the modulation scheme that has been most employed for deep space telemetry [3] and is one that can interface with the NEN stations. Thus, this modulation type was selected and the relevant transmitting and receiving antennas are to be designed to be able to interface with BPSK modulation.

Next, a selected BER is dependent on the accuracy that the user decides, so this is not a value that can be necessarily calculated. Thus, we will assume a conservative estimate with a BER of  $10^{-6}$ . For reference, internet satellites, requiring a higher need for accuracy, usually has a BER of  $10^{-9}$ . With this information, we are now able to get an estimate at the required  $E_b/N_0$  using Fig.

B.2. below. Note that in the figure, BPSK and QPSK modulation are identical, so the purple line is used. In the case of the CHELL-to-WSS pathway, QPSK modulation was used so despite that, the value of required  $E_b/N_0$  will be the same for all five pathways.



**Fig. B.2 BER vs.  $E_b/N_0$  – BPSK/QPSK Modulation**

Thus, after plotting a BER of  $10^{-6}$ , we find a required  $E_b/N_0$  of approximately 10.5 dB.

### 5. Link Budgets

The following subsection reproduces link budgets seen earlier in the report in greater detail:

**Table B.3 Link Budget – WSE-to-NEN Pathway**

| Item                         | Symbol | Unit | Value | Source         |
|------------------------------|--------|------|-------|----------------|
| <b>Frequency</b>             | $f$    | GHz  | 2.200 | NEN User Guide |
| <b>Transmitter Power</b>     | $P$    | W    | 348   | Sec. II.B–C    |
| <b>Transmitter Power</b>     | $P$    | dBW  | 25.41 | $10 \log P$    |
| <b>Transmitter Line Loss</b> | $L_l$  | dB   | -1.0  | NEN User Guide |

|                                            |                |       |           |                                                                                       |
|--------------------------------------------|----------------|-------|-----------|---------------------------------------------------------------------------------------|
| <b>Peak Transmit Antenna Gain</b>          | $G_{pt}$       | dBi   | 42.29     | $G_{pt} \approx 44.3 - 10\log(\theta_t^2)$                                            |
| <b>Transmit Antenna Beamwidth</b>          | $\theta_t$     | deg   | 1.26      | Sec. II.B-C                                                                           |
| <b>Transmit Antenna Diameter</b>           | $D_t$          | m     | 7.57      | $D_t = 21/f_{GHz}\theta_t$                                                            |
| <b>Transmit Antenna Pointing Offset</b>    | $e_t$          | deg   | 0.50      | NEN User Guide                                                                        |
| <b>Transmit Antenna Pointing Loss</b>      | $L_{pt}$       | dB    | -1.89     | $L_{pt} = -12(e_t/\theta_t)^2$                                                        |
| <b>Transmit Antenna Gain (net)</b>         | $G_t$          | dBi   | 40.40     | $G_{pt} + L_{pt}$                                                                     |
| <b>Peak Receive Antenna Gain</b>           | $G_{rp}$       | dBi   | 49.91     | $G_{rp} = -162.18 + 20\log(D_r)$<br>+ 20 log(f)                                       |
| <b>Receive Antenna Beamwidth</b>           | $\theta_r$     | deg   | 0.52      | $\theta_r = 21/f_{GHz}D_r$                                                            |
| <b>Receive Antenna Diameter</b>            | $D_r$          | m     | 18.3      | [11]                                                                                  |
| <b>Receive Antenna Pointing Error</b>      | $e_r$          | deg   | 0.1       | -                                                                                     |
| <b>Receive Antenna Pointing Loss</b>       | $L_{pr}$       | dB    | -0.44     | $L_{pr} = -12(e_r/\theta_r)^2$                                                        |
| <b>Receive Antenna Gain (net)</b>          | $G_r$          | dBi   | 49.47     | $G_{rp} + L_{pr}$                                                                     |
| <b>Equiv. Isotropic Radiated Power</b>     | $EIRP$         | dBW   | 64.99     | $P + L_l + G_t$                                                                       |
| <b>Propagation Path Length</b>             | $S$            | km    | 382,500   | [11]                                                                                  |
| <b>Space Loss</b>                          | $L_s$          | dB    | -210.95   | $147.55 - 20\log(S) - 20\log(f)$                                                      |
| <b>Propagation &amp; Polarization Loss</b> | $L_a$          | dB    | -0.3      | [11]                                                                                  |
| <b>Receiver Gain-to-Noise-Temperature</b>  | $G/T$          | dB/K  | 29.6      | [11]                                                                                  |
| <b>System Noise Temperature</b>            | $T_s$          | K     | 97.05     | $T_s = 10^{\frac{(G_r - G/T)}{10}}$                                                   |
| <b>Data Rate</b>                           | $R$            | Mbps  | 100.0     | Sec. II.A                                                                             |
| <b><math>E_b/N_0</math></b>                | $E_b/N_0$      | dB    | 31.51     | $E_b/N_0 = EIRP + L_{pr} + L_s + L_a + G_r$<br>+ 228.6<br>- 10 log $T_s$ - 10 log $R$ |
| <b>Carrier-to-Noise Density Ratio</b>      | $C/N_0$        | dB-Hz | 111.51    | $C/N_0 = E_b/N_0 + 10\log R$                                                          |
| <b>Bit Error Rate</b>                      | $BER$          | -     | $10^{-6}$ | [11]                                                                                  |
| <b>Required <math>E_b/N_0</math></b>       | Req. $E_b/N_0$ | dB    | 10.5      | [11]                                                                                  |

|                                          |   |    |       |                                                      |
|------------------------------------------|---|----|-------|------------------------------------------------------|
| <b>Implementation Loss</b>               | — | dB | −2    | Estimate                                             |
| <b>Link Margin (dry) (1)</b>             | — | dB | 19.01 | $E_b/N_O - \text{Req. } E_b/N_O + \text{Impl. Loss}$ |
| <b>Link Margin (w/ rain attenuation)</b> | — | dB | 16.01 | (1) − 3 dB                                           |

**Table B.4 Link Budget – CHELL-to-WSS Pathway**

| Item                             | Symbol     | Unit | Value   |
|----------------------------------|------------|------|---------|
| Frequency                        | $f$        | GHz  | 8.20    |
| Transmitter Power                | $P$        | W    | 20.0    |
| Transmitter Power                | $P$        | dBW  | 13.01   |
| Transmitter Line Loss            | $L_l$      | dB   | -4.5    |
| Peak Transmit Antenna Gain       | $G_{pt}$   | dBi  | 16.0    |
| Transmit Antenna Beamwidth       | $\theta_t$ | deg  | 28.45   |
| Transmit Antenna Diameter        | $D_t$      | m    | 0.09    |
| Transmit Antenna Pointing Offset | $e_t$      | deg  | 14.22   |
| Transmit Antenna Pointing Loss   | $L_{pt}$   | dB   | -3.0    |
| Transmit Antenna Gain (net)      | $G_t$      | dBi  | 13.0    |
| Peak Receive Antenna Gain        | $G_{rp}$   | dBi  | 56.46   |
| Receive Antenna Beamwidth        | $\theta_r$ | deg  | 0.26    |
| Receive Antenna Diameter         | $D_r$      | m    | 10.0    |
| Receive Antenna Pointing Error   | $e_r$      | deg  | 0.13    |
| Receive Antenna Pointing Loss    | $L_{pr}$   | dB   | -3.0    |
| Receive Antenna Gain (net)       | $G_r$      | dBi  | 53.46   |
| Equiv. Isotropic Radiated Power  | $EIRP$     | dBW  | 21.51   |
| Propagation Path Length          | $S$        | km   | 1,500   |
| Space Loss                       | $L_s$      | dB   | -174.23 |

|                                    |                |       |           |
|------------------------------------|----------------|-------|-----------|
| Propagation & Polarization Loss    | $L_a$          | dB    | -0.3      |
| Receiver Gain-to-Noise-Temperature | $G/T$          | dB/K  | -         |
| System Noise Temperature           | $T_s$          | K     | 135.0     |
| Data Rate                          | $R$            | Mbps  | 100       |
| $E_b/N_0$                          | $E_b/N_0$      | dB    | 25.03     |
| Carrier-to-Noise Density Ratio     | $C/N_O$        | dB-Hz | 33.0      |
| Bit Error Rate                     | $BER$          | -     | $10^{-6}$ |
| Required $E_b/N_0$                 | Req. $E_b/N_0$ | dB    | 10.5      |
| Implementation Loss                | -              | dB    | -2        |
| Link Margin (dry)                  | -              | dB    | 12.03     |
| Link Margin (w/ rain attenuation)  | -              | dB    | N/A       |

*Note: All sources are the same as Table A2.3*

**Table B.5 Link Budget – CHELL-to-NEN Pathway**

| Item                             | Symbol     | Unit | Value   |
|----------------------------------|------------|------|---------|
| Frequency                        | $f$        | GHz  | 2.200   |
| Transmitter Power                | $P$        | W    | 444.0   |
| Transmitter Power                | $P$        | dBW  | 26.47   |
| Transmitter Line Loss            | $L_l$      | dB   | -1.0    |
| Peak Transmit Antenna Gain       | $G_{pt}$   | dBi  | 46.38   |
| Transmit Antenna Beamwidth       | $\theta_t$ | deg  | 0.79    |
| Transmit Antenna Diameter        | $D_t$      | m    | 0.97    |
| Transmit Antenna Pointing Offset | $e_t$      | deg  | 0.50    |
| Transmit Antenna Pointing Loss   | $L_{pt}$   | dB   | -4.84   |
| Transmit Antenna Gain (net)      | $G_t$      | dBi  | 41.54   |
| Peak Receive Antenna Gain        | $G_{rp}$   | dBi  | 70.50   |
| Receive Antenna Beamwidth        | $\theta_r$ | deg  | 3.81    |
| Receive Antenna Diameter         | $D_r$      | m    | 18.3    |
| Receive Antenna Pointing Error   | $e_r$      | deg  | 0.1     |
| Receive Antenna Pointing Loss    | $L_{pr}$   | dB   | -0.01   |
| Receive Antenna Gain (net)       | $G_r$      | dBi  | 70.49   |
| Equiv. Isotropic Radiated Power  | $EIRP$     | dBW  | 67.01   |
| Propagation Path Length          | $S$        | km   | 381,000 |
| Space Loss                       | $L_s$      | dB   | -232.85 |

|                                    |                |       |           |
|------------------------------------|----------------|-------|-----------|
| Propagation & Polarization Loss    | $L_a$          | dB    | -0.3      |
| Receiver Gain-to-Noise-Temperature | $G/T$          | dB/K  | 29.6      |
| System Noise Temperature           | $T_s$          | K     | 424.0     |
| Data Rate                          | $R$            | Mbps  | 32.0      |
| $E_b/N_0$                          | $E_b/N_0$      | dB    | 31.62     |
| Carrier-to-Noise Density Ratio     | $C/N_0$        | dB-Hz | 106.67    |
| Bit Error Rate                     | $BER$          | -     | $10^{-6}$ |
| Required $E_b/N_0$                 | Req. $E_b/N_0$ | dB    | 10.5      |
| Implementation Loss                | -              | dB    | -2        |
| Link Margin (dry) (1)              | -              | dB    | 19.12     |
| Link Margin (w/ rain attenuation)  | -              | dB    | 13.12     |

*Note: All sources are the same as Table A2.3*

**Table B.6 Link Budget – Rover-to-CHELL Pathway**

| Item              | Symbol | Unit | Value |
|-------------------|--------|------|-------|
| Frequency         | $f$    | GHz  | 2.200 |
| Transmitter Power | $P$    | W    | 35.0  |

|                                    |            |      |         |
|------------------------------------|------------|------|---------|
| Transmitter Power                  | $P$        | dBW  | 15.44   |
| Transmitter Line Loss              | $L_l$      | dB   | -1.0    |
| Peak Transmit Antenna Gain         | $G_{pt}$   | dBi  | 25.53   |
| Transmit Antenna Beamwidth         | $\theta_t$ | deg  | 8.68    |
| Transmit Antenna Diameter          | $D_t$      | m    | 1.10    |
| Transmit Antenna Pointing Offset   | $e_t$      | deg  | 0.50    |
| Transmit Antenna Pointing Loss     | $L_{pt}$   | dB   | -0.04   |
| Transmit Antenna Gain (net)        | $G_t$      | dBi  | 25.49   |
| Peak Receive Antenna Gain          | $G_{rp}$   | dBi  | 3.74    |
| Receive Antenna Beamwidth          | $\theta_r$ | deg  | 106.06  |
| Receive Antenna Diameter           | $D_r$      | m    | 0.09    |
| Receive Antenna Pointing Error     | $e_r$      | deg  | 0.1     |
| Receive Antenna Pointing Loss      | $L_{pr}$   | dB   | 0.0     |
| Receive Antenna Gain (net)         | $G_r$      | dBi  | 3.74    |
| Equiv. Isotropic Radiated Power    | $EIRP$     | dBW  | 39.93   |
| Propagation Path Length            | $S$        | km   | 1,500   |
| Space Loss                         | $L_s$      | dB   | -162.82 |
| Propagation & Polarization Loss    | $L_a$      | dB   | -0.3    |
| Receiver Gain-to-Noise-Temperature | $G/T$      | dB/K | -       |
| System Noise Temperature           | $T_s$      | K    | 135     |
| Data Rate                          | $R$        | Mbps | 10.83   |
| $E_b/N_0$                          | $E_b/N_0$  | dB   | 17.51   |

|                                   |                |       |           |
|-----------------------------------|----------------|-------|-----------|
| Carrier-to-Noise Density Ratio    | $C/N_O$        | dB-Hz | 87.85     |
| Bit Error Rate                    | $BER$          | —     | $10^{-6}$ |
| Required $E_b/N_O$                | Req. $E_b/N_O$ | dB    | 10.5      |
| Implementation Loss               | —              | dB    | -2        |
| Link Margin (dry) (1)             | —              | dB    | 5.01      |
| Link Margin (w/ rain attenuation) | —              | dB    | N/A       |

*Note: All sources are the same as Table A2.3*

**Table B.7 Link Budget – FISTO-to-CHELL Pathway**

| Item                             | Symbol     | Unit | Value   |
|----------------------------------|------------|------|---------|
| Frequency                        | $f$        | GHz  | 2.200   |
| Transmitter Power                | $P$        | W    | 30.0    |
| Transmitter Power                | $P$        | dBW  | 14.77   |
| Transmitter Line Loss            | $L_l$      | dB   | -1.0    |
| Peak Transmit Antenna Gain       | $G_{pt}$   | dBi  | 4.30    |
| Transmit Antenna Beamwidth       | $\theta_t$ | deg  | 100.0   |
| Transmit Antenna Diameter        | $D_t$      | m    | 0.10    |
| Transmit Antenna Pointing Offset | $e_t$      | deg  | 0.50    |
| Transmit Antenna Pointing Loss   | $L_{pt}$   | dB   | -0.0    |
| Transmit Antenna Gain (net)      | $G_t$      | dBi  | 4.30    |
| Peak Receive Antenna Gain        | $G_{rp}$   | dBi  | 4.225   |
| Receive Antenna Beamwidth        | $\theta_r$ | deg  | 100.0   |
| Receive Antenna Diameter         | $D_r$      | m    | 0.10    |
| Receive Antenna Pointing Error   | $e_r$      | deg  | 0.10    |
| Receive Antenna Pointing Loss    | $L_{pr}$   | dB   | -0.0    |
| Receive Antenna Gain (net)       | $G_r$      | dBi  | 4.225   |
| Equiv. Isotropic Radiated Power  | $EIRP$     | dBW  | 18.07   |
| Propagation Path Length          | $S$        | km   | 2,699   |
| Space Loss                       | $L_s$      | dB   | -167.51 |

|                                    |                |       |           |
|------------------------------------|----------------|-------|-----------|
| Propagation & Polarization Loss    | $L_a$          | dB    | -0.3      |
| Receiver Gain-to-Noise-Temperature | $G/T$          | dB/K  | -         |
| System Noise Temperature           | $T_s$          | K     | 135       |
| Data Rate                          | $R$            | Mbps  | 1.1       |
| $E_b/N_0$                          | $E_b/N_0$      | dB    | 18.68     |
| Carrier-to-Noise Density Ratio     | $C/N_0$        | dB-Hz | 95.31     |
| Bit Error Rate                     | $BER$          | -     | $10^{-6}$ |
| Required $E_b/N_0$                 | Req. $E_b/N_0$ | dB    | 10.5      |
| Implementation Loss                | -              | dB    | -2        |
| Link Margin (dry) (1)              | -              | dB    | 6.29      |
| Link Margin (w/ rain attenuation)  | -              | dB    | -         |

*Note: All sources are the same as Table A2.3*

### III.Risk Matrix

**Table C.1 Communications Infrastructure Risk Matrix**

| Ref ID | Intermediate Event                                                | Likelihood | Consequence | Ref ID  | Basic Event                                          | Likelihood | Consequence | Overall | Post-Mitigation                                                         |             |
|--------|-------------------------------------------------------------------|------------|-------------|---------|------------------------------------------------------|------------|-------------|---------|-------------------------------------------------------------------------|-------------|
|        |                                                                   |            |             |         |                                                      |            |             |         | Likelihood                                                              | Consequence |
| CM-1   | No contact with Earth from Moon ground                            | 1          | 4           | OCM-1.1 | Wheatley Station Earth (WSE) transmitter is down     | 1          | 3           | 3       | Accessible placement for easy repair, redundant transmitter if possible | 1 2         |
|        |                                                                   |            |             | OCM-1.2 | WSE antenna is down                                  | 1          | 3           | 3       | Accessible placement for east repair                                    | 1 2         |
|        |                                                                   |            |             |         |                                                      |            |             | 0       |                                                                         |             |
| CM-2   | No contact with CHELL constellation                               | 1          | 3           | OCM-2.1 | Wheatley Station Satellite (WSS) transmitter is down | 1          | 3           | 3       | Accessible placement for easy repair, redundant transmitter if possible | 1 2         |
|        |                                                                   |            |             | OCM-2.2 | WSS antenna is down                                  | 1          | 3           | 3       | Accessible placement for east repair                                    | 1 2         |
|        |                                                                   |            |             |         |                                                      |            |             |         |                                                                         |             |
| CM-3   | Images not successfully crosslinked between Imaging Sat and CHELL | 2          | 2           | OCM-3.1 | Imaging sat transmitter broken                       | 2          | 2           | 4       | commands and telemetry                                                  | 1 2         |
|        |                                                                   |            |             | OCM-3.2 | CHELL receiver broken                                | 2          | 2           | 4       | CHELL can receive in S and X bands                                      | 1 2         |
|        |                                                                   |            |             | OCM-3.3 | imaging sat antenna broken                           | 2          | 2           | 4       | downlink images to moon ground instead                                  | 1 2         |
|        |                                                                   |            |             | OCM-3.4 | CHELL antenna broken                                 | 2          | 2           | 4       | uplink images from moon ground instead                                  | 1 2         |
| CM-4   | Reduced DMR Connectivity                                          | 4          | 3           | OCM-4.1 | handheld radio is broken or dead                     | 1          | 2           | 2       | easy repair                                                             | 1 1         |
|        |                                                                   |            |             | OCM-4.2 | Rover radio is broken or dead                        | 2          | 2           | 4       | easy repair                                                             | 1 1         |
|        |                                                                   |            |             | OCM-4.3 | Repeater unit(s) broken or dead                      | 2          | 3           | 6       | easy repair                                                             | 1 1         |
|        |                                                                   |            |             | OCM-4.4 | Base Station down                                    | 1          | 4           | 4       | easy repair                                                             | 1 1         |
| CM-5   | Reduced Wi-Fi Connectivity                                        | 4          | 3           | OCM-5.1 | Access Point broken or dead                          | 3          | 2           | 6       | very easy repair                                                        | 2 1         |
|        |                                                                   |            |             | OCM-5.2 | worn cable at some point in the network              | 3          | 2           | 6       | easy repair and/or replacement                                          | 2 1         |
|        |                                                                   |            |             | OCM-5.3 | base station down                                    | 1          | 4           | 4       | easy repair                                                             | 1 2         |

#### IV. MATLAB Code

```

%% Link Budget Calculation
% The following program utilizes equations obtained in the downlink link
% budget design as set up in Chapter 13 of "Space Mission Analysis and
% Design."
%
% NOTE: MAKE SURE YOU CLICK 'RUN SECTION' IF CONDUCTING A MULTI-VALUED
% CALCULATION OR A SINGLE-VALUED CALCULATION. DON'T RUN BOTH.
%
% Author: Alec Leven

%%% *** Multi-Valued Calculation ****
% Use this for when you have two parameters of interest and want to see how
% link margin changes with varying ranges of both (e.g. Transmitter Power
% and Transmitter Antenna Diameter).

% Output will be two charts that plots the two ranges with link margin
% denoted by a different color (red/dark red implies a higher value link
% margin and thus a sturdier and more optimal link; blue implies worse
% link). Note that this link margin accounts for worst-case rain
% conditions.

% The second output will also be a contour plot of the same data in 2D
% space. For this plot, by moving the user's cursor over different
% "power-diameter" coordinate pairs within the same color contour-band, the
% link margin can be maintained constant with the power and diameter
% changing (the link margin should be maintained constant because we do not
% want the quality of the link to drop when power and diameter changes).
% When adjusting the power and diameter, a good measure when eyeballing an
% estimate is to look at points in the "left-half" of the parabolic shape
% that is formed (to try to both minimize the antenna diameter and the
% antenna's power).

% For things noted with "Standard Constant from SMAD" -- do not change this value.
% The parameters that will likely need to be changed are: frequency, path
% length, data rate, rain attenuation level (if frequency above 8 GHz),
% and required Eb/N0 (if data modulation is changed)
% ****

```

close all; clc; clear;

```

% Input Parameters
frequency = 2.2;          % Frequency (GHz) -- Single frequency value selected from either S-Band: 2-4
GHz, Ka-Band: 27-40 GHz (or other freq. band)
%trans_power = 500;        % Transmitter Power (W) ** -- Parameter of Interest, keep commented when
parameter of interest
trans_line_loss = -1;       % Transmitter Line Loss (dB) -- Standard Constant from SMAD

```

```
%trans_diameter = .1; % Transmitter Diameter (m) ** -- Parameter of Interest, keep commented
when parameter of interest
trans_point_offset = 0.5; % Transmitter Pointing Offset (deg) -- General assumption for value, constant
can be researched for more specific value
path_length = 382500; % Path Length (km) (Average Distance to Moon) -- Standard Constant
(Moon-Earth: 382500 km, Satellite-Earth: 381000 km)
rec_diameter = 13; % Receiver Diameter (m) -- Obtained from White Sands NEN Station User
Manual
rec_point_error = 0.1; % Receiver Pointing Error (deg) -- General assumption for value, constant
data_rate = 100 * 10^6; % Data Rate (bps) -- Standard constant from either: Main Comm Station:
100 Mbit/s, Satellite: 32 Mbit/s, or Vehicle: 64 kbit/s
BER = 10^-6; % Bit Error Rate (dimensionless) -- Standard Constant from SMAD
prop_pol_loss = -0.3; % Propagation and Polarization Loss (dB) -- Standard Constant from SMAD
rain_val = 0; % Rain Degradation/Attenuation Level (dB) -- Value from SMAD Fig. 13-11
(p.565) (Use frequency used above and assume Elevation Angle 45° for frequencies greater than 8 GHz)
implement_loss = -2; % Implementation Loss Estimate (dB) -- Standard Constant from SMAD
Req_Eb_No = 10.5; % Required Eb/N0 Limit (dB) -- Given 10^-6 BER for BPSK Modulation,
dependant on data modulation choice (choices: BPSK, QPSK, OQPSK, etc.) -- (obtained from BER vs.
Eb/N0 for selected modulation type)
eta = 0.55; % Antenna Efficiency -- Assume: 0.55 for general parabolic antenna
c = 299792458; % Speed of Light (m/s) -- Standard Constant
G_T = 29.6; % G/T (dB/K) (White Sands NEN Station) -- Obtained from White Sands NEN
Station User Manual

% Calculations
points = 50; %Number of points to sample data at
trans_power = linspace(50,500,points); % Range of transmitter power values (in W) - Adjust with this
trans_diameter = linspace(1,14,points); % Range of transmitter diameter values (in m) - and with this

for i = 1:points
    for j = 1:points
        trans_beamwidth = 21 / (frequency * trans_diameter(j));
        peak_trans_gain = 44.3 - 10 * log10(trans_beamwidth^2); % assuming circular beamwidth area
        trans_point_loss = -12 * (trans_point_offset / trans_beamwidth)^2;
        trans_gain = peak_trans_gain + trans_point_loss;
        wavelength = c / (frequency * 10^9);
        space_loss = 20*log10(c) - 20*log10(4*pi) - 20*log10(path_length * 1000) - 20*log10(frequency *
        10^9);

        peak_rec_gain = 20*log10(pi) - 20*log10(c) + 20*log10(rec_diameter) + 20*log10(frequency *
        10^9) + 10*log10(eta);
        rec_beamwidth = 21 / (frequency * rec_diameter);
        rec_point_loss = -12 * (rec_point_error / rec_beamwidth).^2;
        rec_gain = peak_rec_gain + rec_point_loss;
        %rec_gain = 70.5 %Use direct rec_gain for calculation with White Sands NEN
        trans_power_new = 10 * log10(trans_power(i));
        EIRP = trans_power_new + trans_line_loss + trans_gain;

        sys_noise_temp = 10^((rec_gain-G_T)/10);
    end
end
```

```
%sys_noise_temp = 135; %Use direct sys_noise_temp for calculation to Moon or to satellite (S-band)

Eb_No = EIRP + rec_point_loss + space_loss + prop_pol_loss + rec_gain + 228.6 -
10*log10(sys_noise_temp) - 10*log10(data_rate);
C_No = Eb_No + 10*log10(data_rate);

margin = Eb_No - Req_Eb_No + implement_loss;
rain_val_safe = rain_val + 3; % creates a safe 3dB tolerance above the required rain
attenuation limit (link margin, dry conditoins)
margin_diff(i,j) = margin - rain_val_safe; % dB difference of the margin level above the safe rain
attenuation limit (link margin, rainy conditions)

bt = 2 * 10^6; %bandwidth
SNR = Eb_No * (data_rate / bt);
end
end

% Plotting
figure(1) %3D Plot
surf(trans_power, trans_diameter,margin_diff);
surf(trans_power, trans_diameter,margin_diff);
colormap('jet')
hcb = colorbar
grid on;
xlabel("Transmit Power (W)");
ylabel_new = ylabel("Transmit Diameter (m)");
zlabel("Link Margin (dB)");
colorTitleHandle = get(hcb,'Title');
titleString = 'Link Margin (dB)';
set(colorTitleHandle , 'String', titleString);
title('Link Margin Over Transmitter Parameters');
% ylabel_new.Position(2) = ylabel_new.Position(2) - 1.3;
% ylabel_new.Position(1) = ylabel_new.Position(1) - 5.3;
set(gca,'color','none')

figure(2) % 2D Contour Plot
contourf(trans_diameter, trans_power,margin_diff,7);
colormap('jet')
hcb = colorbar
grid on;
xlabel("Transmitter Diameter (m)");
ylabel_new = ylabel("Transmitter Power (W)");
zlabel("Link Margin (dB)");
colorTitleHandle = get(hcb,'Title');
titleString = 'Link Margin (dB)';
set(colorTitleHandle , 'String', titleString);
title('Link Margin for Transmitter Parameters');
```

```

%% *** Calculations for Single Values ****
% Use this for when you want to just see the link margin for the given
% values below (only one set of input parameters). For example, you have
% just found the power-diameter value you want to use - enter those values
% here and press "run section"
%
% Output will be a printed list of all relevant parameters that are
% calculated and involved.
% ****
%
```

% New Input Parameters (Obtained from Section 1: Multi-Valued Calculation)

```

trans_power = 348;          % Transmitter Power (W)
trans_diameter = 7.5;       % Transmitter Beamwidth (deg)
```

% Input Parameters (exact same as before in Section 1)

```

% frequency = 2.1;          % Frequency (GHz)
% trans_line_loss = -1;      % Transmitter Line Loss (dB)
% trans_point_offset = 0.5;   % Transmitter Pointing Offset (deg)
% path_length = 381000;       % Path Length (km) (Average Distance to Moon)
% rec_diameter = .1;         % Receiver Diameter (m) (White Sands NEN)
% rec_point_error = 0.1;      % Receiver Pointing Error (deg)
% data_rate = 1.1 * 10^6;     % Data Rate (bps) Main: 100 Mbit/s Satellite: 32 Mbit/s Vehicle: 64 kbit/s
% BER = 10^-6;               % Bit Error Rate (dimensionless)
% prop_pol_loss = -0.3;       % Propagation and Polarization Loss (dB)
% rain_val = 0;               % Rain Degradation Level (dB)
% implement_loss = -2;        % Implementation Loss Estimate (dB); Constant
% Req_Eb_No = 10.5;           % Required Eb/N0 Limit (dB) (Given 10^-6 BER for BPSK Modulation)
% eta = 0.55;                 % Antenna Efficiency (assume 0.55 for parabolic antenna)
% c = 299792458;              % Speed of Light (m/s)
% G_T = 29.6;                 % G/T (dB/K) (White Sands)
```

% Calculations

```

trans_beamwidth = 21 / (frequency * trans_diameter);
peak_trans_gain = 44.3 - 10 * log10(trans_beamwidth^2); % assuming circular beamwidth area
trans_point_loss = -12 * (trans_point_offset / trans_beamwidth)^2;
trans_gain = peak_trans_gain + trans_point_loss;
wavelength = c / (frequency * 10^9);
space_loss = 20*log10(c) - 20*log10(4*pi) - 20*log10(path_length * 1000) - 20*log10(frequency *
10^9);

peak_rec_gain = 20*log10(pi) - 20*log10(c) + 20*log10(rec_diameter) + 20*log10(frequency * 10^9) +
10*log10(eta);
rec_beamwidth = 21 / (frequency * rec_diameter);
rec_point_loss = -12 * (rec_point_error / rec_beamwidth)^2;
rec_gain = peak_rec_gain + rec_point_loss;
%rec_point_loss = -1;
%rec_gain = 70.5;
trans_power_new = 10 * log10(trans_power);
EIRP = trans_power_new + trans_line_loss + trans_gain;
```

```
% System Noise Temperature based on Table 13-10 in SMAD
% if (2 < frequency <= 13)
%   sys_noise_temp = 135;
% elseif (frequency < 20)
%   sys_noise_temp = 424;
% end
sys_noise_temp = 10^((rec_gain-G_T)/10)
% sys_noise_temp = 135;

Eb_No = EIRP + rec_point_loss + space_loss + prop_pol_loss + rec_gain + 228.6 -
10*log10(sys_noise_temp) - 10*log10(data_rate);
C_No = Eb_No + 10*log10(data_rate);

margin = Eb_No - Req_Eb_No + implement_loss;
rain_val_safe = rain_val + 3;           % creates a safe 3dB tolerance above the required rain attenuation
limit (link margin, dry conditoins)
margin_diff = margin - rain_val_safe; % dB difference of the margin level above the safe rain
attenuation limit (link margin, rainy conditions)

bt = 2 * 10^6; %bandwidth
SNR = Eb_No * (data_rate / bt);

%Output
fprintf("Frequency:      %8.3f GHz\n",frequency);
fprintf("Transmitter Power:    %8.3f W\n",trans_power);
fprintf("Transmitter Power:    %8.3f dBW\n",trans_power_new);
fprintf("Transmitter Line Loss    %8.3f dB\n",trans_line_loss);
fprintf("-----\n")
fprintf("Peak Transmit Antenna Gain:  %8.3f dB\n",peak_trans_gain);
fprintf("Transmit Antenna Beamwidth:  %8.3f deg\n",trans_beamwidth);
fprintf("Transmit Antenna Diameter:  %8.3f m\n",trans_diameter);
fprintf("Transmit Antenna Pointing Offset%8.3f deg\n",trans_point_offset);
fprintf("Transmit Antenna Pointing Loss: %8.3f dB\n",trans_point_loss);
fprintf("Transmit Antenna Gain:      %8.3f dB\n",trans_gain);
fprintf("-----\n")
fprintf("Peak Receive Antenna Gain:  %8.3f dB\n",peak_rec_gain);
fprintf("Receive Antenna Beamwidth:  %8.3f deg\n",rec_beamwidth);
fprintf("Receive Antenna Diameter:  %8.3f m\n",rec_diameter);
fprintf("Receive Antenna Pointing Error: %8.3f deg\n",rec_point_error);
fprintf("Receive Antenna Pointing Loss: %8.3f dB\n",rec_point_loss);
fprintf("Receive Antenna Gain:      %8.3f dB\n",rec_gain);
fprintf("-----\n")
fprintf("Equiv. Isotropic Rad. Power:  %8.3f dBW\n",EIRP);
fprintf("Propagation Path Length:    %0.1f km\n",path_length);
fprintf("Space Loss:                %8.3f dB\n",space_loss);
fprintf("System Noise Temperature:   %8.3f K\n",sys_noise_temp);
fprintf("Eb/No:                      %8.3f dB\n",Eb_No);
fprintf("Carrier-to-Noise Density Ratio: %8.3f dB-Hz\n",C_No);
```

```
fprintf("Required Eb/No:      %8.3f dB\n",Req_Eb_No);  
fprintf("Margin (Dry):        %8.3f dB\n",margin);  
fprintf("Margin (Rain):       %8.3f dB\n",margin_diff);  
fprintf("SNR:                  %8.3f dB\n",SNR);
```

## Vehicles: Satellites Appendix

Samantha Dickmann,<sup>1</sup> Lorin Nugent,<sup>2</sup> Jaxon Connolly,<sup>3</sup> Monica Viz,<sup>4</sup> Matthew Popplewell,<sup>5</sup>

Stephen Grabowski,<sup>6</sup> Gregory Fretti,<sup>7</sup> Alan Gelman,<sup>8</sup> Sanjana Singh,<sup>9</sup> Roberto Cardano<sup>10</sup>

*Purdue University, West Lafayette, IN, 47906, USA*

**This appendix describes the design process and analysis that led to the design of the satellite system for Project NextStep. The satellite system comprises 7 satellites, including 6 communications satellites and 1 observation satellite. We name the communications satellite constellation Circumlunar**

---

<sup>1</sup> Controls, Satellite Team Lead

<sup>2</sup> Controls

<sup>3</sup> Controls

<sup>4</sup> Controls

<sup>5</sup> Mission Design

<sup>6</sup> Mission Design

<sup>7</sup> Propulsion

<sup>8</sup> Power and Thermal

<sup>9</sup> Propulsion

<sup>10</sup> Controls

**High-altitude Extraterrestrial Lunar Link (CHELL) and the observation satellite FirstStep Orbiter (FISTO).** CHELL provides communications infrastructure for the colony and Earth, and FISTO provides optical observation and scientific research. In this appendix, we discuss in detail the work, analyses, and decisions that support the subsystem and overall designs described on the main report. Our work and analyses show that the satellite system accomplishes the mission objectives and supports the Project NextStep colony.

## I. Sizing

We conduct satellite sizing using mass- and power-estimating relationships developed by Springmann and de Weck [13]. These relationships start with the payload of each satellite, which we find by adding the communications component masses and power for the CHELL satellites and by adding the science instrument masses and power for FISTO. We then apply the following equations from [13]:

$$m_{dry} = 7.5P_{PL}^{0.65} \quad (1)$$

$$P_{eol} = 1.3P_{PL} + 261 \quad (2)$$

From these equations, we obtain estimates for the dry mass and the total end-of-life (EOL) power requirements.

Next, we apply the mass allocation model developed by Wijker [14] using the dry mass. We allocate the dry mass into every subsystem and check that the resulting subsystem masses are consistent with our estimates created through other means such as by summing masses of commercially available components. The allocation budget reproduced from Wijker [14] is in Table 1.

**Table 1 Mass allocation breakdown based on dry mass from Wijker [14]**

| Subsystem         | Mass Allocation (%) |
|-------------------|---------------------|
| <b>Structure</b>  | 18                  |
| <b>Propulsion</b> | 12                  |
| <b>AOCS</b>       | 7                   |
| <b>Power</b>      | 23                  |
| <b>TT&amp;C</b>   | 4                   |
| <b>Thermal</b>    | 4                   |
| <b>Payload</b>    | 28                  |
| <b>Wiring</b>     | 4                   |
| <b>Total</b>      | 100                 |

We use Eq. (2) to calculate the EOL power requirement for the CHELL satellites and then use Eq. (1) and Table 1 to calculate the mass of the entire payload. Knowing the mass of the CHELL bus is 450 kg, as discussed in the main report, we sum the bus mass and payload mass to calculate the dry mass of the CHELL satellites. The working spreadsheet used in these calculations is below in Fig. 1.

| Component/Subsystem           | Mass (kg)     | % of Dry Mass | Power (W)    | Source       |
|-------------------------------|---------------|---------------|--------------|--------------|
| Ka-band Antenna               | 4.0           | 0.69          | 444          | Bridget      |
| X-band Antenna                | 0.3           | 0.05          | 20           | Anna         |
| S-band Antenna                | 0.3           | 0.06          | 20           | Anna         |
| Ka-band Transceiver           | 2.3           | 0.39          | 47           |              |
| X-band Transmitter            | 1.0           | 0.16          | 15           |              |
| S-band Receiver               | 0.2           | 0.04          | 2            |              |
| <b>Communications Payload</b> | <b>126.6</b>  | <b>22.0</b>   | <b>548</b>   |              |
| Structure                     | 103.8         | 18            |              | [3]          |
| Prop                          | 69.2          | 12            |              | [3]          |
| ACS                           | 40.4          | 7             |              | [3]          |
| Power                         | 132.6         | 23            |              | [3]          |
| TT&C                          | 23.1          | 4             |              | [3]          |
| Thermal                       | 23.1          | 4             |              | [3]          |
| Wiring                        | 23.1          | 4             |              | [3]          |
| <b>Bus (ELiTebus-1000)</b>    | <b>450.0</b>  | <b>78.0</b>   |              | [5]          |
| <b>Dry Mass</b>               | <b>576.6</b>  | <b>100</b>    | <b>973.4</b> | [3], [4] EOL |
| <b>Propellant Mass</b>        | <b>773.1</b>  |               |              | Sanjana      |
| <b>Wet Mass</b>               | <b>1349.7</b> |               |              |              |

|             | Current Best Estimates (CBE) | Max Expected Value (MEV) |
|-------------|------------------------------|--------------------------|
| Mass (Dry)  | 576.6 kg                     | 691.9 kg                 |
| Mass (Wet)  | 1349.7 kg                    | 1619.7 kg                |
| Power       | 973.4 W                      | 1168.1 W                 |
| Contingency | 0 %                          | 20 %                     |

[3] [https://drive.google.com/file/d/1upaxJl2PXLrm7EgOq8UWH886p\\_J3eJjt/view?usp=sharing](https://drive.google.com/file/d/1upaxJl2PXLrm7EgOq8UWH886p_J3eJjt/view?usp=sharing)

[4] [http://web.mit.edu/deweck/www/PDF\\_archive/2%20Refereed%20Journal/2\\_3\\_JSР\\_parametric\\_NGSO.pdf](http://web.mit.edu/deweck/www/PDF_archive/2%20Refereed%20Journal/2_3_JSР_parametric_NGSO.pdf)

[5] <https://spaceflight101.com/spacecraft/iridium-next/>

**Fig. 1 Full working spreadsheet used for sizing the CHELL satellites.**

We perform preliminary satellite sizing for the FISTO observation satellite using the masses of the scientific payload (optical camera, radar sensor, and mass spectrometer) and standard mass allocation budgets for small to medium-sized spacecraft presented by Table 1.

By assuming the combined mass of the scientific instruments in our payload makes up 28% of our satellite's total mass, we find a preliminary estimate for the overall mass of the satellite. We constrain the orbit altitude due to the resolution requirements for the optical camera, so we determine that electric propulsion is the most appropriate form of propulsion for this application. More information regarding the decision to use electric propulsion is discussed in the Satellite Propulsion section of the report. The electric propulsion system is based off of the NSTAR Xenon thruster. We then calculate the mass of propellant required using an early estimate for the dry mass of the satellite and predicted yearly delta-v requirements.

One of our assumptions for the development of the 7 satellites is that they can all be completely manufactured and tested in the first year of the mission. To make this more feasible, we select commercially-available satellite buses instead of starting from scratch. For the FISTO observation satellite, the Lockheed Martin (LM) 400 Series satellite bus is found as a strong match for our estimated payload mass, size, and power requirements. The LM 400 Series satellite is designed for missions with imaging, remote sensing, communications, and radar. Its intended payload is approximately 400 kg, with dimensions 1.7 m x 1.5m x 1.1 m [1]. Both the intended missions and the vehicle sizing align well with our goals for the observation satellite, so the LM 400 is selected as a base for modeling and analysis.

Once we choose the LM 400 satellite bus, component mass estimates that are based off percentages in Table 1 can now be replaced with more concrete values. The largest variable then becomes the solar panels. As instrument and payload power requirements change, the solar panel requirements change frequently, which in turn affects the mass properties and propellant requirements for the satellite. Once we freeze the payload, instruments, and finalize propellant

masses, we verify the volume of individual components to ensure they fit within the satellite frame, then estimate the overall volume from the dimensions of the LM 400 bus.

The specific breakdown of the mass and power properties of the FISTO observation satellite are laid out in the spreadsheet in Fig. 2.

| Component/Subsystem                                      | Mass (kg) | Power (W) | Qty | Total mass (kg) | % Dry Mass | Total Power (W) | Source      |
|----------------------------------------------------------|-----------|-----------|-----|-----------------|------------|-----------------|-------------|
| <b>Scientific Payload</b>                                |           |           |     | 36.80           | 12.82      | 70              | [3]         |
| Camera                                                   | 15        | 20        | 1   | 15.00           |            | 20              | [2]         |
| MiniRF                                                   | 13.8      | 25        | 1   | 13.80           |            | 25              | [9], [1]    |
| Mass spectrometer                                        | 8.0       | 25        | 1   | 8.00            |            | 25              | [10]        |
| <b>Communications (S-band)</b>                           | 0.07      | 15        | 2   | 0.14            | 0.05       | 30              | Anna        |
| <b>Structure</b>                                         | 72.0      | 0         | 1   | 72.00           | 25.08      | 0               | [3]         |
| <b>Electric Propulsion</b>                               | 26.0      | 2300      | 1   | 26.00           | 9.06       | 2300            | [8], [5]    |
| <b>RWAs</b>                                              | 13.1      | 125       | 4   | 52.50           | 18.29      | 500             | [7]         |
| <b>TT&amp;C</b>                                          | 3.7       | 10        | 2   | 7.36            | 2.56       | 20              | Monica      |
| <b>Thermal</b>                                           | 16.0      | 11.11     | 1   | 16.00           | 5.57       | 11.11           | [3]         |
| <b>Wiring</b>                                            | 16.0      | 0         | 1   | 16.00           | 5.57       | 0               | [3]         |
|                                                          |           |           |     |                 |            |                 |             |
| <b>Dry Properties (before Solar/Prop; used for code)</b> |           |           |     | 226.80          |            | 3001.11         |             |
| <b>Dry Properties (with Solar/Prop)</b>                  |           |           |     | <b>287.05</b>   | 100.00     | <b>3001.11</b>  |             |
|                                                          |           |           |     |                 |            |                 |             |
| <b>Outputs from Greg's Code</b>                          |           |           |     |                 |            |                 |             |
| <b>Solar Panels</b>                                      | 19.80     |           | 2   | 39.60           | 13.80      |                 | Greg's code |
| <b>Battery</b>                                           | 20.65     |           | 1   | 20.65           | 7.19       |                 | Greg's code |
| <b>Main Propellant</b>                                   | 20.19     |           | 1   | 20.19           | 7.03       |                 | Greg's code |

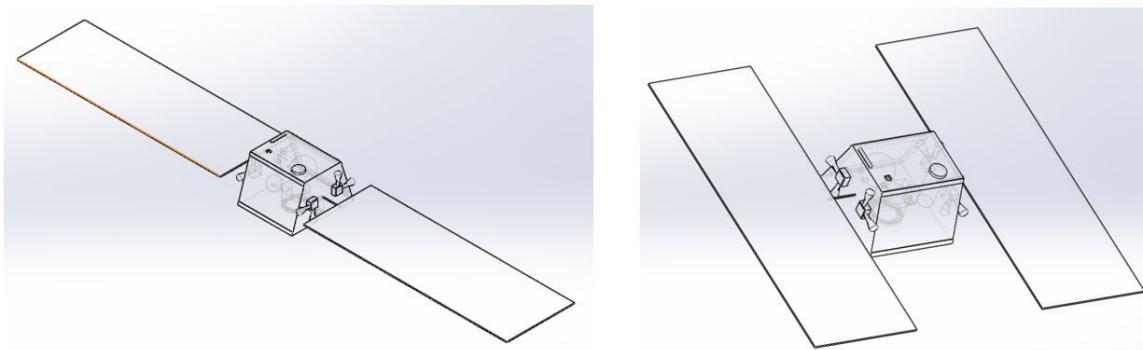
|                                             |                                                           |          |    |               |       |                |     |
|---------------------------------------------|-----------------------------------------------------------|----------|----|---------------|-------|----------------|-----|
| <b>ACS Propellant</b>                       | 63.8                                                      |          | 1  | 63.75         | 22.21 |                | [3] |
|                                             |                                                           |          |    |               |       |                |     |
| <b>Wet Properties (without contingency)</b> |                                                           |          |    | <b>370.99</b> |       | <b>3001.11</b> |     |
|                                             |                                                           |          |    |               |       |                |     |
|                                             | <b>Max Expected Values (MEV)</b><br><b>(Used for PDR)</b> |          |    |               |       |                |     |
|                                             | <b>Wet Mass</b>                                           | 445.188  | kg |               |       |                |     |
|                                             | <b>Power</b>                                              | 3601.332 | W  |               |       |                |     |
|                                             | <b>Contingency</b>                                        | 20       | %  |               |       |                |     |

**Fig. 2 Full working spreadsheet used for sizing the FISTO satellite.**

## II. CAD

We initially create a simplified CAD model of the FISTO satellite while the specifics of the payload and subsystems are still being finalized. As discussed in the Sizing portion of the Appendix, we model the main bus of the satellite after the LM 400 series commercially-available satellite bus. For our initial CAD model, we simplify the satellite as a solid, homogeneous material with an assigned mass property equal to the total mass. Upon each iteration of the model, we add more individual components and updated masses to reflect any changes in the subsystems. Eventually, the driving cause of these refinements is the analysis of the attitude control system - specifically the sizing of the reaction wheel assemblies. We use the attitude control Matlab script in the Appendix to determine that the close proximity of FISTO to the Moon in addition to the large required solar arrays to power the electric propulsion system cause the satellite to require significant reaction control. To reduce these adverse effects, we include as many specific

components with assigned masses as possible, placing them toward the center of the satellite to try to reduce the principal moments of inertia. We also rotate the solar arrays by 90 degrees to again bring more mass toward the center of the satellite and improve the controllability of the satellite's attitude. The solar panels account for a significant portion of the total spacecraft mass and also have a large radius from the center of mass.



**Fig. 3 An early iteration of the CAD model (left) requires more active attitude control than the final iteration (right) due to the orientation of the solar panels (CAD by Nicholas Masso).**

We make this change for two main reasons. Orienting the solar panels as shown in the right image of Fig. 3 reduces the inertia properties along that axis, which improves the controllability of the spacecraft. This change also reduces the strength of disturbances to the spacecraft's attitude caused by the gravity torque from the Moon and solar radiation pressure.

Figure 2 and Table 5 in the main report describe which components end up in the final CAD model. We assign the rest of the total satellite mass to the main satellite structure so that the final mass of the model is correct.

### III. Budget and Costs

#### A. CHELL Satellites

We calculate the cost of the CHELL constellation based on the costs of the Iridium-NEXT constellation. We choose Iridium-NEXT as a basis because Iridium-NEXT is a state-of-the-art, Ka-band communications constellation, like CHELL.

66 Iridium-NEXT satellites were built under a \$2.1 billion contract [15]. We assume that 20% of this \$2.1 billion is fixed cost, or \$420 million is fixed. Subtracting this fixed cost and dividing by the number of satellites gives \$25 million per satellite. The development timeline of Iridium-NEXT was approximately nine years [15], so we calculate a yearly fixed cost of about \$50 million. Since our development timeline is accelerated, we add a factor of 1.5, so the fixed cost of the CHELL constellation is \$75 million.

From \$25 million per satellite and \$75 million in fixed costs, we finally estimate the total development and manufacturing cost for all CHELL satellites to be \$225 million.

We estimate operational costs to be 7% of spacecraft cost [16], so approximately \$2 million per CHELL satellite. Thus we expect the total operational cost each year to be \$12 million.

## IV.Mission Design

### A. FISTO Orbit Design

In order to introduce the orbit design methodology, we first present a general analysis of orbit perturbations for spacecraft in low lunar orbit. The Lunar Reconnaissance Orbiter (LRO) presents an excellent case study, as it required a yearly delta-v budget of approximately 150 m/s per year to maintain a 50 km mean altitude circular polar orbit within 20 km of the nominal orbit. Additionally, its commissioning phase made use of a quasi-frozen orbit of 30 km x 216 km that experiences far less variation in its orbital elements, which we use as inspiration for the orbit design of our satellite [6]. In order to compare these two orbits as well as the lifetimes of circular polar orbits at a range of altitudes, the perturbing effects of the Moon's gravity field up to  $J_{8,8}$  harmonics (with the LP-165 model), the Sun, the Earth, and solar radiation pressure are modeled in GMAT. Using circular polar orbits at altitudes of 50 km, 75 km, 100 km, and 200 km, as well as the LRO commissioning orbit (eccentricity = 0.043, mean altitude = 123 km), we determine that an altitude of approximately 100 km is required for a spacecraft in a circular polar orbit at the Moon to not eventually impact the surface, and natural altitude variations vary wildly, only becoming bounded to within +/- 60 km if the orbit height is increased to 200 km. By contrast, the minimum perilune and maximum apolune of the quasi-frozen orbit used by LRO do not vary significantly from their expected values after half of a year, and this orbit similarly does not impact the Moon despite the extremely low perilune altitude (Table 2). This propagation uses a start date of 12/01/2021 and the following initial conditions: inclination =  $90^\circ$ , right ascension of the ascending node =  $120^\circ$ , argument of periapsis =  $270^\circ$ , and insertion at perilune.

**Table 2 Orbit impact time comparison**

| Initial Orbit Type | Min Altitude (km) | Max Altitude (km) | Impact Date |
|--------------------|-------------------|-------------------|-------------|
| 50 km Circular     | 0                 | 101.420           | 72 Days     |
| 75 km Circular     | 0                 | 151.222           | 129 Days    |
| 100 km Circular    | 22.184            | 178.961           | N/A         |
| 200 km Circular    | 139.130           | 261.942           | N/A         |
| LRO Commissioning  | 34.165            | 212.886           | N/A         |

Instead of directly implementing the LRO commissioning orbit, we choose to change the perilune altitude to 50 km, as this is still well-suited for the operation of the various satellite instruments, but this places the minimum acceptable altitude at 35 km instead of 15 km based on the tolerance defined earlier. With a perilune distance of 35 km and an eccentricity of 0.443566 instead of 0.50 for the LRO commissioning orbit (to account for the change in periapsis distance), all other orbit parameters remain the same for this new quasi-frozen orbit. The orbital elements of the final selected orbit for the observation satellite are summarized in Table 3.

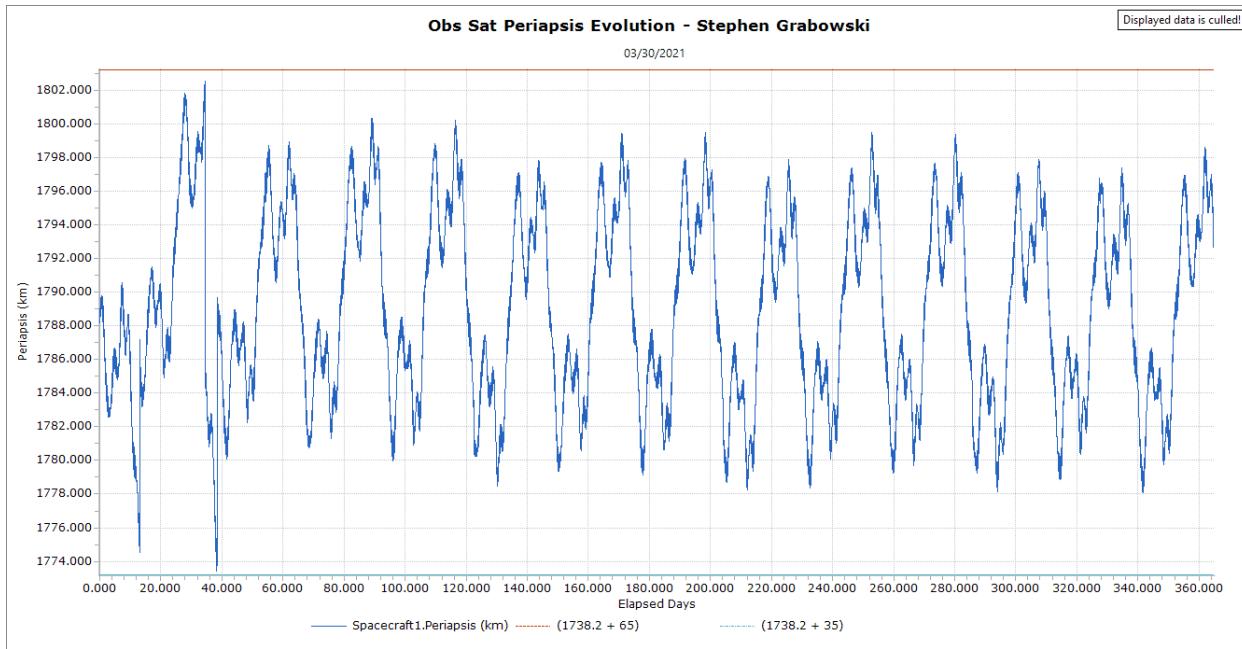
**Table 3 Orbital elements of observation satellite**

| Semi-major axis (km) | Inclination | Right Ascension of the Ascending Node | Argument of Periapsis | Eccentricity |
|----------------------|-------------|---------------------------------------|-----------------------|--------------|
| 1871.2               | 90°         | 190°                                  | 270°                  | 0.0443566    |

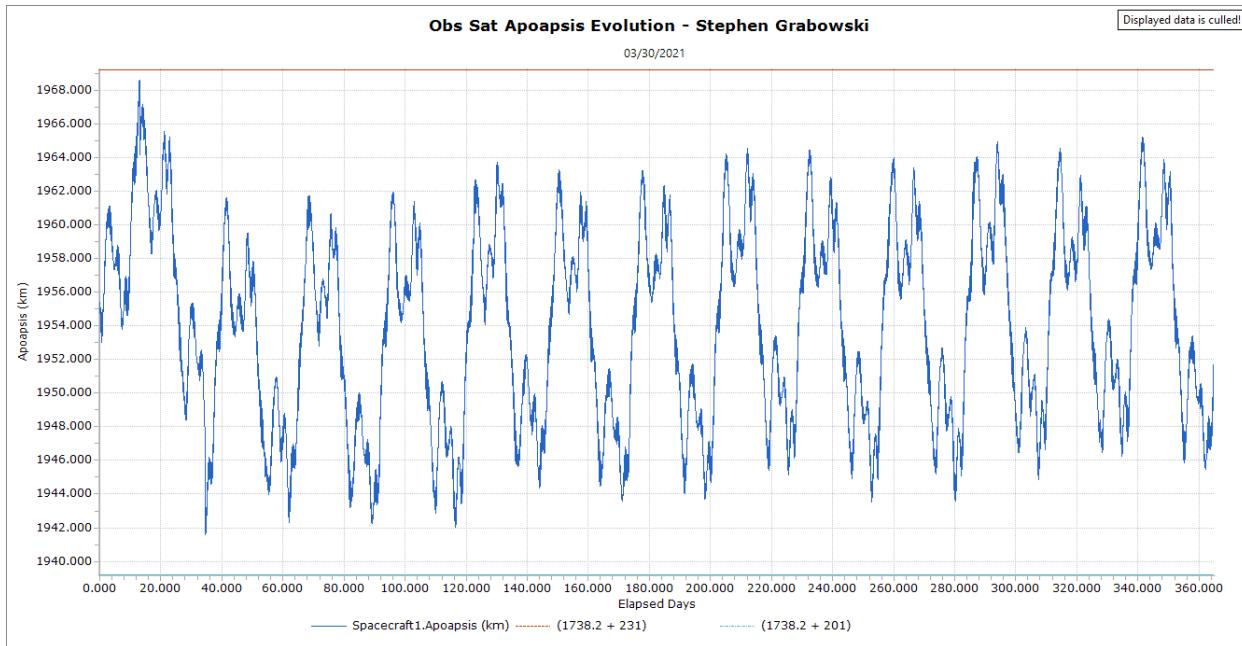
## B. FISTO Orbit Stationkeeping Maneuver Strategy

Having selected the orbit that the FISTO satellite will use, we propagate this chosen orbit for a year, starting from a date of Nov. 01, 2023, which aligns more closely with the launch of the observation satellite in year 3 of the mission timeline. FreeFlyer is used instead of GMAT for this stage of the orbit modeling because the more recent GSFC GRAIL gravity field model for the Moon is available instead of LP-165, and we use a higher fidelity model taking into account zonal and tesseral harmonics of the Moon up to  $J_{20,20}$  as well.

We implement a burn targeting strategy with impulsive maneuvers that places bounds on perilune and apolune ( $\pm 15$  km) per the driving requirement on altitude, argument of periapsis ( $\pm 10^\circ$ ) in order to maintain the location of periapsis approximately above the lunar south pole, and inclination ( $\pm 3^\circ$ ) in order to maintain a polar orbit. The minimum time between burns is 13.5 days, or approximately half of a lunar sidereal period, in order to limit propellant expenditure and because this was roughly the timetable followed by LRO in its initial circular orbit. Right ascension of the ascending node is not targeted, as no requirements were specifically generated for the orientation of the observation satellite orbit with respect to the Earth. Additionally, right ascension of the ascending node is typically one of the orbital elements that experiences the strongest secular impact from non-spherical gravity harmonics, so maintaining it on top of the more important parameters like altitude and argument of periapsis would likely dramatically drive up propellant cost. We see from the simulation results below that with 13 burns totaling 137.2 m/s of delta-v over the course of a year, all of the specified orbit objectives are met.

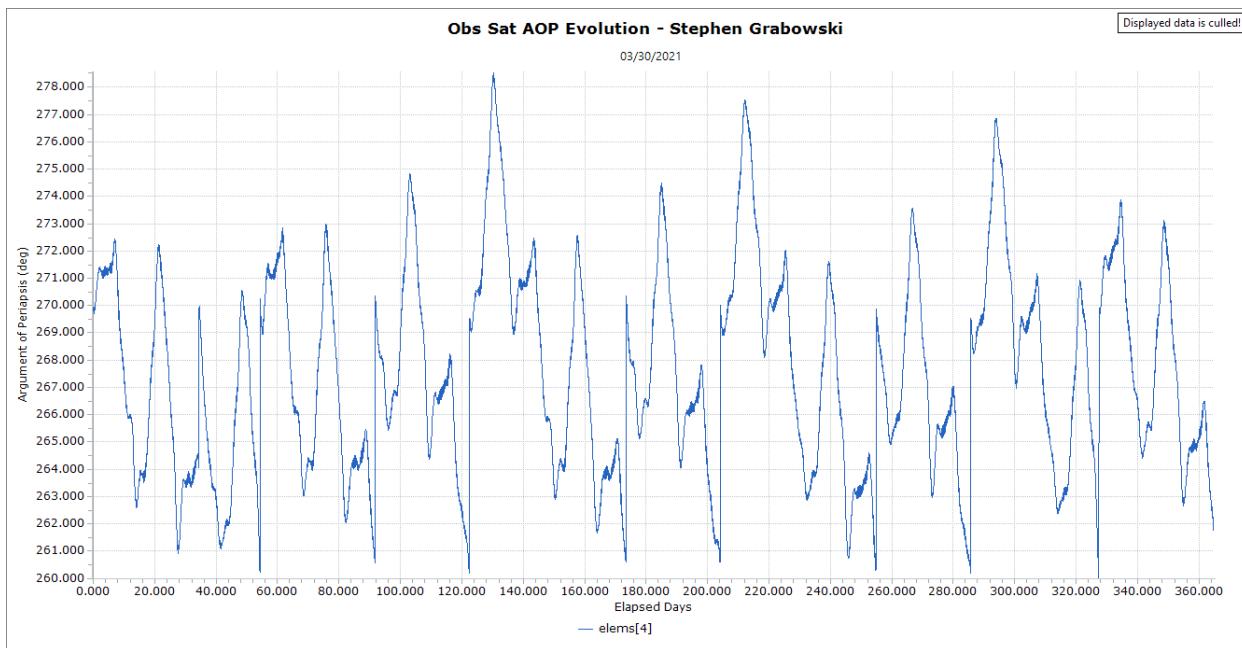


**Fig. 4 Observation satellite perilune is well-bounded with corrective burns.**

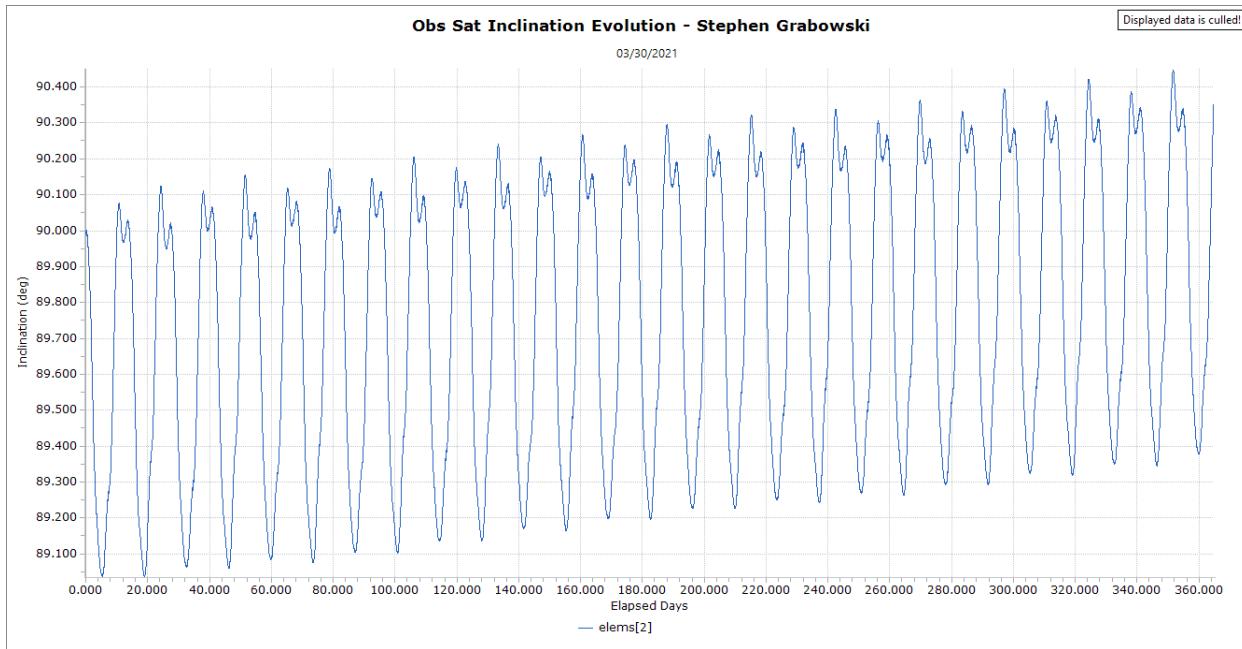


**Fig. 5 Observation satellite apolune is well-bounded with corrective burns.**

We successfully establish well-bounded perilune and apolune distances to within the required tolerance with this burn targeting logic, and in fact for the bulk of the propagation the spacecraft's orbit remains even closer to nominal than the requirement (Figs. 4-5). As it turns out, the majority of the required burns combat the effects of the simulated perturbations on argument of periapsis (Fig. 6), and the perilune and apolune distances remain within tolerance due mainly to the natural behavior of the orbit. Additionally, the inclination only increases by less than  $0.5^{\circ}$  over the course of a year, meaning that the orbit continues to pass over the south pole uninterrupted by perturbing effects (Fig. 7). Hence, we see the value of using a quasi-frozen orbit with relatively stable and predictable behavior, even when subjected to a wildly uneven gravity field like that of the Moon.



**Fig. 6 Observation satellite argument of periapsis is well-bounded with corrective burns.**



**Fig. 7 Observation satellite inclination exhibits a very small secular increase.**

### C. FISTO Orbit Stationkeeping FreeFlyer Script

```

Spacecraft1.SetCentralBody(Moon);

// View window
vw.CurrentViewpoint.ThreeDView.Source = Moon.ObjectId;
vw.CurrentViewpoint.ThreeDView.Target = Moon.ObjectId;
vw.CurrentViewpoint.ThreeDView.TailReference = Moon.ObjectId;

// Variables
Variable dvTot = 0;
Variable numBurns = 0;
Variable tPrevBurn = 0;
Variable minBurnSep = 13.5;
Array elems = Spacecraft1.GetKeplerianState(7);

// Plot Windows
PlotWindow p1({Spacecraft1.ElapsedTime, Spacecraft1.Apoapsis, 1738.2+231,
1738.2+201});
p1.PlotTitle.Text = 'Obs Sat Apoapsis Evolution - Stephen Grabowski';
p1.XAxis.Title.Text = 'Elapsed Days';
p1.YAxis.Title.Text = 'Apoapsis (km)';

```

```
PlotWindow p2({Spacecraft1.ElapsedTime, Spacecraft1.Periapsis, 1738.2+65,
1738.2+35});
p2.PlotTitle.Text = 'Periapsis Evolution';
p2.PlotTitle.Text = 'Obs Sat Periapsis Evolution - Stephen Grabowski';
p2.XAxis.Title.Text = 'Elapsed Days';
p2.YAxis.Title.Text = 'Periapsis (km)';

PlotWindow p3({Spacecraft1.ElapsedTime, elems[1]});
p3.PlotTitle.Text = 'Eccentricity Evolution';
p3.PlotTitle.Text = 'Obs Sat Eccentricity Evolution - Stephen Grabowski';
p3.XAxis.Title.Text = 'Elapsed Days';
p3.YAxis.Title.Text = 'Eccentricity';

PlotWindow p4({Spacecraft1.ElapsedTime, elems[2]});
p4.PlotTitle.Text = 'Inclination Evolution';
p4.PlotTitle.Text = 'Obs Sat Inclination Evolution - Stephen Grabowski';
p4.XAxis.Title.Text = 'Elapsed Days';
p4.YAxis.Title.Text = 'Inclination (deg)';

PlotWindow p5({Spacecraft1.ElapsedTime, elems[3]});
p5.PlotTitle.Text = 'RAAN Evolution';
p5.PlotTitle.Text = 'Obs Sat RAAN Evolution - Stephen Grabowski';
p5.XAxis.Title.Text = 'Elapsed Days';
p5.YAxis.Title.Text = 'Right Ascension (deg)';

PlotWindow p6({Spacecraft1.ElapsedTime, elems[4]});
p6.PlotTitle.Text = 'Argument of Periapsis Evolution';
p6.PlotTitle.Text = 'Obs Sat AOP Evolution - Stephen Grabowski';
p6.XAxis.Title.Text = 'Elapsed Days';
p6.YAxis.Title.Text = 'Argument of Periapsis (deg)';

p1.MaxPoints = 10000;
p2.MaxPoints = 10000;
p3.MaxPoints = 10000;
p4.MaxPoints = 10000;
p5.MaxPoints = 10000;
p6.MaxPoints = 10000;

// Mission
TimeSpan stepEpoch;
stepEpoch = Spacecraft1.Epoch + TimeSpan.FromDays(365);
```

```
While (Spacecraft1.ElapsedTime.ToDays < 365);
Step Spacecraft1;
elems = Spacecraft1.GetKeplerianState(7);
Update vw;

If (Spacecraft1.ElapsedTime.ToDays - tPrevBurn > minBurnSep or numBurns == 0);
If (Spacecraft1.Apoapsis > 1969.2 or Spacecraft1.Apoapsis < 1939.2);
    Target using DifferentialCorrector1;
    Iterate Spacecraft1;
    Vary ImpulsiveBurn1.BurnDirection[0] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[1] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[2] = 0 + .001;
    Maneuver Spacecraft1 using ImpulsiveBurn1;
    Achieve Spacecraft1.Apoapsis = 1954.2 +/- 5;
    If (DifferentialCorrector1.IterationMode == 'Converged');
        numBurns += 1;
        dvTot += (ImpulsiveBurn1.BurnDirection[0]^2 +
ImpulsiveBurn1.BurnDirection[1]^2 + ImpulsiveBurn1.BurnDirection[2]^2)^.5;
        tPrevBurn = Spacecraft1.ElapsedTime.ToDays;
    End;
End;
End;

If (Spacecraft1.Periapsis > 1803.2 or Spacecraft1.Periapsis < 1773.2);
    Target using DifferentialCorrector1;
    Iterate Spacecraft1;
    Vary ImpulsiveBurn1.BurnDirection[0] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[1] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[2] = 0 + .001;
    Maneuver Spacecraft1 using ImpulsiveBurn1;
    Achieve Spacecraft1.Periapsis = 1788.2 +/- 5;
    If (DifferentialCorrector1.IterationMode == 'Converged');
        numBurns += 1;
        dvTot += (ImpulsiveBurn1.BurnDirection[0]^2 +
ImpulsiveBurn1.BurnDirection[1]^2 + ImpulsiveBurn1.BurnDirection[2]^2)^.5;
        tPrevBurn = Spacecraft1.ElapsedTime.ToDays;
    End;
End;
End;
```

```
If (elems[4] > 280 or elems[4] < 260)
    Target using DifferentialCorrector1;
    Iterate Spacecraft1;
    Vary ImpulsiveBurn1.BurnDirection[0] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[1] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[2] = 0 + .001;
    Maneuver Spacecraft1 using ImpulsiveBurn1;
    Achieve Spacecraft1.GetKeplerianState(7)[4] = 270 +/- 1;
    If (DifferentialCorrector1.IterationMode == 'Converged');
        numBurns += 1;
        dvTot += (ImpulsiveBurn1.BurnDirection[0]^2 +
ImpulsiveBurn1.BurnDirection[1]^2 + ImpulsiveBurn1.BurnDirection[2]^2)^.5;
        tPrevBurn = Spacecraft1.ElapsedTime.ToDays;
    End;
End;
End;

If (elems[2] > 93 or elems[2] < 87);
    Target using DifferentialCorrector1;
    Iterate Spacecraft1;
    Vary ImpulsiveBurn1.BurnDirection[0] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[1] = 0 + .001;
    Vary ImpulsiveBurn1.BurnDirection[2] = 0 + .001;
    Maneuver Spacecraft1 using ImpulsiveBurn1;
    Achieve Spacecraft1.GetKeplerianState(7)[2] = 90 +/- .5;
    If (DifferentialCorrector1.IterationMode == 'Converged');
        numBurns += 1;
        dvTot += (ImpulsiveBurn1.BurnDirection[0]^2 +
ImpulsiveBurn1.BurnDirection[1]^2 + ImpulsiveBurn1.BurnDirection[2]^2)^.5;
        tPrevBurn = Spacecraft1.ElapsedTime.ToDays;
    End;
End;
End;

Update p1;
Update p2;
Update p3;
Update p4;
Update p5;
Update p6;
End;
```

```
| Report dvTot, numBurns;
```

## D. Molniya Orbit

Molniya orbits are defined as highly eccentric orbits with an inclination designed to eliminate any precession of the orbit plane due to the J2 gravitational harmonic perturbation. The high eccentricity and apoapsis of the orbit enable long periods of coverage over a specific location. Translating this concept to the Moon allows for long periods of communication with the south pole. Deploying two spacecraft in this configuration allows for continuous contact with the south pole.

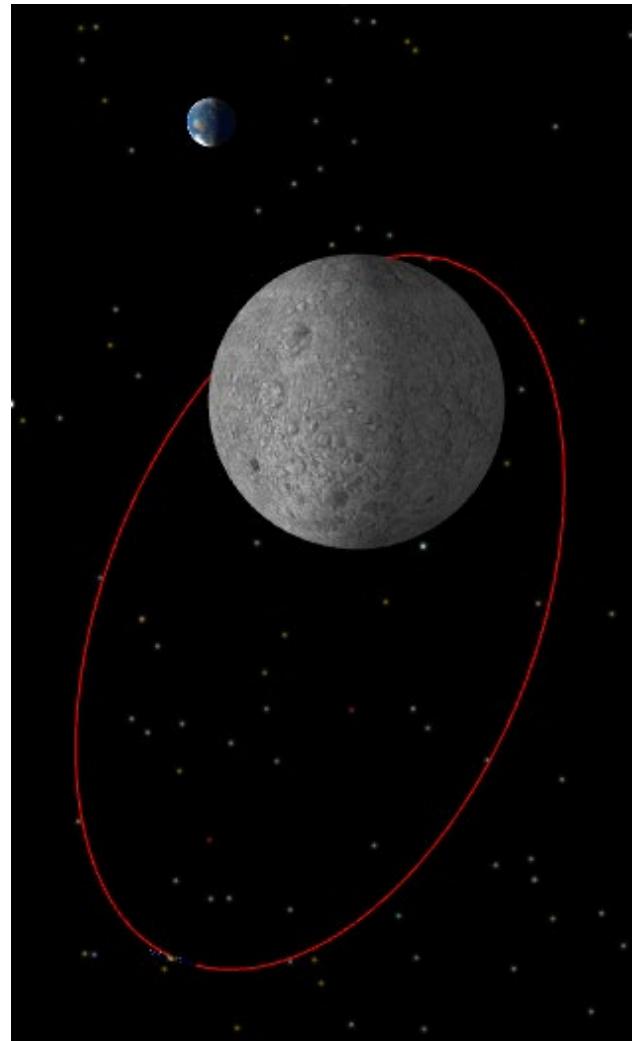
The perilune of the orbit is chosen in order to reduce any perturbing effects due to the nonuniform gravitational field of the Moon caused by large mass concentrations (mascons) beneath the lunar surface. Analysis showed that orbits with an altitude greater than 200 km were largely unaffected by these mascons. The apolune of the orbit is then chosen such that the spacecraft will complete two orbits per one lunar day. As previously mentioned, the inclination of the orbit is set to  $63.435^\circ$  in order to eliminate any precession of the ascending node due to the J2 harmonic term. The argument of periapsis is set such that the apoapsis of the spacecraft is in the southern hemisphere of the Moon. Two spacecraft are phased  $180^\circ$  apart in this orbit.

**Table 4 Molniya Keplerian orbit elements with respect to an inertial Moon-centered frame**

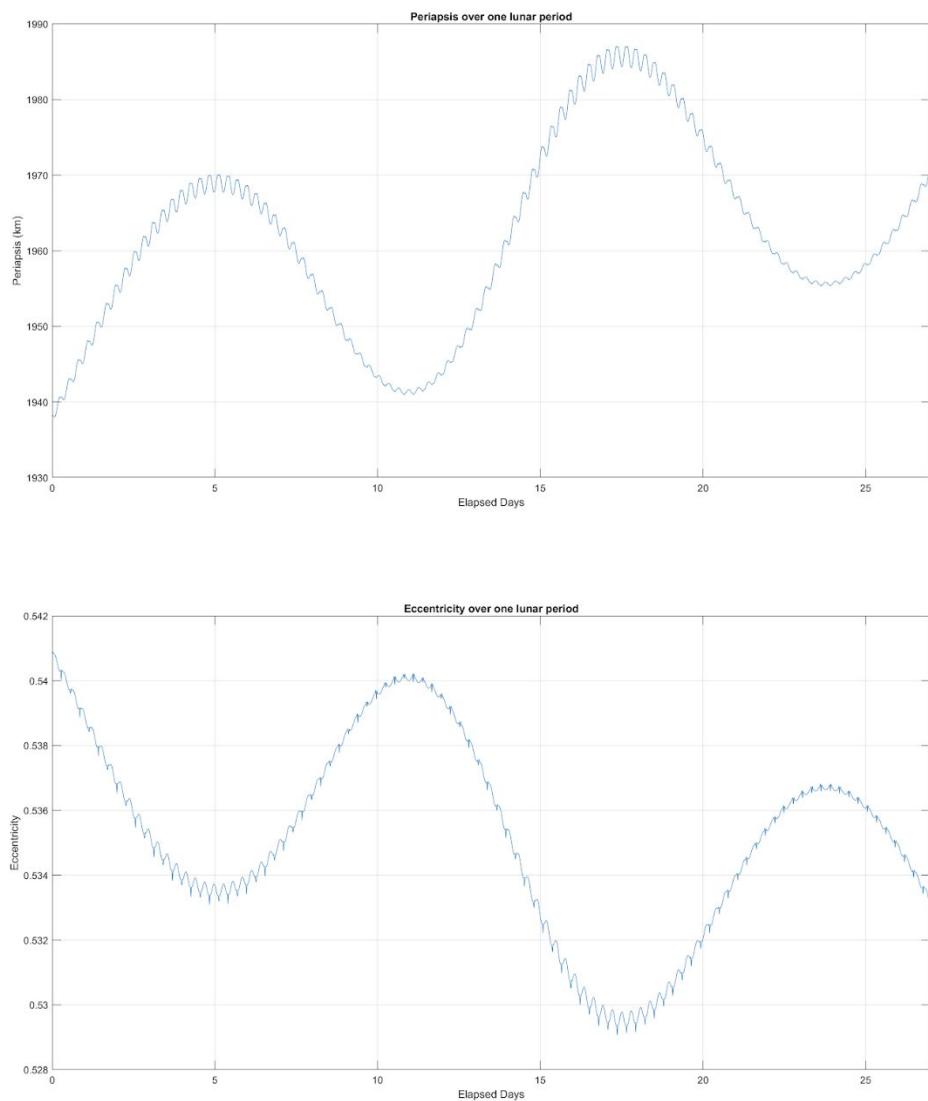
| Orbit Element   | Value          |
|-----------------|----------------|
| Semi-major axis | 4218.8896 km   |
| Eccentricity    | 0.5406         |
| Inclination     | $63.435^\circ$ |

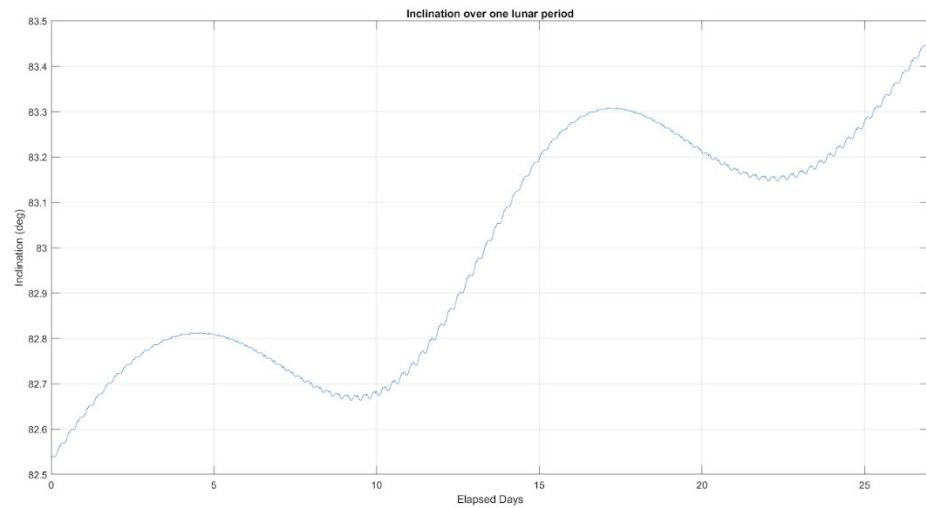
Right ascension of the ascending node  $0^\circ$

Argument of periapsis  $90^\circ$

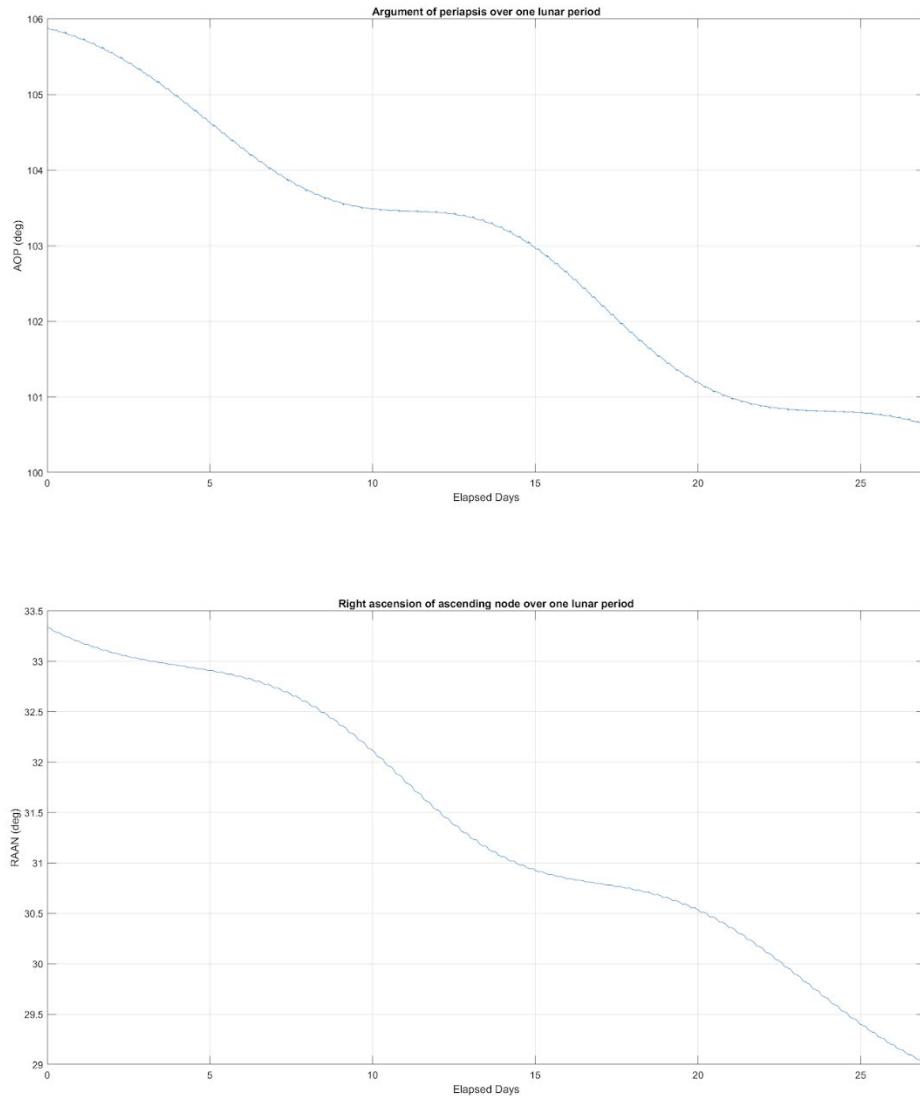


**Fig. 8** 3D view of Molniya orbit in an inertial Moon-centered frame





**Fig. 9 Evolution of Molniya orbit elements over one lunar period**



**Figure 9 (cont.) Evolution of Molniya orbit elements over one lunar period**

The given Molniya orbit was propagated for one lunar period in the presence of perturbing effects. The gravitational field of the Moon up to the J7 gravitational harmonic term, the Earth and Sun as point masses, and solar radiation pressure were included as forces acting on the spacecraft. The introduction of perturbing effects causes the eccentric orbit about the Moon to rapidly degrade over time. This directly contradicts with the fourth design requirement as it requires the satellites

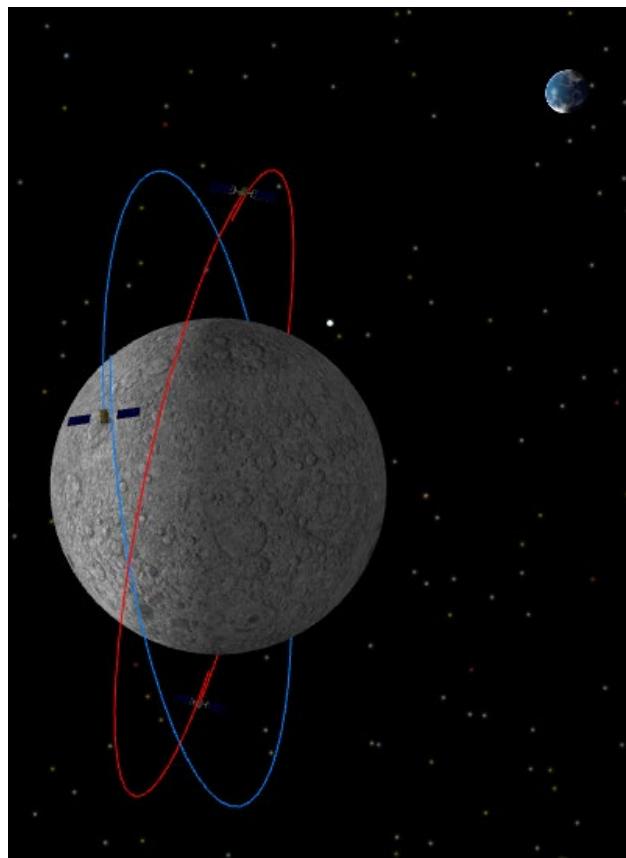
to perform near continuous orbit maintenance in order to continue operations. Additionally, the Molniya configuration lacks significant surface coverage and fails to provide redundancy in case of failure. A failure of even a single spacecraft would result in noncontinuous contact with the south pole and noncontinuous line-of-sight with Earth. Therefore, the Molniya orbit configuration is neglected.

#### E. $80^\circ$ : 4/2/1 Walker Constellation

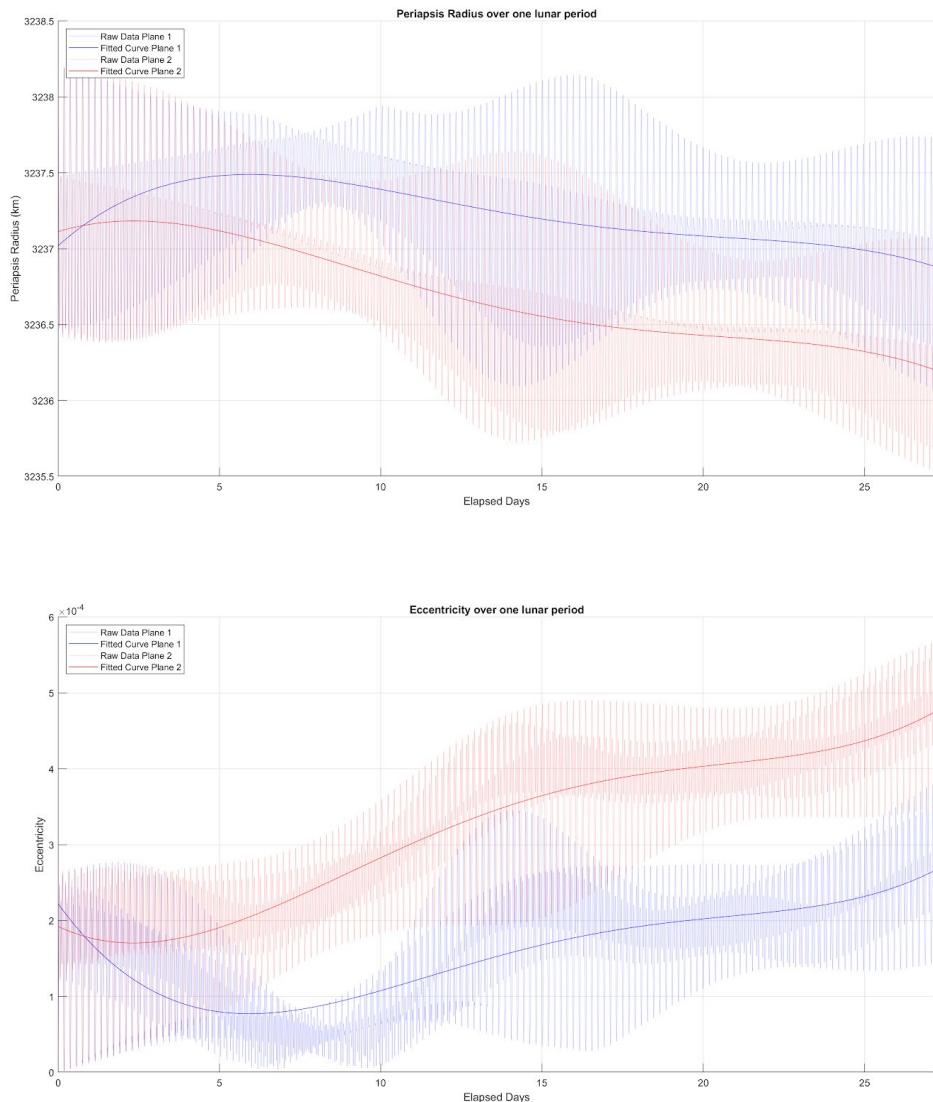
The 4/2/1 configuration consists of four spacecraft divided evenly amongst two orbit planes. The resulting orbital elements for the four spacecraft are given below.

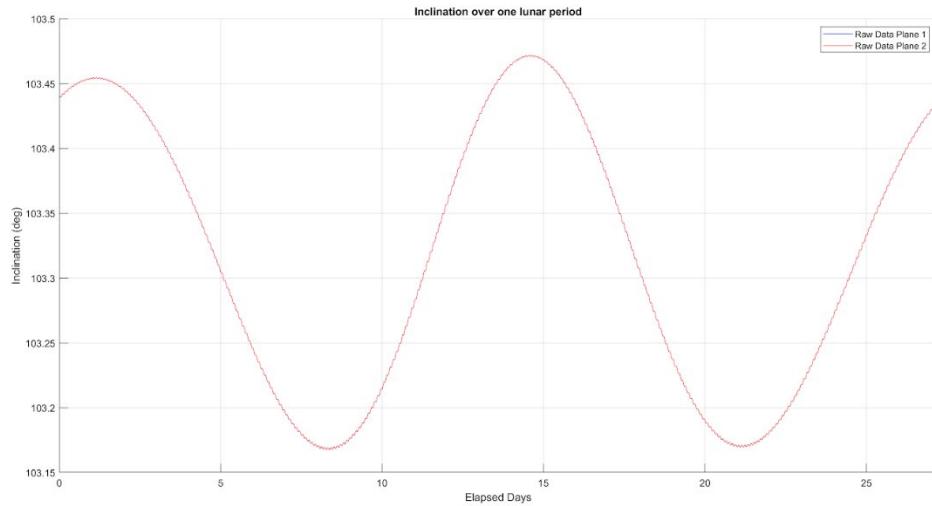
**Table 5 Keplerian orbit elements for Walker  $80^\circ$ : 4/2/1 constellation**

| Satellite   | $a$ (km) | $e$ | $i$ ( $^\circ$ ) | $RAAN$ ( $^\circ$ ) | $TA$ ( $^\circ$ ) |
|-------------|----------|-----|------------------|---------------------|-------------------|
| Spacecraft1 | 3238.2   | 0   | 80               | 0                   | 0                 |
| Spacecraft2 | 3238.2   | 0   | 80               | 180                 | 90                |
| Spacecraft3 | 3238.2   | 0   | 80               | 0                   | 180               |
| Spacecraft4 | 3238.2   | 0   | 80               | 180                 | 270               |



**Fig. 10** 3D view of Walker 80°: 4/2/1 constellation in an inertial Moon frame





**Fig. 11 Evolution of Walker 80°: 4/2/1 constellation  $r$ ,  $e$ ,  $i$  elements over one lunar period**

The given Walker constellation was propagated for one lunar period in the presence of perturbing forces. The gravitational field of the Moon up to the J7 gravitational harmonic term, the Earth and Sun as point masses, and solar radiation pressure were included as forces acting on the spacecraft. Even in the presence of perturbing effects, the stability of the Walker constellation is evident. The largest change is a reduction in periapsis altitude of one kilometer.

Despite the stability of the constellation, the 4/2/1 Walker constellation fails to meet all the mission requirements as it lacks continuous contact with the south pole and global surface coverage of the Moon. The orientation of the orbital planes causes the satellites to neglect a large portion of the lunar surface. Additionally, the 4/2/1 configuration has a satellite with line-of-sight of the south pole ground station, and thus the capability to communicate, only 93% of the time. Furthermore, the use of only four satellites does not include any redundancy. The failure of a single satellite could drastically reduce contact with the south pole and Earth. This configuration does not satisfy mission requirements.

## F. Alternative Deployment Strategy Using Single Launch

The chosen methodology for deploying the communication satellites uses a translunar trajectory discussed in a prior section. This trajectory begins in a low Earth parking orbit and ends in a single orbital plane about the Moon. Upon entering the lunar vicinity, a series of maneuvers are performed to capture about the Moon, adjust to the correct inclination and right ascension values, and finally insert into the mission orbit. Using this strategy, any plane change maneuvers, specifically those to precess the right ascension of the ascending node, are prohibitively expensive. Therefore, each launch can only deploy satellites in a single orbit plane. Since the communication constellation consists of three orbital planes, each containing two satellites, the minimum required number of launches is three. While using three launches is feasible, the discovery of a solution using one launch to deploy all six satellites drastically reduces the launch expenses of the mission. This section discusses an alternative deployment strategy that uses a single launch vehicle to deploy all six communication satellites.

The alternative method consists of a “mothership”, or a vehicle containing the six communication satellites, that jumps between three eccentric orbits and deploys two satellites in each orbit. The satellites then use on-board thrusters to insert into the mission orbit. The mothership follows the same translunar insertion trajectory discussed in prior sections but deviates after the inclination change burn. Rather than immediately reducing the periapsis and apoapsis of the spacecraft, the vehicle remains in an eccentric orbit about the Moon. The orbit elements for this eccentric orbit are given below.

**Table 6 Keplerian orbit elements for “mothership” orbit with respect to lunar body frame**

| Orbit Element                         | Value   |
|---------------------------------------|---------|
| Semi-major axis                       | 8000 km |
| Eccentricity                          | 0.5     |
| Inclination                           | 80°     |
| Right ascension of the ascending node | 0°      |
| Argument of periapsis                 | -90°    |

The goal of the spacecraft is to change the right ascension of the ascending node by 120° and reduce the periapsis of the orbit to the value of the periapsis of the mission orbit without altering the eccentricity or inclination. This can be accomplished in a single, impulsive burn given one knows an intersection point of the current orbit and the orbit after the maneuver. For the current orbit defined above, there are two intersection points: one in the southern hemisphere and one in the northern hemisphere. The intersection point in the northern hemisphere is more desirable as it is near apoapsis and performing a maneuver near apoapsis drastically reduces the size of the maneuver. The exact location of the intersection point was found numerically.

While propagating the mothership to the intersection point, the mothership will deploy the two satellites destined for the current orbit plane. These satellites will ultimately insert themselves into the mission orbit using on-board propulsion. At the intersection point, a nonlinear optimization algorithm is used to find the VNB components of the maneuver that precesses the right ascension

of the ascending node by  $120^\circ$  and reduces the periapsis of the orbit without altering the eccentricity or inclination. The objective of the optimizer is to minimize the magnitude of the maneuver. The optimizer converges on a maneuver of magnitude 0.7932 km/s.

Maneuvering the mothership at the intersection point using the burn found by the optimizer successfully changes the right ascension of the ascending node by  $120^\circ$  and reduces the periapsis of the orbit to the value of the periapsis of the mission orbit. The mothership remains in this second eccentric orbit for one period, during which it deploys the two satellites destined for this orbital plane. These satellites use on-board thrusters to insert themselves into the mission orbit.

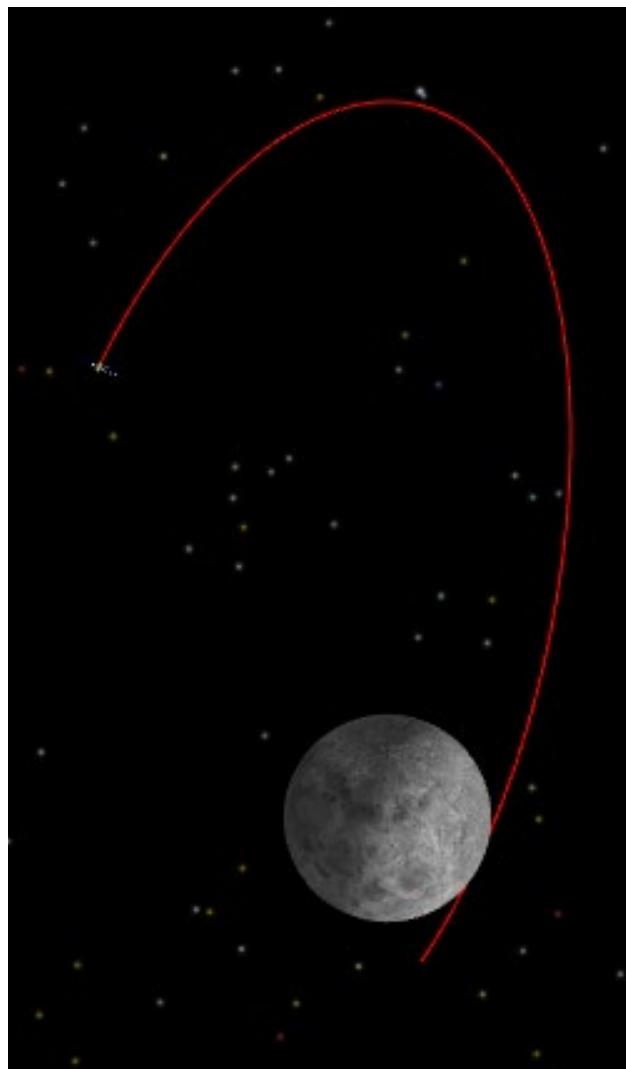
After one period, the above process is used to calculate the intersection point of the second eccentric orbit and a third eccentric orbit that is precessed an additional  $120^\circ$  from the second orbit. Likewise, at this intersection point, the nonlinear optimizer is used to determine the appropriate maneuver to insert the mothership into this third orbit. The mothership will remain in this orbit for one period, during which it will deploy the two satellites destined for this orbital plane. These satellites will insert themselves into the mission orbit. After deploying the final two satellites, the mothership can either remain in this eccentric orbit and be used as an additional communication relay or be decommissioned. The final destination of the mothership will not be discussed in this section.

**Table 7 Maneuver list for “mothership” vehicle beginning at low-Earth parking orbit**

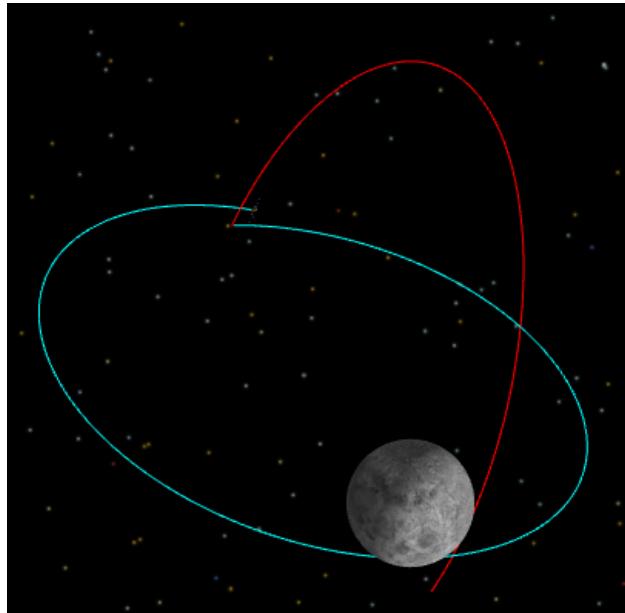
| Maneuver Name        | Maneuver Magnitude (km/s) |
|----------------------|---------------------------|
| TLI                  | 3.047                     |
| Midcourse Adjustment | 0.077                     |

|                        |       |
|------------------------|-------|
| Eccentric Capture      | 0.444 |
| RAAN Adjustment        | 0.063 |
| Inclination Adjustment | 0.068 |
| Plane Hop #1           | 0.793 |
| Plane Hop #2           | 0.793 |
| Total                  | 5.285 |

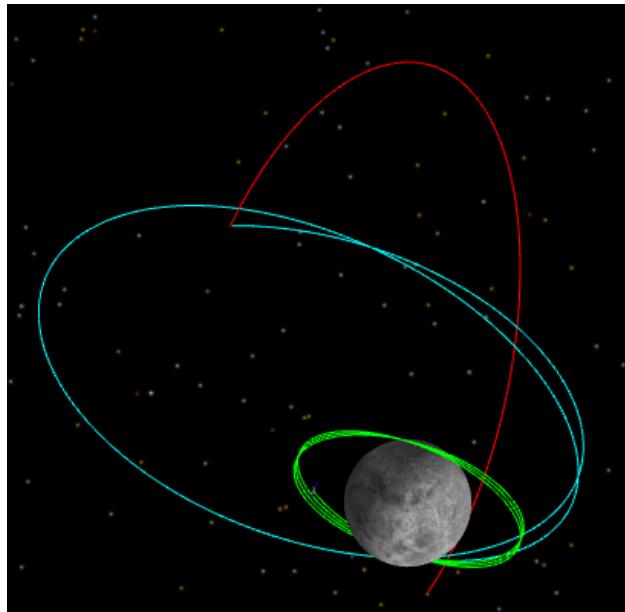
---



**Fig. 12 Eccentric orbit (red) immediately after inclination adjustment. The vehicle propagates to near apoapsis intersection point before maneuvering**



**Fig. 13** After performing the plane hop burn (blue), the vehicle propagates for one period during which the two satellites destined for this orbital plane are deployed



**Fig. 14** Each satellite will use on-board thrusters to reduce its altitude and circularize into the proper mission orbit (green)

Using the process outlined above, the mothership will enter three orbital planes about the Moon with right ascensions of the ascending node of  $0^\circ$ ,  $120^\circ$ , and  $240^\circ$ , each inclined at  $80^\circ$  relative to a lunar inertial frame. Two satellites are deployed in each plane, which insert themselves into the mission orbit. Ultimately, this deployment strategy achieves the desired constellation geometry using a single launch vehicle. In order to employ this strategy, the mothership vehicle must be developed and must be capable of performing a total change in velocity of about 5.3 km/s. A feasibility study would need to be conducted in order to determine the viability of developing and launching such a vehicle, as well as if the vehicle would be able to carry enough propellant to perform the necessary maneuvers. Due to the need for additional investigations, the team has chosen to move forward with the three launch scenario discussed in the body of the paper over the single launch scenario discussed here.

## V. Risk and Fault Analysis

The full fault and risk spreadsheet is in Fig. 15. The spreadsheet breaks the top nine faults into basic events, discusses the likelihood and consequence of each event, and suggests mitigation strategies. Note that top events VS-3, VS-7, and VS-9 can be basic events for other top events. We use this fault and risk spreadsheet to create the fault and risk analysis for the overall mission. Additionally, as we discuss in the main report, we implement fault mitigation strategies after identifying faults in this analysis.

| RefID |                                                   | Likel<br>ihoo quenc<br>d e | RefID    |                                                                  | Likel<br>ihoo quenc<br>d e | Overall | Mitigation/Warning                                                                                                                       |
|-------|---------------------------------------------------|----------------------------|----------|------------------------------------------------------------------|----------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------|
|       | Intermediate Event                                |                            |          | Basic Event                                                      |                            |         |                                                                                                                                          |
| VS-1  | No contact with communications satellite          | 2                          | OVS-41.1 | Transceiver failure                                              | 1                          | 4       | 4Redundant communications components implemented                                                                                         |
|       |                                                   |                            | OVS-1.2  | Incorrect ground station configuration                           | 2                          | 2       | Ground station can be reconfigured, operations procedures should be detailed and tested                                                  |
|       |                                                   |                            | OVS-1.3  | Computer fault                                                   | 1                          | 4       | 4Redundant C&DH system recommended                                                                                                       |
|       |                                                   |                            | OVS-3    | Solar panels do not provide power                                | 2                          | 4       | 8See VS-3                                                                                                                                |
| VS-2  | No contact with observation satellite             | 1                          | OVS-42.1 | Incorrect crosslinking configuration                             | 1                          | 5       | Ensure correct configuration through testing. If incorrect, configuration can be fixed when Wheatley GS is built, but 5significant delay |
|       |                                                   |                            | OVS-2.2  | Computer fault                                                   | 1                          | 4       | 4Redundant C&DH system recommended                                                                                                       |
|       |                                                   |                            | OVS-3    | Solar panels do not provide power                                | 2                          | 4       | 8See VS-3                                                                                                                                |
|       |                                                   |                            | OVS-43.1 | Solar panels do not deploy                                       | 2                          | 5       | 10Ensure through testing<br>Implement low-power mode in attempt to save satellite                                                        |
| VS-3  | Solar panels do not provide power                 | 2                          | OVS-3.2  | Solar panel short                                                | 1                          | 5       | 5If short occurs                                                                                                                         |
|       |                                                   |                            | OVS-7    | Satellite has incorrect attitude                                 | 1                          | 3       | 3See VS-7                                                                                                                                |
|       |                                                   |                            | OVS-44.1 | Error in optics (ie misalignment)                                | 1                          | 5       | 5Ensure through testing                                                                                                                  |
|       |                                                   |                            | OVS-4.2  | Error in data management system                                  | 1                          | 4       | Redundant C&DH systems recommended, reset system if 4occurs                                                                              |
| VS-4  | Camera images are unusable (ie blurry, corrupted) | 1                          | OVS-7    | Incorrect attitude for image collection                          | 1                          | 3       | 3See VS-7                                                                                                                                |
|       |                                                   |                            | OVS-9    | Off-nominal orbit                                                | 1                          | 4       | 4See VS-9                                                                                                                                |
|       |                                                   |                            | OVS-45.1 | Error in instrumentation (ie uncalibrated)                       | 1                          | 5       | 5Ensure through testing                                                                                                                  |
|       |                                                   |                            | OVS-5.2  | Error in data management system                                  | 1                          | 4       | Redundant C&DH systems recommended, reset system if 4occurs                                                                              |
| VS-5  | Sensor data are unusable (ie unusual, corrupted)  | 1                          | OVS-7    | Incorrect attitude for data collection                           | 1                          | 3       | 3See VS-7                                                                                                                                |
|       |                                                   |                            | OVS-9    | Off-nominal orbit                                                | 1                          | 4       | 4See VS-9                                                                                                                                |
|       |                                                   |                            | OVS-36.1 | Error in data management system                                  | 1                          | 4       | 4Redundant C&DH systems recommended                                                                                                      |
|       |                                                   |                            | OVS-6.2  | Antenna not correctly aligned                                    | 1                          | 2       | Change attitude control of satellite to restore correct 2pointing                                                                        |
| VS-6  | Satellite does not successfully transmit data     | 1                          | OVS-6.3  | Incorrect communication system configuration                     | 1                          | 3       | 3Configuration can be fixed                                                                                                              |
|       |                                                   |                            | OVS-7    | Pointing error due to incorrect attitude                         | 1                          | 3       | 3See VS-7                                                                                                                                |
|       |                                                   |                            | OVS-37.1 | Reaction wheel component failure                                 | 2                          | 4       | 8Redundant RWA implemented                                                                                                               |
|       |                                                   |                            | OVS-7.2  | Attitude sensor malfunction (ie blocked or degraded)             | 1                          | 4       | Multiple sensors implemented but in case of event, investigate algorithms and alternate control modes to 4salvage control                |
| VS-7  | Satellite has incorrect attitude                  | 1                          | &VS-7.3  | Reaction wheels saturated                                        | 1                          | 1       | 1Desaturate RWA                                                                                                                          |
|       |                                                   |                            | &VS-7.4  | ACS thruster failure                                             | 1                          | 3       | Investigate alternative thruster combos for desaturation 3and alternative control models                                                 |
|       |                                                   |                            | OVS-58.1 | Battery cell failure                                             | 1                          | 5       | 5Implement low-power mode to salvage mission                                                                                             |
|       |                                                   |                            | OVS-8.2  | Charging failure                                                 | 1                          | 5       | 5Implement low-power mode to salvage mission                                                                                             |
| VS-8  | Battery failure                                   | 1                          | OVS-8.3  | Power distribution, computer failure                             | 1                          | 4       | 4Implement contingency procedures                                                                                                        |
|       |                                                   |                            | OVS-49.1 | Orbit determination system failure                               | 1                          | 2       | 2Update algorithm                                                                                                                        |
|       |                                                   |                            | OVS-9.2  | Propulsion system failure (chemical for comms, electric for obs) | 1                          | 5       | Ensure through testing, redundancy, known good propulsion technology where redundancy isn't present 5(NSTAR)                             |

**Fig. 15** The full fault and risk analysis spreadsheet for the satellite system contains events, likelihood, consequences, and mitigation strategies.

## VI. Subsystems

### A. Payload

#### 1. Camera

To determine feasible altitudes for FISTO, we calculate the camera specifications necessary to reach the required Ground Sample Distance (GSD) of 0.5 m. MATLAB code produces the required instantaneous Field of View (iFOV) to accomplish the 0.5 m GSD for different altitudes. The results of this investigation are in Table 8.

**Table 8 Required camera iFOV at different altitudes**

| Altitude (km) | Required iFOV (microrad) |
|---------------|--------------------------|
| 50            | 10                       |
| 100           | 5                        |
| 150           | 3.3                      |
| 200           | 2.5                      |
| 300           | 1.7                      |
| 400           | 1.25                     |

The current state-of-the-art in satellite-based optics is an iFOV of 1.8 microrad accomplished by the CANON CE-SAT-1 as a technology demonstration [17]. We conclude that the maximum feasible altitude for observation at a GSD of 0.5 m is near 300 km and that ideally a lower altitude should be chosen.

This conclusion led to the design decision of creating a separate observation satellite, now FISTO, and communications constellation, now CHELL, since better GSD and better coverage are

competing interests. Better GSD requires lower altitude, and better coverage requires higher altitude.

Knowing the required GSD and selected altitude, we also select an image swath of 2.5 km to calculate the data rate. We assume a line array CCD, a common optical configuration for satellites, and a pass length of 3 minutes. We also assume a pixel bit depth of 8 bits, also common for optical data. We use these parameters to calculate the data rate necessary to downlink one 2.5 km x 2.5 km image in a 3 minute pass. The calculations show that four such images can be downloaded in a pass with a bit rate of 1.11 Mbps, which is within the bandwidth constraints of the 32 Mbps antenna.

The MATLAB script used for these calculations is the following:

```
%% Camera Specification Calculator
% inputs
desired_GSD = 0.5; % m
altitude = 50; % km
desired_swath = 2.5; %km

% calculations
% specs
required_IFOV = desired_GSD/(altitude*1000); % rad
required_pixels = desired_swath/(desired_GSD/1000);

FOV = required_IFOV*required_pixels*180/pi; % deg

% data size
% (https://4nsi.com/faq/how-do-i-calculate-the-file-size-for-a-digital-image)
bit_depth = 8;
detector_size = required_pixels * 1; % line array CCD
bits_total = bit_depth * detector_size; % bits in single line

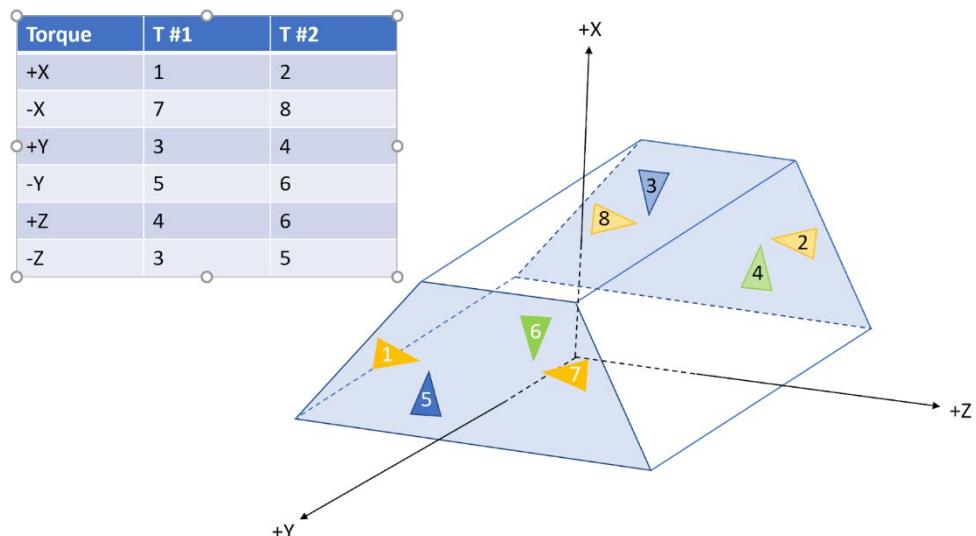
% size of swath x length km pic
desired_length = 2.5; % km
n_pics = desired_length/desired_GSD*1000;
bits_ex = bits_total*n_pics; % bits in image

% data rate
```

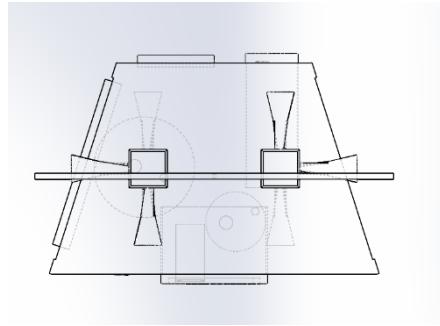
```
pass_length = 3; % min  
bps = bits_ex/(pass_length*60); % bits per second  
  
% equations used  
%GSD = altitude*required_IFOV*1000; % m  
%swath = GSD/1000*pixels; % km
```

## B. Attitude Control System

As mentioned in the attitude control section of the main report, we use four reaction control wheels to provide three-axis attitude control of the FISTO satellite. An additional propulsion system is necessary to periodically desaturate the momentum wheels without the assistance of the Earth's magnetic field. We choose hydrazine as the propellant for this attitude control system in part due to its use in other similar applications, such as the equivalent system on LRO. Based on the available momentum storage of the reaction wheels and the frequency with which momentum dumping is required, we calculate the required propellant mass for the ACS thrusters to be 63.75 kg. This value includes a 12% margin for trapped propellant, boil-off, and initial ullage.



**Fig. 16** The attitude control thrusters on the observation satellite are used for momentum desaturation and can provide torque about any of the three principal axes.



**Fig. 17 A CAD model by Nicky Masso illustrates the implementation of the predetermined ACS thruster orientations.**

We place the thrusters on the vehicle with the goal to be able to provide a moment about the center of mass in any of the three principal directions of the vehicle without creating a net translational force that would perturb the orbit. We accomplish this goal with 8 thrusters located on the sides of the vehicle, away from the scientific equipment. Figure 17 above shows our selected configuration of the attitude control thrusters, where each triangle depicts a diverging nozzle. The table next to the diagram specifies which pairs of thrusters fire at the same time to create a moment around the specified axis, as defined in the diagram.

### 1. Code for Sizing ACS Systems in Lunar Orbit

```
%%%%%%%%
%
% ADCS_sizing.m
%
% Environmental Disturbances and Attitude Determination and Control System
% sizing - this program calculates the maximum disturbance torques (gra
%%%%%%%%
%
% INPUTS
% satellite - observation, communication, or prop_depovity
% gradient torque and solar radiation pressure torque are the only ones
% calculated here due an assumption of no magnetic field torque and
% aerodynamic torque around the moon). The program uses this max torque to
% determine RWA sizing.
% Also determines the force and chemical propellant mass needed from
% thrusters for momentum dumping
%
% AAE450: Project Next Step
%%%%%%%
% Jaxon Connolly -- Controls
% r_a - spacecraft apoapsis
% r_p - spacecraft periapsis
% I_xx, I_yy, I_zz - spacecraft moments of inertia
% theta - maximum angle of deviation from z-axis
% q - reflectance factor
% i - angle of incidence from sun
% A_s - spacecraft surface area
% P - orbital period
% theta_slew - slew angle
% t_slew - time in which slew maneuvers must be performed
% L - distance from center of gravity to thrusters
% t_burn - burn time of thruster
% N_wheels - number of reaction wheels being used
% days - mission duration
% Isp - specific impulse from desired chemical propellant
%
% OUTPUTS
% T_gEarth - gravity gradient from the Earth
% T_gMoon - gravity gradient from the moon
% T_sp - torque from solar pressure radiation
% T_max - maximum projected environmental torque
% T_slew - required slew torque
% T_req - required torque for reaction wheel disturbance mediation
```

```
% h_RW - momentum storage in reaction wheel
% P_RW - reaction wheel power
% F_MD - force required by thrusters for momentum dumping
% M_p - propellant mass required by thrusters
%
% PLOTS
% none
%
% FLAGS
% none
%
% ADDITIONAL DEVELOPMENT NOTES:
% ---Equations for environmental disturbances and reaction wheel sizing
% come from ValiSpace How To's
%%%%%%%%%%%%%%%
%%
clear all;close all;clc;

% Environmental Constants
mu_earth = 398600.4418; %[km^3/s^2] Earth's gravitational parameter
mu_moon = 4902.8695;   %[km^3/s^2] Moon's gravitational parameter
r_earth_moon = 384400;  %[km] distance between Earth and Moon
J_s = 1367;             %[W/m^2] solar constant
c = 3e8;                %[m/s] speed of light
g_earth = 9.81;          %[m/s] acceleration due to earth's gravity

% Inputs
%satellite = "observation";
%satellite = "communication";
satellite = "prop_depot";

if satellite == "observation"
    r_a = 216;      %[km] imaging satellite apoapsis
    r_p = 50;       %[km] imaging satellite periapsis
    I_xx = 100;     %[kg*m^2] imaging satellite moment of inertia x-axis
    I_yy = 360;     %[kg*m^2] imaging satellite moment of inertia y-axis
    I_zz = 370;     %[kg*m^2] imaging satellite moment of inertia z-axis
    A_s = 32.4;     %[m^2] imaging satellite surface area
    P = 2*60*60;   %[s] imaging satellite orbital period
    theta_slew = 0;  %[rad] imaging satellite slew angle
    t_slew = 1;      %[s] imaging satellite time for slew maneuver
    L = 0.8;        %[m] distance from center of gravity to thrusters
    Isp = 225;      %[s] propellant specific impulse
    theta = 0.01;    %[rad] maximum angle of deviation from z-axis
elseif satellite == "communication"
```

```

r_a = 1500; %[km] communication satellite apoapsis
r_p = 1500; %[km] communication satellite periapsis
I_xx = 653.50; %[kg*m^2] communication satellite moment of inertia x-axis
I_yy = 905.56; %[kg*m^2] communication satellite moment of inertia y-axis
I_zz = 1253.90; %[kg*m^2] communication satellite moment of inertia z-axis
A_s = 27.96; %[m^2] communication satellite surface area
P = 4.5*60*60; %[s] communication satellite orbital period
theta_slew = 0; %[rad] communication satellite slew angle
t_slew = 1; %[s] communication satellite time for slew maneuver
L = 1.5; %[m] distance from center of gravity to thrusters
Isp = 225; %[s] propellant specific impulse
theta = 0.5; %[rad] maximum angle of deviation from z-axis
elseif satellite == "prop_depot"
    r_a = 100; %[km] lunar propellant depot apoapsis
    r_p = 100; %[km] lunar propellant depot periapsis
    I_xx = 2.265e7; %[kg*m^2] lunar propellant depot moment of inertia x-axis
    I_yy = 2.931e7; %[kg*m^2] lunar propellant depot moment of inertia y-axis
    I_zz = 3.389e7; %[kg*m^2] lunar propellant depot moment of inertia z-axis
    A_s = 143.314; %[m^2] lunar propellant depot surface area
    P = 89.715; %[s] lunar propellant depot orbital period
    theta_slew = 0; %[rad] lunar propellant depot slew angle
    t_slew = 1; %[s] lunar propellant depot time for slew maneuver
    L = 1.5; %[m] distance from center of gravity to thrusters
    Isp = 225; %[s] propellant specific impulse
    theta = 0.01; %[rad] maximum angle of deviation from z-axis
end

q = 0.6; % reflectance factor
inc = 0; %[rad] angle of incidence from sun
t_burn = 1; %[s] time of burn for thrusters
N_wheels = 4; % number of reaction wheels
days = 15*365; %[day] mission duration

% Initializing
orbital_rad_earth = [r_earth_moon - r_a, r_earth_moon - r_p, r_earth_moon + r_a,
r_earth_moon + r_p];
orbital_rad_moon = [r_a, r_p];

% Gravity Gradient Torque
for i = 1 : length(orbital_rad_earth)
    T_gEarth_vec(i) = 3 * mu_earth * abs(I_zz - I_yy) * sin(2 * theta) / (2 *
orbital_rad_earth(i) ^ 3); %[Nm]
end
T_gEarth_zy = max(T_gEarth_vec);
for i = 1 : length(orbital_rad_earth)

```

```

T_gEarth_vec(i) = 3 * mu_earth * abs(I_zz - I_xx) * sin(2 * theta) / (2 *
orbital_rad_earth(i) ^ 3); %[Nm]
end
T_gEarth_zx = max(T_gEarth_vec);
T_gEarth = max(T_gEarth_zy, T_gEarth_zx); % max gravity gradient due to Earth
for i = 1 : length(orbital_rad_moon)
    T_gMoon_vec(i) = 3 * mu_moon * abs(I_zz - I_yy) * sin(2 * theta) / (2 *
orbital_rad_moon(i) ^ 3); %[Nm]
end
T_gMoon_zy = max(T_gMoon_vec);
for i = 1 : length(orbital_rad_moon)
    T_gMoon_vec(i) = 3 * mu_moon * abs(I_zz - I_xx) * sin(2 * theta) / (2 *
orbital_rad_moon(i) ^ 3); %[Nm]
end
T_gMoon_zx = max(T_gMoon_vec);
T_gMoon = max(T_gMoon_zy, T_gMoon_zx); % max gravity gradient due to Moon

% Solar Radiation Pressure Torque
solar_force = J_s * A_s * cos(inc) * (1 + q) / c; %[N]
diff_cps_cg = 0.1 * A_s; % difference of center of solar pressure and center of gravity
T_sp = abs(solar_force) * diff_cps_cg; %[Nm]

% Slew Torque
T_slew = 4 * theta_slew * (I_xx + I_yy + I_zz) / t_slew ^ 2; %[Nm]

% Required Torque
T_g_max = max(T_gEarth, T_gMoon);
T_max = T_sp + T_g_max;
T_req = max((1.25 * T_max), T_slew); %[Nm]

% Required Momentum Storage
h_RW = T_req * P * 0.707 / 4; %[Nm/s]

% Required Power
P_RW = 1000 * T_req + 4.51 * h_RW ^ 0.47; %[W]

% Momentum Dump with Thrusters
% Required Force
F_MD = h_RW / L / t_burn; %[N]

% Total Impulse
I_t = t_burn * N_wheels * days;

% Propellant Mass
M_p = I_t * F_MD / Isp / g_earth; %[kg]

```

## 2. Equations of Motion

The following is a code to propagate the equations of motion for the communications satellites.

The logic has been inspired by Sadie Holbert's code from AAE 450: Project Artemis in 2014 [48].

The moments of inertia in each direction, the sum of the disturbance torques, the rotation rate of the moon, and the required reaction wheel momentum storage are all defined as global variables in the beginning of the code so that they can be used in another function called later in the main code. The values for the moments of inertia, the sum of anticipated disturbance torques, and the required reaction wheel momentum storage are all taken from the reaction wheel sizing code created by Jaxon, found in Appendix VI.B.1. Next, the initial spin rates and angle perturbations are set. The initial spin rates are determined by the dimensions and weight distribution of the communications satellite itself and the orientation of the satellite with respect to the moon. The farther away from the center of gravity of the satellite a force can be applied, the more potential for spin it has, and the harder to stabilize it will be. For this reason, there is more initial spin in the direction of the longest dimension of the satellite. Additionally, the gravitational pull of the moon is nonnegligible for angle perturbations of this magnitude and will depend on the orientation of the satellite. Effects of the gravity gradient are included in the reaction wheel sizing code.

Next, the numeric integration is performed. The initial spin rates and angle perturbations are used as initial conditions. This allows the attitude control system response to be observed. The simulation runs for 10,000 seconds. MATLAB's ode45 function is used as the numeric integration method. The input function is the equations of motion that define all three angular velocities and all three rotation angles. Because this is complex, these equations are stored in a separate function, called here. The ode45 function returns a matrix with 10,000 rows and 6 columns. The solution matrix is parsed to separate each angular velocity and rotation angle into its own array. Finally,

the solutions are plotted against time. The angular velocities are all plotted on the same figure and can be seen in Fig. 16 in the main report. The rotation angles are plotted separately as subplots and can be seen in Fig. 17 in the main report.

```
%Monica Viz
%AAE 450 Satellite EOMs code

global Ixx Iyy Izz M omega0 hRW %CONSTANTS
Ixx = 636.72; %kg*m^2
Iyy = 758.7; %kg*m^2
Izz = 1284.97; %kg*m^2
M = 0.0021982; %N*m, sum of disturbance torques
hRW = 6.2943; %required RW momentum storage

omega0 = 1.389e-4*pi/180; %rad/s, rotation rate of the moon
w0 = [omega0/5 omega0/8 omega0/13]; %rad/s, initial spin rates
ang0 = [.1 .2 .5]*pi/180; %initial angle perturbation

%Numeric Integration
IC = [w0 ang0];
tspan = [0,10000];
[t, x] = ode45(@attitude, tspan, IC);

wx = x(:,1);
wy = x(:,2);
wz = x(:,3);
phi = x(:,4);
theta = x(:,5);
psi = x(:,6);

%Plots
plot(t,wx,'r',t,wy,'b',t,wz,'g')
title('Body-frame Angular Velocities v. Time')
ylabel('Angular Velocity (rad/s)')
xlabel('Time (s)')
legend('wx','wy','wz')

figure(2)
subplot(3,1,1)
plot(t,phi*180/pi, 'r')
title('Roll (phi)')
xlabel('Time (s)')
ylabel('Angle (deg)')
```

```

figure(2)
subplot(3,1,2)
plot(t, theta*180/pi, 'b')
title('Pitch (theta)')
xlabel('Time (s)')
ylabel('Angle (deg)')

figure(2)
subplot(3,1,3)
plot(t, psi*180/pi, 'g')
title('Yaw (psi)')
xlabel('Time (s)')
ylabel('Angle (deg)')

```

The following function is called in the main code as the `ode45` function input and defines the actual equations of motion. The equations are taken from Purdue University AAE 421: Flight Dynamics and Controls lecture slides by R. Dai [49].

The code is built on the assumption that the products of inertia,  $I_{xy}$ ,  $I_{yz}$ , and  $I_{zx}$ , are all zero. For this to be the case, there must be symmetry in the  $xy$ -,  $yz$ -, and  $zx$ -planes – a reasonable assumption for our communications satellites. It is noted that the reaction wheel required storage is not used for the equation for angular velocity in the  $y$ -direction, the direction moving forward along the orbital path of the satellite. The function returns a column vector of length 6 that the `ode45` function uses as input.

```

function yd = attitude(~,y)

global Ixx Iyy Izz M omega0 hRW

wx = y(1);
wy = y(2);
wz = y(3);
phi = y(4);
theta = y(5);
psi = y(6);

```

```
wxd = (M - (Izz-Iyy*wy*wz - hRW*wz)/Ixx;  
wyd = (M - (Ixx-Izz*wx*wz)/Iyy;  
wzd = (M - (Iyy-Ixx*wx*wy + hRW*wx)/Izz;  
phid = wx*cos(psi) - wy*sin(psi);  
thetad = wx*sin(psi)/cos(phi) + wy*cos(psi)/cos(phi) - omega0;  
psid = wx*sin(psi)*tan(phi) + wy*cos(psi)*tan(phi) + wz;  
  
yd = [wxd;wyd;wzd;phid;thetad;psid];
```

### *3. Attitude Control Scripts Mathematical Breakdown*

The kinematic equations of motions are modelled according to a Body-Two 1-3-1 sequence, they are numerically integrated to yield how angular velocities and Euler parameters evolve over time. This information helps demonstrate how the attitudes of the satellite are changing according to the body's moment of inertia (dyadic within model) along with its rotational motion. To visualize this information the precession and nutation angles are tabulated from the Euler parameter equations after the respective direction cosine matrix is formulated. Applying the components of the direction cosine matrix allows for the nutation, precession, and spin angles to be calculated. To correctly calculate the nutation and precession angles one must compare the quadrant feasibility of inverse cosine and inverse sine angles to ensure meeting their constraints. The correct nutation and precession angles have been found are tabulated because of the while loop within the script. The while loop is responsible for testing whether the precession angles exceed their quadrant limits. Since this is an axis-symmetric rotation the spin angle can be calculated but will not be analyzed. For the purposes of this model, the spin angle remains constant as the nutation and precession angles are changing. Due to the disturbance torque present, the behavior of the angles is non-periodic suggesting an unsteady attitude for the body.

The plots of nutation and precession angles with the total disturbance torque applied can be interpreted to understand the attitudes of the satellite. The bottom limit of the precession graph represents the first undefined angle due to division by zero when nutation angle ( $\kappa$ ) = 0 (degrees), this is a singularity (Fig. 18). Ensuring the inputs of the program are correct is essential as it should not yield any singularities. Another constraint check for singularities is ensuring the magnitude of the four quaternions yield 1. To graphically represent this Fig. 19 depicts the quaternions tabulated within the analysis with limits of -1 and 1 thus satisfying the constraint. The

angle does not return to its initial condition meaning the applied external torque causes non-periodic behaviors to the cuboid satellite.

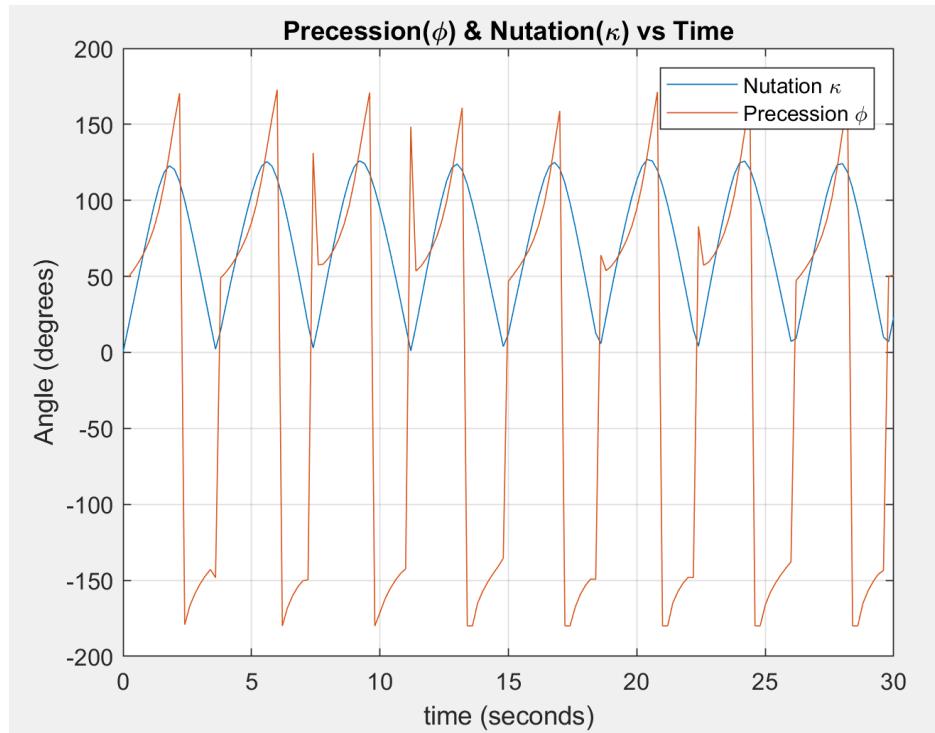
A way to visualize the perturbations within the plot for nutation angle is swinging between Min-Max of graph, these rises and drops within the graph depict the satellite altering its attitude with respect to the axis of rotation of nutation and/or precession. The same type of visualization is possible for the precession angle even though the values skew slightly.

#### *4. Nutation and Precession Angle Visualization Considerations*

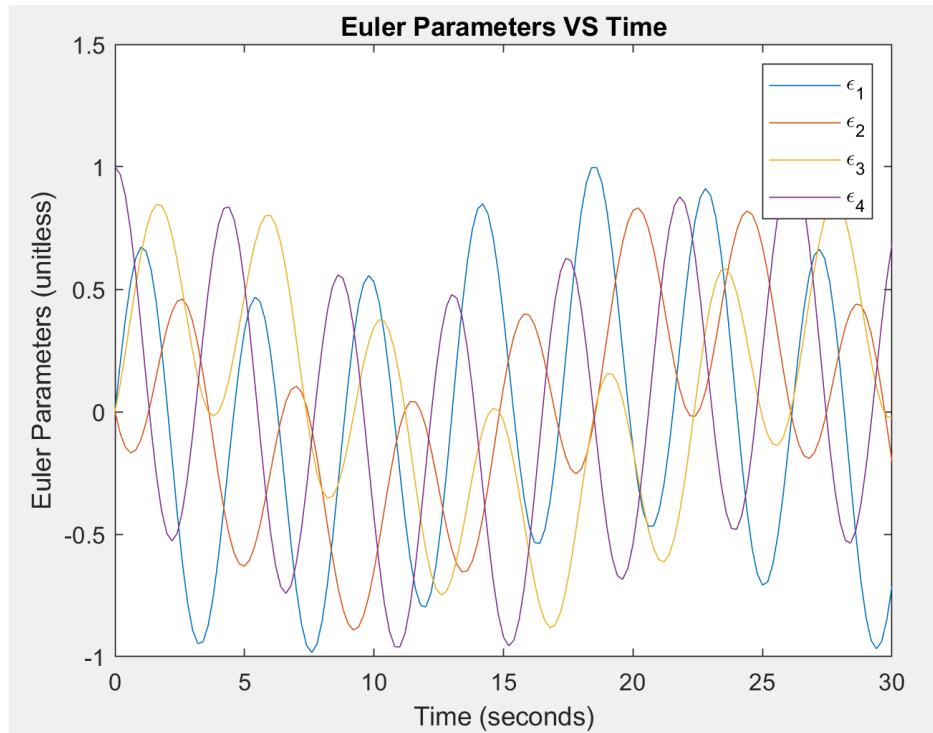
Much like the precession plot, nutation exhibits repeating behavior. The nutation angle starts at 0 as  $\cos(0) = 0$ . Meaning, at the models starting point, the inertial frame is aligned with the body frame (as shown in Fig. 14 in the main report). For the nutation angle, the sudden peak within Fig. 24 corresponds to the torque applied parallel to  $\hat{\mathbf{n}}_3$  to rotate  $\hat{\mathbf{n}}_1$ . Visualizing the motion, it would act as “up and down” movements rotating  $\hat{\mathbf{y}}_1$ . Due to the non-periodic behavior due to the torque the peak of nutation changes but does not return to 0. Any time the nutation angle is equal to 0 the precession plot will highlight a singularity due to dividing by 0 as mentioned before.

The direction cosine matrix plots of C<sub>21</sub>-C<sub>31</sub> (Fig. 25) depicts the satellite attitude projected into the  $\hat{\mathbf{n}}_2$ - $\hat{\mathbf{n}}_3$  plane as viewed from  $\hat{\mathbf{n}}_1$  for a “certain time” of t=10 seconds. The arbitrary time was chosen arbitrarily to explain what information can be extracted from this graph. The path in this plot corroborates the other assumptions regarding visualizing the motion. From the C<sub>21</sub>-C<sub>31</sub> graph the nutation angle can also be described as  $\sin(\kappa) = \sqrt{C_{21}^2 - C_{31}^2}$ . Designating that quantity as h to check for correct angles yields the correct results within the MATLAB script. Since the plots themselves do not address quadrant ambiguities the exact nutation angle cannot be calculated, however, it approximates the nutation angle and should be implemented to check feasibility of solution at the “certain instant”.

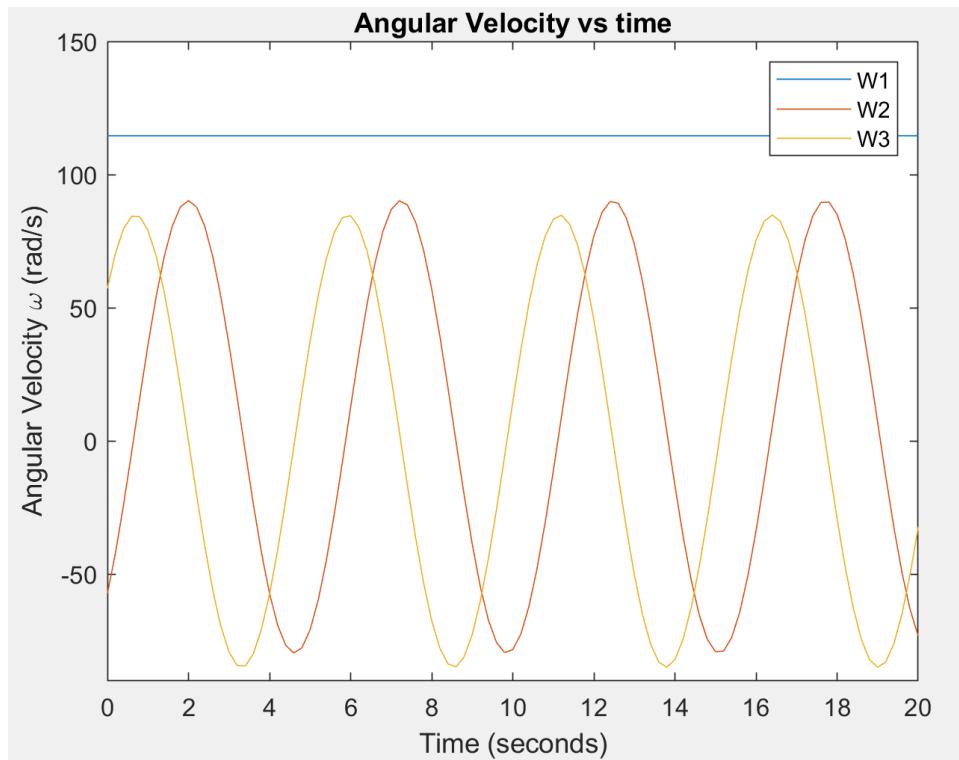
Figures 24-27 within the MATLAB script output the nutation and precession angles evolving over time due to the given total disturbance torques simulated. The satellite attitude dynamics were interpreted for when the observations satellite experiences no disturbance torques, the tabulated external torques for this given satellite, and a disturbance torque above the body's limits. These three scenarios are within the program to test the bounds of how the attitudes react to the total disturbance torque. As Figs. 22 and 23 demonstrate, with no disturbance torque, the nutation and precession angles remain steadily periodic over the simulation time. Next, within Figs. 24 and 25, when the total tabulated disturbance torque is implemented, we can begin to see alteration in nutation and precession angles. These angles are being perturbed due to the external forces the body experiences due to disturbance torque. Although the response is not ideal, a controller can be implemented to mediate the nutation and precession angle fluctuations. Lastly, when the torque surpasses the limit of the satellite design, the nutation and precession angles become erratic and alter attitudes with no pattern as Figs. 26 and 27 suggest with multiple peaks and troughs being very different. With these three iterations enough information about the nutation and precession angles should be recorded as raw data to be implemented into a controller of choice to moderate the perturbations.



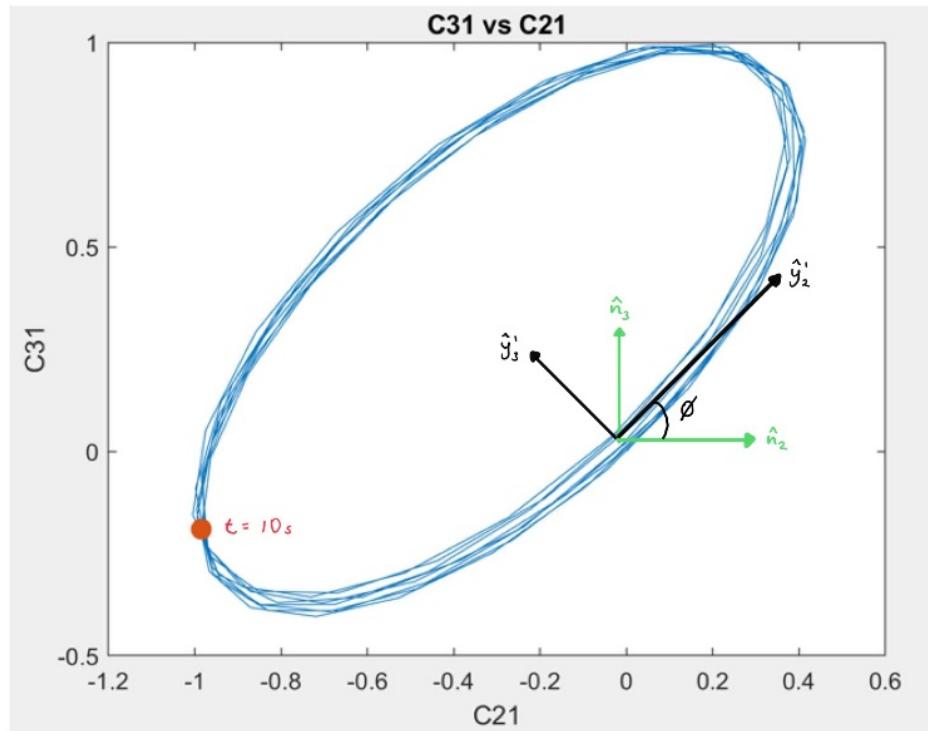
**Fig. 18 Nutation & Precession angles plotted together making the angle quadrant constraints evident for mathematical analysis.**



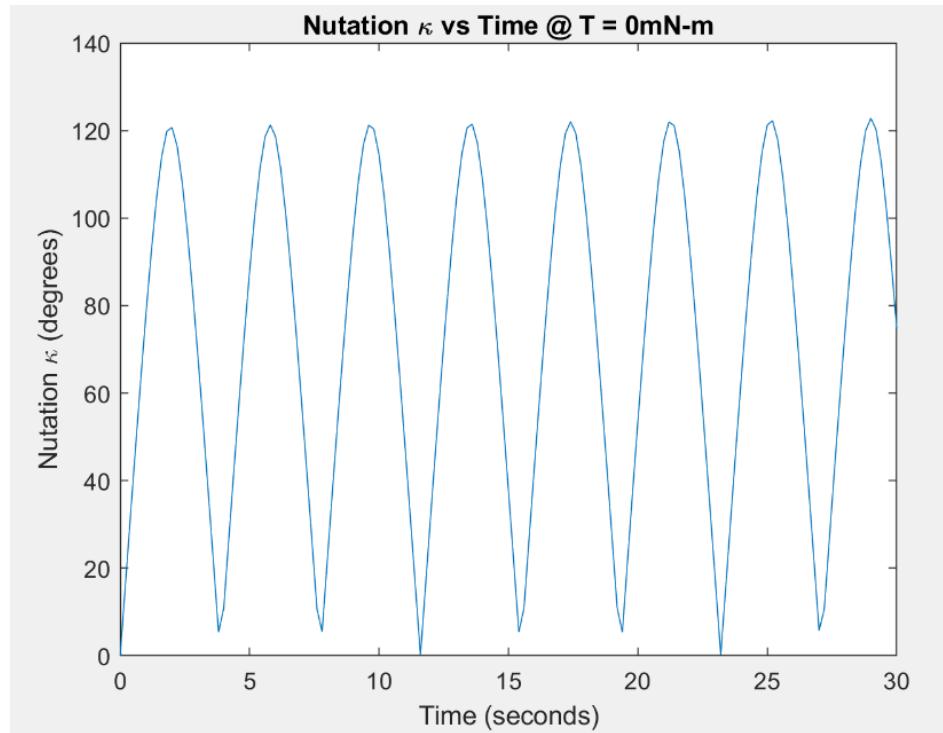
**Fig. 19** This plot determines whether the Euler Parameter constraint to be between the values of 1 or -1. This plot re-confirms the mathematics within the model have followed the correct assumptions.



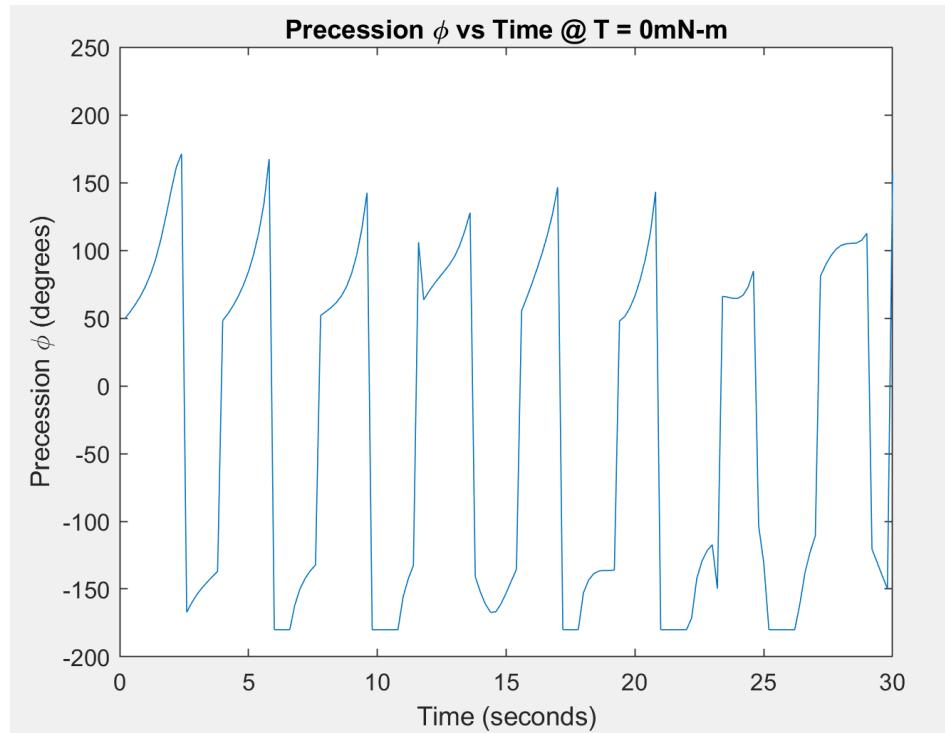
**Fig. 20** Demonstrates angular velocity remaining constant along the spin angle axis which remains consistent with the initial assumptions of the problem.



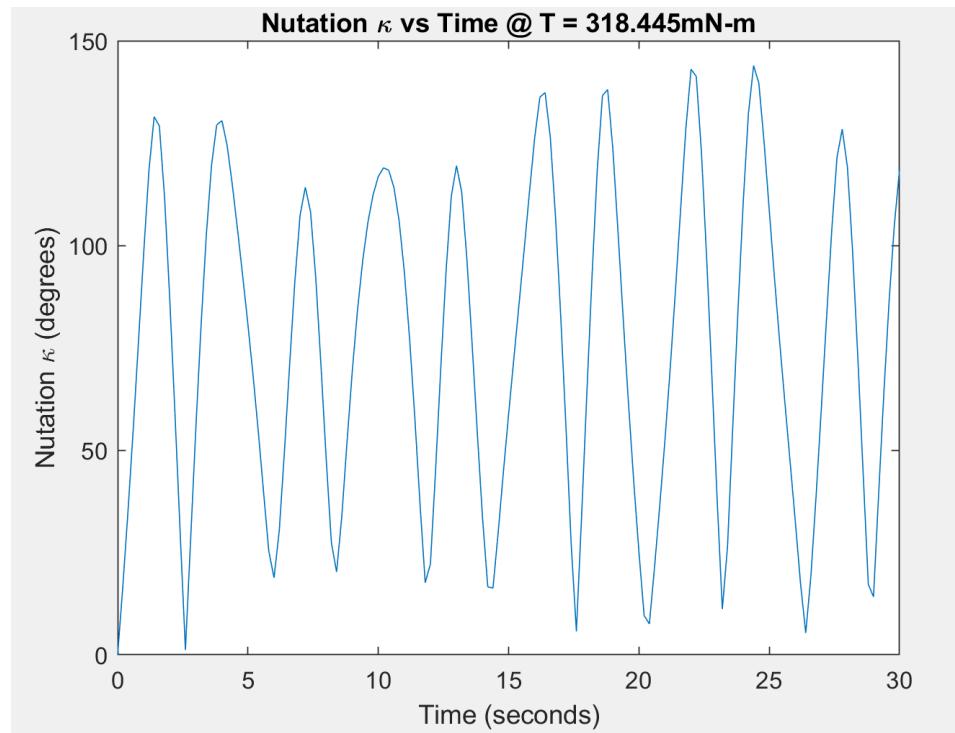
**Fig. 21** Depicts the axes rotation with respect to torque to demonstrate in which ways the attitude of the inertial frames changes due to perturbation forces.



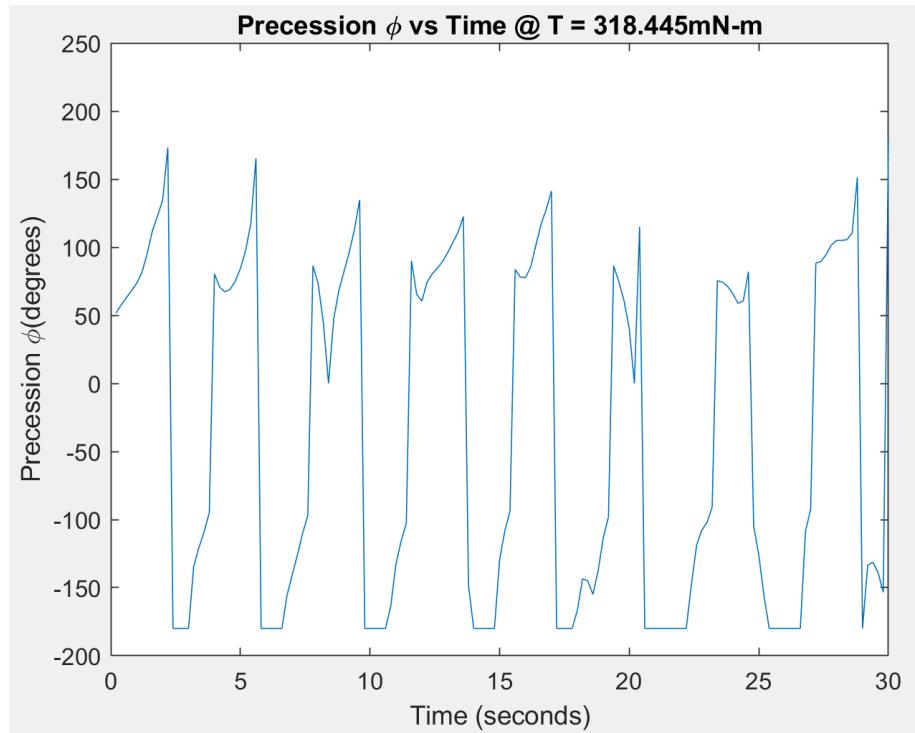
**Fig. 22** Illustrates nutation angle remaining periodic when no torque is present within the simulation.



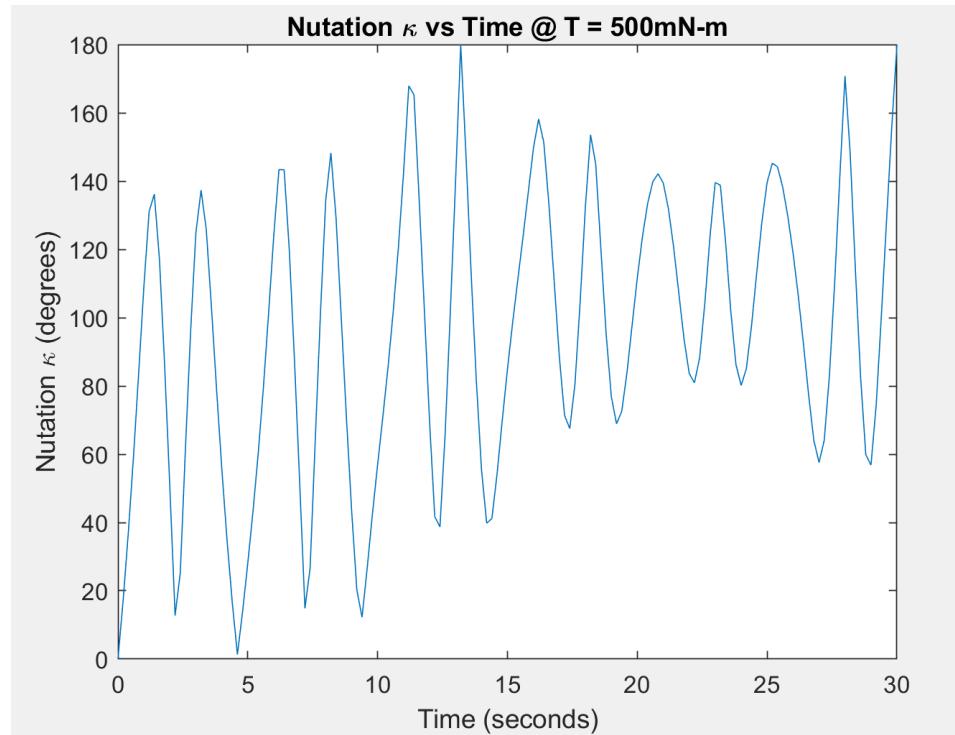
**Fig. 23 Precession angle evolution is based on more complicated quadrant conditions and sees more fluctuation with respect to disturbance torques.**



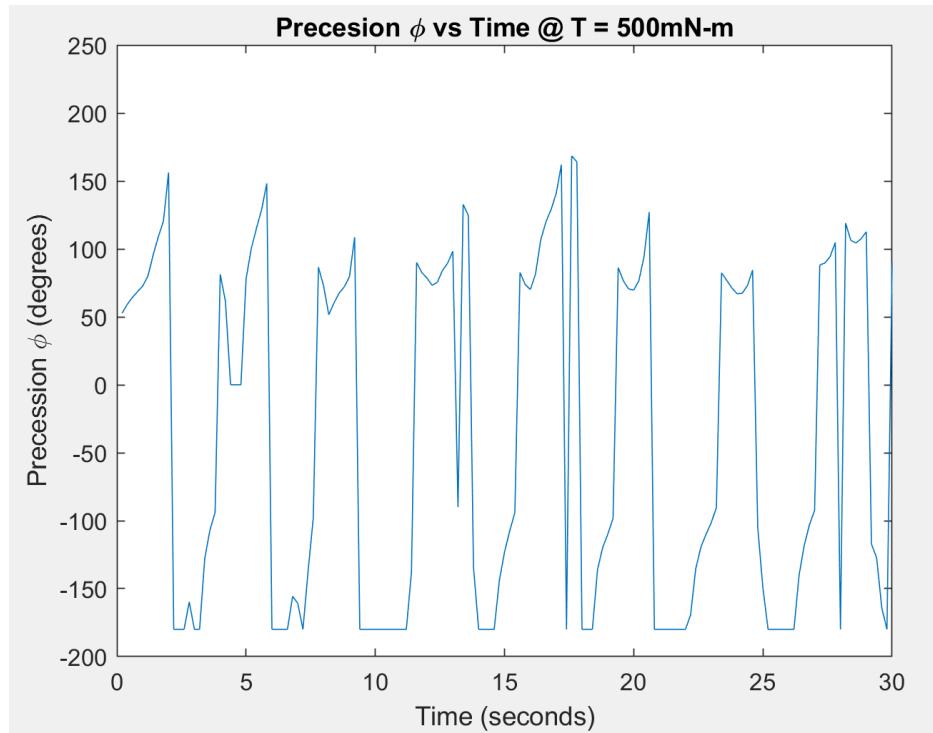
**Fig. 24** Demonstrates evolution of nutation angle as a disturbance torque is applied to the observation's satellites.



**Fig. 25 Suggests how the precession angle will fluctuate according to the disturbance torques inputted.**



**Fig. 26 Suggests how nutation angle would react if the torque limit for the satellite is surpassed without additional attitude control systems.**



**Fig. 27** Demonstrates the precession angle erratic behavior as peaks and troughs become harder to identify and there is no pattern as to how precession angle behaves.

### 5. Attitude Dynamics MATLAB Script

```
%%
% Cardano, Roberto
% AAE450 Attitude Dynamics Program Description: This program is utilized to
determine the precession and nutation angles of an axis-symmetric body that is modeled
after a cuboid satellite. Assuming this rigid body is fixed about an axis, a cuboid is
utilized to determine the inertia dyadic of the rotations. This will take under
consideration the perturbation forces the cuboid experiences according to the input total
disturbance torque of the program. This disturbance torque is comprised of the maximum
gravity gradient due to the earth and moon, alongside the solar radiation pressure torque
as well. The model itself is based on the Body-Two 1-3-1 angle sequence so that the
dynamics may be analyzed and calculated. Next, in order to understand the perturbations,
example numbers are input at a "certain instant" for the initial conditions. These initial
conditions include arbitrary angular velocities, initial Euler parameters, constraint K to
check the feasibility of the Euler parameter computations, and the total disturbance
torque. To make the body perturbed the "certain instant" will be analyzed with respect
to a Space 2-3-1 angles and Body 1-3-1, this allows the script to depict how the torque
is affecting a satellite in motion. Using DCM construction and Euler parameters the
```

orientations and perturbations of the satellite can be better understood applying their equations of motions to determine nutation and precession angle reactions to torque. This model can be implemented to suggest control models/parameters to offset unwanted perturbations while in orbit.

```
%%
```

```
clear
```

```
clc
```

```
%Observations Satellite Inputs
```

```
%The numbers provided below were calculated by other observation satellites
```

```
%team members within AAE450, these numbers were inputted to make this
```

```
%analysis possible.
```

```
%m = 287.1; %kg, dry mass % mdrycont = 344.6; %kg, dry mass with 20% overshoot  
contingency % mwet = 371.0; %kg, wet mass % mwetcont = 445.2; %kg, wet mass with  
20% overshoot contingency
```

```
%h = 1.1; %meters, height of cuboid model
```

```
%w = 1.7; %meters, width of cuboid model
```

```
%d = 1.5; %meters, depth/length of cuboid model
```

```
%At first, contingency weights were going to be inputted for multiple simulations, but  
the overall scope
```

```
%project changed, so the model only took into account the dry mass estimate
```

```
%for the observations satellite (other masses can be input)
```

```
%Calculations for inertia matrix and mass considerations
```

```
%Il = (m * (h^2*w^2 + w^2*d^2 + d^2*h^2))/(6 * (h^2 + w^2 + d^2)); %moment of  
inertia of a solid cuboid (axis of rotation at longest diagonal)
```

```
%Id = 1/12 * m * (h^2 + w^2); %moment of inertia of a solid cuboid (axis of rotation at  
the depth)
```

```
%Iw = 1/12 * m * (h^2 + d^2); %moment of inertia of a solid cuboid (axis of rotation at  
the width)
```

```
%Moment of inertia matrix for given mass, includes [inertia longest
```

```
%diagonal, inertia depth, and inertia width]
```

```
%Idry = [95.87 0 0; 0 98.1 0; 0 0 82.78]; %kgm^2
```

```
%Idrycon = [115.06 0 0; 0 117.74 0; 0 0 99.36]; %kgm^2
```

```
%Iwet = [123.88 0 0; 0 126.76 0; 0 0 106.97]; %kgm^2
```

```
%Iwetcon = [148.66 0 0; 0 152.1 0; 0 0 128.37]; %kgm^2
```

```
%Initialization
```

```
% Given (Simplified Cuboid Model Inertias)
```

```
I = 95.87; %kgm^2, inertia along the longest diagonal (dry mass inertia)
```

```
J = 98.1; %kgm^2, inertia along depth of cuboid (dry mass inertia)
```

```
W1_0 = 2; %rad/s, initial arbitrary angular velocity the cuboid satellite in n1-direction
```

```

W2_0 = -1; %rad/s, initial arbitrary angular velocity the cuboid satellite in n2-direction
W3_0 = 1; %rad/s, initial arbitrary angular velocity the cuboid satellite in n3-direction
E1_0 = 0; %quaternion initial conditions (Euler parameters)
E2_0 = 0; %quaternion initial conditions (Euler parameters)
E3_0 = 0; %quaternion initial conditions (Euler parameters)
E4_0 = 1; %quaternion initial conditions (Euler parameters)
K0 = 1; %constant constraint checking value for Euler parameters
M = 0.318445; %Nm, calculated total torque acting on the observations satellite
(includes maximum gravity gradient torque due to the earth/moon, and solar radiation
pressure torque all tabulated within the reaction wheel environment analysis.

%Calculations of precession, nutation, and spin angle
%Assuming an ideal cuboid, axis-symmetric satellite, next the program takes tabulated
inertia dyadic to perform DCM Construction, EOMs derived according to Body and
Space angles, Euler Parameters solutions for given problem
%EOM
time = 0:0.2:30;
options = odeset('RelTol',1e-3,'AbsTol',1e-3);
initial = [W1_0 W2_0 W3_0 E1_0 E2_0 E3_0 E4_0];
[~, output] = ode45(@EOM, time, initial, options);
W1 = output(:,1);
W2 = output(:,2);
W3 = output(:,3);
E1 = output(:,4);
E2 = output(:,5);
E3 = output(:,6);
E4 = output(:,7);
K = sqrt(E1.^2 + E2.^2 + E3.^2 + E4.^2);

% DCM Construction, Using quaternion method the program calculates the
% appropriate Euler Parameters for the rotation according to angular
% velocity parameters
E = [E1 E2 E3];
C11 = 1-2*E(:,2).^2-2*E(:,3).^2;
C12 = 2*(E(:,1).*E(:,2) - E(:,3).*E4);
C13 = 2*(E(:,3).*E(:,1) + E(:,2).*E4);
C21 = 2*(E(:,1).*E(:,2) + E(:,3).*E4);
C22 = 1 - 2*E(:,3).^2 - 2*E(:,1).^2;
C23 = 2*(E(:,2).*E(:,3) - E(:,1).*E4);
C31 = 2*(E(:,3).*E(:,1) - E(:,2).*E4);
C32 = 2.* (E(:,2).*E(:,3) + E(:,1).*E4);
C33 = 1 - 2*E(:,1).^2 - 2*E(:,2).^2;

%Implementing the rotation matrix found and Euler parameters, now the
%precession, nutation, and spin angle can be calculated

```

```
% Angle Calculations
%Nutation
Nutation = acosd(C11);

%Precession (two formulas are used to provide correct quadrant for angles)
Precession = acosd((C21 ./ sind(Nutation)));
C31test = sind(Precession).*sind(Nutation);
%while loop is implemented to perform quadrant checks for precession angle
%calculations
k = 1;
while k < 151
    if sign(C31test(k)) ~= sign(C31(k))
        Precession(k) = -1* Precession(k);
    else
        end
    k = k+1;
end

%Spin (two formulas are used to provide correct quadrant for angles)
Spin = asind(C13 ./ sind(Nutation));
C13test = sind(Nutation).*sind(Spin);
k = 1;
%while loop is implemented to perform quadrant checks for spin angle
%calculations
while k < 151
    if sign(C13test(k)) ~= sign(C13(k))
        Spin(k) = 180 - Spin(k);
    else
        end
    k = k+1;
end

%Check whether outputted numbers are feasible
%angle1 = atand(C31(mark1)./ C21(mark1));
%(Precession(mark1)); numbers match, reason for commenting

%C31^2 + C21^2 : Calculations to yield nutation angle from precession
%analysis, important way to check whether analysis makes physical sense and
%numbers match
%h_1 = sqrt(C21(mark1).^2 + C31(mark1).^2);
%Check whether outputted numbers are feasible
%gamma1 = asind(h_1);
%Nutation(mark1);

%Plots of precession, nutation, and spin angle
```

```
%Plots
figure(1)
plot(time, Precession)
title('Precession \phi vs. Time')
xlabel('Time (Seconds)')
ylabel('Precession \phi (Degrees)')

figure(2)
plot(time, Nutation)
title('Nutation \kappa vs. Time')
xlabel('Time (Seconds)')
ylabel('Nutation \kappa (Degrees)')

figure(3)
plot(time, Nutation)
hold on
plot(time, Precession)
xlabel('time (seconds)')
ylabel('Angle (degrees)')
title('Precession(\phi) & Nutation(\kappa) vs Time')
grid on
legend('Nutation \kappa', 'Precession \phi')

%In order to check how feasible the solution is the Euler parameters are
%graphed and do not exceed 1 or -1 which follows the mathematical
%constraint the quaternions have to meet in order to not have any
%singularities occur ( $E1^2 + E2^2 + E3^2 + E4^2$ ) $^{.5}$ 
figure(4)
plot(time, E1, time, E2, time, E3, time, E4)
title('Euler Parameters VS Time')
xlabel('Time (seconds)')
ylabel('Euler Parameters (unitless)')
legend('epsilon_1', 'epsilon_2', 'epsilon_3', 'epsilon_4')

%Plots 21-31 matrix coordinates to determine spin orientation, and depicts
%how the satellite is rotating according to its direction cosine matrix.
%This allows for understanding how the body moves along the axes being
%taken into account.
figure(5)
plot(C21, C31)
hold on
title('C31 vs C21')
xlabel('C21')
ylabel('C31')
mark1 = find(time == 10);
```

```

plot(C21(mark1), C31(mark1), '.', 'MarkerSize', 30)

%Plots the angular velocities tabulated throughout the analysis to depict
% at what velocities the perturbations are happening to the cuboid model
% satellite
figure(6)
plot(time, rad2deg(W1), time, rad2deg(W2), time, rad2deg(W3))
title("Angular Velocity vs time")
xlabel('Time (seconds)')
ylabel('Angular Velocity \omega (rad/s)')
legend('W1', 'W2', 'W3')
axis([0 20 -90 150])

%Next, the script is testing different torques to comprehend how the
%attitude shifts according to the nutation angle being altered, when
%nutation angle is utilized for this analysis ADD

% Zero-Torque simulation of nutation and precession angle analysis
[~, output] = ode45(@EOM2, time, initial, options);
E1 = output(:,4);
E2 = output(:,5);
E3 = output(:,6);
E = [E1 E2 E3];
C11 = 1-2*E(:,2).^2- 2*E(:,3).^2;
C21 = 2*(E(:,1).*E(:,2) + E(:,3).*E4);
C31 = 2*(E(:,3).*E(:,1) - E(:,2).*E4);
Nutation = acosd(C11);
Precession = acosd((C21 ./ sind(Nutation)));
C31test = sind(Precession).*sind(Nutation);
%while loop is implemented to perform quadrant checks for precession angle
%calculations
k = 1;
while k < 151
    if sign(C31test(k)) ~= sign(C31(k))
        Precession(k) = -1* Precession(k);
    else
        end
    k = k+1;
end

%Plotting Zero Torque Nutation/Precession Angle
figure
plot(time, Nutation)
title('Nutation \kappa vs Time @ T = 0mN-m')
xlabel('Time (seconds)')

```

```

ylabel('Nutation \kappa (degrees)')

figure
plot(time, Precession)
title('Precession \phi vs Time @ T = 0mN-m')
xlabel('Time (seconds)')
ylabel('Precession \phi (degrees)')
axis([0 30 -200 250])

% Observations Satellite Torque simulation of nutation and precession angle analysis
[~, output] = ode45(@EOM3, time, initial, options);
E1 = output(:,4);
E2 = output(:,5);
E3 = output(:,6);
E = [E1 E2 E3];
C11 = 1-2*E(:,2).^2- 2*E(:,3).^2;
Nutation = acosd(C11);
Precession = acosd((C21 ./ sind(Nutation)));
C31test = sind(Precession).*sind(Nutation);
%while loop is implemented to perform quadrant checks for precession angle
%calculations
k = 1;
while k < 151
    if sign(C31test(k)) ~= sign(C31(k))
        Precession(k) = -1* Precession(k);
    else
        end
    k = k+1;
end

%Plotting tabulated Torque on observations satellite nutation and precession Angle. This
%graph is taking into account the calculated solar radiation pressure
%torque, the maximum gravity gradient torque due to the moon and earth
figure
plot(time, Nutation)
title('Nutation \kappa vs Time @ T = 318.445mN-m')
xlabel('Time (seconds)')
ylabel('Nutation \kappa (degrees)')

figure
plot(time, Precession)
title('Precession \phi vs Time @ T = 318.445mN-m')
xlabel('Time (seconds)')
ylabel('Precession \phi(degrees)')
axis([0 30 -200 250])

```

```
% Torque exceeding momentum storage simulation of nutation & precession angle
% analysis. This plot is implemented as an upper limit to depict how the
% attitude of the cuboid can go out of control when the torque exceeds the
% limits of the spacecraft
[~, output] = ode45(@EOM4, time, initial, options);
E1 = output(:,4);
E2 = output(:,5);
E3 = output(:,6);
E = [E1 E2 E3];
C11 = 1-2*E(:,2).^2- 2*E(:,3).^2;
Nutation = acosd(C11);
Precession = acosd((C21 ./ sind(Nutation)));
C31test = sind(Precession).*sind(Nutation);
%while loop is implemented to perform quadrant checks for precession angle
%calculations
k = 1;
while k < 151
    if sign(C31test(k)) ~= sign(C31(k))
        Precession(k) = -1* Precession(k);
    else
        end
    k = k+1;
end

figure
plot(time, Nutation)
title('Nutation \kappa vs Time @ T = 500mN-m')
xlabel('Time (seconds)')
ylabel('Nutation \kappa (degrees)')

figure
plot(time, Precession)
title('Precession \phi vs Time @ T = 500mN-m')
xlabel('Time (seconds)')
ylabel('Precession \phi (degrees)')
axis([0 30 -200 250])

% Functions of EOMs according to Body-Two 1-3-1 & Space-Two 2-3-1, with
% inputted torque values in order to graph effects of torque on the
% nutation and precession angles.
function xdot = EOM(~, x)
M = 0.318445;
xdot(1) = 0;
xdot(2) = 0.6*x(1)*x(3);
```

```

xdot(3) = (45/400) - 0.6*x(1)*x(2);
xdot(4) = 1/2*(x(1)*x(7) - x(2)*x(6) + x(3)*x(5));
xdot(5) = 1/2*(x(1)*x(6) + x(2)*x(7) - x(3)*x(4));
xdot(6) = 1/2*(-x(1)*x(5) + x(2)*x(4) + x(3)*x(7));
xdot(7) = -1/2*(x(1)*x(4) + x(2)*x(5) + x(3)*x(6));
xdot = xdot';
end

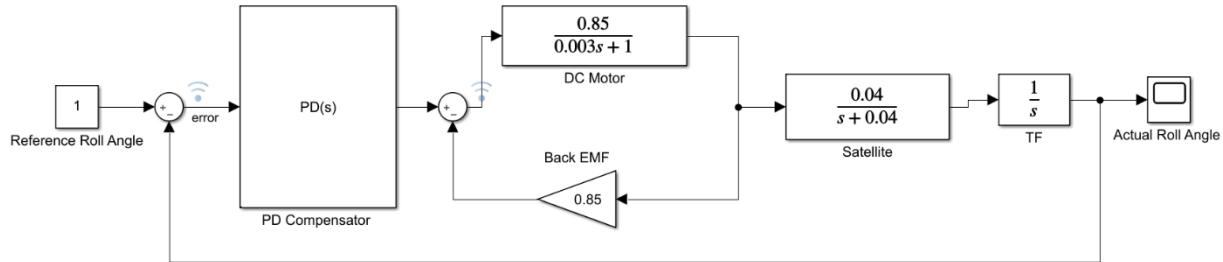
function xdot = EOM2(~, x)
M = 0;
xdot(1) = 0;
xdot(2) = 0.6*x(1)*x(3);
xdot(3) = (M/400) - 0.6*x(1)*x(2);
xdot(4) = 1/2*(x(1)*x(7) - x(2)*x(6) + x(3)*x(5));
xdot(5) = 1/2*(x(1)*x(6) + x(2)*x(7) - x(3)*x(4));
xdot(6) = 1/2*(-x(1)*x(5) + x(2)*x(4) + x(3)*x(7));
xdot(7) = -1/2*(x(1)*x(4) + x(2)*x(5) + x(3)*x(6));
xdot = xdot';
end

function xdot = EOM3(~, x)
M = 318.445;
xdot(1) = 0;
xdot(2) = 0.6*x(1)*x(3);
xdot(3) = (M/400) - 0.6*x(1)*x(2);
xdot(4) = 1/2*(x(1)*x(7) - x(2)*x(6) + x(3)*x(5));
xdot(5) = 1/2*(x(1)*x(6) + x(2)*x(7) - x(3)*x(4));
xdot(6) = 1/2*(-x(1)*x(5) + x(2)*x(4) + x(3)*x(7));
xdot(7) = -1/2*(x(1)*x(4) + x(2)*x(5) + x(3)*x(6));
xdot = xdot';
end

function xdot = EOM4(~, x)
M = 500;
xdot(1) = 0;
xdot(2) = 0.6*x(1)*x(3);
xdot(3) = (M/400) - 0.6*x(1)*x(2);
xdot(4) = 1/2*(x(1)*x(7) - x(2)*x(6) + x(3)*x(5));
xdot(5) = 1/2*(x(1)*x(6) + x(2)*x(7) - x(3)*x(4));
xdot(6) = 1/2*(-x(1)*x(5) + x(2)*x(4) + x(3)*x(7));
xdot(7) = -1/2*(x(1)*x(4) + x(2)*x(5) + x(3)*x(6));
xdot = xdot';
end

```

### 6. Attempted PD-Compensator Implementation

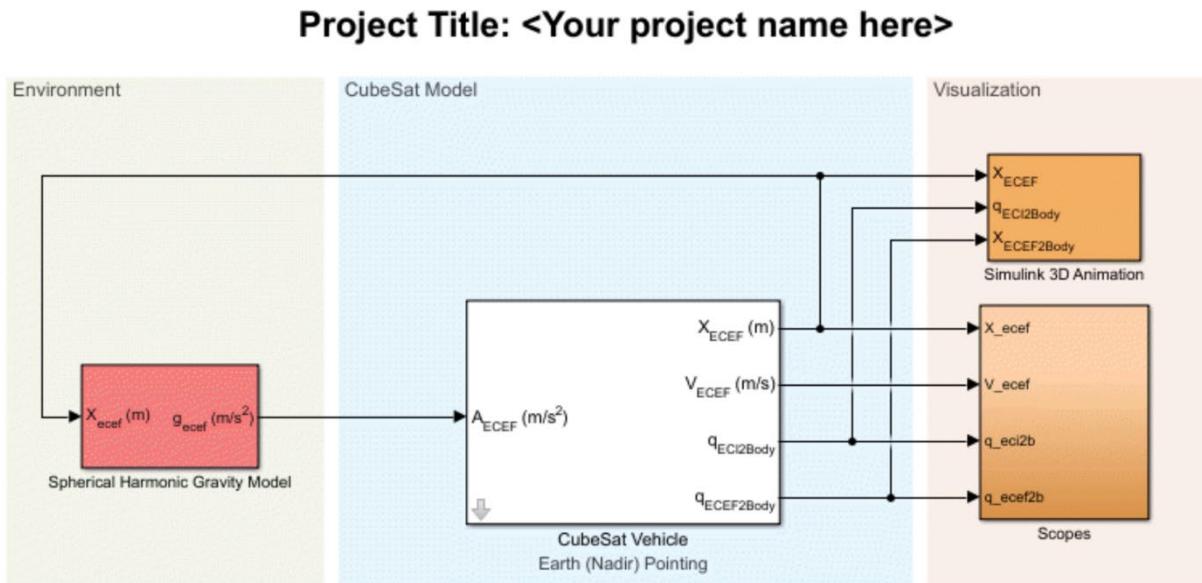


**Fig. 28 Attempted Simulink model of ACS for NPSAT satellite implementing a PD Compensator to control fluctuations in dynamics.**

The problem I pursued was creating an Attitude Control System (ACS) responsible for maintaining satellites in their prescribed orbit from the perturbed orbit. It is necessary to control the orbit of a satellite to reduce the oscillation it experiences due to the perturbation forces. I worked with the satellites team, yet, the necessary numbers were not available in time (inertial matrix), which led me to implement the NPSAT-1 (Naval Postgraduate School Satellite) as an example [18]. The Proportional Derivative Compensator was suggested by the majority of sources to control attitude dynamics and pointing accuracy [19]. A set of specifications are set for a “good controller” that the Simulink model tuning must match once the analysis is complete, this all happens within a feedback loop. The model finds the dominant poles, solves the Root Locus and Phase Margin to check stability of the solution. Then, in order to determine the Dynamic attributes three MATLAB Toolboxes are implemented: Satellite Attitude Determination, Prediction, and Control with inputs from NPSAT-1 [20]. The attitudes being tested were Roll, Pitch, and Yaw angle. With numbers for our observation satellites a similar analysis is possible.

The results of implementing the model for the Roll angle successfully met certain specifications that were set, but the MATLAB Toolbox did not converge for Pitch and Yaw angle calculations. A lot of improvement with the stability of the satellite is visible due to the application of the PD compensator. Next to my models results, the results of a study from the University of Petroleum and Energy Studies in India on NPSAT-1 [21]. To compare it to the study's more complex model, when the ACS is more redefined the improvement of the specifications is noticeable as shown in the table. Decreasing rise time, overshoot percentage, settling time, and peak time were the primary goals of the model. Due to my model's limitations with computing attitude dynamics and only slightly improving the majority of accuracy parameters, the next steps involve implementing MATLAB's "Aerospace Blockset CubeSat Simulation Library" for a more in-depth analysis and determination of pointing requirements.

### 7. Attempted CubeSat Implementation



**Fig. 29 Example CubeSat library Simulink model set up for 6DOF analysis encompassed with visual simulation.**

MATLAB has multiple simulation libraries for analyzing attitude dynamics. After the PD Compensator model failed to create the wanted results, I attempted implemented MATLABs CubeSat Simulation Library. After spending hours researching, testing, and analyzing the model, the wanted inputs for this Simulink block diagram were inaccessible due to lack of knowledge. This analysis is based on quaternion parameters and takes into account the 6DOF which exist in space. When more information is available this would be the ideal model to simulate attitude dynamics as it was built with the spherical harmonic gravity model to emphasize the environment the satellite is placed in. Then the CubeSat vehicle itself asks for in-depth inputs for orbit considerations, attitude considerations, and body frame considerations. With the observations satellite being designed in a simplest of manners this model was too complex to be applicable and was discarded.

### C. Power and Thermal

After receiving initial EOL power consumption requirements, the BOL power consumption is calculated using the equation:

$$BOL = \frac{EOL}{(1 - r)^t}$$

Where  $BOL$  represents the beginning-of-life power consumption,  $EOL$  represents the End-of-life power consumption,  $r$  represents the percent performance degradation rate per year (0.5% per year), and  $t$  represents the desired life span of the solar arrays (15 years). After the BOL power requirements were calculated, the total necessary solar array area was calculated by dividing the BOL power requirement by the  $300 \text{ W/m}^2$  that the solar arrays generate. Two solar arrays are used

per satellite, so the area is divided by two, and the width and length are calculated from this area value per solar array.

The code used for calculating the BOL power requirements for the solar arrays, as well as the sizes of the solar arrays, can be seen below.

```
EOL_comms = 2044.14; % End of Life power requirement for communications sat
EOL_obs = 5881; % End of Life power requirement for observation sat

BOL_comms = EOL_comms/((1-0.005)^15); % Beginning of Life power requirement for comms sat
BOL_obs = EOL_obs/((1-0.005)^15); % Beginning of Life power requirement for obs sat

total_area_comms = BOL_comms/300; % Total solar array area for comms sat
total_area_obs = BOL_obs/300; % Total solar array area for obs sat

per_array_comms = total_area_comms/2; % Area of each solar array on comms sat
per_array_obs = total_area_obs/2; % Area of each solar array on each obs sat

width_comms = sqrt(per_array_comms/3); % Width of each solar array on comms sat
width_obs = sqrt(per_array_obs/3); % Width of each solar array on obs sat

length_comms = width_comms*3; % Length of each solar array on comms sat
length_obs = width_obs*3; % Width of each solar array on obs sat

fprintf('The BOL power requirement for the communications satellite is %f W', BOL_comms)
fprintf('\nThe BOL power requirement for the observation satellite is %f W', BOL_obs)
fprintf('\nThe solar arrays for the communications satellite are %f x %f meters each', width_comms, length_comms)
fprintf('\nThe solar arrays for the observation satellite are %f x %f meters each\n\n', width_obs, length_obs)
```

## D. Propulsion

### 1. FISTO Propulsion System Sizing

propSizeFunc.m is a script written by Gregory Fretti and Lorin Nugent. Our code and several derivatives were used for satellite sizing. The function takes a propulsion system wattage, payload wattage, propulsion specific impulse, dry/dead mass pushed by the system (tanks, scientific payload, attitude control system and fuel, etc.), and required delta-v.

This script is usable not only for electrical propulsion systems, but also chemical propulsion systems. To model a chemical propulsion system, the user sets the propulsion system wattage to zero and sets a specific impulse that is reasonable for a chemical propulsion system.

Several assumptions were made in this code, although the parameters are easily modifiable:

- Power density for the solar arrays is assumed to be 150W per kg [22].
- Panels are sized to produce double the power consumption of the spacecraft, so the spacecraft can charge in half an orbit while operating the onboard systems even if the spacecraft is in eclipse for the entire other half of the orbit.
- Solar panel output decay is assumed to be 0.5% per year in the example code, but this can be modified.
- The mission duration is set to 15 years.
- The orbital period is based off of a 50km low lunar orbit (however, FISTO will operate in an elliptical orbit and the eclipse periods thus will be shorter.) Since it is a low orbit, the worst-case alignment has slightly under 50% of the orbit (rounded up to 50% for the code) in shadow.

- The battery capacity is based on 70% depth of discharge and energy density of 180 Wh/kg [32, 40], along with an extra factor to make a conservative estimate. The 70% limit is intended to provide extra capacity such that the spacecraft can still function as the battery degrades over time.
- The final mass is also multiplied by an extra factor to make a conservative estimate. This is to account for the enlargement of any other systems that were initially assumed in the dry mass, which may be caused by the enlargement of the power and propulsion systems.

PropMassGraph.m is a script, written by Gregory Fretti, that uses propSizeFunc.m to compare electric and chemical propulsion systems visually. This is done by running propSizeFunc.m with a vector of delta-V values and two different propulsion systems. As with propSizeFunc.m. PropMassGraph.m can be treated as a template to base other propulsion system comparisons off of. The advantage of this format is that it intuitively shows whether or not a mission has the delta-V requirements that would justify electric propulsion.

## 2. *propMassGraph.m*

```
clear
clc
clf

dV = [0:1:2500];

wattage = 2300;
payloadWattage=680;
Isp = 3127;
dryMass = 268;

for n = 1:length(dV)

    ElectricProp(n) = propSizeFunc(wattage,payloadWattage,Isp,dryMass,dV(n));
```

```
ConvProp(n) = propSizeFunc(0,payloadWattage,320,dryMass,dV(n));  
end  
  
hold on  
  
plot (dV,ElectricProp,'LineWidth',2,'Color','blue');  
plot (dV,ConvProp,'LineWidth',2,'Color','red');  
  
xline(15*150,'--','LineWidth',1.5);  
  
xline(15*100,'--','LineWidth',1.5);  
  
xlabel('Delta-V, m/s','FontSize',24);  
ylabel('Spacecraft Mass, kg','FontSize',24);  
ax=gca;  
ax.XAxis.FontSize=20;  
ax.YAxis.FontSize=20;  
  
title('Spacecraft Mass vs. \Delta V for Electric and Chemical Propulsion ','FontSize',24);  
  
legend('Electric Propulsion I_s_p = 3127s','Chemical Propulsion I_s_p =  
320s','Location','northwest','FontSize',24);  
  
text(1400,200,'100 m/s per year, 15  
years','FontSize',20,'HorizontalAlignment','center','Rotation',90);  
  
text(2150,200,'150 m/s per year, 15  
years','FontSize',20,'HorizontalAlignment','center','Rotation',90);  
  
text(500,500,'Electric propulsion masses include'  
, 'FontSize',16,'HorizontalAlignment','center');  
  
text(500,480,'solar panels and batteries necessary to'  
, 'FontSize',16,'HorizontalAlignment','center');  
  
text(500,460,'power propulsion system continuously,'  
, 'FontSize',16,'HorizontalAlignment','center');  
  
text(500,440,'including eclipse conditions.'  
, 'FontSize',16,'HorizontalAlignment','center');  
  
ylim([0 inf]);
```

### 3. *propSizeFunc.m*

```
function totalMass = propSizeFunc(propWattage, payloadWattage, Isp, dryMass, dV)

%The first round of dry mass estimates produced an approximate mass
%as well as dry mass for the ACS system.

dry_estimate_total = dryMass; %kg

%Solar panel size estimate to drive EP
n_years = 15; %lifespan

extraFactor = 2.1; %2.1 allows the panels to drive the propulsion system while
charging the batteries in half an orbit

wattage_required = (propWattage*extraFactor + payloadWattage) / (1-
(0.5/100))^n_years; %2.75% per year decay
%The addition of the payload wattage was from Lorin Nugent

powerDensity_panels = 150; %W/kg,
http://www.dept.aoe.vt.edu/~cdhall/courses/aoe4065/power.pdf

panel_mass = wattage_required/powerDensity_panels;

%Battery size estimate for EP

GM=0.00490*10^6; %km^3/s^2,
https://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html
orb_alt = 50; %km
SMA = (1738.1+orb_alt);
orbit_period = 2*pi*sqrt(SMA^3/GM);

%Worst case: roughly 50% of the time is
%eclipse
eclipse_time = orbit_period * 0.5;
eclipse_energy = propWattage * eclipse_time; %Joules

%https://www.sciencedirect.com/science/article/pii/S0378775303002222
%14 year lifespan from 80% depth of discharge, so we'll use 70%
%Thus, 70% of the battery's energy capacity is spent in the time spent
```

```
%running in eclipse

batt_cap = (eclipse_energy * 1.2)/0.7; %J

batt_energy_density = 180; %Wh/kg %https://www.fluxpower.com/blog/what-is-the-
energy-density-of-a-lithium-ion-battery

batt_mass = (batt_cap*(1/3600))/(batt_energy_density);

dead_mass = dry_estimate_total+batt_mass+panel_mass;

%Propulsion mass
%https://www.colorado.edu/faculty/kantha/sites/default/files/attached-
files/arnold_electricprop.pdf

g0 = 9.81;

propmass = dead_mass*exp(dV./(Isp.*g0))-dead_mass;

totalMass = (propmass+dead_mass)*1.1; %Extra factor for extra ACS systems, etc.)

end
```

*4. Decision Matrix for Propellant Type*

| Decision Matrix for propellant type |                      |                                                                                                  |       |                                                                                                                                                              |       |                                                                                                                                                                                                                  |       |
|-------------------------------------|----------------------|--------------------------------------------------------------------------------------------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| Criteria                            | Weight<br>(out of 5) | Cold Gas Propellants<br>(Isobutane, R144a, SO <sub>2</sub> , CO <sub>2</sub> , N <sub>2</sub> O) |       | Monopropellants (N <sub>2</sub> H <sub>4</sub> , H <sub>2</sub> O <sub>2</sub> , N <sub>2</sub> O, AND(Ammonium dinitramide), HAN(Hydroxylammonium nitrate)) |       | Bipropellants<br>(MMH(Monomethylhydrazine) or UDMH(Unsymmetrical dimethylhydrazine) + N <sub>2</sub> O <sub>4</sub> , H <sub>2</sub> O <sub>2</sub> + CH <sub>4</sub> , H <sub>2</sub> O <sub>2</sub> + Kerosene |       |
|                                     |                      | Score (out of 5)                                                                                 | Total | Score (out of 5)                                                                                                                                             | Total | Score (out of 5)                                                                                                                                                                                                 | Total |
| High ΔV                             | 4                    | 3                                                                                                | 12    | 3                                                                                                                                                            | 12    | 4                                                                                                                                                                                                                | 16    |
| High Isp                            | 5                    | 2                                                                                                | 10    | 4                                                                                                                                                            | 20    | 5                                                                                                                                                                                                                | 25    |
| High Thrust                         | 3                    | 3                                                                                                | 9     | 5                                                                                                                                                            | 15    | 4                                                                                                                                                                                                                | 12    |
| Ease of Storage                     | 5                    | 2                                                                                                | 10    | 5                                                                                                                                                            | 25    | 3                                                                                                                                                                                                                | 15    |
| Low complexity of propulsion system | 4                    | 4                                                                                                | 16    | 4                                                                                                                                                            | 16    | 2                                                                                                                                                                                                                | 8     |
| Safety                              | 5                    | 3                                                                                                | 15    | 4                                                                                                                                                            | 20    | 3                                                                                                                                                                                                                | 15    |
|                                     |                      |                                                                                                  |       |                                                                                                                                                              |       |                                                                                                                                                                                                                  |       |
| total score                         |                      |                                                                                                  | 72    |                                                                                                                                                              | 108   |                                                                                                                                                                                                                  | 91    |

**Fig. 30 Decision Matrix for selecting chemical propellant**

### *5. Observation Satellite Propulsion System Analysis*

For initial analysis, we compared popular electric propulsion system and chemical propulsion technologies. For the analysis [23] uses Small monopropellant thruster with hydrazine as propellant for the chemical system and SMART-1 Hall Effect Thruster (PPS-1350) with Xe propellant as the electric propulsion system. As presented in the table in the document, chemical propulsion systems and electric propulsion systems vastly vary in terms of  $I_{sp}$ , Thrusting time and Total Impulse.

From the data provided, it is concluded that the monopropellant thruster has nearly  $\frac{1}{8^{th}}$  the  $I_{sp}$  value as compared to that of the Hall-Effect thruster. Total impulse produced by the Hall-Effect thruster is nearly 10 times that of the monopropellant chemical thruster. The downside of using the electric propulsion system is its long thrusting time required to reach the higher  $\Delta V$  values and the nearly  $10^{-2}$  times the magnitude of Thrust produced by traditional chemical propulsion systems.

For small-satellites, electric propulsion systems prove to be useful because of high  $I_{sp}$  values and low-thrust applications. For maneuvers such as stationkeeping and attitude control, especially in higher (low-perturbation) orbits, thrust requirements are lower. Higher  $I_{sp}$  values help cut down the mass of propellant required. The growing use of electric propulsion for CubeSats and other Small Satellites also make this a reliable technology. These factors establish electric propulsion for small applications to be very desirable.

### *6. Analysis of Various Electric Propulsion Systems*

For short-listing prospective useful electric propulsion technologies, three types of systems are considered - SMART-1 Hall Effect Thruster [23], IFM-Nano thruster for Field Emission Electric

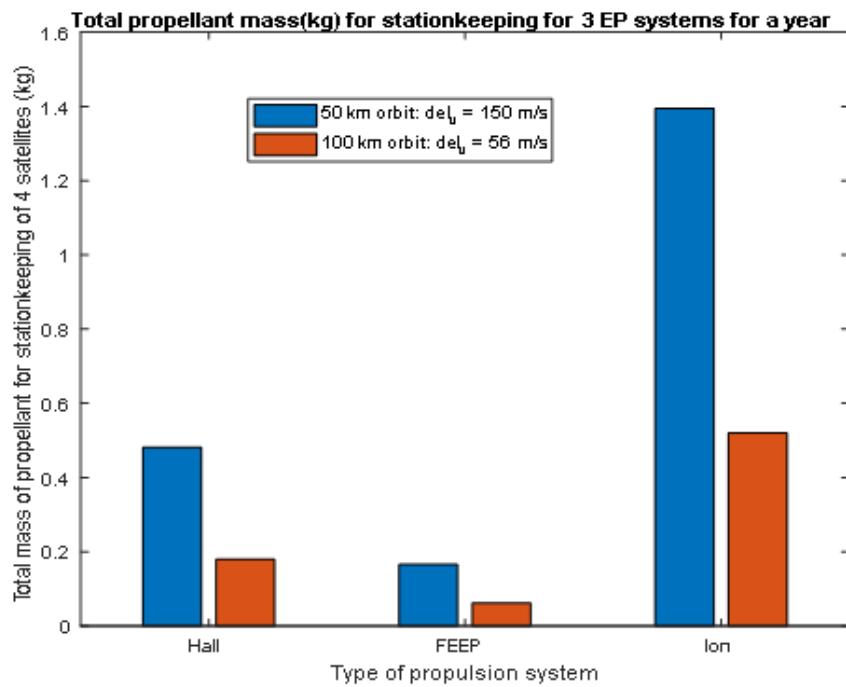
propulsion systems (FEEP) [24], and NSTAR engine from Deep Space 1 mission [25]. Table 9 below provides data on these different technologies.

To begin analysis for considering electric propulsion systems, reasonable estimates about  $\Delta V$  requirements, orbit altitudes and mass of payload are made for a baseline comparison. For reference mass of payload, New Horizon's LORRI camera is used [26] It weighs 8.8 kg. To provide best viewing location for imaging equipment on-board, 50 km and 100 km orbits are considered for the present analysis. For these orbits, the respective  $\Delta V$  estimates are provided by [6] and [27] to be 150 m/s and 29-83 m/s for 50 km and 100 km orbits respectively. The code for the same is provided.

Figure 31 yields that ion-thrusters use a considerable amount of fuel, nearly 21 kg of Xe for 15 years when maintained in a low 50 km altitude orbit. As highlighted by Communications, the orbit should be maintained at 50 km in order to meet good accuracy and precision requirements for the imaging satellites. Therefore, despite the relatively large fuel requirements to maintain the 50 km orbit as opposed to those in a higher orbit, the 50 km altitude orbit is chosen. Since the satellite performance is critical for the Lunar colony's sustenance, it is recommended to use Ion propulsion systems. Frequent use of ion thrusters, their durability and reliability in Deep Space 1 mission and on Dawn Space probes, and their objective benefits over traditional chemical propulsion systems outweigh the small mass benefits of other lesser demonstrated technologies mentioned earlier in this analysis. Another factor that supports this choice of propulsion system is that once the investments is made, there is no additional re-fueling or expenses associated with the propulsion system within the 15-year period. Understanding that propellant mass differences between the various electric propulsion systems are not too extreme, the popular and reliable Ion thrusters were selected for the FISTO satellites.

**Table 9** Here, the data for 3 different electric propulsion devices are listed.

| Propulsion Type      | Satellite/ Mission reference | Propellant  | Avg. Isp(s) | Power(W) | Dry mass of propulsion system(kg) | volume (cubic m.) or size(m) | Max Payload mass (kg) |
|----------------------|------------------------------|-------------|-------------|----------|-----------------------------------|------------------------------|-----------------------|
| Hall-Effect Thruster | SMART-1                      | Xe          | 1800        | 1500     | 5.3                               | 0.1m diameter                | 20                    |
| FEEP                 | IFM-Nano thruster            | liq. Indium | 3500        | 35       | 0.64                              | 0.00168 cubic m.             | 60                    |
| Ion Thruster         | NSTAR engine, Deep Space 1   | Xe          | 2500        | 2100     | 48                                | 0.3m diameter                | Approx. 400           |



**Fig. 31** The figure displays propellant mass for 2 orbits for the 3 different electric propulsion technologies.

## 7. Code for Chemical Propulsion System Mass and Sizing

```
%Communication Satellite Chemical propulsion
%Sanjana Singh
% The code below presents the analysis for calculating propellant mass required for
% the CHELL satellites. Each satellite has N2H4 as propellant. N2H4 has an Isp of 225 s taken
from the average range of 220-230s with a dry mass of 576.6 kg
% The analysis takes two calculations for orbital insertion and stationkeeping requirements
% assuming the delta_v value for stationkeeping and orbital maneuvers to be 100 m/s per year
% and delta_v value for orbital insertion maneuver is 1000 m/s

dry_mass = 576.6 ; %kg
Isp_N2H4 = 225 ; %s
rho_N2H4 = 1008; %kg/m^3
g = 9.81; % m/s^2
delta_v = 100 ; %m/s per year
delta_v_insertion = 1000; % m/s
Propellant mass required

t=15; %years
sk_prop_mass = (exp(delta_v/(g*Isp_N2H4))*dry_mass - dry_mass)*t ;% mass of propellant
required for stationkeeping (kg)
acs_prop_mass = 41.6343 ;% as provided by mission design (kg)
orbit_insertion_prop_mass = (exp(delta_v_insertion/(g*Isp_N2H4))*dry_mass - dry_mass)
;%orbit insertion prop mass (kg)
total_prop_mass = sk_prop_mass + acs_prop_mass + orbit_insertion_prop_mass;
total_prop_sf = 1.0*total_prop_mass ;%with 0% safety factor, factor is adjustable

Volume of propellant tank

Vp_total = total_prop_sf/rho_N2H4 ;% Volume of propellant in prop tank (m^3)
%taking 5%, 1% and 6% for ullage, boiloff and trapped volumes -
Vt_total = Vp_total*1.12 ;% (12% buffer as stated above)
Vt_tank = Vt_total/2 ;% taking 2 spherical tank assembly
d_prop_tank = ((Vt_tank*3/(4*pi))^(1/3))*2; % diameter of each prop tank (m)

Volume of pressurant tank

rho_He = 0.1786; %kg/m^3
% Ignoring trapped volume -
V_He = 0.94*Vt_total; %Volume of helium required to pressurize the 2 tanks (m^3)
% assuming 30% buffer
mass_He = V_He*rho_He*1.3;% (kg)
% can use 16.6" dia X 26.1" Long Helium Tank, P/N 80436 by Northrop Grumman
```

## 8. Code for Electric Propulsion System Propellant Mass Analysis

```
%The code below compares three different electric propulsion systems
%The code uses reference of 50 km and 100 km altitude lunar orbits and assumes
% a propellant mass of 8.8 kgs taken from New Horizon's LORRI. The code predicts which
% Thruster is suitable for each of the orbits. The aim of the code is to suggest an electric
% propulsion
% system with the lowest propellant mass consumption
```

*For h = 50 km perturbations are dominant*

```
m_pl = 8.8; %kg
delta_v_50 = 150 ;% m/s
g0 = 9.81 ;%m/s^2

% for hall effect thruster
md_hall_1 = 5.3 + m_pl ;%kg
Isp_hall_1 = 1800;%s
mp_hall_1 = md_hall_1*(exp(delta_v_50/(g0*Isp_hall_1))-1); % mass of propellant needed
m_hall_1 = mp_hall_1 + md_hall_1;% total mass of 1 satellite
mp_hall_total_1 = mp_hall_1*4 ;%Total propellant mass for 4 satellites
m_total_hall_1 = m_hall_1*4; %Total mass of 4 satellites

%for FEEP
md_feep_1 = 0.64 + m_pl;%kg
Isp_feep_1= 3500;%s
mp_feep_1 = md_feep_1*(exp(delta_v_50/(g0*Isp_feep_1))-1); % mass of propellant needed
m_feep_1 = mp_feep_1 + md_feep_1;% total mass of 1 satellite
mp_feep_total_1 = mp_feep_1*4 ;%Total propellant mass for 4 satellites
m_total_feep_1 = m_feep_1*4; %Total mass of 4 satellites

%for Ion thruster
md_ion_1 = 48 + m_pl;%kg
Isp_ion_1= 2500;%s
mp_ion_1 = md_ion_1*(exp(delta_v_50/(g0*Isp_ion_1))-1); % mass of propellant needed
m_ion_1 = mp_ion_1 + md_ion_1;% total mass of 1 satellite
mp_ion_total_1 = mp_ion_1*4; %Total propellant mass for 4 satellites
m_total_ion_1 = m_ion_1*4; %Total mass of 4 satellites
```

*For h = 100 km perturbations are small*

```
m_pl = 8.8; %kg
delta_v_100 = 56 ;% m/s
g0 = 9.81 ;%m/s^2
```

% for hall effect thruster

```

md_hall_2 = 5.3 + m_pl ; %kg
Isp_hall_2 = 1800;%s
mp_hall_2 = md_hall_2*(exp(delta_v_100/(g0*Isp_hall_2))-1); % mass of propellant needed
m_hall_2 = mp_hall_2 + md_hall_2;% total mass of 1 satellite
mp_hall_total_2 = mp_hall_2*4 ;%Total propellant mass for 4 satellites
m_total_hall_2 = m_hall_2*4; %Total mass of 4 satellites

%for FEEP
md_feep_2 = 0.64 + m_pl;%kg
Isp_feep_2= 3500;%s
mp_feep_2 = md_feep_2*(exp(delta_v_100/(g0*Isp_feep_2))-1); % mass of propellant needed
m_feep_2 = mp_feep_2 + md_feep_2;% total mass of 1 satellite
mp_feep_total_2 = mp_feep_2*4 ;%Total propellant mass for 4 satellites
m_total_feep_2 = m_feep_2*4; %Total mass of 4 satellites

%for Ion thruster
md_ion_2 = 48 + m_pl;%kg
Isp_ion_2= 2500;%s
mp_ion_2 = md_ion_2*(exp(delta_v_100/(g0*Isp_ion_2))-1); % mass of propellant needed
m_ion_2 = mp_ion_2 + md_ion_2;% total mass of 1 satellite
mp_ion_total_2 = mp_ion_2*4; %Total propellant mass for 4 satellites
m_total_ion_2 = m_ion_2*4; %Total mass of 4 satellites
mp_1 = [mp_hall_total_1,mp_hall_total_2; mp_feep_total_1,mp_feep_total_2;
mp_ion_total_1,mp_ion_total_2];
prop = categorical({'Hall','FEEP','Ion'});
prop = reordercats(prop,['Hall','FEEP','Ion']);
figure(1)
bar(prop,mp_1);
ylabel('Total mass of propellant for stationkeeping of 4 satellites (kg)');
xlabel('Type of propulsion system');
legend('50 km orbit: del_v = 150 m/s','100 km orbit: del_v = 56 m/s');
title('Total propellant mass(kg) for stationkeeping for 3 EP systems for a year')
ylim([0 1.6])

```

## VII. Potential and Future Work

### A. Lunar Positioning System

While we were still in the early phases of mission planning, we considered the concept of incorporating a satellite network that would provide positioning and navigation services for surface operations. This satellite network, nicknamed ‘Lunar Positioning System’ (LPS), would serve very similar functions to the Earth-based Global Positioning System (GPS) which has become integral in everyday life. We could use this to track the movements of surface rovers and any human activity necessary outside of the Lunar base. This would also allow us to more safely perform research on the far side of the Moon if desired.

There are currently 31 GPS satellites around the Earth in a medium-Earth orbit and 24 GLONASS satellites (the Russian equivalent) which make up the Global Navigation Satellite System (GNSS). A standard GPS receiver can provide roughly 8 meter location accuracy anywhere on the Earth 95% of the time. Each GPS satellite uses a highly accurate on-board atomic clock to transmit time data to the GPS receiver, which then calculates the distance to that satellite based on its position during transmission. A minimum of four concurrent satellite fixes is required to perform trilateration and determine the 3D position of the receiver. In addition to the large satellite network, GPS systems require monitor stations to keep track of the satellite orbits [28].

To hypothetically implement a LPS satellite network around the Moon, we deem the following components and characteristics necessary to provide precise positioning information to rovers performing surface operations.

- 12 to 15 LPS satellites (include atomic clock, antennas, radio transmitters, and solar panels)

- 4 to 5 orbital planes, arranged in a Walker constellation
- On-board propulsion and reaction control
- Ground stations on the Lunar surface
- GPS receivers aboard rovers

The satellites would be placed in a medium Lunar orbit to increase each satellite's field of view and reduce the effects of perturbations caused by the uneven mass distribution of the Moon. A Walker constellation is used for the GPS satellite network around the Earth, and would also be ideal for this application. The benefits of Walker constellations are described in the Walker Constellation section of the appendix.

We consider the Lockheed Martin GPS III as our satellite of choice, as it is state of the art modern GPS technology. The satellite is 2,200 kg, draws 4,500 W from solar power, and is approximately 15 m<sup>3</sup> volume [29]. Assuming 12 to 15 satellites, this totals to 26,400 to 33,000 kg of GPS of launch payload. Since we have 4 to 5 orbital planes each with 3 satellites, we must consider our deployment options. The simplest option is to deploy satellites directly from the upper stage of the launch vehicle. If this is the case, each satellite must additionally carry enough propellant to place itself into the correct location in its orbital plane, which reduces the propellant left for general stationkeeping and thus the lifespan of the satellite. If this option were to be selected, we would likely still need one launch per orbital plane. Since we are launching from the Earth to a lunar orbit, each launch incurs a substantial cost, reducing our budget for the development of other necessary systems. This does not factor in the cost of launching the Lunar ground stations for the satellites. Another satellite deployment option is to have a 'mothership' transportation vehicle which places each satellite into its intended orbit using only its own

propellant. This would preserve the stationkeeping propellant aboard the LPS satellites, however it is still a new technology with no viable commercial option for this application yet on the market.

Due in large part to the drawbacks of deployment methods and costs, we decide not to move forward with a Lunar Positioning System satellite network for our lunar colony. The majority of surface operations likely occur on the Earth-facing side of the Moon, which has a direct line of sight with Earth-based observation stations. Additionally, there is research currently underway suggesting that GNSS satellites in orbit around Earth might be able to provide positioning information for high orbit altitudes, or potentially even Lunar surface vehicles [30]. The development of this technology is something that can also be aided by the rovers in the NextStep colony.

## References

- [1] Satellite Solutions. Lockheed Martin Corporation, 2021, [www.lockheedmartin.com/en-us/products/satellite.html](http://www.lockheedmartin.com/en-us/products/satellite.html).
- [2] Walker, J. G., “Continuous Whole-Earth Coverage by Circular-Orbit Satellite Patterns”, RAE TR 77044, 1977.
- [3] J. Schwartz, “The Distributed Spacecraft Attitude Control System Simulator: From Design Concept to Decentralized Control”, published July 7th, 2004. <https://hanspeterschaub.info/Papers/grads/JanaSchwartz.pdf>
- [4] Anvari, A., Farhani, F., Niaki, K., “Comparative Study on Space Qualified Paints Used for Thermal Control of a Small Satellite,” Iranian Research Organization for Science and Technology, Department of Mechanical Engineering, Jan. 2009.
- [5] Bell, T., and Phillips, T. Bizarre Lunar Orbits | Science Mission Directorate. NASA. [https://science.nasa.gov/science-news/science-at-nasa/2006/06nov\\_loworbit](https://science.nasa.gov/science-news/science-at-nasa/2006/06nov_loworbit). Accessed Apr. 4, 2021.
- [6] Beckman, M., and Lamb, R. “Stationkeeping for the Lunar Reconnaissance Orbiter (LRO).” Proceedings of the 20th International Symposium on Space Flight Dynamics, 2007.
- [7] 400 N Bipropellant Apogee Motors. Arianegroup Orbital Propulsion Centre. <https://www.space-propulsion.com/spacecraft-propulsion/apogee-motors/index.html>. Accessed Apr. 4, 2021.
- [8] Brophy, J. “The Dawn Ion Propulsion System.” Space Science Reviews, Vol. 163, No. 1, 2011, pp. 251–261. <https://doi.org/10.1007/s11214-011-9848-y>.
- [9] Brophy, J., Marcucci, M., Ganapathi, G., Garner, C., Henry, M., Nakazono, B., and Noon, D. “The Ion Propulsion System for Dawn.” 2003. <https://doi.org/10.2514/6.2003-4542>.
- [10] Ion Propulsion | Technology. NASA Solar System Exploration. <https://solarsystem.nasa.gov/missions/dawn/technology/ion-propulsion>. Accessed Apr. 4, 2021.
- [11] Saleh, J. H., Geng, F., Ku, M., and Walker, M. L. R. “Electric Propulsion Reliability: Statistical Analysis of on-Orbit Anomalies and Comparative Analysis of Electric versus Chemical Propulsion Failure Rates.” Acta Astronautica, Vol. 139, 2017, pp. 141–156. <https://doi.org/10.1016/j.actaastro.2017.06.034>.
- [12] Schmitz, H., and Steenborg, M., “Augmented electrothermal hydrazine thruster development,” Journal of Spacecraft and Rockets, vol. 20, 1983, pp. 178–181.
- [13] Springmann, P. N. and de Weck, O. L., “Parametric Scaling Model for Nongeosynchronous Communications Satellites,” *Journal of Spacecraft and Rockets*, Vol. 41, No. 3, May-June 2004, pp. 472-477.
- [14] Wijker, J. J., *Spacecraft Structures*, Springer Berlin/Heidelberg, 2008, pp. 216.
- [15] “Iridium-NEXT,” Spaceflight101.com, URL: <https://spaceflight101.com/spacecraft/iridium-next/> [retrieved 3 April 2021].

Samantha Dickmann, Lorin Nugent, Jaxon Connolly, Monica Viz, Matthew Popplewell, Stephen Grabowski, Gregory Fretti, Alan Gelman, Sanjana Singh, Roberto Cardano

- [16] Nag, S., LeMoigne, J., de Weck, O., "Cost and Risk Analysis of Small Satellite Constellations for Earth Observation," *IEEE Aerospace Conference*, Big Sky, MT, 2014, pp. 1-16.  
doi: 10.1109/AERO.2014.6836396
- [17] "Satellite Development," Canon Global, URL: <https://global.canon/en/technology/frontier13.html> [retrieved 3 April 2021].
- [18] A. Fleming, "Real-time Optimal Slew Maneuver and Control", Naval Postgraduate School, published December 2004.  
<https://apps.dtic.mil/dtic/tr/fulltext/u2/a477384.pdf>
- [19] S. Aslam, M. Hamza, M. Moazzam, Z. Abbas, F. S. Ali and S. Hannan, "Development of Satellite Attitude Simulink Model to Study the Rotational Degrees of Freedom," 2020 International Conference on Engineering and Emerging Technologies (ICEET), Lahore, Pakistan, 2020, pp. 1-5, doi: 10.1109/ICEET48479.2020.9048202.  
<https://ieeexplore.ieee.org/document/9048202>
- [20] P. Grenfel, M. Long, "Pointing, Acquisition, and Tracking for Small Satellite Laser Communications", MIT, published August 2018. <https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=4230&context=smallsat>
- [21] M. Raja, O. Prakash, "Design of High Pointing Accuracy NPSAT-1 Satellite Attitude Systems of Armature Controlled DC Motor with utilization for PD Controller", published by INCAS March 2020, doi: 10.13111/2066-8201.2020.12.1.14  
[https://www.researchgate.net/publication/339613460\\_Design\\_of\\_High\\_Pointing\\_Accuracy\\_NPSAT-1\\_Satellite\\_Attitude\\_Systems\\_of\\_Armature\\_Controlled\\_DC\\_Motor\\_with\\_utilization\\_for\\_PD\\_Controller](https://www.researchgate.net/publication/339613460_Design_of_High_Pointing_Accuracy_NPSAT-1_Satellite_Attitude_Systems_of_Armature_Controlled_DC_Motor_with_utilization_for_PD_Controller)
- [22] AOE 4065 Space Design. Spacecraft Power Systems. Virginia Tech.
- [23] "Navigation," ESA Science & Technology - Electric Spacecraft Propulsion Available: <https://sci.esa.int/web/smart-1-/34201-electric-spacecraft-propulsion>.
- [24] "One-stop webshop for CubeSats & Nanosats," CubeSatShop.com Available: <https://www.cubesatshop.com/>.
- [25] NSTAR Available: <http://www.astronautix.com/n/nstar.html>.
- [26] New Horizons: Beyond Pluto Available: <http://pluto.jhuapl.edu/Mission/Spacecraft/Payload.php>.
- [27] Knežević, Z., and Milani, A., "Orbit maintenance of a lunar polar orbiter," Planetary and Space Science, vol. 46, 1998, pp. 1605–1611.
- [28] "Satellite Navigation - GPS - How It Works", Federal Aviation Administration, published online 22 Jul. 2020,  
[www.faa.gov/about/office\\_org/headquarters\\_offices/ato/service\\_units/techops/navservices/gnss/gps/howitworks/#:~:text=GP%20satellites%20carry%20atomic%20clocks,time%20the%20signal%20was%20broadcast.&text=Thus%2C%20the%20reciever%20uses%20four,longitude%2C%20altitude%2C%20and%20time.](http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks/#:~:text=GPS%20satellites%20carry%20atomic%20clocks,time%20the%20signal%20was%20broadcast.&text=Thus%2C%20the%20reciever%20uses%20four,longitude%2C%20altitude%2C%20and%20time.)

- [29] Steigenberger, P., Thoelert, S., and Montenbruck, O.. “GPS III Vespucci: Results of Half a Year in Orbit.” *Advances in Space Research*, Vol. 66, No. 12, 2020, pp. 2773-2785.
- [30] Keesey, L., “NASA Eyes GPS at the Moon for Artemis Missions”, NASA TV, published online 28 Jun. 2019.  
<https://www.nasa.gov/feature/goddard/2019/nasa-eyes-gps-at-the-moon-for-artemis-missions>
- [31] Kerslake, T., Gustafson, E., “On-Orbit Performance Degradation of the International Space Station P6 Photovoltaic Arrays,” Glenn Research Center, Cleveland, Ohio, Jul. 2003.
- [32] Cloud, M. What Is the Energy Density of a Lithium-Ion Battery? <https://www.fluxpower.com/blog/what-is-the-energy-density-of-a-lithium-ion-battery>. Accessed Apr. 4, 2021.
- [33] “Momentum Control Systems,” Honeywell Aerospace Available:  
<https://aerospace.honeywell.com/en/learn/products/space/momentum-control-systems>.
- [34] “RDR 68 Momentum and Reaction Wheels 14 – 68 Nms with external Wheel Drive Electronics,” satsearch Available:  
<https://satsearch.co/products/collins-aero-rdr-68>.
- [35] Simpson, J., Badgley, J., McCaughey, K., Brown, K., Calhoun, P., Davis, E., Garrick, J., Gill, N., Hsu, O., Jones, N., Ortiz-Cruz, G., Raymond, J., Roder, R., Shah, N., and Wilson, J., “Integration and Testing of the Lunar Reconnaissance Orbiter Attitude Control System,” 2011 Aerospace Conference, 2011.
- [36] Ackley, M., “How to (III): Calculate Satellite Disturbance Torques,” Valispace Available: <https://www.valispace.com/how-to-iii-calculate-satellite-disturbance-torques/>.
- [37] Ackley, M., “How To (IV): Size an Attitude Control System,” Valispace Available: <https://www.valispace.com/how-to-size-an-attitude-control-system/>.
- [38] Kovo, Y., “5.0 Guidance, Navigation & Control,” NASA Available: <https://www.nasa.gov/smallsat-institute/sst-soa-2020/guidance-navigation-and-control>.
- [39] Brophy, J., Kakuda, R., Polk, J., Anderson, J., Marcucci, M., Brinza, D., Henry, M., Fujii, K., Mantha, K., Stocky, J., Sovey, J., Patterson, M., Rawlin, V., Hamley, J., Bond, T., Christensen, J., Cardwell, H., Benson, G., and Gallagher, J. “Ion Propulsion System (NSTAR) DS1 Technology Validation Report.”
- [40] Fellner, J. P., Loeber, G. J., Vukson, S. P., and Riepenhoff, C. A. “Lithium-Ion Testing for Spacecraft Applications.” *Journal of Power Sources*, Vols. 119–121, 2003, pp. 911–913. [https://doi.org/10.1016/S0378-7753\(03\)00222-2](https://doi.org/10.1016/S0378-7753(03)00222-2).
- [41] Potrivity, G.-C., Sun, Y., Rohaizat, M. W. A. bin, Cherkun, O., Xu, L., Huang, S., and Xu, S. “A Review of Low-Power Electric Propulsion Research at the Space Propulsion Centre Singapore.” *Aerospace*, Vol. 7, No. 6, 2020, p. 67.  
<https://doi.org/10.3390/aerospace7060067>.
- Samantha Dickmann, Lorin Nugent, Jaxon Connolly, Monica Viz, Matthew Popplewell, Stephen Grabowski, Gregory Fretti, Alan Gelman, Sanjana Singh, Roberto Cardano

- [42] Saleh, J. H., Geng, F., Ku, M., and Walker, M. L. R. “Electric Propulsion Reliability: Statistical Analysis of on-Orbit Anomalies and Comparative Analysis of Electric versus Chemical Propulsion Failure Rates.” *Acta Astronautica*, Vol. 139, 2017, pp. 141–156. <https://doi.org/10.1016/j.actaastro.2017.06.034>.
- [43] Shah, N., Calhoun, P., Garrick, J., Hsu, O., and Simpson, J., “Launch and Commissioning of the Lunar Reconnaissance Orbiter (LRO),” NASA AAS 10-085.
- [44] de Weck, O. L., “Attitude Determination and Control (ADCS),” Massachusetts Institute of Technology 16.684 Space Systems Product Development, 2001.
- [45] Adcole Space, “Coarse Sun Sensor,” URL: <https://adcolespace.com/product/coarse-sun-sensor/> [retrieved 5 April 2021].
- [46] satsearch.co, “Leonardo Finmeccanica A-STR,” URL: <https://satsearch.co/products/leonardo-finmeccanica-a-str>. [retrieved 5 April 2021].
- [47] Digi-Key Electronics, “Honeywell HG4930 Inertial Measurement Unit,” URL: <https://www.digikey.com/en/product-highlight/h/honeywell-microelectronics-precision-sensors/hg4930-inertial-measurement-unit> [retrieved 5 April 2021].
- [48] Holbert, S., “Project Artemis,” Purdue University AAE 450, 2014, pp.423-424, 430-431.
- [49] Dai, R., “AAE 421 Lecture 6: Angular Velocities and Flight Dynamics Equations of Motion,” Purdue University, 2021.
- [50] “Multi-Layer Insulation,” Meyer Tool & MFG, Oak Lawn, IL
- [51] Plachta, D. W., Feller, J. R., Guzik, M. C., “In-Space Cryogenic Propellant Storage Applications for a 20 W at 20 K Cryocooler”, NASA Glenn Research Center, Cleveland, OH, NASA Ames Research Center, Moffett Field, CA

## **Systems: Habitats**

## I. Matlab Scripts

### A. Structures

#### Program 1 Estimation of Mass and Volume Using Cylindrical Approximation

```
%Wilson's code (and research) makes use of diffusion which lowers the  
%energy of impacting micrometeoroids. Derek has added a layer of  
%aluminum to the top of the habitat to further reduce velocity and improve  
%the longevity of materials and habitat operations.  
  
%%Update 02/21/2021 Removed two top layers of the inflatable structure,  
%%still left shield.  
  
%All habitats are approximated as closed systems and the material gained  
%back from the connector modules is considered extra. This is not removed  
%from the formula in case of material irregularities or shipping damage.  
%Therefore, an inherent safety factor is included.  
num_modules = 2;  
num_airlocks = 2;  
  
rad_inner = 3 %inner radius in meters  
Reg_Thickness = 1 %regolith thickness in meters  
Webbing_thickness = 0.0254 %Thickness in meters  
  
%Wilson's Code from Week 4 - directly reused  
length = 21; %length of a module in meters  
  
d_liner = 960;  
d_bladder_packed = 940;  
d_restraint = 1400;  
d_thermal = 1390;  
d_alum = 2700; %density of aluminum in kg/m^3  
d_reg = 1600; %density of regolith in kg/m^3  
  
t_liner = 1e-3;  
t_bladder_packed = 6*13*1e-6;  
t_bladder = 6*25*1e-3;  
t_restraint = 10*1e-3;  
t_thermal = 5.34*1e-3;  
  
t_total = t_liner + t_bladder + t_restraint + t_thermal;
```

```

t_packed = t_liner + t_bladder_packed + t_restraint + t_thermal;

rad_outer = rad_inner + t_total;

h_packed = pi*rad_inner + 2*t_packed;
w_packed = 2*t_packed;
v_packed = length * h_packed * w_packed * 2;

v_inner = rad_inner^2 * pi * length;
v_1 = (rad_inner + t_liner)^2 * pi * length;
v_2 = (rad_inner + t_liner + t_bladder)^2 * pi * length;
v_3 = (rad_inner + t_liner + t_bladder + t_restraint)^2 * pi * length;
v_4 = (rad_inner + t_liner + t_bladder + t_restraint + t_thermal)^2 * pi * length;
v_outer = (rad_inner + t_liner + t_bladder + t_restraint + t_thermal)^2 * pi * length;

v_liner = v_1 - v_inner;
v_bladder = v_2 - v_1;
v_restraint = v_3 - v_2;
v_thermal = v_4 - v_3;

d_bladder = d_bladder_packed*t_bladder_packed/t_bladder;

m_liner = d_liner * v_liner;
m_bladder = d_bladder * v_bladder;
m_restraint = d_restraint * v_restraint;
m_thermal = d_thermal * v_thermal;

tf_liner = t_liner / t_total;
tf_bladder = t_bladder / t_total;
tf_restraint = t_restraint / t_total;
tf_thermal = t_thermal / t_total;

% End of Wilson's code -
% Derek's code from this point forward

t_shield = 0.0025 %Thickness of aluminum shielding
v_shield = (rad_outer + 0.5*t_shield)*2*pi*length*t_shield;
m_shield = v_shield * d_alum;

m_hab_mod = m_liner + m_bladder + m_restraint + m_thermal + m_shield;

a_cap = (rad_outer + t_shield)^2 .*pi;

```

```

v_cap_liner = t_liner * a_cap;
v_cap_bladder = t_bladder * a_cap;
v_cap_restraint = t_restraint * a_cap;
v_cap_thermal = t_thermal * a_cap;
v_cap_shield = t_shield * a_cap;

m_cap_liner = d_liner * v_cap_liner;
m_cap_bladder = d_bladder * v_cap_bladder
m_cap_restraint = d_restraint * v_cap_restraint;
m_cap_thermal = d_thermal * v_cap_thermal;
m_cap_shield = d_alum * v_cap_shield;

m_cap_total = m_cap_liner + m_cap_bladder + m_cap_restraint + m_cap_thermal
+ m_cap_shield;

m_module = m_hab_mod + (2*m_cap_total);

%Interface between modules
mod_in_rad = 1.5; %inner diameter in meters
mod_in_length = 2; %module interface length in meters
mod_in_vol = (mod_in_rad)^2 * pi * mod_in_length; %inner volume of interface

%airlock
airlock_in_length = 2.3; %airlock height in meters
airlock_in_rad = 1.4; %Airlock inner radius
airlock_in_vol = (airlock_in_rad)^2 * pi* airlock_in_length; %inner volume of
airlock

airlock_liner_vol = (airlock_in_rad + 0.5*t_liner)*2*pi*t_liner*airlock_in_length;
airlock_bladder_vol      =      (airlock_in_rad      +      t_liner      +
0.5*t_bladder)*2*pi*t_bladder*airlock_in_length;
airlock_restraint_vol   =   (airlock_in_rad   +   t_liner   +   t_bladder   +
0.5*t_restraint)*2*pi*t_restraint*airlock_in_length;
airlock_thermal_vol = (airlock_in_rad + t_liner + t_bladder + t_restraint +
0.5*t_thermal)*2*pi*t_thermal*airlock_in_length;
airlock_shield_vol = (airlock_in_rad + t_liner + t_bladder + t_restraint + t_thermal +
0.5*t_shield)*2*pi*t_shield*airlock_in_length;

airlock_cap_area = (mod_in_rad + t_liner + t_bladder + t_restraint + t_thermal +
t_shield)^2 * pi;

airlock_liner_cap_vol = airlock_cap_area * t_liner;

```

```

airlock_bladder_cap_vol = airlock_cap_area * t_bladder;
airlock_restraint_cap_vol = airlock_cap_area * t_restraint;
airlock_thermal_cap_vol = airlock_cap_area * t_thermal;
airlock_shield_cap_vol = airlock_cap_area * t_shield;

airlock_liner_mass = airlock_liner_vol * d_liner;
airlock_bladder_mass = airlock_bladder_vol * d_bladder;
airlock_restraint_mass = airlock_restraint_vol * d_restraint;
airlock_thermal_mass = airlock_thermal_vol * d_thermal;
airlock_shield_mass = airlock_shield_vol * d_alum;

airlock_cylinder_mass = airlock_liner_mass + airlock_bladder_mass +
airlock_restraint_mass + airlock_thermal_mass + airlock_shield_mass;

airlock_liner_cap_mass = airlock_liner_cap_vol * d_liner;
airlock_bladder_cap_mass = airlock_bladder_cap_vol * d_bladder;
airlock_restraint_cap_mass = airlock_restraint_cap_vol * d_restraint;
airlock_thermal_cap_mass = airlock_thermal_cap_vol * d_thermal;
airlock_shield_cap_mass = airlock_shield_cap_vol * d_alum;

airlock_cap_mass = airlock_liner_cap_mass + airlock_bladder_cap_mass +
airlock_restraint_cap_mass + airlock_thermal_cap_mass +
airlock_shield_cap_mass;

airlock_total_mass = airlock_cylinder_mass + 2*airlock_cap_mass;

transport_mass = num_modules*m_module + num_airlocks*airlock_total_mass;
total_interior_volume = mod_in_vol*4*num_airlocks +
airlock_in_vol*num_airlocks + v_inner*num_modules;

%Regolith on main habitats

rad_minus_reg = rad_outer + t_shield;
rad_w_reg = rad_minus_reg + Reg_Thickness;

vol_reg_module = ((pi*(rad_w_reg)^2) - (pi*(rad_minus_reg)^2)) * length;
vol_reg_cap = a_cap * Reg_Thickness * 2;

vol_reg_airlock = ((airlock_in_rad + t_liner + t_bladder + t_restraint + t_thermal +
t_shield + Reg_Thickness)^2 - (airlock_in_rad + t_liner + t_bladder + t_restraint +
t_thermal + t_shield)^2)*pi*airlock_in_length;
vol_reg_airlock_cap = (airlock_cap_area * Reg_Thickness) *2;

```

```
Total_reg_volume = vol_reg_module + vol_reg_cap + vol_reg_airlock +  
vol_reg_airlock_cap;  
Total_reg_mass = Total_reg_volume * d_reg;
```

## Program 2 Habitat Optimization Scripts

### Script 1: CallSA.m

```

close all;
clear all;
global numpeople
format short g
bounds = [1 5; 1 11; 0.01 3]; % upper and lower bounds for each of the three variables:
radius, length, and floor height

X0 = [3; 5; 1.5]; %Possible initial design

options = zeros(1,9); % set up options array for non-default inputs

options(1) = 200; % initial temperature (default = 50)
options(6) = 0.5; % cooling rate (default = 0.5)
options(8) = 1000000; %Maximum allowed iterations (default = 100000)
options(9) = 0;

%calls the SA550.m function
numpeople = 8;
[x,fstar,count,accept,oob]=SA_550('SAfunc',bounds,X0,options);
constraints = finalcon(x)
inner_radius = x(1)
cylinder_length = x(2)
total_module_length = x(2)+2*x(1)
floor_height = x(3)
module_mass = fstar
volume = pi*x(1)^2*x(2)+4/3*pi*x(1)^3

```

### Script 2: SAfunc.m

```

function phi = SAfunc(x)
%finds the value of the objective function
f = objfun(x);
%finds the value of the constraint function
g = finalcon(x);

%determines penalty to be added if there are violated constraints
ncon = length(g);
P = 0;
for j = 1:ncon
    if g(j)>0

```

```

P = P + 10000*(1+g(j));
end
end

%adds penalty
phi = f + P;
end

```

**Script 3: objfun.m**

```

function f = objfun(x)
%objective function
d_liner = 960;
d_bladder_packed = 940;
d_restraint = 1400;
d_thermal = 1390;
t_liner = 1e-3;
t_bladder_packed = 6*13*1e-6;
t_restraint = 10*1e-3;
t_thermal = 5.34*1e-3;

d = d_liner*t_liner + d_bladder_packed*t_bladder_packed + d_restraint*t_restraint + d_thermal*t_thermal;

f = d*(2*pi*x(1)*x(2)+4*pi*x(1)^2);
end

```

**Script 4: finalcon.m**

```

function [g,h] = finalcon(x)
global numpeople
%list of constraints
g(1) = 1-(2*x(1)-2*x(3))/3.1;
g(2) = 1-findvolume(x)/(67*numpeople);
g(3) = x(3)-x(1);
g(4) = 1-(pi*x(1)^2*x(2)+4/3*pi*x(1)^3)/(103*numpeople);

%This problem has no equality constraints
h = [];
end

```

**Script 5 findvolume.m**

```
function V = findvolume(x)
r = x(1);
l = x(2);
f = x(3);
w = 2*sqrt(r^2-(r-f)^2);
Abox = w*(2*r-2*f);
Atop = acos((r-f)/r)/2*r^2-w/2*(r-f);
Vcyl = (Abox+Atop)*l;
Vhemicyl = 0.3*2*pi*(w/2)^2*(2*r-2*f);
Vhemitop = 0.3*2*(acos((r-f)/r)/2*r^2-w/2*(r-f))/2*pi*w;
V = Vcyl+Vhemicyl+Vhemitop;
end
```

### Program 3 Thermal Analysis Script

```

%% Introduction
% Author      Elijah Weinstein
% Team       Power and Thermal
% Date Completed  4/1/2021
% Purpose     Complete a thermal analysis of the 3 thermal
% systems that were considered through the span of this project.

% How it works This code takes values from hand calculations
% performed outside of the code using equations 1, 2, and 3 from
% the habitats section of the report then rescales the
% computed values for the final structure.

%% Initialize

modxD = 4.5          ; % m
modxL = 13.3         ; % m
Ax   = modxD*pi*modxL ; % m^2
Vint = 540.2          ; % m^3
Dint = 3.8*2          ; % m
L    = Vint/pi*4/Dint^2 ; % m
Dext = Dint+.3        ; % m
Aext = L*pi*Dext     ; % m^2
Ti   = 294            ; % K
Td   = 393            ; % K
Tn   = 120            ; % K
Q1d  = 2.2*1000       ; % W
Q1n  = -1.7*1000      ; % W
Q2d  = .05*1000       ; % W
Q2n  = -.11*1000      ; % W
Q3d  = .08*1000       ; % W
Q3n  = -.17*1000      ; % W
Vair = 16.2           ; % m^3
Lair = 2.5            ; % m
Rair = (Vair/Lair/pi)^.5 ; % m
Aair = 2*pi*Rair*Lair ; % m^2

%% Case 1: No Regolith Layer

dtxd = 294-358       ; % K
dtxn = 294-98          ; % K
phid = Q1d/Ax/(dtxd)*(Ti-Td) ; % W/m^2
phin = Q1n/Ax/(dtxn)*(Ti-Tn) ; % W/m^2
Q1dr = phid*Aext      ; % W

```

```

Q1nr = phin*Aext      ; % W
Q1da = phid*Aair      ; % W
Q1na = phin*Aair      ; % W

fprintf('\nFor no regolith daytime Q is %.*f kW nighttime Q is %.*f kW', 3,Q1dr/1000,
3,Q1nr/1000)

%% Case 2: Regolith Layer Directly on Habitat

dtxd = 294-374          ; % K
dtxn = 294-120          ; % K
Dreg = linspace(Dext+.2,Dext+8,2^8) ; % m
rreg = Dreg/2           ; % m
rmod = Dext/2           ; % m
m1  = log(rreg/rmod)    ; %
m2  = log((modxD/2+2)/(modxD/2))   ; %
phid = Q2d/Ax/(dtxd)*(Ti-Td)./m1*m2 ; % W/m^2
phin = Q2n/Ax/(dtxn)*(Ti-Tn)./m1*m2 ; % W/m^2
Q2dr = phid*Aext       ; % W
Q2nr = phin*Aext       ; % W
Q2da = phid*Aair       ; % W
Q2na = phin*Aair       ; % W
reg_thickness = rreg-rmod ; % m

figure(1)
plot(reg_thickness,Q2dr/1000)
hold on
plot(reg_thickness,Q2da/1000,'-b')
title('Regolith on Habitat')
xlabel('Regolith Thickness (m)')
ylabel('Daytime Heat Transfer (kW)')
grid on
yyaxis right
plot(reg_thickness,Q2nr/1000)
plot(reg_thickness,Q2na/1000)
ylabel('Nighttime Heat Transfer (kW)')
legend('Habitat Daytime','Airlock Daytime','Habitat Nighttime','Airlock Nighttime')

i = 1                   ;
while reg_thickness(i) < 1.165
    i = i+1             ;
end

fprintf('\nFor regolith on habitat day time Q is %.*f kW night time Q is %.*f
kW',3,Q2dr(i)/1000,3,Q2nr(i)/1000)

```

```
%% Case 3: Regolith Layer Separated from Habitat
```

```
dtxd = 294-374 ; % K
dtxn = 294-120 ; % K
Dreg = linspace(Dext+.5,Dext+8.3,2^8) ; % m
rreg = Dreg/2 ; % m
rmod = Dext/2 ; % m
m1 = log(rreg/(rreg(1)-.03)) ; %
m2 = log((modxD/2+4)/(modxD/2+2-.03)); %
phid = Q3d/Ax/(dtxd)*(Ti-Td)./m1*m2 ; % W/m^2
phin = Q3n/Ax/(dtxn)*(Ti-Tn)./m1*m2 ; % W/m^2
Q3dr = phid*Aext ; % W
Q3nr = phin*Aext ; % W
Q3da = phid*Aair ; % W
Q3na = phin*Aair ; % W
reg_thickness = rreg-rmod ; % m

figure(2)
hold on
plot(reg_thickness,Q3dr/1000)
plot(reg_thickness,Q3da/1000,'-b')
title('Regolith Separated from Habitat')
xlabel('Regolith Thickness (m)')
ylabel('Daytime Heat Transfer (kW)')
grid on
yyaxis right
plot(reg_thickness,Q3nr/1000)
plot(reg_thickness,Q3na/1000)
ylabel('Nighttime Heat Transfer (kW)')
legend('Habitat Daytime','Airlock Daytime','Habitat Nighttime','Airlock Nighttime')

j = 1 ;
while reg_thickness(j) < 1.165
    j = j+1 ;
end

fprintf('\nFor regolith seperated from habitat day time Q is %.*f kW Night time Q is
%.*f kW\n', 3,Q3dr(i)/1000, 3,Q3nr(i)/1000)
```

## Program 4 Power Analysis Scripts

```
% %
% Author: Dylan Anderson
% Last Revision: 3/13/21
%
% Program Description: This program will plot the power
% system cost for a moon colony for nuclear and solar power
% for AAE 450 Senior Design Project. It will output all
% requirements for power allowing for optimization of mass
% and dollar value budget for the system.
%
clc;clear;
m_colonists=25;
colonists = linspace(0,m_colonists,m_colonists+1);
power_per_colonist=10;%kw seen figures from 13-30kw/person
time=29.5*24*0.15*1.25%hours spent on
batteries=lunar_cycle_days*hours*darkness_percent*safety_factor
kg_to_moon = 27000;
format
battery_kWh = (colonists*power_per_colonist*time)
b_cost_kWh=200;
b_kWh_per_kg=0.3;
b_kg=(battery_kWh/b_kWh_per_kg)
battery_cost = b_kg*b_cost_kWh+ b_kg*kg_to_moon
r_kW = 40;
num_reactors = ceil(colonists*power_per_colonist/r_kW)
r_kg = 3300;
r_cost=215000000;
reactor_cost=1400000000+num_reactors*r_cost+num_reactors*r_kg*kg_to_moon
plot(colonists,battery_cost)
hold on
grid on
plot(colonists,reactor_cost)
title('Nuclear vs Solar power Cost for each colonist')
xlabel('Number of Colonists')
legend('Solar Power','Nuclear Power')
ylabel('Cost')
```

```
% %
% Author: Dylan Anderson
% Last Revision: 2/3/21
```

```

%
% Program Description: This program will plot the power
% generated, used, and what is stored for a moon colony
% for AAE 450 Senior Design Project. It will output all
% requirements for power allowing for optimization of mass
% and dollar value budget for the system.
%
% %
clc
clear
t_span = 29.5*24; %1 lunar period

%Power_Budget= another power option if we have power required for all subsystems.
Same as total power required
Number_Colonists = 20;
Power_Per_Colonist= 7; %kWe/colonist basic approx found online
S_Power_Per_kg= 0.08; %kWe/kg basic value found online
S_kg_Per_m2= 15; %kg/m^2
S_Power_Per_m2=0.04; %kW/m^2 basic average from ISS
N_Power_Per_kg=0.005556; %kWe/kg %based on 10KW Kilopower Reactor
B_Effic=0.9; %base figure

Nuclear_kg=0; %kg 1800kg per 10 KW Kilopower Reactor 0 as we are not using nuclear
power
Solar_kg=2500; %kg

%Calculate power generated by Nuclear and Solar Power
Nuclear_Power = N_Power_Per_kg*Nuclear_kg;
Solar_Power = S_Power_Per_kg*Solar_kg;

storage_need=(Number_Colonists*Power_Per_Colonist-
Nuclear_Power)*t_span*0.2/B_Effic;
battery_kg=storage_need/0.3;

t = linspace(0,t_span,t_span)
Solar_Efficiency = 0.90*(t<0.8*t_span)+(90*(t/t_span).^2-
162.*((t/t_span)+72.9).*(t>0.8*t_span)); %basic sine wave for now

Power_Generated = Nuclear_Power+Solar_Power.*Solar_Efficiency;
%Calculate Power consumption by average per colonist
Power_Used=Number_Colonists*Power_Per_Colonist.*ones(1,length(t)); %Total
power required kWe

%Calculate Power Storage
Power_Storage_Rate = (Power_Generated-Power_Used).*B_Effic;

```

```
Z = Power_Storage_Rate.*ones(1,length(t));
Power_Stored = cumtrapz(t,Z(1,:));
figure
subplot(2,2,1)
plot(t,Power_Stored)
title('Power Stored KWh')
xlabel('Time (Hours)')
ylabel('KW-hours stored')
grid on
subplot(2,2,2)
plot(t,Power_Generated)
title('Power Generated KW')
xlabel('Time (Hours)')
ylabel('KWs generated')
subplot(2,2,3)
plot(t,Power_Used)
title('Power Used KW')
xlabel('Time (Hours)')
ylabel('KWs used')
```

## Program 5 Estimation of Solar Power Cost

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as sts

sns.set_theme()
def gen_beta(mode, avg):
    a = (-1 * avg + 2 * mode * avg) / (mode - avg)
    b = (a - 1 - (a - 2) * mode) / mode

    return sts.beta(a, b)

def gen_fig(data, fig_title):
    fig = plt.figure(figsize=(8, 6))
    ax = plt.gca()
    sns.histplot(data, stat='probability')
    ax.set_xlabel('Marginal Cost [$USD 2021/W]')
    ax.set_title(fig_title)

    return fig

# 702 cost (2013)
# https://spacenews.com/34454new-boeing-satellite-platform-drawing-lots-of-customer-interest/

# assuming beta distribution with 4% cost median, 5% avg [recurring]
# assuming 25% is development cost, over 45 vehicles, scaling for 100kW
sat_panels = 85000000 * gen_beta(.04, .05).rvs(size=2**14) / 5500 * .75
sat_panels_dev = sat_panels / 3 * 100 * 1000

fig = gen_fig(sat_panels, '702SP Cost Estimate (2013)')
fig.savefig('satellite_estimate.png', dpi=300)
pass

# ISS Cost (2021)
# https://spaceflightnow.com/2021/01/13/boeing-says-assembly-complete-on-first-set-of-new-space-station-solar-arrays/

# assuming normal distribution with 65% cost median, 7.5% standard deviation [recurring]
iss_panels1 = 103000000 * sts.norm.rvs(.65, .075, size=2**14) / 120000

# assuming 25% dev cost with 5% standard deviation

```

```
iss_panels1_dev = 103000000 * sts.norm.rvs(.25, .05, size=2**14)

fig = gen_fig(iss_panels1, 'ISS Solar Expansion Estimate (2021)')
fig.savefig('iss_estimate.png', dpi=300)
pass

# NASA Estimate (1981)
# https://ntrs.nasa.gov/citations/19810011645

# assuming a standard normal with $400 median, $75 std, brought forward to 2020 dollars
nse1 = sts.norm.rvs(400, 50, size=2**14) * 2.8655

# assuming a 1.5-5x reduction in cost (mode=2, average=3)
nse1 /= 10 * (.1 + gen_beta(.1, .2).rvs(size=2**14))

# assuming 100kW arrays at 25% development cost
nse1_dev = .25 * 100 * 1000 * nse1

fig = gen_fig(nse1, 'NASA Solar Estimate with Cost Reduction (1981)')
fig.savefig('nasa_estimate.png', dpi=300)
pass

# Final Estimate

rng = np.random.default_rng()

pre_arr = [iss_panels1, sat_panels, nse1]
weights = np.array([5, 4, 2])

arrs = [None, None, None]
for i in range(len(pre_arr)):
    num_to_gen = int(2**14 / weights.sum() * weights[i])
    arrs[i] = pre_arr[i][rng.integers(0, 2**14, size=num_to_gen)]

res = np.hstack(arrs)

final_fig = gen_fig(res, 'Weighted Marginal Estimate')
pass

final_fig.savefig('res.png', dpi=300)

# Final Estimate Development

rng = np.random.default_rng()
```

```
pre_arr = [iss_panels1_dev, sat_panels_dev, nse1_dev]
weights = np.array([5, 2, 1])

arrs = [None, None, None]
for i in range(len(pre_arr)):
    num_to_gen = int(2**14 / weights.sum() * weights[i])
    arrs[i] = pre_arr[i][rng.integers(0, 2**14, size=num_to_gen)]

res_dev = np.hstack(arrs)

final_fig_dev = plt.figure(figsize=(8, 6))
ax = plt.gca()
sns.histplot(res_dev / 1e6, stat='probability')
ax.set_xlabel('Fixed Cost [$MM USD 2021]')
ax.set_title('Fixed Solar Investment at Current Production Rates')
pass

final_fig_dev.savefig('res_dev.png', dpi=300)

final_fig_dev = plt.figure(figsize=(8, 6))
ax = plt.gca()
sns.histplot(res_dev / 1e6 * ((2 - 1) * .3 + 1), stat='probability')
ax.set_xlabel('Fixed Cost [$MM USD 2021]')
ax.set_title('Fixed Solar Investment at 2x Production Rates')
pass

final_fig_dev.savefig('res_dev_speed.png', dpi=300)

line_samples = 2 ** 4
x = np.repeat(np.linspace(20, 400, 25), line_samples)

rng = np.random.default_rng()

ms = rng.choice(res, size=x.shape[0], replace=True)
bs = rng.choice(res_dev, size=x.shape[0], replace=True) * ((2 - 1) * .3 + 1)

a = .9
top_ms = np.sort(ms)[int(ms.shape[0] * (.5 + a / 2) ** 2)] # $/kW
top_bs = np.sort(bs)[int(bs.shape[0] * (.5 + a / 2) ** 2)] # $

y = bs + x * ms * 1000

fig = plt.figure(figsize=(10, 8))
ax = plt.gca()
sns.regplot(x=x, y=y / 1e6, ci=99, x_jitter=10)
```

```

ax.set_xlabel('Capacity Purchased [kW]')
ax.set_ylabel('Acquisition Cost [$MM USD 2021]')
ax.set_title('Solar Panel Monte Carlo Costing Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
ytxt2 = np.ptp(ax.get_ylim()) * .6 + ax.get_ylim()[0]
plt.text(10, ytxt, f'Mean Curve:\n$ r^2={lr.rvalue**2:.5f} $' " + 
r"$\frac{\sigma}{\sqrt{n}}=$" + f'$\{lr.stderr:.5f} $')
plt.plot(x, (top_bs + x * top_ms * 1000) / 1e6, 'r--')
plt.legend([f'Mean Fit: $y={lr.slope:.5f}*x+{lr.intercept:.5f}$',
            f'$100 * a:.0f}^{th} $ Percentile: $y={top_ms / 1e3:.5f}*x+{top_bs / 1e6:.5f}$'])

fig.savefig('linear_model.png', dpi=300)

fig = plt.figure(figsize=(8, 6))
ax = plt.gca()
sns.regplot(x=x, y=y / 1e6, ci=99, x_jitter=10)
ax.set_xlabel('Capacity Purchased [kW]')
ax.set_ylabel('Acquisition Cost [$MM USD 2021]')
ax.set_title('Solar Panel Monte Carlo Costing Result')
# lr = sts.linregress(ax.collections[0].get_offsets().data)
# ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
# ytxt2 = np.ptp(ax.get_ylim()) * .6 + ax.get_ylim()[0]
# plt.text(10, ytxt, f'Mean Curve:\n$ r^2={lr.rvalue**2:.5f} $' " + 
# r"$\frac{\sigma}{\sqrt{n}}=$" + f'$\{lr.stderr:.5f} $')
plt.plot(x, (top_bs + x * top_ms * 1000) / 1e6, 'r--')
# plt.legend([f'Mean Fit: $y={lr.slope:.5f}*x+{lr.intercept:.5f}$',
#             f'$100 * a:.0f}^{th} $ Percentile: $y={top_ms / 1e3:.5f}*x+{top_bs / 1e6:.5f}$'])

fig.savefig('linear_model_small.png', dpi=300)

```

**Program 6 Estimation of Solar Power Volume & Area**

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as sts
import pandas as pd
import statsmodels.formula.api as smf
import scipy

sns.set_theme()

class Model:
    def __init__(self, fnctn, confidence):
        self.model = fnctn
        self.confidence = confidence

model_list = []

# Rosa images from
# https://www.nasa.gov/planetarydefense/dart
# https://dart.jhuapl.edu/Gallery/

def rosa_model(num_samples):
    # guy looks shorter than his coworker, assume 5' 8" +- 5" for 3 sigma
    # assuming everything has like +- 10% error in 3 sigma

    person_h = 1.7 + sts.norm(loc=0, scale=.127 / 3).rvs(num_samples)
    sc_h = person_h * 675 / 725 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)

    bbox_h = sc_h * 294 / 192 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)
    bbox_d = bbox_h * 37 / 259 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)

    v = bbox_h * np.pi * (bbox_d / 2) ** 2

    power = 6.6 / 2 * np.ones(num_samples) # kw https://www.dss-space.com/post/dss-to-
    # provide-rosa-solar-array-for-nasas-double-asteroid-redirection-test-dart-mission

    deployed_l = sc_h * 805 / 142 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)
    deployed_a = deployed_l * sc_h * sts.norm(loc=1, scale=.1/3).rvs(num_samples)

    return np.vstack([v, power, bbox_h, bbox_d, deployed_a, deployed_l])

model_list.append(Model(rosa_model, 2))
```

```

# image from
# https://www.dss-space.com/bl-themes/dss/img/slide-1.jpg

def iss_model(num_samples):
    person_h = 1.8 + sts.norm(loc=0, scale=.15 / 3).rvs(num_samples)

    array_h = person_h * 470 / 169 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)

    # .5m width estimate, 20% error in 3 sigma
    cell_1 = .5 * sts.norm(loc=1, scale=.15/3).rvs(num_samples)
    array_1 = 32 * cell_1

    bbox_d = 3 * cell_1 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)

    bbox_h = (person_h / 2 * sts.norm(loc=1, scale=.1/3).rvs(num_samples)) + array_h

    v = (bbox_d / 2) ** 2 * np.pi + bbox_h

    power = 25 * np.ones(num_samples) # https://www.dss-space.com/products-flex-blanket

    array_a = array_1 * array_h

    return np.vstack([v, power, bbox_h, bbox_d, array_a, array_1])

model_list.append(Model(iss_model, 2))

samples = 2 ** 16

model_arr = [m.model(samples) for m in model_list]
res = np.hstack(model_arr)

stats_table = pd.DataFrame(data=res.T, columns=['Volume', 'Power', 'bbox_h', 'bbox_d', 'array_area', 'array_length'])

sns.histplot(res[1, :] / res[4, :])

x=res[1, :]
y=res[4, :]

x_model = np.linspace(2, 30)
a = .9
mod = smf.quantreg('array_area ~ Power', stats_table)
fit = mod.fit(q=a)

```

```

top_bs = fit.params['Intercept']
top_ms = fit.params['Power']
y_model = top_bs + x_model * top_ms

# Begin Plot
fig = plt.figure(figsize=(10, 8))
ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=.5)
plt.plot(x_model, y_model, 'r--')

# Helper Functions
ax.set_xlabel('Array Capability [kW]')
ax.set_ylabel('Array Size [$m^2$]')
ax.set_title('Solar Panel Monte Carlo Sizing Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
x_txt = np.ptp(ax.get_xlim()) * .05 + ax.get_xlim()[0]
plt.text(x_txt, ytxt, f"Mean Curve: \n r^2={lr.rvalue**2:.5f} $" +
          r"\frac{\sigma}{\sqrt{n}} = " + f"${lr.stderr:.5f}$")
plt.legend([f"Mean Fit: y={lr.slope:.5f}*x+{lr.intercept:.5f}",
            f"100 * a:.0f}^{th} Percentile: y={top_ms:.5f}*x+{top_bs:.5f}"])

fig.savefig('area_vs_power.png', dpi=300)

a_arr = np.linspace(.05, .95)
eff_arr = np.zeros(a_arr.shape)

for idx, a in enumerate(a_arr):
    mod = smf.quantreg('array_area ~ Power', stats_table)
    fit = mod.fit(q=a)
    top_bs = fit.params['Intercept']
    top_ms = fit.params['Power']

    eff_arr[idx] = 1 / (top_ms + top_bs / 15)

plt.figure(figsize=(8, 6))
plt.plot(eff_arr, a_arr)
plt.xlabel('Efficiency [kW/m^2]')
plt.ylabel('Probability P(Efficiency > x)')
plt.title('Probability of Solar Efficiency')

x=res[1, :]
y=res[4, :]

```

```

x_model = np.linspace(2, 30)
a = .9
mod = smf.quantreg('array_area ~ Power', stats_table)
fit = mod.fit(q=a)
top_bs = fit.params['Intercept']
top_ms = fit.params['Power']
y_model = top_bs + x_model * top_ms

# Begin Plot
fig = plt.figure(figsize=(6, 5))
ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=.5)
plt.plot(x_model, y_model, 'r--')

# Helper Functions
ax.set_xlabel('Array Capability [kW]')
ax.set_ylabel('Array Size [$m^2$]')
ax.set_title('Solar Panel Monte Carlo Sizing Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
# ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
# x_txt = np.ptp(ax.get_xlim()) * .05 + ax.get_xlim()[0]
# plt.text(x_txt, ytxt, f'Mean Curve:\n{lr.rvalue**2:.5f} + \n{r"\frac{\sigma}{\sqrt{n}} = " + f"\${lr.stderr:.5f}"')
# plt.legend([f'Mean Fit: $y={lr.slope:.5f}*x+{lr.intercept:.5f}$',
#            f'$100 * a:.0f}^{th} Percentile: $y={top_ms:.5f}*x+{top_bs:.5f}$'])

fig.savefig('area_vs_power_small.png', dpi=300)

x = stats_table['Power']
# y = stats_table['bbox_d'] ** 2 * stats_table['bbox_h']
y = stats_table['Volume']
x_model = np.linspace(2, 30)

a = .9
lumped_table = stats_table.copy()
lumped_table['Lumped'] = y
mod = smf.quantreg('Lumped ~ Power', lumped_table)
fit = mod.fit(q=a)
model_const = fit.params['Intercept']
mult_const = fit.params['Power']
y_model = mult_const * x_model + model_const

```

```

# Begin Plot
fig = plt.figure(figsize=(10, 8))
ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=.5)
plt.plot(x_model, y_model, 'r--')

# Helper Functions
ax.set_ylabel('Volume [m^3]')
ax.set_xlabel('Power [kW]')
ax.set_title('Solar Panel Monte Carlo Sizing Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
x_txt = np.ptp(ax.get_xlim()) * .05 + ax.get_xlim()[0]
plt.text(x_txt, ytxt, f"Mean Curve: \n r^2={lr.rvalue**2:.5f} " +
          r"\sigma = " + f"\sqrt{n} = {lr.stderr:.5f}")
plt.legend([f"Mean Fit: y={lr.slope:.5f}*x+{lr.intercept:.5f}",
            f"100 * a:.0f}^{th} Percentile: " +
            f"y={mult_const:.5f}*x+{model_const:.5f}"])

fig.savefig('power_vs_volume.png', dpi=300)

x = stats_table['Power']
# y = stats_table['bbox_d'] ** 2 * stats_table['bbox_h']
y = stats_table['Volume']
x_model = np.linspace(2, 30)

a = .9
lumped_table = stats_table.copy()
lumped_table['Lumped'] = y
mod = smf.quantreg('Lumped ~ Power', lumped_table)
fit = mod.fit(q=a)
model_const = fit.params['Intercept']
mult_const = fit.params['Power']
y_model = mult_const * x_model + model_const

# Begin Plot
fig = plt.figure(figsize=(6, 5))
ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=.5)
plt.plot(x_model, y_model, 'r--')

# Helper Functions

```

```
ax.set_ylabel('Volume [m^3]')
ax.set_xlabel('Array Capability [kW]')
ax.set_title('Solar Panel Monte Carlo Sizing Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
# ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
# x_txt = np.ptp(ax.get_xlim()) * .05 + ax.get_xlim()[0]
# plt.text(x_txt, ytxt, f"Mean Curve:\n{lr.rvalue**2:.5f} + \n"
#           r"\frac{\sigma}{\sqrt{n}} = " + f"${lr.stderr:.5f}")
# plt.legend([f'Mean Fit: y={lr.slope:.5f}*x+{lr.intercept:.5f}', 
#            f'{100 * a:.0f}^{th} Percentile: 
#            y={mult_const:.5f}*x+{model_const:.5f}'])
fig.savefig('power_vs_volume_small.png', dpi=300)
```

### Program 7 Estimation of Solar Power Mass

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as sts
import pandas as pd
import statsmodels.formula.api as smf

sns.set_theme()

class Model:
    def __init__(self, fnctn, confidence):
        self.model = fnctn
        self.confidence = confidence

model_list = []

# https://space.nss.org/the-case-for-solar-power-from-space/
# Point estimate 20 kg / kW, assuming 2 sigma 13-25

def nss_model(num_samples):
    norm = sts.norm(loc=19, scale=3)
    return norm.rvs(num_samples)

model_list.append(Model(nss_model, 1))

# https://en.wikipedia.org/wiki/Boeing_702_3kW_to_12_kW_from_1500kg_to_2300kg
# https://www.nasa.gov/smallsat-institute/sst-soa-2020/power mass up to 1/3 of total
# weight

def b702_model(num_samples):
    sc_mass = sts.norm(loc=1900, scale=130).rvs(num_samples) # assume normal
    centered at center
    solar_mass = sc_mass * sts.norm(loc=.225, scale=.0275).rvs(num_samples) # 3 sigma
    normal 15% - 33% mass

    marginal = (3 + 18) / (1500 + 2300) # slope of scale
    sig = (marginal * 1500 - 3) / 3 # 3 sigma to min/max
    sig_rvs = sts.norm().rvs(num_samples)
    sig_rvs[np.abs(sig_rvs) > 3] = 0
    power_delivered = marginal * sc_mass + sig_rvs * sig # all together

    return solar_mass / power_delivered
```

```
model_list.append(Model(b702_model, 2))

# https://lasp.colorado.edu/home/maven/about/spacecraft/ 1.5kW
# https://www.nasa.gov/smallsat-institute/sst-soa-2020/power mass up to 1/3 of total
weight

def maven_model(num_samples):
    sc_mass = 809
    solar_mass = sc_mass * sts.norm(loc=.225, scale=.036).rvs(num_samples) # 3 sigma
normal 15% - 33% mass

    return solar_mass / 1.5

# Doesn't seem to work too well
# model_list.append(Model(maven_model, 2))

#
https://www.jpl.nasa.gov/nmp/st8/tech_papers/2005%20IEEE%20Aerospace%20Confe
rence%20_Big%20Sky_%20Paper-%20NGU%20ST8.pdf
# assuming normal distribution centered at 85 W/kg with 2 sigma at 103 W/kg

def ultraflex_model(num_samples):
    mid = 1e3 * 1 / 85
    high = 1e3 * 1 / 103
    norm = sts.norm(loc=mid, scale=(mid-high) / 2)
    print(mid)
    print((high-mid) / 2)
    return norm.rvs(num_samples)

model_list.append(Model(ultraflex_model, 2))

n = 2**8

mod_arr = [x.model(int(n * x.confidence)) for x in model_list]
res = np.hstack(mod_arr)

fig = plt.figure(figsize=(8, 6))
ax = plt.gca()
sns.histplot(mod_arr[1], stat='probability')
ax.set_xlabel('Marginal Mass [kg/kW]')
ax.set_title("Model Marginal Mass")
fig.savefig('marginal.png', dpi=300)

line_samples = 2 ** 4
x = np.repeat(np.linspace(20, 400, 25), line_samples)
```

```

rng = np.random.default_rng()

ms = rng.choice(res, size=x.shape[0], replace=True)
bs = 0

a = .9
top_ms = np.sort(ms)[int(ms.shape[0] * (.5 + a / 2) ** 2)] # $/kW
top_bs = 0

y = bs + x * ms

fig = plt.figure(figsize=(10, 8))
ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=10)
ax.set_xlabel('Capacity Purchased [kW]')
ax.set_ylabel('Mass [kg]')
ax.set_title('Solar Panel Monte Carlo Mass Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
ytxt2 = np.ptp(ax.get_ylim()) * .6 + ax.get_ylim()[0]
plt.text(10, ytxt, f'Mean Curve:\n{lr.rvalue**2:.5f} $\n" + r"\frac{\sigma}{\sqrt{n}} = $" + f'${lr.stderr:.5f}$')
plt.plot(x, (top_bs + x * top_ms), 'r--')
plt.legend([f'Mean Fit: $y={lr.slope:.5f}*x+{0:.5f}$',
            f'{100 * a:.0f}^{th} Percentile: $y={top_ms:.5f}*x+{top_bs:.5f}$'])

fig.savefig('linear_model.png', dpi=300)

fig = plt.figure(figsize=(6, 5))
ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=10)
ax.set_xlabel('Capacity Purchased [kW]')
ax.set_ylabel('Mass [kg]')
ax.set_title('Solar Panel Monte Carlo Mass Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
ytxt = np.ptp(ax.get_ylim()) * .75 + ax.get_ylim()[0]
ytxt2 = np.ptp(ax.get_ylim()) * .6 + ax.get_ylim()[0]
# plt.text(10, ytxt, f'Mean Curve:\n{lr.rvalue**2:.5f} $\n" + r"\frac{\sigma}{\sqrt{n}} = $" + f'${lr.stderr:.5f}$')
plt.plot(x, (top_bs + x * top_ms), 'r--')

fig.tight_layout()

fig.savefig('linear_model_small.png', dpi=300)

```

```
stats_table = pd.DataFrame(data=np.vstack([x, y]).T, columns=['Power', 'Mass'])
```

### Program 8 Estimation of Battery System Mass

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as sts
import pandas as pd
import statsmodels.formula.api as smf

sns.set_theme()

class Model:
    def __init__(self, fnctn, confidence):
        self.model = fnctn
        self.confidence = confidence

model_list = []

#
https://www.nasa.gov/sites/default/files/atoms/files/ves16_cell_batt_design_yborthomieu.pdf
# Slide 24

def saft_10s8p(num_samples):
    energy = sts.norm.rvs(loc=1300 / 1e3, scale=75 / 1e3, size=num_samples)
    mass = sts.norm.rvs(loc=11.5, scale=.5, size=num_samples)

    return np.vstack([energy, mass])

model_list.append(Model(saft_10s8p, 4))

#
https://www.nasa.gov/sites/default/files/atoms/files/ves16_cell_batt_design_yborthomieu.pdf
# Slide 7

def saft_cells(num_samples):
    mass = sts.uniform.rvs(loc=2, scale=30, size=num_samples)
    energy = sts.norm.rvs(loc=169, scale=4, size=num_samples) / 1e3 * mass

    mass = 1.05 * mass
```

```
return np.vstack([energy, mass])

model_list.append(Model(saft_cells, 3))

# https://eepower.com/news/saft-li-ion-batteries-selected-for-geo-telecom-satellites/#

def saft_orbital(num_samples):
    mass = sts.uniform.rvs(loc=2, scale=30, size=num_samples)
    energy = sts.norm.rvs(loc=120, scale=10, size=num_samples) / 1e3 * mass

    return np.vstack([energy, mass])

model_list.append(Model(saft_orbital, 1))

n = 2**8

mod_arr = [x.model(int(n * x.confidence)) for x in model_list]
res = np.hstack(mod_arr)

stats_table = pd.DataFrame(data=res.T, columns=['Energy', 'Mass'])

x_name = 'Energy'
y_name = 'Mass'

x=stats_table[x_name]
y=stats_table[y_name]

# Model Fit
x_model = np.linspace(0, 5)
a = .9
mod = smf.quantreg(y_name + ' ~ ' + x_name, stats_table)
fit = mod.fit(q=a)
top_bs = fit.params['Intercept']
top_ms = fit.params[x_name]
y_model = top_bs + x_model * top_ms

# Best case scenarios
y250 = x_model / (250 / 1e3)
y300 = x_model / (300 / 1e3)

# Begin Plot
fig = plt.figure(figsize=(10, 8))
```

```

ax = plt.gca()
sns.regplot(x=x, y=y, ci=99, x_jitter=.5)
plt.plot(x_model, y_model, 'r--')

plt.plot(x_model, 3.5 + y250, ls='--', c=sns.color_palette()[2])
plt.plot(x_model, 3.5 + y300, ls='--', c=sns.color_palette()[4])

# Helper Functions
ax.set_xlabel('Battery Size [kWh]')
ax.set_ylabel('Battery Mass [kg]')
ax.set_title('Battery System Mass Result')
lr = sts.linregress(ax.collections[0].get_offsets().data)
ytxt = np.ptp(ax.get_ylim()) * .65 + ax.get_ylim()[0]
x_txt = np.ptp(ax.get_xlim()) * .05 + ax.get_xlim()[0]
plt.text(x_txt, ytxt, f"Mean Curve: \n r^2={lr.rvalue**2:.5f} " +
          r"\sigma = $" + f"\u03c3 = {lr.stderr:.5f} ")
plt.legend([f"Mean Fit: y={lr.slope:.5f}*x+{lr.intercept:.5f}", 
           '250 Wh/kg Opportunity', '300 Wh/kg Opportunity'])

fig.savefig('battery_mass.png', dpi=300)

```

## Program 9 Solar/Nuclear Trade Study

Noah Stockwell

```

import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.colors import Normalize, ListedColormap
import numpy as np
import seaborn as sns

def solar_model(launch_cost, capacity, launches_per_year,
                launch_capacity_non_starship, starship=False, amt_before_starship=.25):
    year = np.linspace(0, 10, 2**10)
    cost = np.zeros(year.shape)
    day_capacity = cost.copy()
    night_capacity = cost.copy()

    if = (1 + 6 / 28)

    tech_dev = 0

```

Noah Stockwell

```

procurement = lf * capacity * 0.53357 + 31.91107 + capacity * 24 * 5 * 8500 / 1e6
mass = capacity * 24.83783 * lf + capacity * 120 * 1000 / 250

tech_dev_horizon = 0
procurement_horizon = 120 / capacity # 120 kW / year capable

starship_introduction = 3
launch_capacity_starship = 67500
launch_cost_starship = 875 / launch_capacity_starship

print(procurement)
print(mass)

def day_capacity_model(mass_on_moon):
    per_array_capacity = 25
    per_array_mass = per_array_capacity * 24.83783
    tot_arrays = np.ceil(capacity / per_array_capacity)

    arrays_allowed = np.ceil(mass_on_moon / mass * tot_arrays)

    if mass_on_moon != mass:
        incremental_array_mass = min(per_array_mass, np.mod(mass_on_moon, mass / tot_arrays))

        num_complete_arrays_present = np.floor(incremental_array_mass / per_array_mass)
        num_complete_arrays_present += max(0, arrays_allowed - 1)
    else:
        num_complete_arrays_present = tot_arrays

    # tot_array_mass_delivered = incremental_array_mass + per_array_mass * max(0, arrays_allowed - 1)

    return num_complete_arrays_present * per_array_capacity

def night_capacity_model(mass_on_moon):
    battery_capacity = .250 # kWh/kg

    per_array_capacity = 25
    per_array_mass = per_array_capacity * 24.83783
    tot_arrays = np.ceil(capacity / per_array_capacity)

    arrays_allowed = np.ceil(mass_on_moon / mass * tot_arrays)

```

```

incremental_array_mass = min(per_array_mass, np.mod(mass_on_moon, mass / tot_arrays))

# num_complete_arrays_present = np.floor(incremental_array_mass / per_array_mass)
# num_complete_arrays_present += max(0, arrays_allowed - 1)

tot_array_mass_delivered = incremental_array_mass + per_array_mass * max(0, arrays_allowed - 1)

battery_mass_delivered = mass_on_moon - tot_array_mass_delivered

battery_capacity = (battery_mass_delivered * battery_capacity) / (5 * 24)

return min(capacity, battery_capacity)

## Calculations
tech_dev_mask = np.logical_and(0 <= year, year < tech_dev_horizon)
procurement_mask = np.logical_and(tech_dev_horizon <= year, year < procurement_horizon)

frac_year = year / tech_dev_horizon if tech_dev_horizon > 0 else 0
cost += tech_dev * (.75 * tech_dev_mask * (frac_year)**2 + .25)
cost[year > tech_dev_horizon] += tech_dev

contract_issued = np.where(year > tech_dev_horizon)[0][0]
contract_finished = np.where(year > tech_dev_horizon + procurement_horizon)[0][0]
cost[contract_issued:contract_finished] += procurement * .25
cost[contract_finished:] += procurement * .75

# Launch calcs
launches = np.arange(year[contract_finished], 10, 1 / launches_per_year)
launch_capacity = launch_capacity_non_starship * np.ones(launches.shape)

if starship:
    launch_capacity[launches > starship_introduction] = launch_capacity_starship
    launched_mass = np.cumsum(launch_capacity)

last_falcon_launch = np.where(launched_mass > mass * amt_before_starship)[0][0]
starship_cts = mass * (1 - amt_before_starship) / launch_capacity_starship

on_falcon_launches = launched_mass < mass * amt_before_starship
on_falcon_launches[:last_falcon_launch + 1] = True

```

```

for launch_num, launch_date in enumerate(launches[on_falcon_launches]):
    date_idx = np.where(year > launch_date)[0][0]

    if launch_num == 0:
        mass_on_vehicle = min(launch_capacity_non_starship, mass)
    else:
        mass_on_vehicle = min(launch_capacity_non_starship, mass - 
launched_mass[launch_num-1])

    cost[date_idx:] += launch_cost * mass_on_vehicle

    day_capacity[date_idx:] = day_capacity_model(min(mass,
launched_mass[launch_num]))
    night_capacity[date_idx:] = night_capacity_model(min(mass,
launched_mass[launch_num]))

first_starship_launch = np.where(launches > starship_introduction)[0][0]

starship_mass_delivered = 0
starship_num = 0
payload_to_deliver = mass * (1 - amt_before_starship)

while starship_mass_delivered < payload_to_deliver:
    starship_num += 1
    mass_on_vehicle = min(launch_capacity_starship, payload_to_deliver - 
starship_mass_delivered)
    starship_mass_delivered += mass_on_vehicle

    date_idx = np.where(year > launches[first_starship_launch + 
starship_num])[0][0]

    cost[date_idx:] += launch_cost_starship * mass_on_vehicle

    day_capacity[date_idx:] = day_capacity_model(min(mass,
starship_mass_delivered + mass * amt_before_starship))
    night_capacity[date_idx:] = night_capacity_model(min(mass,
starship_mass_delivered + mass * amt_before_starship))
else:
    launched_mass = launch_capacity * np.arange(1, launches.shape[0] + 1)

last_launch = np.where(launched_mass > mass)[0][0]
on_launches = launched_mass < mass
on_launches[:last_launch + 1] = True

for launch_num, launch_date in enumerate(launches[on_launches]):
```

```

date_idx = np.where(year > launch_date)[0][0]

if launch_num == 0:
    mass_on_vehicle = min(launch_capacity_non_starship, mass)
else:
    mass_on_vehicle = min(launch_capacity_non_starship, mass -
launched_mass[launch_num-1])

cost[date_idx:] += launch_cost * mass_on_vehicle

day_capacity[date_idx:] = day_capacity_model(min(mass,
launched_mass[launch_num]))
night_capacity[date_idx:] = night_capacity_model(min(mass,
launched_mass[launch_num]))

return year, cost, day_capacity, night_capacity

launch_cost = 30000

incremental_val = 1250
rocket_vals = np.arange(15000, 100000 + incremental_val, incremental_val)

least_diff = np.abs(rocket_vals - launch_cost)
launch_cost_dollars = rocket_vals[least_diff.min() == least_diff]

color_vals = np.ones((rocket_vals.shape[0], 3))
color_vals[rocket_vals == launch_cost_dollars] = [0, 0, 0]
cmap = ListedColormap(color_vals, N=color_vals.shape[0])

# ts, cs, ss, bs = solar_model(launch_cost_dollars / 1e6, 200, 10, 4000)
ts, cs, ss, bs = solar_model(launch_cost_dollars / 1e6, 200, 10, 4000, True, .25)
# tn, cn, sn, bn = nuclear_model(launch_cost_dollars / 1e6, 175, 10, 4000)
# tc, cc, sc, bc = np.array(solar_model(launch_cost_dollars / 1e6, 175 / 2, 10, 4000)) +
np.array(nuclear_model(launch_cost_dollars / 1e6, 175 / 2, 10, 4000))

fig, axes = plt.subplots(1, 3, figsize=(14, 4))

ax = axes[0]
ax.plot(ts+2021, cs, color=sns.color_palette("tab10")[0], lw=2)
# ax.plot(tn+2021, cn)
# ax.plot(ts+2021, cc)
ax.set_ylabel('Cost [$ USD Millions]')
ax.set_xlabel('Year')
# ax.set_xlim([0, 4500])

```

```

ax.set_title('Cost')

ax = axes[1]
ax.plot(ts+2021, ss, color=sns.color_palette("tab10")[1], lw=2)
# ax.plot(tn+2021, sn)
# ax.plot(ts+2021, sc)
ax.set_ylabel('Daytime Power [kW]')
ax.set_xlabel('Year')
ax.set_title('Daytime Power')

ax = axes[2]
ax.plot(ts+2021, bs, color=sns.color_palette("tab10")[2], lw=2)
# ax.plot(tn+2021, bn)
# ax.plot(ts+2021, bc)
ax.set_ylabel('Nighttime Power [kW]')
ax.set_xlabel('Year')
ax.set_title('Nighttime Power')

fig.suptitle('Parameters over Program Timeline', size='xx-large')
fig.tight_layout()

# norm = Normalize(15000, 100000)
#     fig.colorbar(cm.ScalarMappable(norm=norm,           cmap=cmap),      ax=axes,
orientation='horizontal', fraction=.05, aspect=60)

fig.savefig('parametersOverTimeline.jpg', dpi=180)

def nuclear_model(launch_cost, capacity, launches_per_year, launch_capacity):
    year = np.linspace(0, 10, 2**10)
    cost = np.zeros(year.shape)
    day_capacity = cost.copy()
    night_capacity = cost.copy()

    tech_dev = 1300
    procurement = capacity / 10 * 215
    mass = capacity / 10 * 3300

    tech_dev_horizon = 3.5
    procurement_horizon = 1

    def day_capacity_model(mass_on_moon):
        per_reactor_capacity = 10
        per_reactor_mass = 3300

```

```

num_reactors = np.floor(mass_on_moon / per_reactor_mass)

return min(capacity, num_reactors * per_reactor_capacity)

def night_capacity_model(mass_on_moon):
    return day_capacity_model(mass_on_moon)

## Calculations
tech_dev_mask = np.logical_and(0 <= year, year < tech_dev_horizon)
procurement_mask = np.logical_and(tech_dev_horizon <= year, year < procurement_horizon)

frac_year = year / tech_dev_horizon if tech_dev_horizon > 0 else 0
cost += tech_dev * (.75 * tech_dev_mask * (frac_year) ** 2 + .25)
cost[year > tech_dev_horizon] += tech_dev

contract_issued = np.where(year > tech_dev_horizon)[0][0]
contract_finished = np.where(year > tech_dev_horizon + procurement_horizon)[0][0]
cost[contract_issued:contract_finished] += procurement * .25
cost[contract_finished:] += procurement * .75

launches = np.arange(year[contract_finished], 10, 1 / launches_per_year)
launched_mass = launch_capacity * np.arange(1, launches.shape[0] + 1)

last_launch = np.where(launched_mass > mass)[0][0]
on_launches = launched_mass < mass
on_launches[:last_launch + 1] = True

for launch_num, launch_date in enumerate(launches[on_launches]):
    date_idx = np.where(year > launch_date)[0][0]

    if launch_num == 0:
        mass_on_vehicle = min(launch_capacity, mass)
    else:
        mass_on_vehicle = min(launch_capacity, mass - launched_mass[launch_num-1])

    cost[date_idx:] += launch_cost * mass_on_vehicle

    day_capacity[date_idx:] = day_capacity_model(min(mass,
launched_mass[launch_num]))
    night_capacity[date_idx:] = night_capacity_model(min(mass,
launched_mass[launch_num]))

```

```

return year, cost, day_capacity, night_capacity

incremental_val = 1250
rocket_vals = np.arange(15000, 100000 + incremental_val, incremental_val)

for val, launch_cost_dollars in enumerate(rocket_vals):
    color_vals = np.ones((rocket_vals.shape[0], 3))
    color_vals[rocket_vals == launch_cost_dollars] = [0, 0, 0]
    cmap = ListedColormap(color_vals, N=color_vals.shape[0])

    ts, cs, ss, bs = solar_model(launch_cost_dollars / 1e6, 175, 10, 4000)
    tn, cn, sn, bn = nuclear_model(launch_cost_dollars / 1e6, 175, 10, 4000)
    tc, cc, sc, bc = solar_model(launch_cost_dollars / 1e6, 175 / 2, 10, 4000) +
    nuclear_model(launch_cost_dollars / 1e6, 175 / 2, 10, 4000)

    fig, axes = plt.subplots(1, 3, figsize=(14, 5))

    ax = axes[0]
    ax.plot(ts+2021, cs)
    ax.plot(tn+2021, cn)
    ax.plot()
    ax.set_ylabel('Cost [$ USD Millions]')
    ax.set_xlabel('Year')
    ax.set_ylim([0, 6500])

    ax = axes[1]
    ax.plot(ts+2021, ss)
    ax.plot(tn+2021, sn)
    ax.set_ylabel('Daytime Power [kW]')
    ax.set_xlabel('Year')

    ax = axes[2]
    ax.plot(ts+2021, bs)
    ax.plot(tn+2021, bn)
    ax.set_ylabel('Nighttime Power [kW]')
    ax.set_xlabel('Year')

    fig.tight_layout()

    norm = Normalize(15000, 100000)
    fig.colorbar(cm.ScalarMappable(norm=norm,           cmap=cmap),           ax=axes,
    orientation='horizontal', fraction=.05, aspect=60)

```

```
fig.savefig(f'Animations/sim-{val+1}.jpg', dpi=300)
plt.close()

import shutil

new_v = rocket_vals.shape[0] + 1 + np.arange(rocket_vals.shape[0])
old_v = np.arange(rocket_vals.shape[0], 0, -1)

for idx in range(len(new_v)):
    new_num = new_v[idx]
    old_num = old_v[idx]

    shutil.copy(f'Animations/sim-{old_num}.jpg', f'Animations/sim-{new_num}.jpg')
```

**Script 5 Regolith Load Distribution Calculation**

```
global rho g h d omega phi
% Cage geometry (in mm)
z1 = -1126.91; % left z coordinate of the cage
z2 = 7309.13; % right z coordinate of the cage
L = z2-z1; % Cage width

% Parameters
rho = 1.6e-6; %regolith density (kg/mm^3)
g = 1675; %lunar gravity (mm/s^2)
h = 1000; %regolith layer thickness (mm)
d = 1500; %depth of fem (mm)

% sine wave parameters
% f = rho*g*h*d*sin(omega*t + phi)
omega = pi/L;
phi = -omega*z1;

% fun = @(z) rho*g*h*d*sin(omega*z + phi);

F_tot = integral(f, z1, z2)/1000; %total distributed load (N)

function f_out = f(z1, z2)
global rho g h d omega phi

% distributed force function
f_out = rho*g*h*d*sin(omega*z + phi);
end
```

## B. EVA Support

### Allocation of xEMU Suit Resources

```

%% EVA Supply Storage

%Airlock Configurations
total_volume = 7.26; %per airlock (m3)
num_airlocks = 2; %number of airlocks

%EVA Requirements
EVA_res = [4 5 6 7]; %EVAs for research
crew_res = 6; %crewmembers per EVA for research
EVA_rec = [1 1 2 3]; %EVAs for recreation per week
crew_rec = 4; %crewmembers per EVA for recreation
EVA_duration = 8; %max duration per EVA

%Supply Configurations
req_water1 = 0.454; %recharge water (kg/hour)
req_water2 = 13.608; %waste water (kg/suit)
rho_H2O = 1000; %density of water (kg/m3)

req_O2 = 6.35; %total O2 per EVA (kg)
rho_O2 = 1.4290; %starting density of O2 (kg/m3)

%Totals
total_EVA = (EVA_res.*crew_res) + (EVA_rec.*crew_rec); %total number of EVAs per week
total_time = total_EVA.*EVA_duration; %hours of EVA time per week (all crew included)

total_water_mass = (total_time*req_water1) + (req_water2*total_EVA); %kg of water per week needed
total_water_volume = total_water_mass/rho_H2O;

total_ox_mass = req_O2*total_EVA;
total_ox_volume = total_ox_mass/rho_O2;

total_airlock_vol = total_volume*num_airlocks;

%Reassess Volume Allocation based on different oxygen compression levels
rho_O2_req = total_ox_mass/(total_airlock_vol - total_water_volume); %minimum density o2 (kg/m3)

```

```
rho_lox = 1141; %liquid oxygen density (kg/m3)
rho_O2_vary = [rho_O2_req:10:rho_lox]; %increase oxygen by 10kg/m3 from minimum required up to density of standard lox used in industry

O2_vol_vary = total_ox_mass./rho_O2_vary';
total_water_vary = total_water_volume'*ones(size(rho_O2_vary)); %simply assembling an array for total water for later comparison// does not vary in density

remaining_vol = total_airlock_vol - (total_water_vary+O2_vol_vary'); %remaining volume in the airlock available after density has been adjusted

%Additional EVAs enabled at each compression level
EVA_req = total_water_vary + O2_vol_vary';
enabled = remaining_vol./EVA_req;

%Plot to Visualize Utility of Compressing Oxygen to Different Degrees
plot(rho_O2_vary, O2_vol_vary)
hold on
plot(rho_O2_vary, total_water_vary)
title("Volume Allocation for Resources")
plot(rho_O2_vary, remaining_vol, 'r')
xlabel("Density of Oxygen (kg/m3)")
ylabel("Volume of Resource (m3)")
legend("Phase 1", "Phase 2", "Phase 3", "Phase 4")

%Plot to See How Many EVAs are enabled due to compression of oxygen
figure(2)
plot(rho_O2_vary, enabled)
title("EVAs Enabled by Oxygen Compression")
ylabel("Number of EVAs (per week)")
xlabel("Oxygen Density (kg/m3)")
```

## Total Power Requirements

```
% James Pannullo
```

```
% AAE 450 Senior Design
```

```
clc; clear
```

```
%% Power Requirements including Efficiencies
```

```
DC_convert = 0.9; % DC-DC converter efficiency
```

```
DC_switch = 0.9; % DC switching unit efficiency
```

```
charge = 0.97; % Battery charge efficiency
```

```
discharge = 0.98; % Battery discharge efficiency
```

```
lunarDay = 709; % Total number of hours in lunar day
```

```
time_dark = lunarDay * 0.15; % Dark hours in lunar day
```

```
time_light = lunarDay - time_dark; % Light hours in lunar day
```

```
power_used = linspace(50,250,100); % Vector of habitat power (kW) from 50-250
```

```
battery_power = power_used * time_dark / (discharge * DC_switch); % Calculates how many kWh needed for batteries
```

```
battery_power_gen = battery_power / (time_light * charge); % Calculates the rate we need to charge batteries
```

```
power_req = power_used / (DC_convert * DC_switch) + battery_power_gen; % Calculates total power required from solar panels
```

```
power_loss = power_req - power_used; % Difference Between power required and power needed creates power lost
```

```
figure
```

```
plot(power_used,power_req, 'linewidth', 2); grid on;
```

```
xlabel("Habitat Power Needs (kW)");
ylabel("Total Power Required from Solar Panels (kW)");
title("Total Power Requirements including inefficiencies");

figure
plot(power_used,battery_power, 'linewidth', 2); grid on;
xlabel("Habitat Power Needs(kW)");
ylabel("Battery Power (kWh)");
title("Total Battery Power Requirements including inefficiencies");

figure
plot(power_used,battery_power_gen, 'linewidth', 2); grid on
xlabel("Habitat Power Needs (kW)");
ylabel("Battery Power Generation (kW/h)");
title("Total Battery Power Generation Requirements including inefficiencies");

figure
plot(power_used,power_loss, 'linewidth', 2); grid on;
xlabel("Habitat Power Needs (kW)");
ylabel("Power Losses (kW)");
title("Total Solar Power Losses due to inefficiencies");
```

## II.Tables

### 1. Structures

### 2. Habitat Design Decision

| <b>Habitat Optimization Configurations</b>       |                                             |                                                                |                          |            |                 |              |                          |                 |                  |                        |               |                                    |
|--------------------------------------------------|---------------------------------------------|----------------------------------------------------------------|--------------------------|------------|-----------------|--------------|--------------------------|-----------------|------------------|------------------------|---------------|------------------------------------|
| Inner<br>Radius<br>range<br>optimizati<br>on (m) | Floor<br>height<br>from<br>bottom<br>of Hab | Central<br>Cylinde<br>r length<br>range of<br>optimiz<br>ation | People<br>per<br>habitat | Num<br>mod | Inner<br>volume | Mass<br>(Mg) | Full<br>Module<br>Length | Inner<br>Radius | Packed<br>Length | Packed<br>Diamet<br>er | Pack<br>Volum | Floor<br>Heigh<br>t from<br>Bottom |
|                                                  |                                             |                                                                |                          |            |                 |              | Length                   | Radius          | Packed<br>Length | Diamet<br>er           | Volum         | Bottom                             |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>0]                                | [1,100<br>0]                                                   | 10                       | 1209.1     | 15.56           | 3            | 28.73                    | 2               | 14.365           | 5.7618                 | 324.5         | 1.552                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>0]                                | [1,100<br>0]                                                   | 8                        | 989.03     | 12.95           | 3            | 23.9                     | 6               | 11.95            | 3.4929                 | 103.4         | 1.529                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>0]                                | [1,100<br>0]                                                   | 6                        | 712.77     | 11.51           | 1            | 27.177                   | 3               | 13.5885          | 5                      | 218.6         | 1.392                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>0]                                | [1,100<br>0]                                                   | 5                        | 651.62     | 9.087           | 8            | 17.044                   | 7               | 8.522            | 5                      | 161.4         | 1.591                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>0]                                | [1,100<br>0]                                                   | 4                        | 529.45     | 7.789           | 15.202       | 1                        | 7.601           | 5                | 135.6                  | 8             | 1.653                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,20]]                           | [1,100<br>[1,20]]                                              | 10                       | 1030.2     | 25.05           | 8            | 24.779                   | 3.841           | 12.3897          | 5.7625                 | 273.0         | 1.524                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,20]]                           | [1,100<br>[1,20]]                                              | 8                        | 987.43     | 12.97           | 5            | 24.055                   | 9               | 12.0275          | 5                      | 261.3         | 1.504                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,20]]                           | [1,100<br>[1,20]]                                              | 6                        | 759.27     | 10.44           | 6            | 19.83                    | 9               | 9.915            | 5                      | 198.2         | 1.574                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,20]]                           | [1,100<br>[1,20]]                                              | 5                        | 651.33     | 17.053          | 9.087        | 8                        | 6               | 8.5269           | 5.6649                 | 167.3         | 1.613                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,20]]                           | [1,100<br>[1,20]]                                              | 4                        | 536.08     | 7.791           | 5            | 15.260                   | 3.618           | 7.6302           | 5.4279                 | 134.7         | 1.702                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,15]]                           | [1,100<br>[1,15]]                                              | 10                       | 933.04     | 35.74           | 2            | 6                        | 8               | 11.3418          | 5.7627                 | 245.7         | 1.545                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,15]]                           | [1,100<br>[1,15]]                                              | 8                        | 824.01     | 22.88           | 4            | 20.33                    | 9               | 10.165           | 5                      | 215.0         | 1.574                              |
| [0.01,<br>[1, 3.842]<br>3]                       | [1,100<br>[1,15]]                           | [1,100<br>[1,15]]                                              | 6                        | 767.01     | 10.36           | 6            | 19.15                    | 9               | 9.575            | 5                      | 199.0         | 1.584                              |

|                         |        |    |                                                                    |
|-------------------------|--------|----|--------------------------------------------------------------------|
| [0.01,<br>[1, 3.842] 3] | [1,15] | 5  | 653.815<br>8 9.076 16.898 3.806<br>8.449 5.709 167.6 1.622         |
| [0.01,<br>[1, 3.842] 3] | [1,15] | 4  | 529.02 7.789 15.225 3.626<br>7.6125 5.439 134.7 1.653              |
| [0.01,<br>[1, 3.842] 3] | [1,11] | 10 | 37.04 3.841<br>747.47 9 18.68 6<br>9.34 5.7624 193.5 6<br>1.601    |
| [0.01,<br>[1, 3.842] 3] | [1,11] | 8  | 33.78 3.841<br>747.4 1 18.68 4<br>9.34 5.7621 193.5 6<br>1.601     |
| [0.01,<br>[1, 3.842] 3] | [1,11] | 6  | 20.04 3.841<br>747.58 2 18.68 9<br>9.34 5.7628 193.5 7<br>1.601    |
| [0.01,<br>[1, 3.842] 3] | [1,11] | 5  | 3.829<br>655.46 9.065 16.779 6<br>8.3895 5.7444 167.8 1<br>1.639   |
| [0.01,<br>[1, 3.842] 3] | [1,11] | 4  | 3.784 5.6761<br>540.2 7.758 14.531 1<br>7.2655 5 136.0 4<br>1.701  |
| [0.01,<br>[1, 3.842] 3] | [1,8]  | 10 | 38.02 3.841<br>608.44 2 15.68 8<br>7.84 5.7627 154.4 4<br>1.673    |
| [0.01,<br>[1, 3.842] 3] | [1,8]  | 8  | 35.40 3.841<br>608.32 4 15.68 4<br>7.84 5.7621 154.4 2<br>1.673    |
| [0.01,<br>[1, 3.842] 3] | [1,8]  | 6  | 31.03<br>608.53 5 15.684 3.842<br>7.842 5.763 154.4 5<br>1.673     |
| [0.01,<br>[1, 3.842] 3] | [1,8]  | 5  | 19.35 3.841 5.7628<br>608.49 7 8 9<br>7.8419 5 154.4 5<br>1.673    |
| [0.01,<br>[1, 3.842] 3] | [1,8]  | 4  | 7.762 3.760<br>542.73 6 14.629 8<br>7.3145 5.6412 135.8 6<br>1.623 |
| [0.01,<br>[1, 3.842] 3] | [1,6]  | 10 | 38.65 3.841<br>515.72 5 13.683 8<br>6.8415 5.7627 128.3 1.75       |
| [0.01,<br>[1, 3.842] 3] | [1,6]  | 8  | 36.47 3.841<br>515.72 7 5 8<br>6.84175 5.7627 128.3 1.75           |
| [0.01,<br>[1, 3.842] 3] | [1,6]  | 6  | 32.83 3.841<br>515.52 4 13.682 2<br>6.841 5.7618 128.3 1.749       |
| [0.01,<br>[1, 3.842] 3] | [1,6]  | 5  | 19.92 3.841 5.7625<br>515.7 7 4 7<br>6.8417 5 128.3 1.749          |
| [0.01,<br>[1, 3.842] 3] | [1,6]  | 4  | 18.05 3.841<br>515.7 4 13.683 8<br>6.8415 5.7627 128.3 1.75        |

**Table 31 The Starship Habitat Optimization Configuration Raw Data Output by the Optimization Algorithm**  
**Derek Carpenter**

**Table 32 Starship Habitat Optimization Configurations Summary of Options**

|          | Inner Volume [m <sup>3</sup> ] | Mass [Mg] | Inner Inflated Length [m] | Inner Radius [m] | Packed Length [m] | Packed Diameter [m] | Packed Volume [m <sup>3</sup> ] | Total Cost / Module [\$] |
|----------|--------------------------------|-----------|---------------------------|------------------|-------------------|---------------------|---------------------------------|--------------------------|
| Option 1 | 540.2                          | 7.758     | 14.531                    | 3.784            | 7.266             | 3.784               | 21.6                            | 726,395.88               |
| Option 2 | 767.01                         | 10.366    | 19.150                    | 3.835            | 9.575             | 3.835               | 28.8                            | 963,738.40               |
| Option 3 | 1209.1                         | 15.563    | 28.730                    | 3.841            | 14.365            | 3.841               | 43.3                            | 1,445,479.86             |

**Table 33 Weighted Decision Matrix of Options on One to Three Scale**

|               | Weight | Option 1 | Option 1 | Option 2 | Option 2 | Option 3 | Option 3 |
|---------------|--------|----------|----------|----------|----------|----------|----------|
|               |        |          | Weighted |          | Weighted |          | Weighted |
| Cost          | 0.1    | 3        | 0.3      | 2        | 0.2      | 1        | 0.1      |
| Mass          | 0.4    | 3        | 1.2      | 2        | 0.8      | 1        | 0.4      |
| Inner Volume  | 0.2    | 1        | 0.2      | 2        | 0.4      | 3        | 0.6      |
| Packed Volume | 0.3    | 3        | 0.9      | 2        | 0.6      | 1        | 0.3      |
| Total         |        |          | 2.6      |          | 2        |          | 1.4      |

### 3. Cost Estimates

| Cost Matrix     |                   |                     |                   |                  |                    |                        |                    |                               |                |      |  |
|-----------------|-------------------|---------------------|-------------------|------------------|--------------------|------------------------|--------------------|-------------------------------|----------------|------|--|
| Liner Mass (Mg) | Bladder Mass (Mg) | Restraint Mass (Mg) | Thermal Mass (Mg) | Liner Cost (USD) | Bladder Cost (USD) | Restraint Cost (USD)   | Thermal Cost (USD) | Rigidization Estimation (USD) | Total (USD)    | Cost |  |
| 0.3317          | 0.0253<br>4       | 4.873               | 2.565             | \$21,26<br>5.29  | \$491.6<br>0       | \$670,0<br>37.50       | \$33,60<br>1.50    | \$1,000<br>.00                | \$726,395.88   |      |  |
| 0.4432          | 0.0338<br>3       | 6.4638<br>52        | 3.427             | \$28,41<br>5.48  | \$656.7<br>3       | \$888,7<br>72.50       | \$44,89<br>3.70    | \$1,000<br>.00                | \$963,738.40   |      |  |
| 0.6650          | 0.0507<br>2       | 9.6982<br>91        | 5.1418            | \$42,63<br>4.43  | \$985.3<br>5       | \$1,333<br>,502.5<br>0 | \$67,35<br>7.58    | \$1,000<br>.00                | \$1,445,479.86 |      |  |

**Table 34 Cost of Top Three Habitat Options as Computed by Mass**

#### 4. Risk and Fault

| Ref ID | Intermediate Event     | Ref ID   | Basic Event                   | Likelihood (1-5) | Consequence (1-5) | Ov era II | Mitigation/Warning                                                                                                                      |
|--------|------------------------|----------|-------------------------------|------------------|-------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| SH-1   | Power system Failure   | SH - 1.1 | Solar panel failure           | 1                | 1                 | 1         | Multiple Solar Panel Systems for Redundancy                                                                                             |
|        |                        | SH - 1.2 | Battery failure               | 1                | 1                 | 1         | Multiple Battery Locations for Redundancy                                                                                               |
|        |                        | SH - 1.3 | Short circuit                 | 1                | 1                 | 1         | Proper Maintenance of Electrical Systems                                                                                                |
| SH-2   | Electrical Fire        | SH - 2.1 | Short circuit/sparking        | 1                | 4                 | 4         | FPE/ Fire prevention systems. Ability to shutdown electricity to individual habitats in event of emergency                              |
|        |                        | SH - 2.2 | Component Failure             | 1                | 4                 | 4         | Active inspection of components for signs of wear                                                                                       |
| SH-3   | Depressurization Event |          |                               |                  |                   | 0         |                                                                                                                                         |
|        |                        | SH - 3.1 | Minor Leak                    | 3                | 2                 | 6         | Readily repairable, quality control, regular monitoring of system pressure, maintenance                                                 |
|        |                        | SH - 3.2 | Intermediate Leak             | 1                | 4                 | 4         | Evacuation procedures, sealing off depressurized area, may or may not be repairable. Safe activities and monitoring of meteor activity. |
|        |                        | SH - 3.3 | Catastrophic Depressurization | 1                | 5                 | 5         | Obeying maximum system habitats, designing habitat to deal with minor meteoroids, monitoring substantial meteorite activity             |
| SH-4   | Airlock Door Jamming   |          |                               |                  |                   | 0         |                                                                                                                                         |
|        |                        | SH - 4.1 | Failure to Close              | 3                | 2                 | 6         | Proper maintenance of door's moving parts and mechanisms                                                                                |

|      |                           |      |                                                    |   |   |   |                                                                                                                                                                                                     |
|------|---------------------------|------|----------------------------------------------------|---|---|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      |                           | SH - | Failure to Open                                    | 3 | 2 | 6 | Proper maintenance of door's moving parts and mechanisms                                                                                                                                            |
|      |                           | 4.2  |                                                    |   |   |   |                                                                                                                                                                                                     |
| SH-5 | Airlock Fireproof Failure |      |                                                    |   | 0 |   |                                                                                                                                                                                                     |
|      |                           | SH - | Fire jumps across closed door                      | 2 | 4 | 8 | Proper maintenance and replacement of fireproof seals on airlock door                                                                                                                               |
|      |                           | 5.1  |                                                    |   |   |   |                                                                                                                                                                                                     |
| SH-6 | Airlock Airtight Failure  |      |                                                    |   | 0 |   |                                                                                                                                                                                                     |
|      |                           | SH - | Failure to contain depressurization                | 1 | 5 | 5 | Proper maintenance of airlock doors                                                                                                                                                                 |
|      |                           | 5.2  |                                                    |   |   |   |                                                                                                                                                                                                     |
| SH-7 | Structural Failure        |      |                                                    |   | 0 |   |                                                                                                                                                                                                     |
|      |                           | SH - | Regolith cage bends                                | 2 | 1 | 2 | Stress sensors added to the structure to monitor real-time stresses                                                                                                                                 |
|      |                           | 7.1  |                                                    |   |   |   |                                                                                                                                                                                                     |
|      |                           | SH - | Regolith cage punctures habitat                    | 1 | 3 | 3 | Monitor stress sensors, Check to ensure within acceptable tolerances, measure height of structure                                                                                                   |
|      |                           | 7.2  |                                                    |   |   |   |                                                                                                                                                                                                     |
|      |                           | SH - | Regolith cage blocks exit                          | 1 | 1 | 1 | Use other exits to mitigate, Warning includes bending of the structure and higher stress tolerances                                                                                                 |
|      |                           | 7.3  |                                                    |   |   |   |                                                                                                                                                                                                     |
|      |                           | SH - | Regolith cage damages ECLSS system                 | 1 | 1 | 1 | Backup or additional ECLSS system components fulfill the role of the missing ECLSS system component.                                                                                                |
|      |                           | 7.4  |                                                    |   |   |   |                                                                                                                                                                                                     |
| SH-8 | Assembly Failure on Moon  | SH - | Fabric tears during unpacking                      | 1 | 4 | 4 | Live in other habitats if possible, potentially catastrophic if all modules are destroyed                                                                                                           |
|      |                           | 8.1  |                                                    |   |   |   |                                                                                                                                                                                                     |
|      |                           | SH - | Interlocking components are not machined correctly | 1 | 5 | 5 | Live in existing Starship module for as long as possible, Attempt to assemble regardless of component interlocking concerns. Potentially inflate habitats by themselves instead of connecting them. |
|      |                           | 8.2  |                                                    |   |   |   |                                                                                                                                                                                                     |
|      |                           | SH - | Crane cannot lift habitats from starship           | 1 | 4 | 4 | Live in existing starship, Monitor and assess all available resources, attempt to deploy                                                                                                            |
|      |                           | 8.3  |                                                    |   |   |   |                                                                                                                                                                                                     |

|       |                                 |           |                                           |   |   |   | habitats using secondary deployment method                                                                                         |
|-------|---------------------------------|-----------|-------------------------------------------|---|---|---|------------------------------------------------------------------------------------------------------------------------------------|
|       |                                 |           |                                           |   |   |   | Move crew to living in other modules where radiation was not detected, isolate module from habitat system                          |
| SH-9  | Radiation Shield Failure        | SH - 9.1  | Radiation detected within crew quarters   | 1 | 3 | 3 | Note results of experiments may be affected, limit crew exposure in the module                                                     |
|       |                                 | SH - 9.2  | Radiation detected in lab module          | 1 | 3 | 3 | Test amount of radiation, (depending on amount) prepare emergency return to earth                                                  |
|       |                                 | SH - 9.3  | Radiation detected within food            | 1 | 4 | 4 |                                                                                                                                    |
|       | Tripping Hazards                |           |                                           |   | 0 |   |                                                                                                                                    |
| SH-10 |                                 | SH - 10.1 | Floor debris and tripping hazards present | 4 | 1 | 4 | Strict adherence to OSHA standards for tripping hazards.                                                                           |
|       |                                 |           |                                           |   |   |   | Basic medical equipment present for treating sprains, stabilizing more serious injuries such as breaks.                            |
| SH-11 | Fire Suppression System Failure |           |                                           |   | 0 |   |                                                                                                                                    |
|       |                                 | SH - 11.1 | Failure to detect fire                    | 1 | 5 | 5 | Regular inspection and testing of fire detection system. Built in redundancy.                                                      |
|       |                                 | SH - 11.2 | Failure to deploy                         | 1 | 5 | 5 | Regular inspection and maintenance of fire suppression system. Built-in sprinkler redundancy and parallel deployment architecture. |
|       |                                 |           |                                           |   |   |   | Presence of manually operated fire suppression equipment.                                                                          |

**Table 35 Risk and Fault Analysis for Habitat System**  
**William Stahlschmidt, Derek Carpenter**

| Initial Habitat Weighted Decision Matrix         |               |                  |                    |               |            |         |        |
|--------------------------------------------------|---------------|------------------|--------------------|---------------|------------|---------|--------|
| Habitat Type                                     | Survivability | Ease of Analysis | Ease of Deployment | Vehicle Usage | Modularity | Comfort | Totals |
| Rigid Premade Small Premade with Inflatable Arms | 1             | 1                | 0.2                | 0.286         | 0.8        | 0.5758  | 3.862  |
| Exclusively Inflatable Habitat                   | 0.8           | 1                | 0.8                | 1             | 0.8        | 0.7251  | 5.125  |
|                                                  | 0.5           | 0                | 1                  | 1             | 0.2        | 1       | 3.700  |

**Table 36 Initial WDM to Decide Habitat Type**

**Table 37 Exercise Equipment Sizing Calculations**

| Equipment Type | Physiologic                | Deployed**               | Installed                |                    |              |
|----------------|----------------------------|--------------------------|--------------------------|--------------------|--------------|
|                | Physiological*             | Physical                 | Hardware                 | **                 | Total Volume |
|                | Operational Dimensions (m) | Operational Volume (m^3) | Operational Volume (m^3) | Volume Range (m^3) | Range**      |
| Treadmill      | 2.3 H x 1.0 W x 1.8 D      | 4.14                     | 0.4                      | 0.6-2.8            | >5.5         |
| Resistance     | 2.3 H x 2.3 W x 1.8 D      | 9.522                    | 10.2                     | N/A                | >11.6        |
| Cycle          | 2.0 H x 1.5 W x 1.3 D      | 3.9                      | 4.2                      | N/A                | >4.4         |

\*Empty space needed by crew member to perform exercise

\*\*Installed and Deployed Volumes include the primary exercise device, crew stabilization hardware (handrails, benches, etc.), control panels (laptop), crew monitoring equipment, Vibration Isolation System or Mounting System

**Table 38 Habitat Power Requirements**

| Power Consuming Equipment             | Quantity  | Power Used      |            | Mission Critical |
|---------------------------------------|-----------|-----------------|------------|------------------|
|                                       |           | Unit Power (kW) | Total (kW) |                  |
| <b>Comms</b>                          |           |                 |            |                  |
| Habitat Antenna (to Wheatley Station) | 1         | 0.444           | 0.444      | Yes              |
| Interpersonal Radio                   | 19        | 0.004           | 0.076      | Yes              |
| Wifi Network                          | 1         | 1               | 1          | Yes              |
| DMR Radio Network                     | 1         | 1               | 1          | Yes              |
| <b>Life Support</b>                   |           |                 |            |                  |
| Water Recovery System                 | 1/habitat | 2.1             | 16.8       | Yes              |
| Air Filtration System                 | 1/habitat | 2.08            | 16.64      | Yes              |
| Active Thermal Control System         | 1/habitat | 0.5             | 4          | Yes              |
| Emergency Heating System              | 1         | 7.29            | 7.29       | Yes              |
| Bosh Reactor (Oxygen Recovery)        | 1/habitat | 1.5             | 12         | Yes              |
| Electrolysis System                   | 1         | 4.91            | 4.91       | Yes              |
| Solid Waste Digester                  | 1/habitat | 4.925           | 39.4       | Yes              |
| Liquid Waste Recycling                | 1/habitat | 0.424           | 3.392      | Yes              |
| <b>Habitat Systems</b>                |           |                 |            |                  |
| Individual Module Lighting System     | 1/habitat | 1.625           | 13         | Yes              |

| Personal Power Requirements | 1/colonist | 0.5   | 9.5   | No |
|-----------------------------|------------|-------|-------|----|
| Agriculture                 |            |       |       |    |
| Watering System             | 1          | 2.91  | 2.91  | No |
| Lighting System             | 1          | 1.33  | 1.33  | No |
| Control System              | 1          | 0.3   | 0.3   | No |
| Research and Science        |            |       |       |    |
| Habitat Research            | 1          | 2.405 | 2.405 | No |
| Rovers*                     |            |       |       |    |
| Clearing Rover*             | 1          | 1.04  | 1.04  | No |
| Sample Collection Rover*    | 1          | 0.95  | 0.95  | No |
| Mining Rover *              | 1          | 1     | 1     | No |
| Scouting Rover*             | 1          | 0.75  | 0.75  | No |

\*Rovers are capable of powering themselves however they may be charged using habitat power systems at times as such these values can be under both habitats and non-habitat power

**Table 39 Non-Habitat Power Requirements**

| Power Consuming Equipment | Quantity | Power Used      |            | Mission Critical |
|---------------------------|----------|-----------------|------------|------------------|
|                           |          | Unit Power (kW) | Total (kW) |                  |
|                           |          |                 |            |                  |

|                      |                            |   |      |      |     |
|----------------------|----------------------------|---|------|------|-----|
| Comms                |                            |   |      |      |     |
|                      | Wheatley Station           | 2 | 1.22 | 2.44 | Yes |
|                      | Communications Satellite   | 6 | 1    | 6    | Yes |
|                      | Observation Satellite      | 1 | 1    | 1    | No  |
|                      | Rover Radio                | 4 | 0.04 | 0.16 | No  |
| Research And Science |                            |   |      |      |     |
|                      | Rover Research Systems     | 1 | 0.23 | 0.23 | No  |
|                      | Satellite Research Systems | 1 | 0.14 | 0.14 | No  |
| Rovers               |                            |   |      |      |     |
|                      | Clearing Rover             | 1 | 1.04 | 1.04 | No  |
|                      | Sample Collection Rover    | 1 | 0.95 | 0.95 | No  |
|                      | Mining Rover               | 1 | 1    | 1    | No  |
|                      | Scouting Rover             | 1 | 0.75 | 0.75 | No  |

### III. Design Process and Theory

#### A. Airlock Research

Two potential airlock types were explored: 1) Airlock (AL) and 2) Suitlock (SL). The Airlock is an independent pressure vessel with multiple hatches leading to both the habitat and external environment. The Suitlock has suits integrated into the actual pressure bulkhead, allowing suits to stay on the dusty side of the hatch.

Six main objectives of an airlock were identified to compare the two options and guide the rest of the design process: 1) minimize mass, 2) minimize air loss, 3) minimize the transference of lunar dust to the habitat, 4) accommodate crew and cargo transport, 5) modularity and 6) reliability. An additional factor, habitat safety would affect the layout of airlocks and habitats.

The most basic AL consists of two hatches: one leading to the external environment and one leading to the habitat. The SL, on the other hand, requires four hatches. Two hatches would be used as rear entry suit hatches, in addition to hatches serving the same purpose as in the AL. One important factor for reliability is the seal length required for each airlock type. The seals pose the greatest risk for air leaks, therefore airlocks with longer seal length have higher leak risk. To minimize seal length at each hatch, while also maximizing the hatch height for convenience, the submarine-style hatch is the best solution. This hatch shape is similar to a vertical rectangle with semicircles at the top and bottom end. Also, since the AL requires only two hatches while the SL requires four, the AL is the favored design with respect to air loss and reliability.

Another key difference between the two airlock types is suit stowage. For the SL design, suits are entered directly from the habitat as they are integrated into the pressure hatches. Likewise, the suits are excited and left in the hatch after the EVA. There is much more flexibility in suit

stowage for the AL model. The suits could be donned, doffed and stored either in the AL itself or in a habitat anteroom. This difference in suit stowage results in a difference in dust controllability. Since the suits in the SL always remain within the airlock, dust is already mitigated almost completely. However, for the AL, extra dust control measures must be taken.

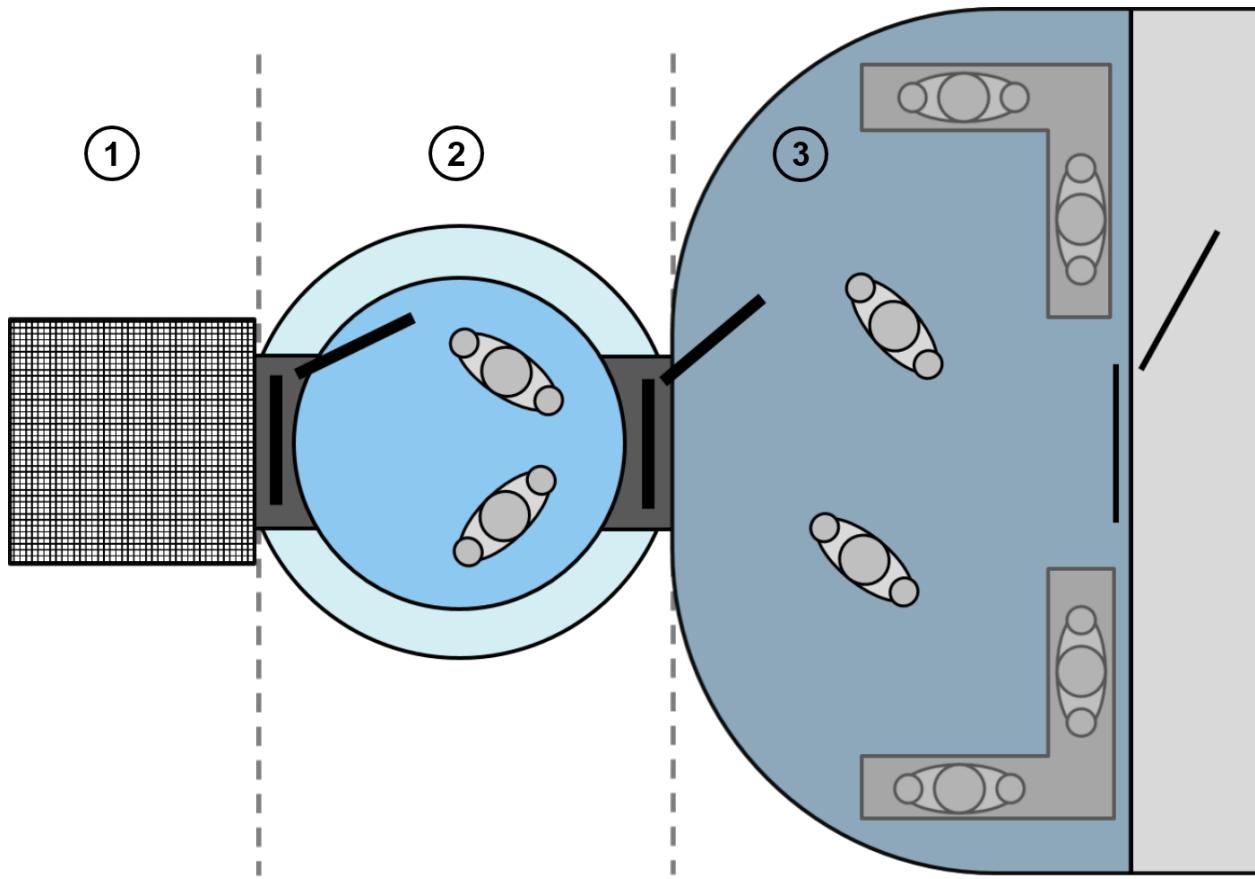
Since the AL is considered an independent component, while the suit ports in the SL need to be more integrated with the habitat, the AL is a more modular airlock. It can connect to any of the standard hatches being used in the habitat.

The final comparison to be made deals with airlock capacity and emergency situations. In an emergency, the maximum number of people able to use the SL is the same as the number of integrated suit hatches in the airlock (likely two). On the other hand, the maximum number of people able to use the AL is only limited by the number of people who can fit inside the airlock. This is because the suits would likely be stored in the habitat, allowing more people to don suits and be prepared to enter the airlock.

Based on the design objectives and the airlock type comparisons, the AL airlock was chosen to be used on the lunar colony. The AL is more modular, flexible and reliable than the SL. The only drawback of the AL is that it has less dust control inherent to its design than the SL. This will be mitigated by a habitat anteroom and various other dust control measures outside the airlock, inside the airlock and in the habitat.

## B. Initial Airlock Design Concept

The preliminary airlock design is based on the AL airlock system presented in Griffin [GC 1].



**Fig 81 Preliminary airlock design concept. 1) external dust removal station, 2) airlock, 3) EVA anteroom.**

Outside the airlock is the external dust removal station (1). This will be the raised platform at the door of the airlock. Here, anyone entering will try to shake off as much dust and debris from their suits as possible. It will also serve as a staging area for preparation of personnel and cargo to enter the airlock.

The next section of the airlock system is the airlock itself (2). This is a separate structure from the habitat that can be pressurized and de-pressurized to allow for habitat entry and exit without lowering pressure in the habitats. The airlock is equipped with two pressure sealed hatches

which will be in the shape of submarine style hatches. Within the airlock will likely be the best place for an air shower for thorough dusting off of the suits before entering the habitat.

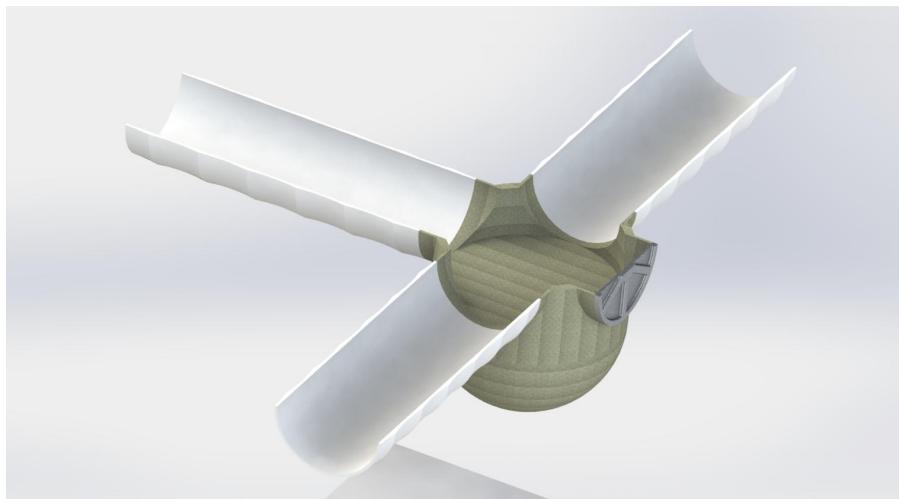
The final stage of the airlock system is an EVA anteroom (3). This will be a vestibule within the habitat from the airlock to the rest of the habitat which will serve several important purposes. First, it will be the last level of dust control because this is where the EVA suits will be taken off before entering the rest of the habitat. There will also be storage space in the anteroom for space suits and life support equipment. Since suits will be stored here, dust transference from the suits to the rest of the habitat should be minimal. The anteroom would also serve as a controlled mission preparation and debrief area. Lastly and most importantly, the EVA anteroom is where people will put on and take off the space suits.

There will be two use cases for the airlock system. First, the airlock system will be used as an interface between the habitats and the exterior. Second, the airlock system will be used as a safety precaution, connecting each of the habitat modules. In case of an emergency, these airlocks would be able to maintain the pressure seal of any individual habitat module. They will also allow ingress/egress at that module even if the neighboring module has had catastrophic failure. These intra-habitat airlocks likely will not include dust off and anteroom components.

Based on the proposed sizing of the preliminary design, the airlock would operationally serve two people at a time. But in the case of an emergency, four people will be able to fit inside the airlock.

### *1. Airlock Design Process*

After three major design iterations to reduce volume and mass whilst retaining usability for the crew, the final design was settled on. The first iteration was a connector-hub design that was not inflatable. This made the habitat overall less compact in transportation, especially given the volume-constrained launch schedule. The habitat was also more difficult to traverse in a timely manner, and did not provide a quickly accessible door in the case of fire or depressurization. Finally, the tube had connectors that had no flat floor and were only two meters in diameter. Therefore, traversing them would be a challenge, and additional support structures may have been necessary. Thus, the initial design did not satisfy later-refined safety concerns nor was it efficient in terms of mass or, in particular, volume.



**Fig 82 Initial Airlock Layout (CAD by John Matuszewski).**

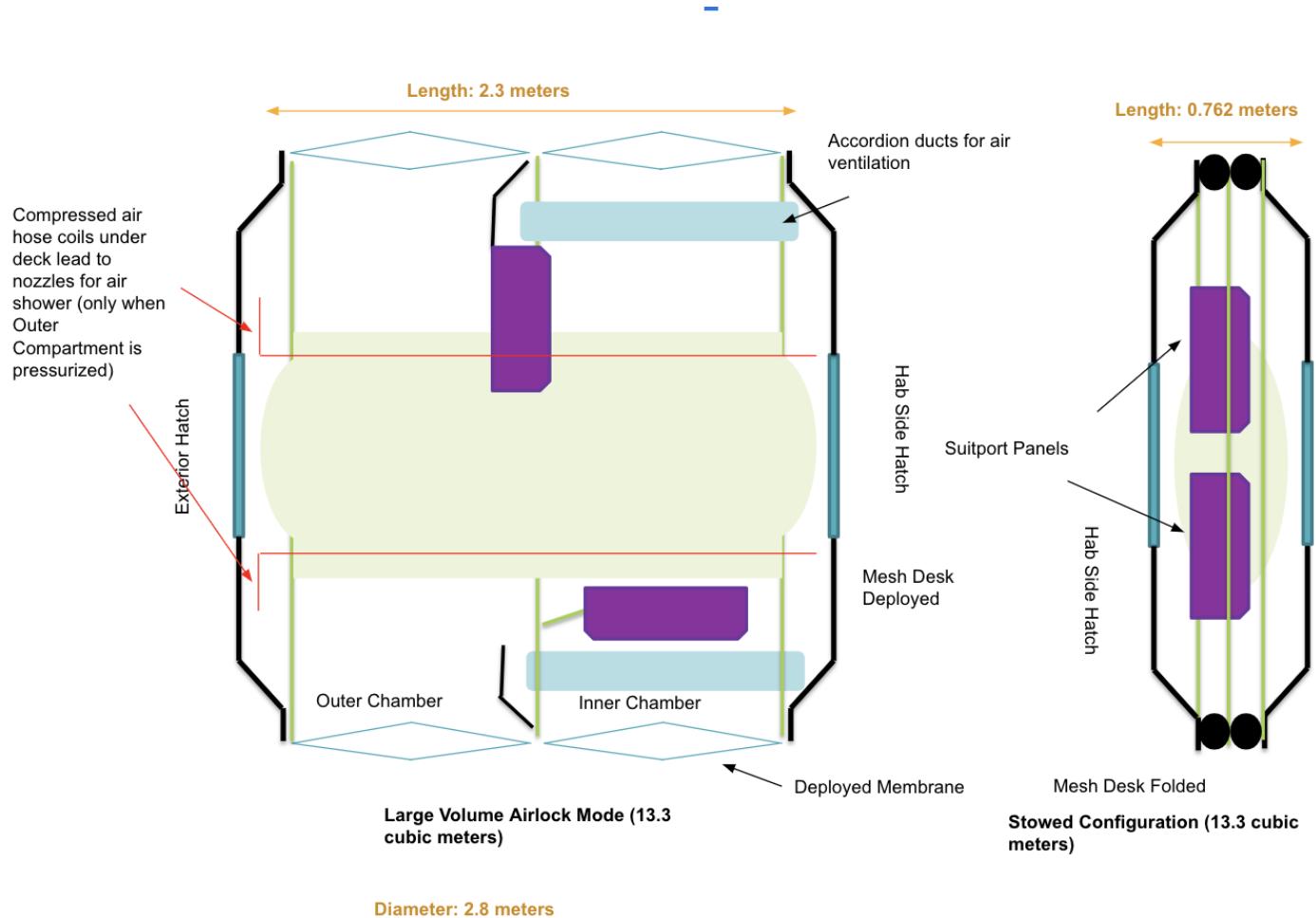
The second design was a radical departure from the initial connector-hub configuration. Instead of having several hallways connect to a central airlock hub, a much shorter hallway that was in itself the airlock was considered. This would allow for the elimination of the central hub component, saving mass and volume. Additionally, the safety concerns of not being able to rapidly seal off a room would be addressed. The efficiency in terms of mass and volume inherent to the

layout also permitted an enlargement of the internal radius as well as the installation of a floor. For this intermediate design iteration, the floor was at a height of .5 m above the bottom of the cylinder. The dimensions, outside of the floor height, were identical to the final configuration and are listed in the main report.

The final design iteration incorporated two minor design changes. The first change was lowering the floor height to 0.11 m above the bottom of the cylinder to make it flush with the habitat module's floor. The second change was a minor revision to the door design, transforming it from a nominal geometry into a fully constrained one within the CAD model.

## *2. Ingress/Egress Airlock Design (Alternate 1)*

An alternative design for the egress/ingress airlocks that was proposed is an inflatable variation, similar to the Dual-Chamber Hybrid Inflatable Suitlock (DCIS) design by NASA. This design is equipped with an air shower and UV lights to clean crew members and the airlock itself upon entrance and remove any contaminants. Two suitports are included to aid in donning and doffing the xEMU suits as well. Whenever the outer compartment of the airlock is pressurized, the ventilation system circulates the air in that chamber. Return air ventilation is under the perforated deck and filters dust-filled air from the air shower that will be used to clean additional dust off the EVA suits while the suits are mounted on the suitports for maintenance. An attractive feature of this design, as you can see in Fig. 83 below, is that it may be collapsed into a stowable configuration of length 0.762 m (from an extended length of 2.3 m), which will make transport and storage much easier, in addition to reconfiguration in layout of the habitat much simpler as more structures are added.



**Fig 83 Concept for alternate Inflatable Egress/Ingress Airlock.**

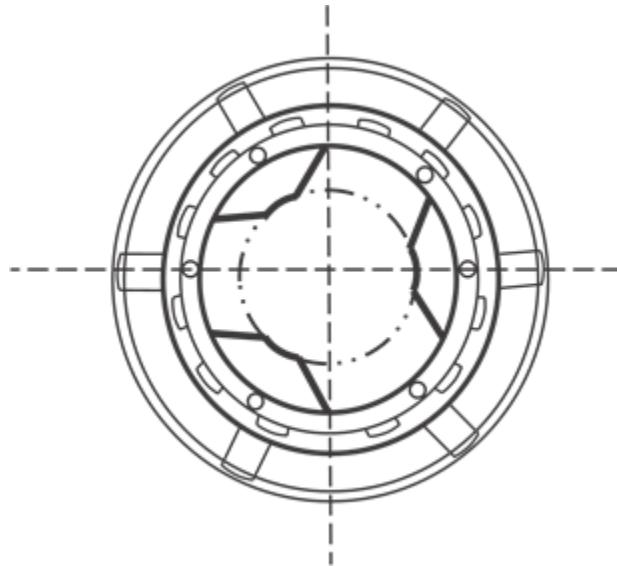
Within this airlock, additional space must be made for suit storage, suit maintenance, logistics and tools, with volume requirements of about  $1.75 \text{ m}^3$ ,  $1.1 \text{ m}^3$ , and  $2.3 \text{ m}^3$ , respectively, otherwise the suits and support supplies would have to be stored within the nearest room inside the habitat. Further, the resources required by the xEMU suits to resupply before and after each mission could either be stored beneath the mesh deck floor or in the room directly inside the habitat past the airlock. This design was discontinued at the point in the design process in which we chose to have all structures similar in build and non-inflatable for structural reasons.

An alternative design for the dust mitigation aspect that was not chosen for the finalized system included an air shower for the egress/ingress airlock modules. This alternate was not selected because of the desire to limit the use and loss of air. The air shower works by using high velocity (at least 0.9 cubic meters per second) to push particulates to the low-pressure side of the room over a period of roughly 45 seconds. The air shower's main filtration system employs a fused carbon nanofiber filter developed by Seldon Technologies designed specifically for lunar dust. This filter yielded an efficiency of 99.6% for particles of sizes 0.3-0.5 micron at elevated velocities of 12 m/min. The nozzle pattern on the air shower points downwards from the top at a 30 degree angle to produce the desired shearing, wash down effect, and all other nozzles on the other walls would point perpendicular to the wall surface. Alternatively, an ultra low particulate air filter was also considered to meet the stringent requirements of 99% efficient at 0.12 micron, though it is unknown whether it can handle the sharp, silica-like quality of lunar regolith. The estimated cost of each air shower unit that would fit 1-2 came out to approximately \$2,000 each, while taking up a space of 2.73 m<sup>3</sup> and consuming 3.5W of power max.

### *3. Rover Docking*

When considering lunar dust as one of the main hazards to health for a lunar colony, the best form of mitigation would be to minimize exposure to the lunar environment. Many EVA tasks can be accomplished with one of rovers, which includes the modular, manned rover BTB. Having a direct docking system between the manned rover and our external airlocks would allow EVAs to be completed without the need for additional areas for dust mitigation such as the showers and remove the need for the xEMU suit and all the resources and maintenance that comes with it. A design for this system was selected to meet both the needs of the rover mission and the airlock

design, though could not be implemented due to unexpected incompatibility in shape, requiring too much redesign of the manned rover system. The docking system, which is similar in design to the NASA Docking System (NDS), allows the rover to berth directly with the entrance to the airlock.



**Fig 84 Concept for Docking System (Active Side).**

This system was studied further because the design allows for a low-impact interaction, which will be important since it will have to go through quite a few docking cycles. The system includes two sides, a passive and an active side, which will go on the rover and airlock, respectively. Mechanical latches on the guide petals in the active ring clamp onto the passive section for contact and capture. This function supports both autonomous and piloted dockings, so it would potentially be compatible with both our autonomous and crewed rovers. Additionally, the passage diameter of 27", including the guide petals, is large enough to allow for individuals without suits to pass through easily. Once mated, the interface is also designed to transfer power, data, commands, air, communications, water, and pressurant to the rover, if necessary. In future development in this colony, an alternative may be to design a larger external airlock that will allow

for the rover to drive directly inside and then pressurize. Table 40 below lists the changes in resources that would be made to the existing systems based on the two options presented.

| <b>System</b>    |         | <b>Mass Increase</b>                  | <b>Power Increase</b>             | <b>Volume Increase</b>                                                    |
|------------------|---------|---------------------------------------|-----------------------------------|---------------------------------------------------------------------------|
| Rover Mechanism  | Docking | Active +319.3 kg<br>Passive +340.2 kg | Side: 220W nominal,<br>1.4kW peak | Active Side: ≈ 1.07 m <sup>3</sup><br>Passive Side: ≈ 1.07 m <sup>3</sup> |
| Drive-In Airlock |         | +2692.4 kg                            | +0kW                              | +187.26 m <sup>3</sup>                                                    |

**Table 40 Increase in Mass, Power, and Volume to Accommodate Rovers in External**

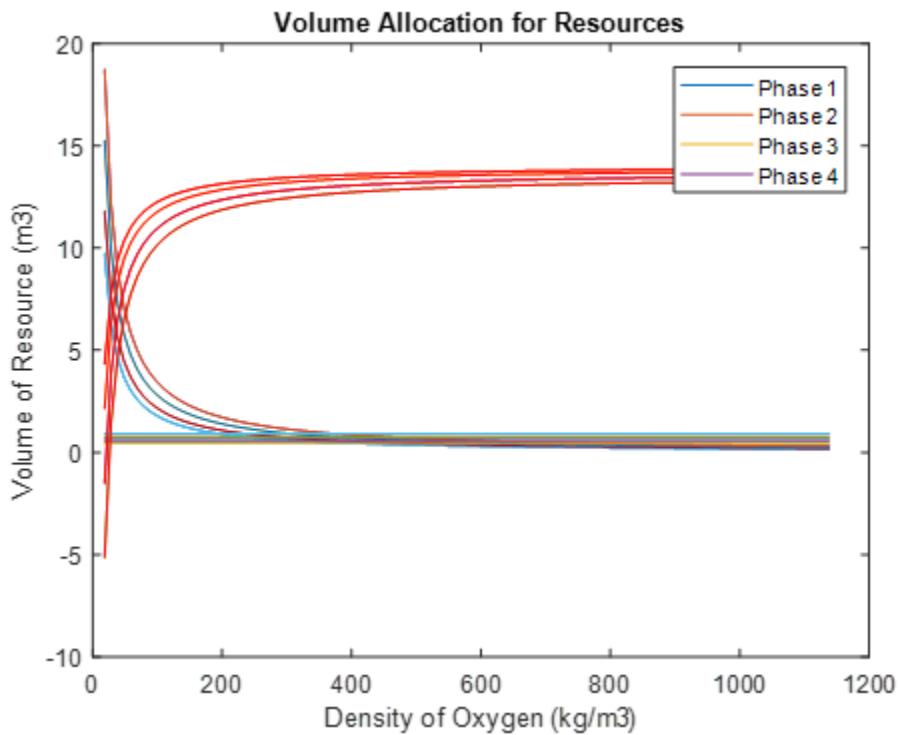
### Airlocks

Looking at the power increase column of Table 40, there is an additional though redundant system for the rover docking mechanism that does not run simultaneously, and it draws 100W nominal and 500W at peak. A compatible heater for the system can also be included for an additional 400W power. For the drive-in airlock power section, only structural aspects are considered, not any systems within the airlock that may be used to provide power or maintenance to the rover systems or for pressurization. The drive-in airlock design is much less favorable than the rover docking mechanism, for numerous reasons including the amount of additional resource and construction required to build such a large airlock. Further, this entire structure needs to be pressurized and unpressurized many times, requiring a decently large amount of air and potentially for air loss over the course of many cycles. Future consideration for a docking system will also need to account for the height of the door on the airlock, which is currently established for the

egress airlock, and the respective height that the door will reach on the rover. One simplified solution is to align the rover door to be similar to the height of the airlock door. For more precision, the BTB or any other future manned rover could have controls to adjust height to best lock into the airlock door.

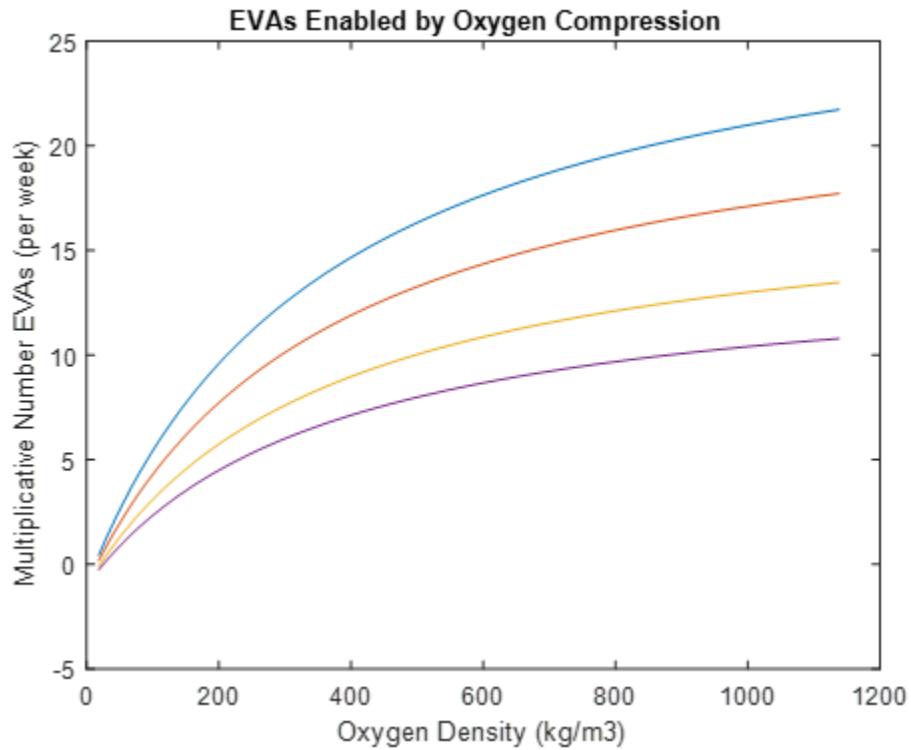
#### *4. EVA Suit Resource Storage*

At the calculated minimum compression requirement of  $18.3 \text{ kg/m}^3$ , there is exactly enough volume to meet the total EVAs expected at the colony's largest size, including the EVAs required solely for research and the EVAs expected for recreation activities. An important assumption here is that all missions that can be completed with the rover do not require the use of the xEMU suit and therefore do not require any of the supply allocated here. Keeping in mind that extra EVAs should be expected in the case of emergency or routine maintenance of the structures or rovers, the compression of oxygen should go beyond the minimum  $18.3 \text{ kg/m}^3$ , and ideally reach at least the  $150 \text{ kg/m}^3$  level to maximize available space.



**Fig 85 Volume Remaining (Red) and Volume Used (Colors) based on Oxygen Density.**

The red lines in Fig. 85 above represent the remaining volume available once the minimum requirements are met. As you can see, there are a few cases in which these lines become negative, and the intersection to once there is zero volume remaining is established as the minimum requirement for oxygen density. Each “phase” is defined by the number of EVAs required for scientific missions, and the respective amounts of resources required for them. The lines with the exponential decay trends are the oxygen requirements, while the straight constant lines are water requirements, which makes sense because only the density of oxygen is being varied.

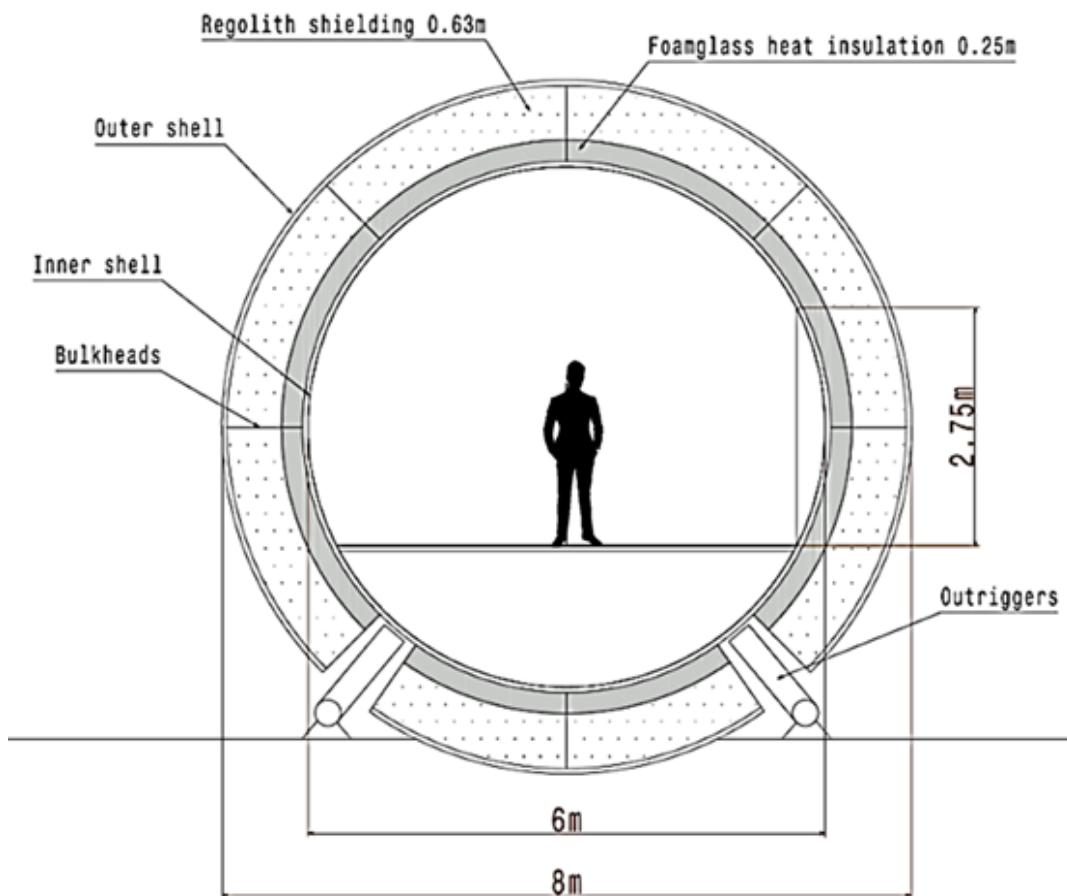


**Fig 86 Number of EVAs enabled by increasing density of stored O<sub>2</sub>.**

The above Fig. 86 shows the number of EVAs that can be done given the resources now available due to the compression of oxygen. This assumes that extra volume that becomes available once the resources are compressed will be filled with additional amounts of these same resources. The y-axis shows the “multiplicative” number of EVAs per week; for example, if the oxygen has a density of 150 kg/m<sup>3</sup>, then around 5x the number of required EVAs can be performed with lasting supplies. In the case that oxygen is chosen not to be compressed at all, the oxygen will have to be stored in additional areas within the habitat. All of the figures above were produced by the code found in the Appendix under the section titled “EVA Support”.

### C. Alternative Design Study for a Rigid Lunar Habitat

Prior to selecting an inflatable habitat for the colony, we considered designing a rigid habitat. When this study was conducted the requirements were for the colony to house 19 colonists with a habitable volume of  $65 \text{ m}^3$  per person. The habitat also had to withstand the load of the regolith and an internal pressure of 101.325 kPa with a safety factor of 1.65. The concept for the rigid habitat can be seen in Fig. 87.



**Fig 87 2D CAD Drawing of a rigid habitat for preliminary design consideration (CAD by Christian Mandrell).**

The habitat would be prefabricated on Earth and transported in one piece; therefore, it is designed to fit inside Starship's payload fairing. There is an inner and outer shell made from sheets of aluminum that surround the interior and exterior of the habitat respectively. The area between the shells is divided into sections by bulkheads, also made of aluminum, that provide structure to the habitat. Between the shells is a 0.25 m thick foam glass heat insulation layer. Once the habitat is on the Moon, regolith will be loaded into the structure through ports in the outer shell of the habitat. The regolith will fill the remainder of the space between the inner and outer shell. This regolith will provide radiation shielding as well as additional thermal insulation. Outriggers provide support and are used level the habitat.

The length of the habitat is 18 m, which is based on the size of Starship's payload fairing. From the model, we find that this habitat has a floor area of 85.76 m<sup>2</sup> and a habitable volume of 353.44 m<sup>3</sup>. Therefore, to support 19 colonists 3.5 habitats are needed. The payload mass is 81.36 Mg and 804.25 m<sup>3</sup> per habitat.

### *1. Alternative Design Studies for Regolith Radiation Shielding*

As mentioned in Section XVI of the report, large bags (1.165 m x 1.165 m x 1.5 m) filled with regolith are used to provide radiation shielding for the habitat. However, two alternate designs were considered: traditional sandbags and continuous tubular bags.

The first design study we conducted was using traditional sandbags. For this design, small sandbags are filled using a hopper and then carried and stacked by hand on the support structure. According to the Occupational Safety & Health Administration (OSHA), a single person can lift up to 22.68 kg on Earth [59]. However, given the difficulty of working in a spacesuit, this weight was reduced to 11.34 kg. Factoring in the reduced gravity on the lunar surface, we established that the colonists could lift bags up to 68.65 kg. With a total of 999.68 Mg of regolith required, approximately 14,500 sandbags are needed per module. Given the large quantity of sandbags required, we decided against this design. The physical strain on the colonists to fill, carry, and stack that high quantity of sandbags is unrealistic. Additionally, to stack sandbags on the upper portions of the structure, a ladder is required, creating additional physical and logistical challenges. This sandbag design also requires a large amount of EVA time for the colonists, exposing them to unnecessary risk. Finally, until the regolith is installed the colony is exposed to radiation, and the time required to completely shield a single module via sandbags leaves the colony exposed to radiation for too long.

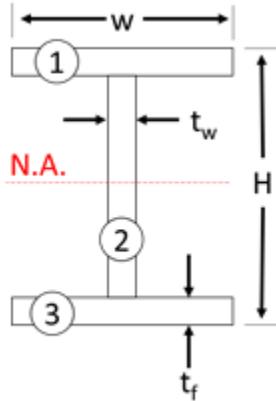
Another design alternative we explored was continuous tubular bags filled with regolith. This design is based on a patented technology from Superior Sandbag Systems [77] that uses specialized equipment to fill tubes up to 75 m long with sand or gravel (Fig. 88). On Earth, this system is capable of filling and installing a 0.36 m wide by 75 m long tube in 90 seconds with only a two-man crew. It is possible that the equipment by Superior Sandbag Systems could be modified

for use on the Moon to apply regolith bags to the habitat's support structure. However, its performance may be impacted by the Moon's reduced gravity. One issue for this application is that the regolith structure must be at least as wide as it is tall. Considering the height of the habitat, this results in a substantial increase in the amount of regolith required to cover the base. This extra regolith provides minimal additional radiation shielding, and requires more time and resources to gather and install. Finally, this system involves specialized equipment that would serve no further purpose once the regolith shielding has been completed.



**Fig 88 Equipment from Superior Sandbag System [77] creating a structure by applying layers of continuous tubular sandbags.**

## 2. Moment of Inertia for Regolith Support Structure I-Beams



**Fig 89 I-beam separated into three segments to calculate the moment of inertia. (Created by Christian Mandrell).**

To find the moment of inertia of an I-beam, it is split into three rectangular segments shown in Fig. 89. Because the section is symmetric, the Neutral Axis (N.A.) is at the center of segment 2. Using Parallel Axis Theorem, we can calculate the total moment of inertia.

$$I_{total} = \sum (\bar{I}_i + A_i d_i^2) \quad (5)$$

Where  $I_i$  is the segment's moment of inertia about its centroid,  $A_i$  is the segment's area, and  $d_i$  is the distance from the segment's centroid to the N.A. These values are calculated for each segment using Eq. 6 - 10. Because of the symmetry across the neutral axis, the values for segment 1 and 3 are the same.

$$\bar{I}_1 = \bar{I}_3 = \frac{1}{12} b h^3 = \frac{1}{12} w t_f^3 \quad (6)$$

$$\bar{I}_2 = \frac{1}{12} t_w (H - 2t_f)^3 \quad (7)$$

$$A_1 = A_3 = wt_f \quad (8)$$

$$A_2 = (H - 2t_f)t_w \quad (9)$$

$$d_1 = d_3 = \frac{1}{2}H - \frac{1}{2}t_f, d_2 = 0 \quad (10)$$

Substituting Eq.6 – 10 into Eq. 5 and simplifying, we find the final expression for the moment of inertia of an I-beam.

$$\begin{aligned} I_{total} &= \sum (\bar{I}_i + A_i d_i^2) = 2(\bar{I}_1 + A_1 d_1^2) + (\bar{I}_2 + A_2 d_2^2) \\ I_{total} &= 2 \left( \left( \frac{1}{12}wt_f^3 \right) + (wt_f) \left( \frac{1}{2}H - \frac{1}{2}t_f \right)^2 \right) + \left( \frac{1}{12}t_w(H - 2t_f)^3 + 0 \right) \\ I_{total} &= \frac{1}{6}wt_f(3H^2 - 6Ht_f + 4t_f^2) + \frac{1}{12}t_w(H - 2t_f)^3 \end{aligned} \quad (11)$$

### *3. Alternate Colonist Makeup Research*

Initially the design had considered a colony of 19 individuals. This number was selected from anticipated Starship capabilities as well as human factor research regarding team dynamic. An odd number was selected in hopes to better settle team related disputes that might occur on the lunar colony. Since the colony in the first phase of habitation would primarily be a research outpost the majority of individual roles were decided from scientific research goals assuming a 40% resource allocation to human and biomedical sciences, 40% to agriculture, and an initial 20% to geology and planetary science. The remaining roles were selected based on current ISS standards and historic crew makeup. There will be a commander, two psychologists, one practicing doctor, three technicians, five biomedical researchers, five botanists and pedologists, and then two geologists or volcanologists. Two psychologists are required due to the size of the crew and the fact that with only one psychologist there would be no one for them to speak with. Currently on the ISS crew members communicate with multiple psychologists at a minimum of once a week, due to the increased distance from Earth the colony hopes to provide better resources than what is currently available to the crews of the ISS. One practicing doctor is included, this is necessary because of the increased distance from the moon to the Earth as compared to low earth orbit. Additionally, the increased number of EVAs anticipated for the colony will increase the chance of injury. Three technicians were deemed necessary for habitat setup, rover setup, and troubleshooting faulty systems around the colony. The specialized research crew is broken down exactly the same way the resources for research are broken down.

After colonist roles have been selected it will be important to gather estimates for crew mass. The first assumption is that all of the crew in the first phase will adhere to the same astronaut physical standards that are currently held by NASA. This means that colonists must be between

62 and 75 inches and in good physical form. The average height for men in the United States is 69 inches. At that height healthy body mass indexes range from 65 kilograms to 80 kilograms. To provide a buffer for calculations the high end of the healthy BMI range will be used for calculations. For women in the United States the average height is 64 inches. At that height a healthy BMI would correspond to 49 kilograms to 60 kilograms. Once again the heavier end of the range will be used for calculations. The colonist make up is required to have both genders represented. The ratio of one gender to another may not exceed 15:4, this value was determined from psychology studies looking at minority representation. [46] The study looked at groups of 20 to 30 with a predetermined “small number of confederates” that attempted to change whatever convention the group had set. For groups with up to 24% confederates there were nearly no successful group conversions but starting at 25% group conversion became not just a possibility but common. It is important that whichever gender is less represented in the colony still doesn’t feel ostracized in the group dynamic and can still affect change. Now with the gender breakdown a mass estimate can be made for the entire group. For a 10 male nine female group the group mass can be estimated to be 1340 kg. For a four male 15 female group a mass of 1220 can be anticipated. The heaviest group that can be expected is 15 male four female which totals 1440 kg.

#### *4. Alternate design concepts*

The team also focuses on examining rigid structures assembled on Earth to reduce mass and volume estimations. Initial habitat payload requirements are estimated at 81.36 Mg and transport volume of 804.25 m<sup>3</sup>. However, these data points are based on all habitable crew space operating as livable space. This estimate minimizes the volume required per person and does not account for requirements outside of survival applications. The initial estimate temporarily suspends the data for required lab space, gym facilities, food generation habitat modules, and food preparation. This is justified for the time, since the systems and vehicles groups due to these numbers being an early estimate. Neither the concept of these modules nor the respective data is available early in the design process, so the team ignored these requirements.

### *5. EVA suit selection and details*

In order to be able to conduct crew activity outside the habitat, including construction, science, maintenance, and personal recreation, it is necessary that EVA (Extravehicular Activity) suits be included in the plans for the colony. While many tasks can be completed by specialized rovers, EVA suits are needed in order to solve problems that rovers cannot. In addition to the functional aspect of needing EVA suits, having EVA suits enables the astronauts the ability to leave the habitat, which will allow for the viewing and enjoyment of the lunar surface, a luxury not available inside the habitat due to the structural problems associated with windows. In addition to the functional and luxury aspects of having EVA suits, they will also simply be required at some point to transfer crew between the landing vehicles and the habitat. Also, having EVA suits can serve as a last resort of sorts in the event of a catastrophic system failure, wherein life support systems inside the habitat are knocked offline.

In past lunar missions, the space suit used was the A7L EMU system. This space suit had a mass of 91 kilograms and was able to support an EVA of 6 hours [66]. While these spacesuits kept the astronauts in the Apollo missions safe and allowed them to complete their missions, these suits had numerous complaints associated with them. These complaints included excessive dust collection on the suit, poor mobility, and poor flexibility. As a response to these problems, NASA has developed a new generation of spacesuits specifically designed for lunar missions.

These suits, called the xEMU, are structurally very different from the original Apollo era suits, and allow for much greater flexibility. In addition to improvements in mobility, these suits also have improved thermal capabilities, CO<sub>2</sub> scrubbing systems, communications, and computer interfaces. In terms of the CO<sub>2</sub> scrubbing technology, the xEMU suits differ greatly from the original Apollo era suits. In the original suits, the CO<sub>2</sub> scrubbing system used filters which had to be switched out after a period of time and could not be easily reused. In the xEMU suit, the CO<sub>2</sub> is scrubbed from the air and expelled out of the suit instead of keeping it in a filter or tank. This means that the astronaut can conduct longer EVA missions, and also allot more mass to other systems that improve functionality. This initially reduced mass also means that the suit can afford more redundant systems. In the original A7L suit, many of the systems did not have a backup, as there simply was not enough room. In the current xEMU suit, improvements in electronics and other systems have allowed for many of the systems to have duplicates and redundancies, which greatly increased their safety and the safety of our astronauts.

While there are other space suit options, there are none that compare to the xEMU system as a realistic option for the mission. Some of the potential alternatives include the current EMU system used on board the International Space Station and the Orlan line of space suits used by the Russian Space Agency. While these suits are very good for in-orbit spacewalks, they were not designed with planetary EVAs in mind, and are thus deficient in key areas. The most obvious area of deficiency is the comparative mobility and flexibility of the systems. According to experiments done by NASA scientists at Johnson Space Center, the xEMU prototype has a 25% higher reach envelope compared to the current EMU design, and has a 3.4 times larger intersection between the left and right hand reach envelopes [66]. This improvement is crucial for a planetary EVA mission, where falling over can have major consequences. A lack of mobility and flexibility led to many

astronauts falling over on previous Apollo missions . Falling on the moon can have significant consequences, such as damaging the suit or getting stuck. With the new xEMU, not only will falls be reduced, but the astronauts will also be able to stand up more easily if there was a fall. As a conclusion of these various reasons, the xEMU will be used for the lunar colony. While the xEMU is currently still in development and testing, it will be ready by FY 2023 according to NASA's schedule. Each suit will have a mass of approximately 120 kg, a cost of approximately \$12,000,000, and will be able to comfortably sustain an EVA of 8 hours [66].



**Fig 90 xEMU suit prototype from NASA press release.**

### **Spacesuit dust mitigation selection**

There are several potential future improvements to the xEMU suit that would allow for less initial dust accumulation on the suit. It is also important to note that several improvements have been already included in the xEMU system that are geared towards dust mitigation, with the most notable being that the suit is designed with very few potential dust entry points such as zippers or seams. This leads to less dust getting inside the suit's layers, which would lead to a much more difficult cleaning process. In terms of future improvements, there are several potential ideas, each with their own merit. The first potential design improvement would be the use of a "bunny suit". A bunny suit would be an extra thin fabric layer that the astronauts would wear overtop of their xEMU suits, with the hope that the bunny suit would intercept a majority of the lunar dust. This design would be lightweight, require no redesign of the suit itself, and greatly limit the amount of dust accumulated on the suit itself. However, this design would not limit the amount of dust that would be brought back into the habitat. This could lead to increased strain on the dust filters and removal systems. In addition, the act of donning and doffing a bunny suit before and after each EVA would increase the time necessary for these actions. The act of doffing the bunny suit after an EVA could also lead to the transfer of the accumulated dust to the xEMU itself, and thus would render the bunny suit useless.

The second potential improvement is the use of an electrostatic charge in the suit, which can be implemented through the use of copper wires or carbon nano-tubes. Lunar dust carries a significant electric charge, and thus if the suit is also charged, the suit can repel the dust with significant efficiency [46]. This method has numerous benefits, including the reduction of lunar dust that reenters the habitat, a much higher reusability than the bunny suit method, and a massive

increase in ease of use. While the bunny suit method would require many disposable suits, leading to high costs and mass, the charged suit method has only an initial installation cost, and has barely any added mass. All of these factors were used to develop a decision matrix, which is shown below.

| <b>Bunny Suit vs. Charged Suit</b> |              |             |                    |                                      |                    |              |
|------------------------------------|--------------|-------------|--------------------|--------------------------------------|--------------------|--------------|
| <b>Design</b>                      | <b>Price</b> | <b>Mass</b> | <b>Ease of use</b> | <b>Dust removal prior to reentry</b> | <b>Reusability</b> | <b>Total</b> |
| Bunny Suit                         | 5            | 2           | 2                  | 1                                    | 2                  | 12           |
| Charged Suit                       | 1            | 4           | 4                  | 5                                    | 5                  | 19           |

**Table 41 Bunny Suit and Charged Suit Design Considerations**

Due to this reasoning, the best dust mitigation technique for the suits before the astronaut reenters the habitat is the charged suit design, which is detailed in section IX B.

## D. Inflatable Habitat Material Selection

An inflatable habitat suitable for a space environment consists of six functional layers. The innermost of these layers is the “inner liner”, and acts as a barrier between the crew and the rest of the habitat. The next layer is the gas retention layer, or bladder, which is intended to prevent the atmosphere inside the habitat from escaping. Surrounding the gas retention layer is the structural restraint layer which is meant to withstand the internal pressure force of the air within the habitat and prevent the gas retention layer from bearing pressure. The Habitats team also included a thermal protection layer that provides insulation for the inflatable habitat and is meant to maintain the internal temperature of the habitat within a suitable range. We also considered surrounding the habitat with a micrometeoroid impact protection layer which is intended to shield the interior layers of the inflatable habitat and inhabitants from impacts and punctures from micrometeoroids that impact the lunar surface. The team then considered a radiation shielding layer in order to mitigate the harmful effects of solar radiation. However, due to mass constraints, the radiation and micrometeoroid shielding were excluded from the final design.

The inner liner, which acts as the boundary between the people within the inflatable habitat and the gas retention layer, provides two basic functions: create a suitable living environment, and protect the gas retention layer from internal hazards. For the extended amount of time that the inflatable habitat will be occupied, a material that is a suitable surrounding for living cannot be detrimental to the health of the inhabitants after extended exposure. Any space occupied for a significant amount of time becomes dirty, so the material of the inner liner must also be washable and to microbial growth. The inner liner must also protect the gas retention layer from internal hazards, which means it must be puncture and chemical resistant as well as flame retardant to

prevent any breaches in the gas retention layer. A material that satisfies all of these conditions was chosen for the TransHab concept design, and that material is Nomex paper which consists of aromatic polyamide fibers [1]. The specific type of Nomex selected is Nomex Type 410 paper with a nominal thickness of ten millimeters in order to provide a sufficient buffer to prevent punctures in the bladder.

The gas retention layer, which is responsible for maintaining the internal pressure of the habitat, must prevent the internal atmosphere from escaping the habitat. A good measure of a material's ability to retain pressure is a material's oxygen transmission rate. Any material with an oxygen transmission rate less than a  $15.15 \text{ cc-mm/m}^2\text{-24hr-atm}$  is considered to be a high oxygen barrier. The material used for the gas retention layer does not need to have the tensile strength to bear the internal pressure of the habitat which allows for a wider range of materials to be selected. A good material to be used in the gas retention layer is Mylar M34 polyester film coated with a PVDC copolymer [2]. This polyester has an oxygen transmission rate of  $0.100 \text{ cc-mm/m}^2\text{-24hr-atm}$  and a thickness of 13.0 microns. The low oxygen transmission rate means that the film is a high oxygen barrier, and its low thickness is advantageous for decreasing the volume required. To ensure that pressure in the habitat is maintained, two additional layers of the gas retention layer should be included. These added layers also reduce the oxygen transmission without significantly adding to the mass of the inflatable habitat.

The structural restraint layer is meant to withstand the internal pressure force of the air within the habitat and prevent the gas retention layer from bearing pressure and subsequently failing. To withstand the pressure of the inflated habitat, the material used in the structural restraint layer must have a high tensile strength. To prevent the gas retention layer from bearing

too much pressure, the structural restraint layer should be composed of a webbing that would not allow any section of the gas retention layer to support a force outside of its operable range. There are several materials that satisfy requirements for use in the restraint layer such as Kevlar, Vectran, or Spectra. Properties of these materials taken from [1] can be seen in Table 42.

| Fiber type   | Tensile strength ( $\frac{N}{m^2} \times 10^6$ ) | Low temp. brittleness ( $^{\circ}C$ ) | Resistance to flex cracking | Abrasion resistance |
|--------------|--------------------------------------------------|---------------------------------------|-----------------------------|---------------------|
| Kevlar       | 3.38                                             | < -196                                | Fair                        | Fair                |
| Spectra 1000 | 2.99                                             | < -25                                 | Excellent                   | Excellent           |
| Vectran HS   | 2.84                                             | < -160                                | Good                        | Good                |

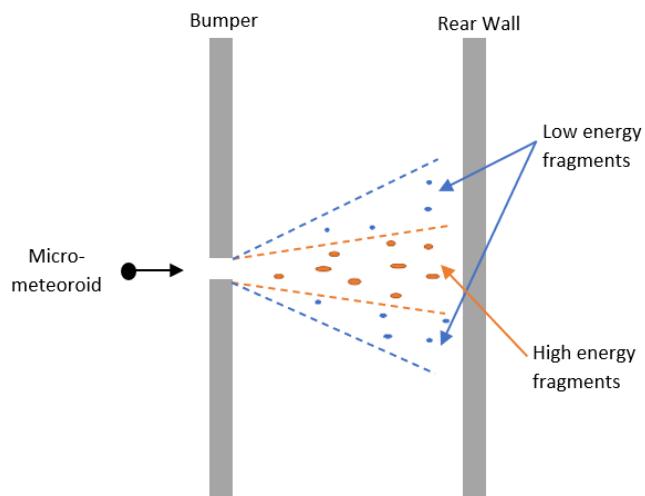
**Table 42 Comparative evaluation of materials for the inflatable habitat restraint**

To mitigate the possibility of the gas restraint layer failing due to the internal pressure of the habitat, a narrow webbing should be used for the restraint layer. A narrow webbing would require a greater number of fiber cables with a smaller size, so the tensile strength of each type of fiber would only affect the diameter of cable. During both the transit and deployment on the lunar surface, the restraint layer may be exposed to temperatures well below one hundred degrees Celsius, so the temperature at which the material becomes brittle is critical. Also, since the inflatable will be tightly packaged during its transit to the Moon, the internal material may be subject to a significant amount of flexing. And, due to the nature of creating a webbing of cables, the cables will overlap and may be subject to abrasion especially while the habitat is being inflated. With all of these factors in mind, the Spectra fiber can be immediately ruled out as an option due to its brittleness at low temperatures. Despite the better tensile strength and operable range of

temperature of Kevlar, the better resistance of Vectran to flex cracking and abrasion make it the favorable choice.

The thermal protection layer is meant to insulate and maintain the internal temperature of the habitat. Currently, the International Space Station uses a multilayer insulation consisting primarily of sheets of metallized film to prevent radiative heat loss. For the inflatable habitat, multiple layers of aluminized Mylar would be sufficient since the majority of the heat loss would most likely be thermal radiation.

The micrometeoroid impact protection layer is responsible for shielding the interior layers of the inflatable habitat. The design of a micrometeoroid shield is similar to a Whipple shield, which consists of two walls with space between them. The outermost wall of a Whipple shield is called the bumper, and the wall must fragment and spread out the micrometeoroid impact. The broken up debris then traverses the empty space between the walls, spreading out more as the distance increases, until impacting the rear wall, as you can see in Fig. 91 which was adapted from [3].



**Fig 91 Diagram of a Whipple shield.**

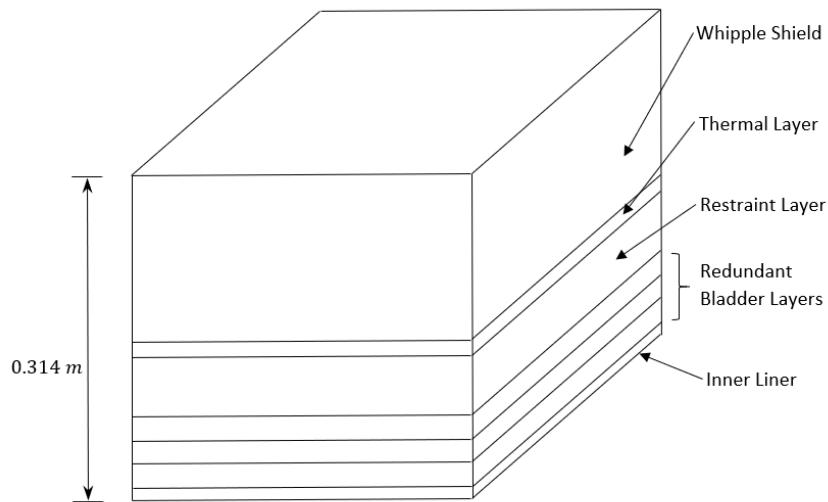
The bumper must be stiff and strong enough to actually break up any projectile, and the rear wall must be very resistant to punctures. However, it is preferable that the walls of the Whipple shield are flexible enough to allow for the inflatable habitat to be tightly packed during transit. Some of the fabrics that have been tested against impacts at extremely high velocities are Kevlar, Zylon fabrics, and Twaron fabrics. Of these materials, Zylon fabric exhibited greater shielding capabilities and lower amount of bumper mass loss. The spacing between these layers of Zylon can be conveniently packed with polyurethane foam since it can be tightly packaged and expanded once the habitat is deployed, thus reducing the volume required for transportation.

With these materials that were chosen, the mass and volume of the inflatable habitat walls can be estimated using the following properties shown in Table 43 that were curated from the materials resources.

| Layer          | Thickness (mm) | Density (kg/m <sup>3</sup> ) | Number of Layers |
|----------------|----------------|------------------------------|------------------|
| Liner          | 1              | 960                          | 1                |
| Bladder        | 0.013          | 940                          | 3                |
| Restraint      | 10             | 1400                         | 1                |
| Thermal        | 5.34           | 1390                         | 1                |
| Micrometeoroid | 23.9           | 1560                         | 2                |

**Table 43 Layer materials and properties**

A depiction of a cutout of the wall of the inflatable habitat can be seen in Fig. 92, including layer dimensions and the number of layers utilized.

**Fig 92 Cutout of a wall section of the inflatable habitat.**

Using the values and information shown in Table 42, and assuming an inflatable module to be ten meters long with an inner radius of three meters, the mass and volume of a habitat module both packed and deployed can be estimated, the results of which are shown in Table 43.

|                 |                    |
|-----------------|--------------------|
| <b>Mass</b>     | <b>29,259 kg</b>   |
| Deployed Volume | 345 m <sup>3</sup> |
| Packed Volume   | 38 m <sup>3</sup>  |

**Table 44 Inflatable habitat mass and volume estimations.**

| Quantity | Item               | Cost \$ (total) | Weight kg (total) |
|----------|--------------------|-----------------|-------------------|
| 3        | Storage Shelves    | 150             | 105               |
| 2        | Storage Shelves    | 90              | 20                |
| 1        | Washer and Dryer   | 1259            | 200               |
| 1        | Floor Lamp         | 20              | 3.6               |
| 1        | Couch              | 1300            | 227               |
| 7        | Pillow             | 49              | 3.85              |
| 3        | Blankets           | 18              | 1.36              |
| 1        | Basket             | 10              | 0.3               |
| 3        | Tall Vase          | 60              | 10.8              |
| 1        | Pampas grass       | 10              | 0.7               |
| 1        | Little Flower      | 10              | 0.5               |
| 7        | Books              | 35              | 3.5               |
| 1        | Living Room Carpet | 150             | 15.5              |
| 1        | Tv Stand           | 80              | 34                |
| 1        | TV                 | 1998            | 45.6              |
| 2        | Fruit Bowls        | 10              | 1.3               |
| 2        | Fake Tree          | 100             | 3.86              |

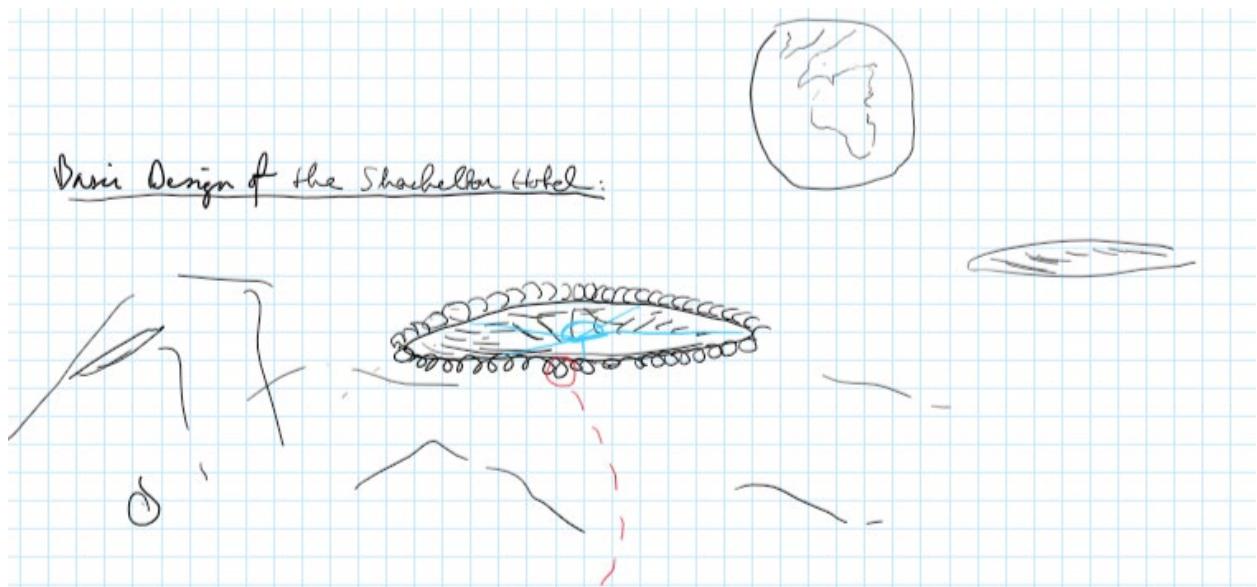
|    |                   |       |      |
|----|-------------------|-------|------|
| 1  | Full body Mirror  | 40    | 10   |
| 1  | Sideboard         | 320   | 18.2 |
| 1  | Dining Carpet     | 120   | 9    |
| 1  | Table             | 600   | 59   |
| 12 | Chair             | 660   | 87.6 |
| 2  | Highchairs        | 170   | 23.5 |
| 1  | Kitchen           | 1115  | 294  |
| 1  | Knife block       | 15    | 3.2  |
| 1  | Microwave         | 55    | 15   |
| 1  | Fridge            | 2700  | 132  |
| 1  | Toilet            | 277   | 36.3 |
| 1  | Side cabinet (WC) | 40    | 8.1  |
| 1  | Sink (WC)         | 409   | 15.8 |
| 1  | Mirror (WC)       | 15    | 3.3  |
| 25 | Plates            | 19.75 | 9    |
| 25 | Bowls             | 19.75 | 5.5  |
| 24 | Silverware        | 30    | 2.7  |
| 12 | Wine Glasses      | 10    | 2    |

|                   |                  |      |                   |
|-------------------|------------------|------|-------------------|
| 20                | Glasses          | 15.8 | 8                 |
| 3/2               | Pots, Pans       | 100  | 10                |
| 1                 | Kitchen Utensils | 5    | 0.25              |
| <b>\$ 12085.3</b> |                  |      | <b>1465.32 kg</b> |

**Table 45 detailed cost and weight overview of Collins Module Interior**

### E. Lunar Hotel -Conceptual Drawings

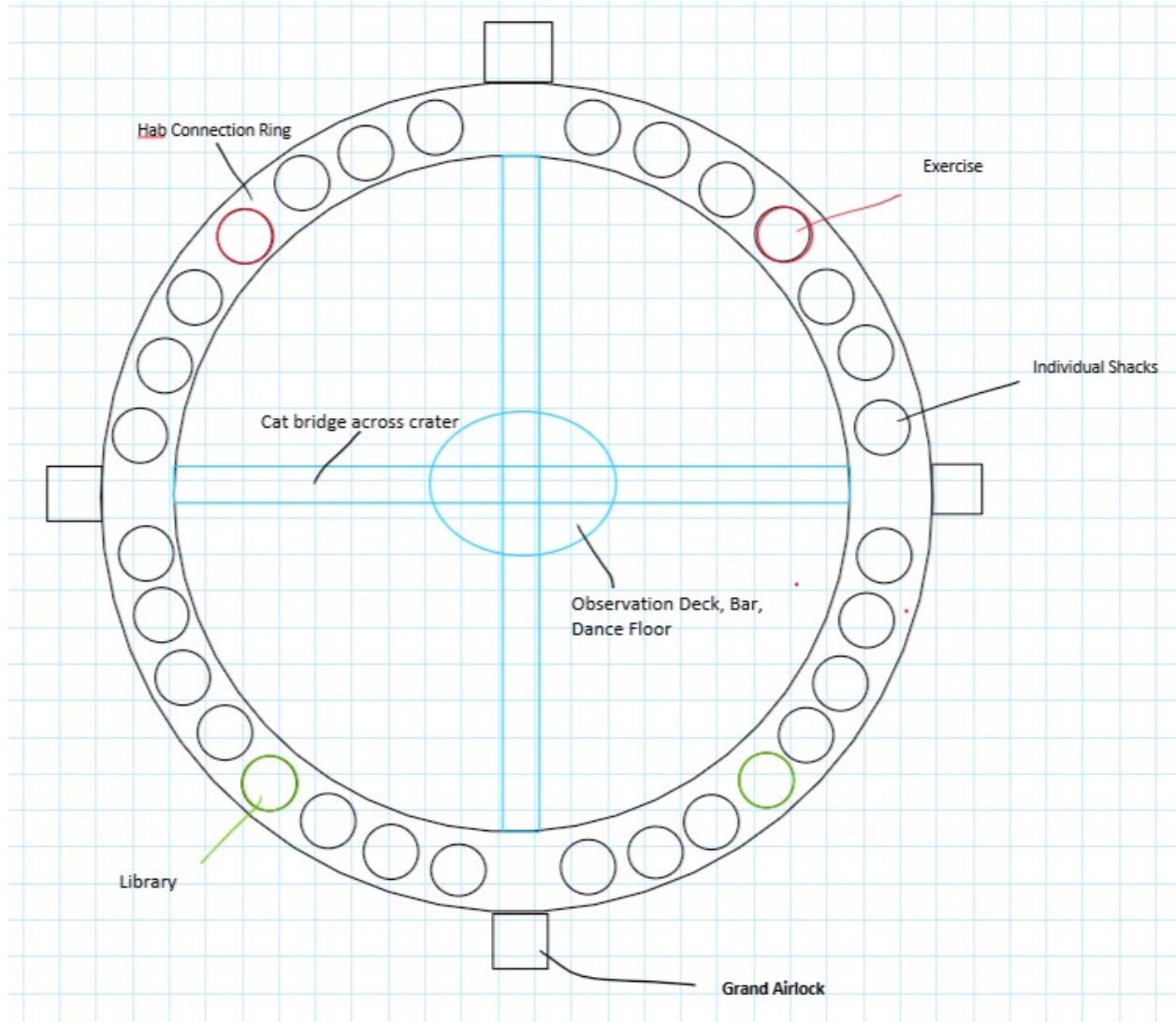
Some initial consideration is given to a potential lunar hotel in this section of the paper. The idea for a lunar hotel stems from the desire to make living on the lunar colony as comfortable as possible. The design presented here is named *Shackleton Shack*. This modern hotel can host up to 70 guests comfortably.



**Fig 92 *Shackleton Shack*.**

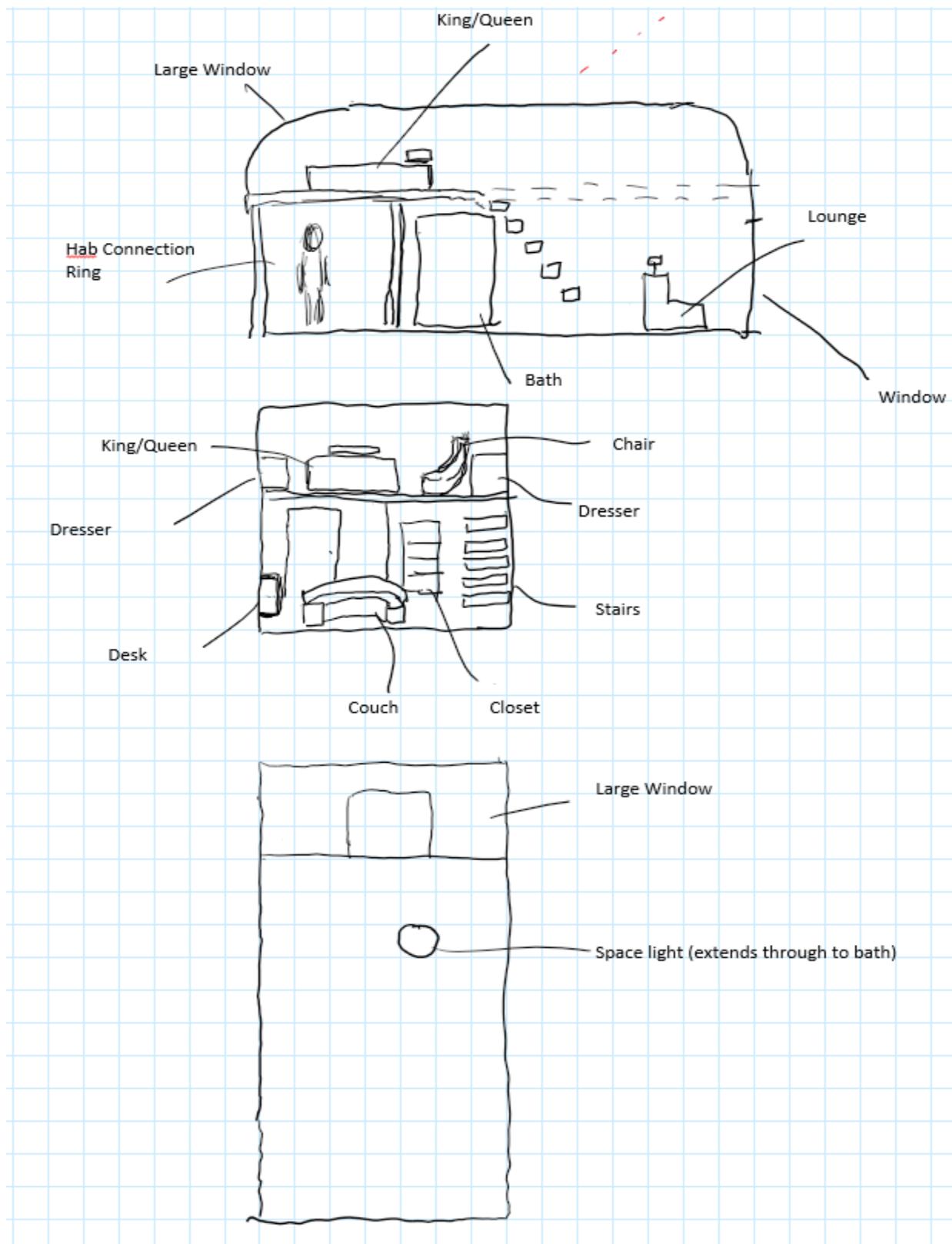
This grand hotel features rooms around a crater on the moon all connected via passageways around the circle. It also features a giant glass dome in the center of the crater for entertainment

and eating which is accessed by similarly clear pathways crossing the crater. There is a bottom tier of rooms as well. They are built into the ground just below the upper ring.



**Fig 93 Blueprint for Upper Ring.**

Each room has a queen or king-sized bed and fold out Murphy beds and mattress couches for extra capacity. The lower ring rooms have a generous window looking out onto the crater. The upper ring rooms feature two windows, one looking into the crater and one looking out of the circle hotel.



**Fig 94 Interior Layout of Upper Ring Room.**

Many activities are offered at *Shackleton Shack*. They include an observation deck with a bespoke bar. The nights are often filled with Karaoke and dancing in the center of the crater. Guests can explore lunar lava tubes with expert guides. They can take advantage of lightweight habitat designs and go high adventure camping. One of the most exciting activities is crater jumping. Making the most of the low gravity environment, guests will spring across these impressive impact craters and land safely on the other side. For those guest who are more mechanically inclined, the *Shackleton Shack* offers All Lunar Vehicle (ALV) trips to really kick up some dust. Of course, not everything is done on a machine, but the exercise facilities are reason alone to visit! For an exclusive event, the hotel offers personal miniature rocket launches for sending small payloads into space such as time capsules or a loved ones ashes. Finally, the hotel offers the guests the chance to leave their footprint behind in the history of the colonization of the moon by helping construct a lunar city and mining for critical ice.

The designs for the hotel are not seriously considered in Project Next Step, however, it is a fun exercise to think of such a venue and what it might look like. Much more research needs to be done to make this a viable investment.

## Systems - Life Support Appendices

Aaron Baum<sup>11</sup>

*Purdue University, West Lafayette, Indiana, 47906, United States*

Blerton Ferati, Hunter Mattingly, Wilson Barce, Riley Harwood, Sanjidah Hossain<sup>12</sup>

*Purdue University, West Lafayette, Indiana, 47906, United States*

---

<sup>11</sup> Life Support Systems Team Lead

<sup>12</sup> Life Support Systems Team Members

## I. Appendix A: Medical Equipment Appendix

### *1. Why a Storage Solution of a Medicine Cabinet is Developed*

We develop a medicine cabinet for the purpose of storing a large amount of medications and medical equipment, an amount that far exceeds what is brought on the International Space Station. The International Space Station has a smaller storage solution with very compact compartments labeled with letters to allow for crew to easily access medications they may not be experts with. We need a larger storage solution since the lunar colony mission will employ significantly more than five crew members that exist on the International Space Station. However, we have a similar need to develop a way to allow for crew to easily access medications that they may not be familiar with which is why we will label the compartments existent in the drawers seen in Fig. 2 and have a guide to access the medications needed.

## II. Appendix B: ATCS Appendix

This section discusses the code and theory behind the thermal analysis completed for the habitat ATCS. Table #1 below shows the inputs and outputs of the analysis.

| Inputs                                                                                                                                                                                                                                                                                  | Outputs                                                                                                                                                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Coolant tubing specs</li> <li>• Coolant fluid specs</li> <li>• IFHX inlet temperature</li> <li>• IFHX heat transfer surface area</li> <li>• Coolant flow rates</li> <li>• Pump efficiency</li> <li>• Radiator surface coating specs</li> </ul> | <ul style="list-style-type: none"> <li>• Required pump power</li> <li>• Required radiator surface area</li> <li>• Maximum heat load capability</li> </ul> |

**Table 1: Inputs and Outputs of the Thermal Analysis**

The calculations for the required pump power and thermal capability were based on an analysis written by members of The Aerospace Corporation and NASA's Jet Propulsion Laboratory (JPL) on pumped fluid loops<sup>[27]</sup>. The calculations for the radiator sizing algorithm were based on MIT lecturer John Keesee's notes on spacecraft thermal control systems<sup>[26]</sup>.

### Pump Power

To find the required pump power, the following equation was used:

$$P_p = \Delta P \frac{\dot{m}}{\rho} \cdot \frac{1}{\eta_p} \quad (1)$$

where  $P_p$  is the required pump power to support the coolant flowing in each loop,  $\Delta P$  is the pressure loss throughout the loop,  $\dot{m}$  is the mass flow rate of the coolant,  $\rho$  is the average density of the fluid, and  $\eta_p$  is the pump efficiency. The mass flow rate, average coolant density, and pump efficiency are all inputs to the analysis, meaning the only thing missing in this equation is  $\Delta P$ . A

few steps are needed to find this value. First the Reynolds number of the fluid is needed and found using

$$Re = \frac{\rho V D}{\mu} \quad (2)$$

where  $Re$  is the Reynolds number,  $V$  is the average fluid velocity,  $D$  is the coolant pipe diameter, and  $\mu$  is the dynamic viscosity of the fluid. It is important to note that a Reynolds number less than 2000 is considered laminar,  $2000 < Re < 4000$  is considered transitioning flow, and  $Re > 4000$  is considered turbulent. The average velocity of the fluid is found by rewriting the mass flow rate

equation:  $V = \frac{\dot{m}}{\rho A}$ .

The majority of pressure change within the loop is a result of friction, changes in elevation, and changes in flow velocity. Assuming the flow velocity is  $\ll 100 \frac{m}{s}$ , the flow is assumed incompressible and the Bernoulli equation can be used, which accounts for the pressure change due to changes in elevation and velocity. In this analysis these values are assumed to be negligible since the flow velocity is very low and the potential energy change due to the moon's gravity is also very low. However, that does not mean the pressure change across the loop is 0. In real flow, one must account for pressure loss due to friction as well. This is known as head loss and is broken down into two terms, minor head loss and major head loss. Major head loss results from friction in fully developed, constant-area portions of the fluid loop. For the sake of simplicity, the cross-sectional area of the ATCS tubing is considered constant. The minor head loss results from frictional effects in bends, elbows, and splits in the tubing. Only the minor head loss from the tube splitting within the radiators and cold plates is considered for this analysis. The equation for major head loss is given by

$$h_{lM} = f \frac{L}{D} \cdot \frac{V^2}{2} \quad (3)$$

where  $f$  is the friction factor for the flow through the pipe and  $L$  is the total length of the pipe. Unfortunately, the value for  $f$  is difficult to find for a real system, but the following equations are used to approximate the friction factor based on the Reynolds number.

$$\text{For } Re < 4000: \quad f = \frac{64}{Re} \quad (4)$$

$$\text{For } 4000 < Re < 20,000: \quad f = \frac{0.079}{Re^{0.25}} \quad (5)$$

$$\text{For } 20,000 < Re < 300,000: \quad f = \frac{0.184}{Re^{0.2}} \quad (6)$$

In the case of the habitat ATCS, the Reynolds number is turbulent but still relatively low due to the small diameter and low flow velocity. Using Equations 2 through 6, the cooling tube length and diameter, and the fluid velocity found earlier, the major head loss can be found.

The minor head loss is found using Equation 7,

$$h_{l_m} = nf \frac{L_e}{D} \cdot \frac{V^2}{2} \quad (7)$$

where  $n$  is the number of bends, elbows, or splits and  $L_e$  is the equivalent length which is found experimentally. In this case, an equivalent length of 30 is used to account for the 90° bends of the fluid loop in the radiators and cold plates. This number was chosen based on previous experimental data. The value of  $n$  is assumed to be 100 to represent these bends.

Finally, the major and minor head losses are combined to find the total head loss using Equation 8 below.

$$h_{l_t} = h_{l_M} + h_{l_m} \quad (8)$$

where  $h_{l_t}$  is the total head loss. Using this, the total pressure loss is found using Equation 9.

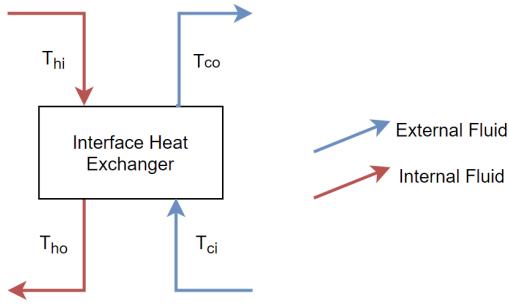
$$\Delta P = P_1 - P_2 = -\rho(h_{l_t}) \quad (9)$$

This change in pressure is then plugged back into Equation 1 to find the required power supplied to the loop's pump. It is important to keep in mind that both the internal and external loops are pumped separately, thus calculations are run to find the required power for each.

### **Thermal Capability**

To find the maximum heat load carried by the designed coolant loops, the temperatures before and after the interface heat exchanger for both internal and external loops are needed. The external inlet temperature is an input to the analysis and is set to some low operating temperature of the external fluid. Once the external fluid passes through the IFHX, it will accept the heat load from the internal loop, thus raising the temperature of the fluid. However, at a high enough heat load the external fluid will rise above its upper operating temperature limit, causing it to boil. Therefore, the heat load must remain below this value.

The internal fluid constrains this heat load even more. The internal fluid inlet and exit temperature must fall within the coolant's operating temperature, to prevent freezing, slush, and boiling. In the case of the habitat ATCS, the internal inlet temperature of the coolant was the largest constraint. At this location, right before entering the IFHX, the internal coolant is at its maximum temperature because it has accepted the heat load from the CHX and cold plates. Therefore, different values of  $Q$  are tested to find the maximum heat load accepted before the internal coolant inlet temperature exceeds its upper operating limit (90°C for water). Figure #2 shows these described reference locations.



**Figure #1: The internal and external fluid exchange heat loads within the IFHX.**

Below are the equations for the temperatures at each of these locations.

$$T_{hi} = T_{ci} + \frac{Q}{\varepsilon C_1} \quad (10)$$

$$T_{ho} = T_{ci} + Q \left( \frac{1}{\varepsilon C_1} - \frac{1}{C_h} \right) \quad (11)$$

$$T_{co} = T_{ci} + \frac{Q}{C_c} \quad (12)$$

Where  $T_{hi}$  is the internal coolant inlet temperature,  $T_{ho}$  is the internal coolant exit temperature,  $T_{co}$  is the external coolant exit temperature,  $T_{ci}$  is the external coolant inlet temperature (input),  $Q$  is the heat load accepted from the cold plates and CHX (input),  $\varepsilon$  is the heat exchanger effectiveness, and  $C_c$  and  $C_h$  are the heat capacitance for the internal and external fluid, respectively. Heat capacitance is the product of the mass flow rate and specific heat of the fluid ( $\dot{m}c_p$ ).  $C_1$  is the smaller of the two values of  $C_c$  and  $C_h$ . Using the given inputs, all of the preceding variables are known except for the heat exchanger effectiveness. Before this value is found however, one must first understand forced convection within coolant tubes using the governing equation below.

$$q_x = h_x(T_w - T_b) \quad (13)$$

where  $q_x$  is axial heat load,  $h_x$  is the axial heat transfer coefficient, and  $T_w$  and  $T_b$  are the tube wall temperature and fluid-bulk temperature, respectively. Since the heat transfer coefficient is only

found experimentally, it is often related to the dimensionless Nusselt number for convenience using the following equation.

$$Nu = \frac{hD}{k} \quad (14)$$

where  $Nu$  is the Nusselt number,  $D$  is the diameter of the coolant tube, and  $k$  is the thermal conductivity of the fluid. Equations 15 through 17 below give the Nusselt number for different conditions in a circular cross-section tube.

For laminar flow with uniform heat flux at the tube wall:  $Nu = 4.364$  (15)

For laminar flow with constant tube-wall temperature:  $Nu = 3.660$  (16)

For turbulent flow:  $Nu = \left(\frac{f}{8}\right) Pr^{1/3}$  (17)

where  $Re$  is the Reynolds number (see Equation 2),  $f$  is the friction factor (see Equations 4 through 6), and  $Pr$  is the Prandtl number given as the ratio of kinematic viscosity and thermal diffusivity ( $\frac{\nu}{\alpha}$ ) and assumed constant throughout the loop. Now working backwards from Equation 14, one can solve for the heat transfer coefficient,  $h$ . Keep in mind the internal and external fluid have different heat transfer coefficients, therefore they must be solved separately.

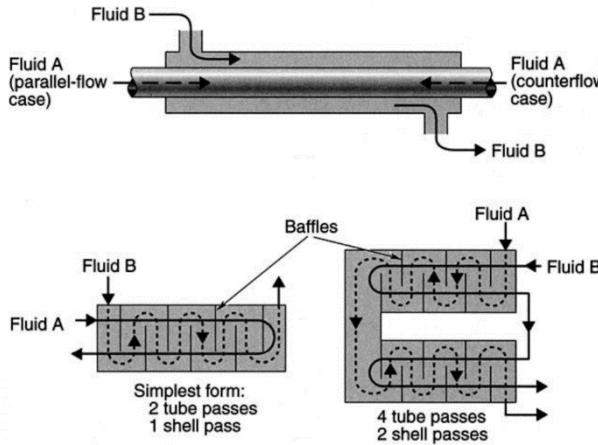
Once internal and external heat transfer coefficients are found, the overall heat transfer coefficient,  $U$ , is found using the following equation,

$$U = \left( \frac{1}{h_0} + \frac{t}{k} + \frac{1}{h_i} \right)^{-1} \quad (18)$$

where  $t$  is the thickness of the tube,  $k$  is the thermal conductivity of the tube material, and  $h_i$  and  $h_0$  are the internal and external fluid heat transfer coefficients, respectively.

Next, the type of heat exchanger is chosen. The three most common types are parallel-flow, counterflow, and shell-and-tube exchangers seen in the figure below. In the case of the habitat

ATCS design, only the counterflow and shell-and-tube (1 shell pass, 2 tube passes) exchangers were considered. Figure #3 below shows the most common types of heat exchangers.



**Figure #2:** The three most common types of heat exchangers are parallel-flow, counter-flow, and shell-and-tube [27].

The effectiveness of each considered heat exchanger is given below.

$$\text{Counterflow: } \varepsilon = \frac{1 - \exp[-NTU(1-C)]}{1 - C \exp[-NTU(1-C)]} \quad (19)$$

$$\text{Shell-and-tube: } \varepsilon = 2 \left[ 1 + C + \frac{1 + \exp[-NTU(1+C^2)^{1/2}]}{1 - \exp[-NTU(1+C^2)^{1/2}]} (1 + C^2)^{1/2} \right]^{-1} \quad (20)$$

where  $C$  is the ratio of the minimum and maximum values for heat capacitance for the internal and external fluid ( $\frac{C_{min}}{C_{max}}$ ).  $NTU$  is considered a heat exchanger size-factor and is given by the following equation:

$$NTU = \frac{UA_{hx}}{C_{min}} \quad (21)$$

where  $A_{hx}$  is not the cross-sectional area of the tube, but the outer area of the tubing within the heat exchanger (input). Given the above equations the effectiveness of the heat exchanger can now

be calculated. Furthermore, the temperature at each of the reference equations in Figure #1 can be found using Equations 10 through 12 by plugging in values of  $Q$  to ensure the temperatures at the inlet and outlet of the IFHX never exceed the internal or external coolant's operating temperature boundaries.

## Radiator Sizing

To perform this analysis one first needs to examine the simple conservation of energy equation for a radiator system, given below.

$$\frac{\partial E}{\partial t} = q_{in} - q_{out} + q_{dis} \quad (22)$$

where  $q_{in}$  is the heat absorbed by the radiators due to thermal energy from the Sun,  $q_{out}$  is the emitted heat from the radiators,  $q_{dis}$  is the heat dissipated from the external coolant loop, and  $\frac{\partial E}{\partial t}$  is the change in this energy balance over time, which is assumed negligible for this analysis. These heat loads are calculated using Equations 23 through 25.

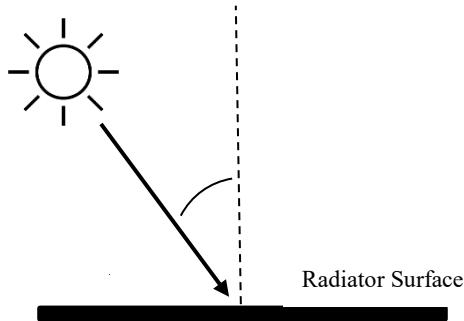
$$q_{in} = G_s A_R \alpha_R \cos(\theta_i) \quad (23)$$

$$q_{out} = \varepsilon_R \sigma T_R^4 A_R \quad (24)$$

$$q_{dis} = Q \quad (25)$$

where  $G_s$  is the solar flux (roughly  $1418 \frac{W}{m^2}$ ),  $A_R$  is the radiator area,  $\alpha_R$  is the radiator surface absorptivity,  $\varepsilon_R$  is the radiator surface emissivity,  $\theta_i$  is the solar incidence angle,  $\sigma$  is the Stefan-Bolzmann constant ( $5.67 \cdot 10^{-8} \frac{W}{m^2 K^4}$ ),  $T_R$  is the average temperature of the radiator surface, and  $Q$  is the heat load carried by the external coolant loop. This is the same heat load accepted from the internal coolant loop that needs to be rejected into space. The maximum  $Q$ , which was calculated in the previous section, is used such that the radiator surface area is large enough to

dissipate the maximum heat load carried by the external loop. The radiator surface absorptivity and emissivity relate to how well the surface absorbs and emits heat, respectively. To dissipate more heat, a surface with high emissivity and low absorptivity is desired. In the ATCS, the outside surfaces of the radiators are covered with white paint to achieve this. To reduce the amount of heat absorbed from the sun, a high solar incidence angle is required. To accomplish this in the ATCS, the radiators are placed on top of the habitat, nearly parallel to the Sun's rays. Below, Figure #3 shows the solar incidence reference angle.



**Figure #3: Solar Incidence**

Reworking Equation 22 gives,

$$A_R = \frac{-(Q)}{G_s \cos \cos(\theta_i) \alpha_R - \sigma \varepsilon_R T_R^4} \quad (26)$$

Assuming  $T_R = \max(T_{co})$  found in the previous section, the rest of the values are known besides the solar incidence angle. Here different value of  $\theta_i$  are used to determine the minimum and maximum radiator surface area required to dissipate the external loop's maximum heat load into space.

The following is a list of assumptions needed to complete the analyses for pump power, thermal capability, and radiator sizing:

|             |
|-------------|
| Assumptions |
|-------------|

- Steady state flow – flow rate is not a function of time
- Steady state energy balance– radiator internal energy is not a function of time
- Incompressible flow – density change due to pressure and temperature change is small
- Negligible potential energy change
- IFHX tubing subject to uniform wall heat flux in the flow direction while the wall temperature remains uniform around the periphery
- Constant tube diameter throughout each loop
- Constant specific heat
- Constant fluid density
- Constant fluid thermal conductivity
- Constant fluid viscosity

**Table #2: Assumptions made while designing ATCS.****Code**

```

%% Inputs
% Tubing
diam = .0125; % Tube diameter in m
area = pi*(diam/2)^2; % Tube cross section area in m^2
thick = .00125; % Tube thickness in m
tub_k = 385; % Copper tubing thermal conductivity in W/m*K
rr = .0015/1000; % Copper roughness coefficient
tub_rho = 8900; % Copper tubing density in kg/m^3

len_hxhx = 5; % Tube length from IFHX to CHX in m
len_hxcp = 10; % Tube length from CHX to cold plates in m
len_cppa = 10; % Tube length from cold plates to pump assembly in m
len_pahx = 2; % Tube length from pump assembly to IFHX in m
len_hxrd = 2; % Tube length from IFHX to radiators in m
len_rdpa = 2; % Tube length from radiators to pump assembly in m

% Cold Plate
len_cp = 4; % Tube length per cold plate in m

```

```

area_cp = .5; % Average cold plate area in m^2
thick_cp = .01; % Average cold plate thickness in m
rho_cp = 2700; % Density of aluminum cold plates in kg/m^3
k=4; % Number of cold plates

% Radiators
len_rad = 8; % Tube length per radiator in m
rad_rho = 50; % Aluminum radiator density in kg/m^3
rad_em = .94; % Radiator white paint (Z-93) coating
emissivity
rad_al = .15; % Radiator white paint (Z-93) coating
absorptivity
Gs = 1418; % Solar flux in W/m^2
theta = linspace(0,180,360); % Angle of incidence
SBC = 5.67e-8; % Stefan-Bolzmann's Constant in W/(m^2-K^4)

% IFHX
Te_in = 273+15; % IFHX Inlet temperature in K
len_ifhx = .5*17; % IFHX internal tube length in m
IFHX_a = (len_ifhx)*(2*pi*((diam/2)+thick)); % IFHX heat transfer
surface area in m^2

% CHX
len_chx = 2; % CHX internal tube length in m

% Fluid Specifications
% Water
cp_i = 4179; % Specific heat of water at 25C in
J/(kgK)
bp_i = 373; % Boiling point of water in K
fp_i = 273; % Freezing point of water in K
rho_i = 997; % Density of water at 25C in kg/m^3
dvis_i = .847/1000; % Dynamic viscosity of water at 25C in
Pa*s
kvis_i = dvis_i/rho_i; % Kinematic viscosity of water at 25C
in m^2/s
k_i = .598; % Thermal conductivity of water at 25C
in W/(m*K)
alp_i = k_i/(rho_i*cp_i); % Thermal diffusivity of water at 25C

% ammonia
cp_e = 4740; % Specific heat of ammonia at 25C in
J/(kgK)
bp_e = -33.3+273; % Boiling point of ammonia in K
fp_e = -77.7+273; % Freezing point of ammonia in K
rho_e = 609; % Density of ammonia at 25C in kg/m^3
dvis_e = .138/1000; % Dynamic viscosity of ammonia at 25C
in Pa*s
kvis_e = dvis_e/rho_e; % Kinematic viscosity of ammonia at 25C
in m^2/s

```

```

k_e = .521; % Thermal conductivity of ammonia at
25C in W/(m*K)
alp_e = k_e/(rho_e*cp_e); % Thermal diffusivity of ammonia at
25C

% Pumps
m_doti = .125; % Mass flow rate of internal fluid in kg/s
m_dote = .15; % Mass flow rate of external fluid in kg/s
np = .75; % Pump efficiency

% Totals
len_i =
len_hhx+len_hxcp+len_cppa+len_pahx+len_chx+len_ifhx+(k*len_cp);
len_e = len_hxrd+len_rdpa+len_pahx+(len_rad*4);
Q = linspace(0,20000);

%% Analysis
vel_i = m_doti/(rho_i*area); % Average internal flow
velocity in m/s
Re_i = (rho_i*vel_i*diam)/dvis_i; % Reynolds number of internal
fluid, <~3000 for laminar flow
Pr_i = kvvis_i/alp_i; % Prandtl Number of internal
fluid

vel_e = m_dote/(rho_e*area); % Average external flow
velocity in m/s
Re_e = (rho_e*vel_e*diam)/dvis_i; % Reynolds number of external
fluid, <~3000 for laminar flow
Pr_e = kvvis_e/alp_e; % Prandtl Number of external
fluid

if Re_i <= 4000
    f_i = 64/Re_i; % Friction factor
    Nu_i = 4.364; % Nusselt number for uniform heat flux at tube
wall for circular CS
elseif (4000 < Re_i) && (Re_i <= 20000)
    f_i = .079/(Re_i^.25);
    Nu_i = Re_i*(f_i/8)* Pr_i^(1/3);
else
    f_i = .184/(Re_i^.2);
    Nu_i = Re_i*(f_i/8)* Pr_i^(1/3);
end

if Re_e <= 4000
    f_e = 64/Re_e; % Friction factor
    Nu_e = 4.364; % Nusselt number for uniform heat flux at tube
wall for circular CS
elseif (4000 < Re_e) && (Re_e <= 20000)
    f_e = .079/(Re_e^.25);
    Nu_e = Re_e*(f_e/8)* Pr_e^(1/3);

```

```

else
    f_e = .184/(Re_e^.2);
    Nu_e = Re_e*(f_e/8)* Pr_e^(1/3);
end

% Pump Power
hl_maji = f_i*(len_i/diam)*(5*vel_i.^2); % Major head loss in
m^2/s^2
hl_mini = 100*f_i*30*(5*vel_i.^2); % Minor head loss in
m^2/s^2
hl_ti = hl_maji + hl_mini; % Total head loss in
m^2/s^2
hl_maje = f_e*(len_e/diam)*(5*vel_e.^2); % Major external head
loss in m^2/s^2
hl_mine = 100*f_e*30*(5*vel_e.^2); % Minor external head loss
in m^2/s^2
hl_te = hl_maje + hl_mine; % Total external head loss
in m^2/s^2

del_Pi = rho_i*hl_ti; % Internal pressure drop in Pa
del_Pe = rho_e*hl_te; % External pressure drop in Pa
pow_pi = (del_Pi*m_doti)/(rho_i*np); % Internal pump power in W
pow_pe = (del_Pe*m_dote)/(rho_e*np); % External pump power in W

% Heater Power
pow_sh = 45*2*8; % Total shell heater power in W assuming 8
modules
pow_rh = 25*7*8; % Total radiator heater power in W assuming 7
panels and 8 modules
pow_fh = 5*4*8; % Total fluid line heater power in W assuming 8
modules

% Heat Load
htc_i = (Nu_i*k_i)/diam; % Heat transfer constant for internal
fluid
htc_e = (Nu_e*k_e)/diam; % Heat transfer constant for external
fluid
U = 1/((1/htc_e)+(thick/tub_k)+(1/htc_i)); % Overall heat
transfer coefficient in IFHX, needs refining

C_i = m_doti*cp_i; % Internal fluid heat capacitance
C_e = m_dote*cp_e; % External fluid heat capacitance
Cmax = max(C_i,C_e); % Max C
Cmin = min(C_i,C_e); % Min C
C = Cmin/Cmax; % Ratio of min to max C
NTU = (U*IFHX_a)/(Cmin); % Heat exchanger size factor
eff = 2*(1+C*((1+exp(-NTU*(1+C^2)^(1/2)))/(1-exp(
-NTU*(1+C^2)^(1/2))))*(1+C^2)^(1/2))^(1/2); % Effectiveness for
shell-and-tube (one shell pass; 1 tube passes) heat exchanger

```

```
%eff = (1-exp(-NTU*(1-C)))/(1-C*exp(-NTU*(1-C))); % Effectiveness
of counterflow heat exchanger

Te_out = Te_int+(Q/C_e); % Temperature of the external
fluid leaving the IFHX
Ti_in = Te_int+(Q/(eff*Cmin)); % Temperature of the internal
fluid entering the IFHX
Ti_out = Te_int+(Q/(eff*Cmin))-(Q/C_i); % Temperature of the
internal fluid leaving the IFHX

%% External Loop Analysis
Temax = 300; % Temperature of the outer loop at max heat load in
K
Qmax = 16000; % Max heat load in W
rad_area = -Qmax./(Gs*rad_al*cosd(theta)-SBc*rad_em*Temax^4); % Radiator area needed to support max heat load with varying
incidence angles

%% Volume, Mass, and Cost Calculations
% Radiators
rad_num = max(rad_area)/(17.44); % Number of radiators per
module based on ISS radiator sizing, not important
rad_v = max(rad_area)*(1.7/100); % Total radiator panel volume
in m^3
rad_m = rad_v*rad_rho; % Total radiator mass in kg

% Piping
tub_vi = len_i*((pi*((diam/2)+thick)^2)-(pi*(diam/2)^2)); % Internal tubing total volume in m^3
tub_ve = len_e*((pi*((diam/2)+thick)^2)-(pi*(diam/2)^2)); % External tubing total volume in m^3
tub_m = (tub_vi+tub_ve)*tub_rho; % Total tubing mass in kg

% Fluid
tub_vie = len_i*(pi*(diam/2)^2); % Empty internal tubing total
volume in m^3
tub_vee = len_e*(pi*(diam/2)^2); % Empty external tubing total
volume in m^3
ifl_m = tub_vie*rho_i; % Total internal fluid mass
in kg
efl_m = tub_vee*rho_e; % Total external fluid mass
in kg
fl_m = ifl_m+efl_m; % Total internal fluid mass
in kg

% Cold Plates
vol_cp = area_cp*thick_cp; % Volume of individual cold plate in
m^3
vol_cp = vol_cp*k; % Total volume of cold plates in m^3
cp_m = vol_cp*rho_cp; % Total mass of cold plates in kg
```

```
% Heat Exchangers
ifhx_m = 5; % IFHX mass estimation in kg
chx_m = 4; % CHX mass estimation in kg
ifhx_v = .5*.5*1; % IFHX volume estimation in m^3
chx_v = .5*.5*.5; % CHX volume estimation in m^3

% Pump Assembly
pai_m = 3; % Internal pump assembly mass estimation in kg
pae_m = 3; % External pump assembly mass estimation in kg
pa_m = pae_m+pai_m; % Total pump assembly mass estimation in kg

% Total Mass
atcs_m = 2*(tub_m+f1_m+ifhx_m+chx_m+pa_m)+cp_m+rad_m; % Total mass in kg including redundant system per module
atcs_total_m = 8*atcs_m % Total mass of ATCS in kg assuming 8 modules

% Total Volume
atcs_v = 2*(tub_ve+tub_vie+tub_veet+ifhx_v+chx_v)+vol_cp+rad_v; % Total volume in m^3 including redundant system per module
atcs_total_v = 8*atcs_v % Total volume of ATCS in m^3 assuming 8 modules

% Total Power
pow_sc = (pow_pi*2*8)+(pow_pe*2*8)+pow_fh % Safety critical power of ATCS in W assuming 8 modules
pow_con = pow_sh+pow_rh % Contingency power of ATCS in W assuming 8 modules

% Total Cost
atcs_launchcost = atcs_total_m*8750; % Launch cost assuming the use of Starship and 8 modules
atcs_prodcost = 2*(atcs_total_m*8750); % Approximated production cost of assuming 8 modules
atcs_cost = atcs_launchcost+atcs_prodcost % Total cost of ATCS in $

%% Plots
figure(1)
plot(Q,Ti_in,'k-','LineWidth', 2)
hold on
plot(Q,Ti_in,'b-','LineWidth', 2)
yline(bp_i,'--','Water Boiling Point');
yline(bp_i-10,'--','Max Water Op Temp');
title('Comparing Heat Load and Internal Fluid Temp')
xlabel('Heat Load [W]')
ylabel('Internal Inlet Temp [K]')
grid on
legend('Nominal Loop Rejection','Faulted Loop Rejection')
```

```
figure(2)
plot(theta, rad_area,'LineWidth',2)
title('Radiator Sizing')
xlabel('Solar Incidence Angle [deg]')
ylabel('Radiator Area Needed for Max Heat Load [m^2]')
grid on
```

### III. Appendix C: ECLSS Layout Simulation Code

The Matlab script shown below is used to run the failure propagation simulation to test the robustness of our proposed ECLSS layouts and to determine the final orientation of the ECLSS. The code uses failure occurrence data collected from the ISS ECLSS subsystem taken from 2002 to 2018 to estimate the probability of a system failure event over time for our subsystems. A failure event in the simulation occurs when both a subsystem and its neighboring redundant system both fail. Redundancy built into particular subsystems was also taken into account by constituting subsystem failure when two separate failures had occurred simultaneously within that component. The assumptions made during the simulations was that failure modes on the habitat are similar to failure modes on the ISS, the OGS subsystem is completely replaced around year 7 or 8, the Bosch reactor in our ECLSS has a similar failure occurrence rate as the Sabatier reactor on the ISS and, finally, TCC failures were fixed fast enough to avoid any failure propagation. The code returned the number of failure events per subsystems over the course of the 500 weeks and exactly which week they would occur.

```
t = 1;
ECLSS_Failure = 0; %initializing ECLSS failure count
TCC_failure = 0; %initializing TCCS failure count
ATCS_failure = 0; %initializing ATCS failure count
CDRA_failure = 0; %initializing CDRA failure count
OGS_failure = 0; %initializing OGS failure count
WPA_failure = 0; %initializing WRS failure count
PCS_failure = 0; %initializing PCS failure count
WMS_failure = 0; %initializing WMS failure count

for t = 1:1:500 %simulation over 500 weeks
    TCC_Farm = TCCfail(t); TCC_Food = TCCfail(t); TCC_Crew1 = TCCfail(t);
    TCC_Crew2 = TCCfail(t); TCC_Lab = TCCfail(t); %Number of TCCS failures in a week

    ATCS_Farm = ATCSfail(t)+ATCSfail(t); ATCS_Food = ATCSfail(t)+ATCSfail(t);
    ATCS_Crew1 = ATCSfail(t)+ATCSfail(t); ATCS_Crew2 = ATCSfail(t)+ATCSfail(t);
    ATCS_Crew3 = ATCSfail(t)+ATCSfail(t); ATCS_Crew4 = ATCSfail(t)+ATCSfail(t);
```

```

ATCSfail(t)+ATCSfail(t); ATCS_Crew4 = ATCSfail(t)+ATCSfail(t);
ATCS_Lab = ATCSfail(t)+ATCSfail(t); ATCS_Fun =
ATCSfail(t)+ATCSfail(t); %Number of ATCS failures in a week

CDRA_Farm = CDRAfail(t); CDRA_Food = CDRAfail(t); CDRA_Crew1 =
CDRAfail(t); CDRA_Crew2 = CDRAfail(t); CDRA_Crew3 =
CDRAfail(t); CDRA_Crew4 = CDRAfail(t); CDRA_Fun =
CDRAfail(t); CDRA_Lab = CDRAfail(t); %Number of CDRA failures in a
week

Bosch_Farm = Boschfail(t)+Boschfail(t); Bosch_Food =
Boschfail(t)+Boschfail(t); Bosch_Crew1 =
Boschfail(t)+Boschfail(t); Bosch_Crew2 =
Boschfail(t)+Boschfail(t); Bosch_Crew3 =
Boschfail(t)+Boschfail(t); Bosch_Crew4 =
Boschfail(t)+Boschfail(t); Bosch_Fun =
Boschfail(t)+Boschfail(t); Bosch_Lab = Boschfail(t)+Boschfail(t);
%Number of Bosch failure in a week

OGS1 = OGSfail(t); OGS2 = OGSfail(t); OGS3 = OGSfail(t); OGS4 =
OGSfail(t); %Number of OGS failures in a week
WPA1 = WPAfail(t); WPA2 = WPAfail(t); WPA3 = WPAfail(t); %Number of
WPA failures in a week
PCS1 = PCSfail(t); PCS2 = PCSfail(t); %Number of PCS failures in a
week
UWPA1 = UWPAfail(t); UWPA2 = UWPAfail(t); %Number of UWPA failures
in a week
UPA1 = UPAfail(t); UPA2 = UPAfail(t); %Number of UPA failures in a
week

%Total number of Failed subsystem components in the Habitat for
week t
CDRA_active =
CDRA_Farm+CDRA_Food+CDRA_Fun+CDRA_Lab+CDRA_Crew1+CDRA_Crew2+CDRA_ =
Crew3+CDRA_Crew4;
Bosch_active =
Bosch_Farm+Bosch_Food+Bosch_Fun+Bosch_Lab+Bosch_Crew1+Bosch_Crew2 +
+Bosch_Crew3+Bosch_Crew4;
OGS_active = OGS1+OGS2+OGS3+OGS4;
ATCS_active =
ATCS_Farm+ATCS_Food+ATCS_Lab+ATCS_Crew1+ATCS_Crew2+ATCS_Crew3+ATC_ =
S_Crew4+ATCS_Fun;
PCS_active = PCS1+PCS2; WPA_active = WPA1+WPA2+WPA3; WMS_active =
UWPA1+UWPA2+UPA1+UPA2;
TCC_active = TCC_Farm+TCC_Food+TCC_Lab+TCC_Crew1+TCC_Crew2;
%Condition for complete ECLSS failure
if CDRA_active == 8 || OGS_active == 4 || ATCS_active == 16 ||
Bosch_active == 16 || TCC_active == 5 || PCS_active == 2 ||
WPA_active == 3 || WMS_active == 4

```

```

        fprintf('ECLSS  failure  at  week  %f\n',t);ECLSS_Failure  =
ECLSS_Failure+1;
    end
%Condition for CDRA failures
    if CDRA_Farm == 1 && CDRA_Food == 1 || Bosch_Farm == 2 &&
Bosch_Food == 2
        fprintf('Left Wing CDRA failure at week %f\n',t);CDRA_failure
= CDRA_failure+1;
    end
    if CDRA_Fun == 1 && CDRA_Lab == 1 || Bosch_Fun == 2 && Bosch_Lab
== 2
        fprintf('Right      Wing      CDRA      failure      at      week
%f\n',t);CDRA_failure = CDRA_failure+1;
    end
    if CDRA_Crew1 == 1 && CDRA_Crew2 == 1 || CDRA_Crew3 == 1 &&
CDRA_Crew4 == 1 || Bosch_Crew1 == 2 && Bosch_Crew2 == 2 || Bosch_Crew3 == 2 && Bosch_Crew4 == 2
        fprintf('Crew  CDRA  failure  at  week  %f\n',t);CDRA_failure =
CDRA_failure+1;
    end
%Condition for OGS failures
    if OGS1 == 1
        fprintf('Left Wing  OGS  failure  at  week  %f\n',t);OGS_failure
= OGS_failure+1;
    end
    if OGS2 == 1 && OGS3 == 1
        fprintf('Central  OGS  failure  at  week  %f\n',t);OGS_failure =
OGS_failure+1;
    end
    if OGS4 == 1
        fprintf('Right Wing  OGS  failure  at  week  %f\n',t);OGS_failure
= OGS_failure+1;
    end
%Condition for ATCS failures
    if ATCS_Farm == 2
        fprintf('Farm  ATCS  failure  at  week  %f\n',t);ATCS_failure =
ATCS_failure+1;
    end
    if ATCS_Lab == 2
        fprintf('Lab  ATCS  failure  at  week  %f\n',t);ATCS_failure =
ATCS_failure+1;
    end
    if ATCS_Crew1 == 2 && ATCS_Crew2 == 2 && ATCS_Food == 2 &&
ATCS_Crew3 == 2 && ATCS_Crew4 == 2 && ATCS_Fun == 2
        fprintf('Crew  ATCS  failure  at  week  %f\n',t);ATCS_failure =
ATCS_failure+1;
    end
%Condition for PCS failures
    if PCS1 == 1

```

```

        fprintf('Left  PCS  failure  at  week  %f\n',t);PCS_failure =
PCS_failure+1;
    end
    if PCS2 == 1
        fprintf('Right  PCS  failure  at  week  %f\n',t);PCS_failure =
PCS_failure+1;
    end
%Condition for WRS failures
    if WPA1 == 1
        fprintf('Left  WPA  failure  at  week  %f\n',t);WPA_failure =
WPA_failure+1;
    end
    if WPA2 == 1
        fprintf('Central WPA  failure  at  week  %f\n',t);WPA_failure =
WPA_failure+1;
    end
    if WPA3 == 1
        fprintf('Right  WPA  failure  at  week  %f\n',t);WPA_failure =
WPA_failure+1;
    end
%Condition for WMS failures
    if UPA1 == 1 && UWPA1 == 1
        fprintf('Left  WMS  failure  at  week  %f\n',t);WMS_failure =
WMS_failure + 1;
    end
    if UPA2 == 1 && UWPA2 == 1
        fprintf('Right  WMS  failure  at  week  %f\n',t);WMS_failure =
WMS_failure + 1;
    end
%Condition for TCC failures
    if TCC_Farm == 1 && TCC_Food == 1
        fprintf('Left Wing  TCC  failure  at  week  %f\n',t);TCC_failure =
TCC_failure+1;
    end
    if ATCS_Lab == 1
        fprintf('Right Wing  TCC  failure  at  week  %f\n',t);TCC_failure =
TCC_failure+1;
    end
    if TCC_Crew1 == 1 && TCC_Crew2 == 1
        fprintf('Crew  TCC  failure  at  week  %f\n',t);TCC_failure =
TCC_failure+1;
    end
end
%ATCS failure probability function over 500 weeks
function select = ATCSfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < (3/365)
        select = 1;
    else

```

```
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
if x < (5/365)
    select = 1;
else
    select = 0;
end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
if x < (5/365)
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
if x < (3/365)
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
if x < (3/365)
    select = 1;
else
    select = 0;
end
else
    select = 0;
end
%CDRA failure probability function over 500 weeks
function select = CDRAfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < ((91/200)*(5/365))
        select = 1;
    else
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
    if x < ((91/200)*(13/365))
        select = 1;
    else
        select = 0;
    end
```

```
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
if x < ((91/200)*(25/365))
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156) && t <= (209+156+156+209)%failure
probability between week 521 and 730
if x < ((91/200)*(12/365))
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
if x < ((91/200)*(10/365))
    select = 1;
else
    select = 0;
end
else
    select = 0;
end
%Bosch failure probability function over 500 week
function select = Boschfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < ((4/200)*(5/365))
        select = 1;
    else
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
    if x < ((4/200)*(13/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
    if x < ((4/200)*(25/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
    if x < ((4/200)*(12/365))
```

```
        select = 1;
else
    select = 0;
end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
if x < ((4/200)*(10/365))
    select = 1;
else
    select = 0;
end
else
    select = 0;
end
%PCS failure probability function over 500 weeks
function select = PCSfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < ((9/200)*(5/365))
        select = 1;
    else
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
    if x < ((9/200)*(13/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
    if x < ((9/200)*(25/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
    if x < ((9/200)*(12/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
    if x < ((9/200)*(10/365))
        select = 1;
    else
        select = 0;
```

```
end
else
    select = 0;
end
%OGS failure probability function over 500 weeks
function select = OGSfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < ((44/200)*(5/365))
        select = 1;
    else
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
    if x < ((44/200)*(13/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
    if x < ((44/200)*(25/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
    if x < ((44/200)*(12/365))
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
    if x < ((44/200)*(10/365))
        select = 1;
    else
        select = 0;
    end
else
    select = 0;
end
%TCC failure probability function over 500 weeks
function select = TCCfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < ((11/200)*(5/365))
        select = 1;
```

```
else
    select = 0;
end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
if x < ((11/200)*(13/365))
    select = 1;
else
    select = 0;
end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
if x < ((11/200)*(25/365))
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
if x < ((11/200)*(12/365))
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
if x < ((11/200)*(10/365))
    select = 1;
else
    select = 0;
end
else
    select = 0;
end
%UPA failure probability function over 500 weeks
function select = UPAfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
if x < 0
    select = 1;
else
    select = 0;
end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
if x < (2/365)
    select = 1;
else
    select = 0;
end
```

```
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
if x < (7/365)
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
if x < (3/365)
    select = 1;
else
    select = 0;
end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
if x < (4/365)
    select = 1;
else
    select = 0;
end
else
    select = 0;
end
%UWPA failure probability function over 500 weeks
function select = UWPAfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < 0
        select = 1;
    else
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
    if x < (2/365)
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
    if x < (8/365)
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
    if x < (4/365)
```

```
        select = 1;
else
    select = 0;
end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
if x < (3/365)
    select = 1;
else
    select = 0;
end
else
    select = 0;
end
%WRS failure probability function over 500 weeks
function select = WPAfail(t)
x = rand;
if t <= 156 %failure probability before 156 weeks
    if x < (2/365)
        select = 1;
    else
        select = 0;
    end
elseif t > 156 && t <= (209+156) %failure probability between week
156 and 365
    if x < (2/365)
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156) && t <= (209+156+156) %failure probability
between week 365 and 521
    if x < (2/365)
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156) && t <= (209+156+156+209) %failure
probability between week 521 and 730
    if x < (3/365)
        select = 1;
    else
        select = 0;
    end
elseif t > (209+156+156+209) && t <= (209+156+156+209+209) %failure
probability between week 730 and 939
    if x < (3/365)
        select = 1;
    else
        select = 0;
```

```
end
else
    select = 0;
end
```

## References

- [1] <https://www.governmentattic.org/> 2016. *National Aeronautics and Space Administration (NASA) Emergency Medical Procedures Manual for the International Space Station (ISS)*. [online] Available at: <-ISSmedicalEmergManual\_2016.pdf> [Accessed 30 March 2021].
- [2] <https://www.bbc.com/>. 2016. How to deal with a medical emergency on the Space Station. [online] Available at: <<https://www.bbc.com/news/health-35254508>> [Accessed 30 March 2021].
- [3] Spina, L., & Lee, M. (1985). Comparison of Co<sub>2</sub> Reduction Process — Bosch and Sabatier. SAE Transactions, 94, 262-271. Retrieved April 5, 2021, from <http://www.jstor.org/stable/44724026>
- [4] Norcross J., Norsk P., Jennifer Law J., Arias D., Conkin J., Perchonok M., Menon A., Huff J., Fogarty J., Wessel J., Whitmire S. “Effects of the 8 psia / 32% O<sub>2</sub> Atmosphere on the Human in the Spaceflight Environment” *NASA*. June 2013.
- [5] Carmo M., Fritz D., Mergel J., Stolten D., “A comprehensive review on PEM water electrolysis” International Journal of Hydrogen Energy, Volume 38, Issue 12, 2013. Pages 4901-4934, ISSN 0360-3199, <https://doi.org/10.1016/j.ijhydene.2013.01.151>.
- [6] Hintze P., Muscatello A. “Self-Cleaning Boudouard Reactor for Full Oxygen Recovery from Carbon Dioxide”. *International Conference on Environmental Systems*. 10-14 July 2016.
- [7] Holmes R., Keller E., King C. “A CARBON DIOXIDE REDUCTION UNIT USING BOSCH REACTION AND EXPENDABLE CATALYST CARTRIDGES”. *NASA Contractor Report*. November 1970.

- [8] Caston R., Luc K., Hendrix D., Hurowitz J., Demple B. “Assessing Toxicity and Nuclear and Mitochondrial DNA Damage Caused by Exposure of Mammalian Cells to Lunar Regolith Simulants” *AGU 100*. 3 April 2018.
- [9] Environmental Control and Life Support System. *NASA*. 23 October 2006.
- [10] Hayne P., Bandfield J., Siegler M., Vasavada A., Ghent R., William J., Greenhagen B., Aharonson O., Elder C., Lucey P., Paige D. “Global regolith thermophysical properties of the Moon from the Diviner Lunar Radiometer Experiment” *Jet Propulsion Lab*. July 2017.
- [11] Kiang C. “EQUIVALENT SYSTEM MASS ANALYSIS OF ASTRONAUT DIETS FOR LONG-DURATION SPACE MISSIONS” *Cornell University*. January 2017.
- [12] “HUMAN INTEGRATION DESIGN HANDBOOK (HIDH)” *NASA*. 27 January 2010. Revised 5 June 2014.
- [13] Do S. “Towards Earth Independence – Tradespace Exploration of Long-Duration Crewed Mars Surface System Architectures” *Massachusetts Institute of Technology*. June 2016.
- [14] Anderson M., Ewert M., Keener J., Wagner S. “Life Support Baseline Values and Assumptions Document” *NASA*. March 2015.
- [15] Greenwood Z. “Plasma Methane Pyrolysis for Spacecraft Oxygen Loop Closure” *11th International Conference on Plasma Assisted Technologies*. January 2018.
- [16] Eckart P. “Spaceflight Life Support and Biospherics”. *Space Technology Library*. 1994.
- [17] Sherif D., Knox J. “International Space Station Carbon Dioxide Removal Assembly (ISS CDRA) Concepts and Advancements”. *NASA*. 2005.
- [18] Peters W., Knox J. “4BMS-X Design and Test Activation” *47th International Conference on Environmental Systems*. July 2017.

- [19] Agui J., Stocker D. “NASA Lunar Dust Filtration and Separations Workshop Report”. *NASA*. December 2009.
- [20] “Ductwork Sizing Computation For Your HVAC: All You Need to Know”  
<https://sandium.com/general-hvac/ductwork-sizing-computation-for-your-hvac-all-you-need-to-know.html>
- [21] “Going to the Moon“ American Museum of Natural History.  
<https://www.amnh.org/exhibitions/beyond-planet-earth-space-exploration/returning-to-the-moon/going-to-the-moon>
- [22] “Ammonia Properties at Gas-Liquid Equilibrium Conditions” Engineering Toolbox.  
[https://www.engineeringtoolbox.com/ammonia-gas-liquid-equilibrium-condition-properties-temperature-pressure-boiling-curve-d\\_2013.html](https://www.engineeringtoolbox.com/ammonia-gas-liquid-equilibrium-condition-properties-temperature-pressure-boiling-curve-d_2013.html)
- [23] Eshima, S. & Nability, J. “Failure Mode and Effects Analysis for Environmental Control and Life Support System Self-Awareness” International Conference on Environmental Systems. July 2020
- [24] Stapleton, T. J. & Heldmann M. “Environmental Control and Life Support for Deep Space Travel” 46th International Conference on Environmental Systems. July 2016.
- [25] Matelli J. A. & Goebel K., ”Resilience evaluation of the environmental control and life support system of a spacecraft for deep space travel” Acta Astronautica, Volume 152, Pages 360-369, 2018.
- [26] Keesee J. “Spacecraft Thermal Control Systems” Massachusetts Institute of Technology. Fall 2003.
- [27] Lam T., Birur G., Bhandari P. Pumped Fluid Loops. Summer 2008.
- [28] Allen, B. and Dunbar, B., “Human Needs: Sustaining Life During Exploration”, 2007
- [29] Carter, D. L., “Status of the Regenerative ECLSS Water Recovery System”, 2009
- [30] Dhoble, A. S. and Pullammanappallil, P. C., “Design and Operation of an Anaerobic Digester

for Waste Management and Fuel Generation during Long Term Lunar Mission”, 2014

[31] Holder, D. W. and Hutchens C. F., “Development Status of the International Space Station Urine Processor Assembly”, 2003

[32] National Overview: Facts and Figures on Materials, Wastes and Recycling, 2021

[33] Prater, T., Edmunsson, J., Fiske, M., Ledbetter, F., Hill, C., Meyyappan, M., Roberts, C., Huebner, L., Hall, P., and Wekheiser, N., “NASA’s In-Space Manufacturing Project: Update on Manufacturing Technologies and Materials to Enable More Sustainable and Safer Exploration,” IAC-19.D3.2B.5, 2019

[34] Prater, T., Werkheiser, N., Ledbetter, F., and Morgan, K., “In-Space Manufacturing at NASA Marshall Space Flight Center: A Portfolio of Fabrication and Recycling Technology Development for the International Space Station,” 2018

[35] Rose, C., Parker, A., Jefferson, B., and Cartmell, E., “The Characterization of Feces and Urine:

A Review of the Literature to Inform Advanced Treatment Technology”, 2015

[36] Ryba, J. and Dunbar, B., “Recycling Water is not Just for Earth Anymore”, 2008.

[37] Doty, Reiley. “New Brine Processor Increases Water Recycling on International Space Station”. Feb 25, 2021. NASA. <https://www.nasa.gov/feature/new-brine-processor-increases-water-recycling-on-international-space-station>

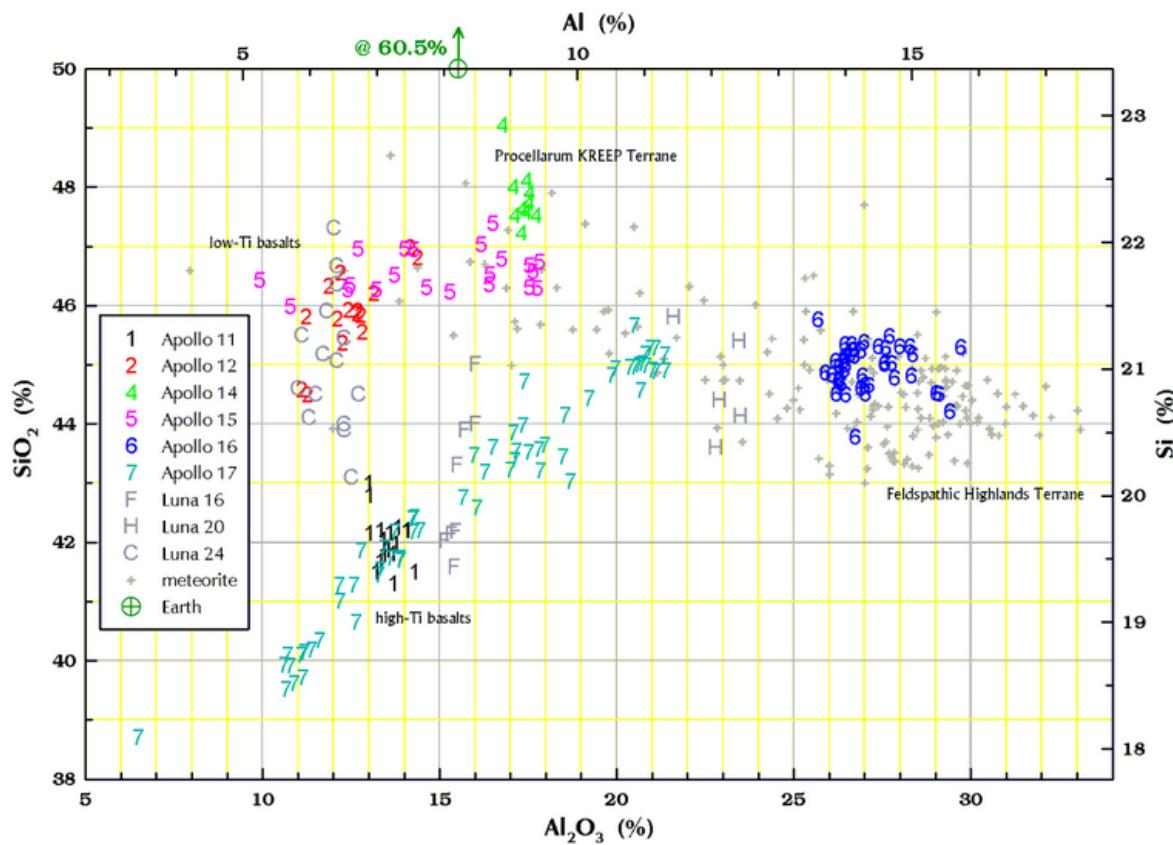
[38] Kortenkamp, D.; Bonasso, R. P.; and Subramanian, D. 2001. Distributed, Autonomous Control of SpaceHabitats, 2752–2762. In Proceedings of the IEEE Aerospace Conference. Washington, D.C.: IEEEComputer Society.

[39] Oren, J., & Howell, H. (1995). Space Station Heat Rejection Subsystem Radiator Assembly Design and Development. *SAE Transactions*, 104, 1086-1095. <http://www.jstor.org/stable/44612020>

## **Systems: Research Facilities**

## I. Regolith Composition

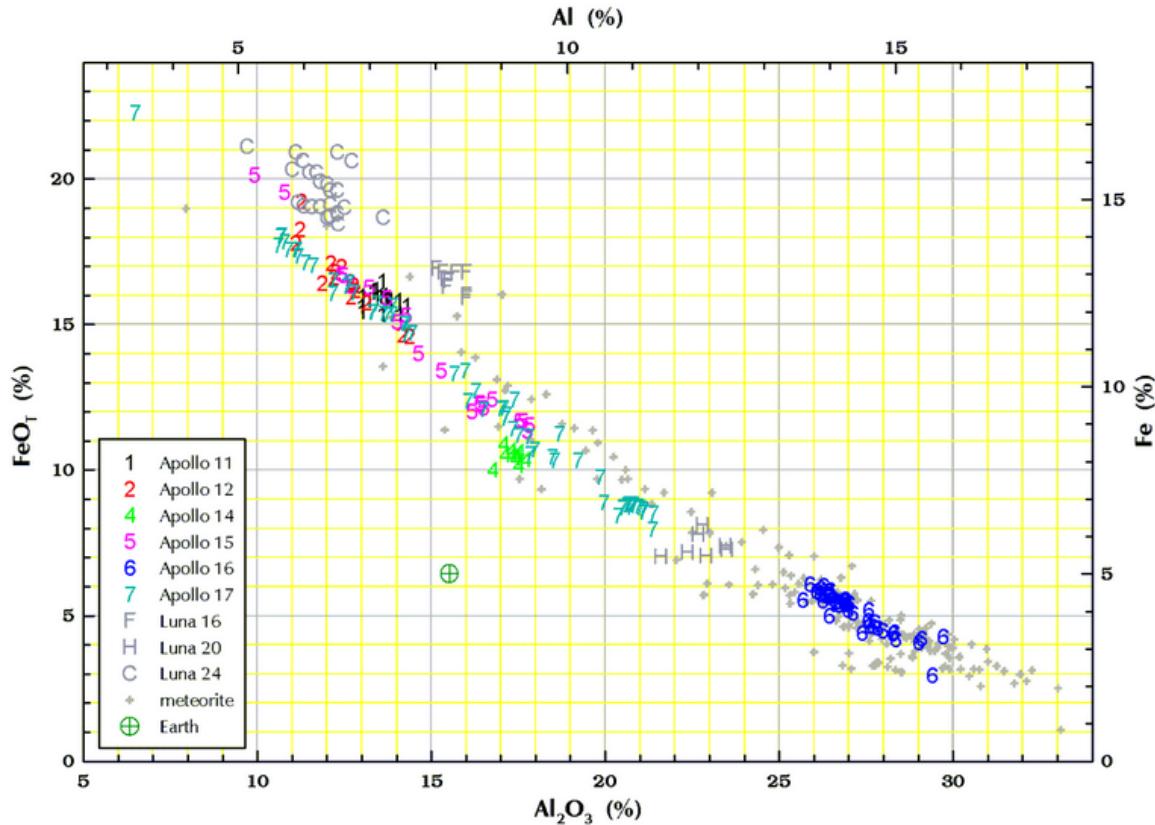
To gain a better understanding of how regolith can be utilized for mission purposes, it is valuable to take a closer look at the composition of the lunar soil. Seen in Figure 15, Dr. Randy Korotev has compiled data from Apollo and Luna missions as well as meteorites to determine the regolith composition [11].



**Fig. 15** Silicon concentration in samples of lunar regolith [11].

The most abundant element present in regolith is oxygen at an estimated 41-45%. The second most abundant is silicon. Figure 15 shows the concentration of silicon oxide and silicon on the left

and right axes, respectively. The graph is normalized by plotting aluminum on the horizontal axes because the aluminum concentration varies over a wide range in lunar samples.



**Fig. 16** Iron concentrations in regolith samples [11].

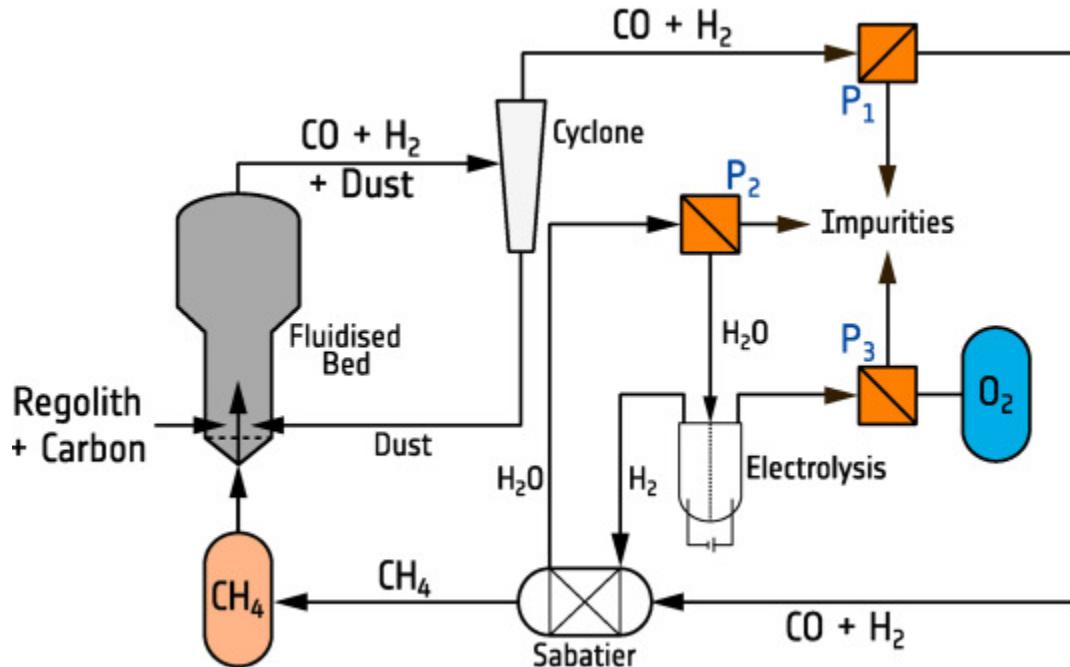
For comparison, Figure 16 shows the concentration of iron in lunar samples. We see the median level of iron (about 9%) is significantly lower than the median concentration of silicon (about 21%). Lunar soil also contains significant amounts of aluminum, calcium, magnesium, and titanium. A very small amount of the regolith is composed of manganese, sodium, potassium, and phosphorus. A knowledge of regolith composition is useful for oxygen extraction and solar cell fabrication in the proceeding Appendix Section B.

## A. ISRU Applications

In-Situ Resource Utilization (ISRU) is critical to sustainable space exploration to limit resupply needs from Earth. Ice deposits offer an attractive source of mission material. Project NextStep focuses on the research of lunar ice volatiles. There are other options, though, for using regolith, but these have been largely neglected from the report due to a lack of space heritage.

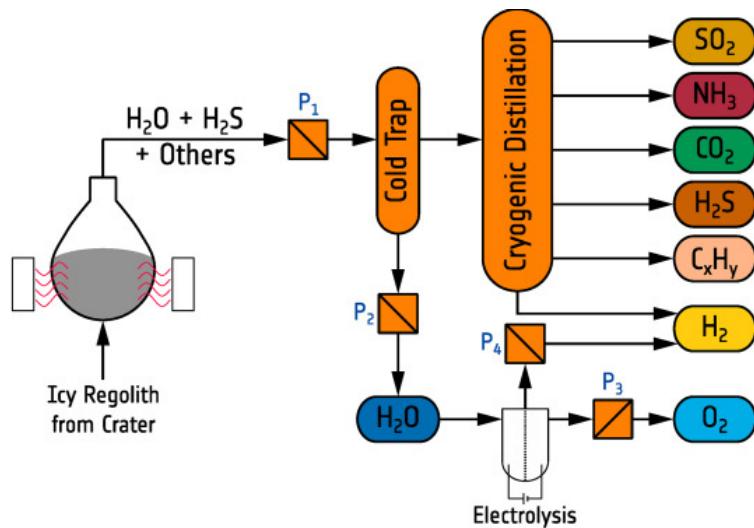
One area of interest for using regolith lies in oxygen extraction. There are some existing designs that utilize gas purification techniques to extract useful materials from regolith. For example, NASA has matured a carbothermal reduction reactor that sources concentrated sunlight to melt regolith to a TLR of five [17]. A brief explanation of oxygen extraction follows.

A thermochemical process employs reactants with higher affinity to oxygen than the other metals present in the regolith such as iron or titanium. Figure 17 shows an architecture using methane as the reducing agent for extracting oxygen.



**Fig. 17** Illustration of Carbothermal Reduction of Regolith using Methane [18]

This design could be interfaced with the Sabatier Reactor used in the ECLSS on base. Being able to produce oxygen on the Moon takes stress off of supplies shipped from Earth and further improves mission sustainability. Beyond oxygen extraction from solid regolith, individual compounds could be extracted from ice deposits on the Moon using cryogenic distillation. A simplified architecture of this process is included in Figure 18.



**Fig. 18** Volatile Ice Extraction Using Cryogenic Distillation [18].

This design could be especially useful for fabricating materials on the Moon. For example, one of the main ingredients in solar cells is silicon. Dr. Alex Ignatiev proposes manufacturing solar cells using silicon, aluminum, calcium, oxygen, and iron found in lunar regolith. Some Earth materials would be required to reach high solar energy efficiency, but the abundance of lunar soil would allow for the mass production of thin solar cells that, in theory, provide the bulk of power needed in the colony.

Production of solar cells using ISRU is purely theoretical. While this would be a paradigm shift in terms of mission sustainability, the lack of practical testing precludes this technology from

being relevant for Project NextStep. Future researchers should, however, strongly consider ISRU techniques for power and other colony needs.

# Systems: Agriculture and Food

Nicholas T. Masso<sup>13</sup>

*Purdue University, West Lafayette, IN, 47907, USA*

Kyle Alvarez<sup>2</sup>, Kartik Bhardwaj<sup>3</sup>, E. Wellington Froelich<sup>4</sup>, Reece Hansen<sup>5</sup>, Jugal Hemani<sup>6</sup>

*Purdue University, West Lafayette, IN, 47907, USA*

## Nomenclature

*m* = amplitude of oscillation

### *Subscripts*

*cg* = center of gravity

---

<sup>13</sup> Systems – Agriculture Team Lead.

<sup>2-6</sup>Systems – Agriculture Team Members.

## I. Determining Caloric Intake

One of the first things we need to estimate how much food the colonists need is an estimation of caloric intake on a daily basis. With a caloric estimate in hand, one can estimate the amount of food needed. The Estimated Energy Requirement (EER) equations found in NASA's Human Integration Design Handbook estimate the number of kilocalories needed per day [1]. The equations are separated based on gender. For men 19 years and older, the EER equation is:

$$EER = 622 - 9.53 * \text{age}(\text{years}) + 1.25 * [15.9 * \text{mass}(\text{kilograms}) + 539.6 * \text{height}(\text{meters})] \quad [1]$$

For women 19 years and older, the EER equation is:

$$EER = 354 - 6.91 * \text{age}(\text{years}) + 1.25 * [9.36 * \text{mass}(\text{kilograms}) + 726 * \text{height}(\text{meters})] \quad [2]$$

At the early stages of planning, we do not know who will be sent to the base. As such, assumptions will have to be made. Assuming the crew will represent the average American, estimations on caloric intake can be computed. The crew will have the average heights of Americans and have healthy body weight. According to the CDC, Americans' average height is 1.726 meters for men and 1.6129 meters for women [2]. Assuming a BMI of 21, the average mass of men comes out to 64.4 kg, and the average mass of women comes out to 53.5 kg. With these values in hand and using the EER equations, one can estimate 2800 kcal per day for men and 2200 kcal per day for women.

## II. Preliminary Food and Diet Research

Researching the types of diets that astronauts needed to maintain in space began with initially researching what construes as a healthy diet on Earth. The next logical step is then researching the differences and limitations that space travel can have on these diets. Only then can food storage be discussed.

A healthy diet here on Earth starts with a couple of things. Obviously, it shifts from person to person, so while every case is different, these basics can be the foundation of any astronaut's diet. First, this diet primarily includes eating meats, fish, eggs, vegetables, fruit, nuts, seeds, high-fat dairy, general fats, and healthy oils. What it is generally devoid of is sugars, wheat, seed oils and trans fats, low-fat products, and highly processed foods. This ideal diet was found from Healthline [3]. There is good reason for avoiding all of these, which this report will now explore.

### *1. Breads and their Components*

Breads and grains are generally avoided due to their high carbs content. This is the case for both whole-grain varieties as well as those in "refined" flours. Grains like rice, wheat, and oats are also dense in carbohydrates and should be avoided on adopted low-carb diets for healthy living in space. Below is Table 1 with a chart describing how many grams of carbohydrates and fiber are within popular bread choices, and the information was gathered from Healthline [4].

**Table 1: Popular bread items and their respective carbohydrates and fiber components**

| Type of Bread               | Carbs (grams) | Fiber (grams) |
|-----------------------------|---------------|---------------|
| White Bread (1 slice)       | 13            | 1             |
| Whole-wheat bread (1 slice) | 15            | 2             |
| Flour tortilla(10-inch)     | 34            | 2             |

|               |    |   |
|---------------|----|---|
| Bagel(3-inch) | 28 | 1 |
|---------------|----|---|

Bread has already been long avoided by the space program due to its lack of structure and easily promoted shelf-life. However, it cannot be eliminated from the space diet entirely, so astronauts actually rely heavily on tortillas. As can be seen from the table, tortillas provide the highest number of carbs and fibers, which when combined with their easily storable shape and size, makes them ideal for transportation and storage for astronauts who want to maintain some level of carbs within their diets.

## *2. Fruits and their Components*

Fruits would be an ideal component of most diets, and a diet with a high intake of fruits and vegetables has consistently been linked to a lower risk of cancer and heart disease according to PubMed [5]. Below in Table 2 is a table with the associated amounts of carbohydrates and fiber for different fruits, with data gathered from Healthline [6].

**Table 2: Popular fruit items and their respective carbohydrates and fiber components**

| Type of Fruit                       | Carbs (grams) | Fiber (grams) |
|-------------------------------------|---------------|---------------|
| Banana (One medium)                 | 24            | 3             |
| Raisins (One oz/28 grams)           | 21            | 1             |
| Dates (Two large)                   | 32            | 4             |
| Mango, sliced (One cup / 165 grams) | 25            | 3             |
| Blueberries (One cup/148 grams)     | 21            | 3.6           |
| Raspberries (One cup/ 123 grams)    | 15            | 8             |
| Strawberries (One cup/ 144 grams)   | 11            | 3             |

Interestingly, berries are lower in sugar, but higher in fiber than most fruits. They would make for the most easily enjoyed fruits in space, as they are smaller but denser in their carbohydrates and fibers compounds. Blueberries, raspberries, and strawberries specifically have been shown to

contain vitamin C, antioxidant anthocyanins, and more, from PubMed [7]. They would be difficult to keep fresh aboard launches but stand to make for a great farming opportunity.

### *3. Vegetables and their Components*

The next ideal food group for healthy diets is the starchy vegetables category. These are very high in fiber, which is beneficial to blood sugar control and weight control, so they can be incredibly helpful to the astronauts in space as they work out and perform tasks for the habitat. Some vegetables should be avoided, however, as they can be composed of more carbs than fiber which is an adverse effect of the healthy carb diet here on Earth. The Table 3 below is a congregation of the data from Healthline [8] that shows some great options and some less than stellar options for the healthy diet of a low-carb person.

**Table 3: Popular vegetable items and their respective carbohydrates and fiber components.**

| Type of Vegetable                  | Carbs (grams) | Fiber (grams) |
|------------------------------------|---------------|---------------|
| Corn (One cup/ 175 grams)          | 36            | 5             |
| Potato (One medium)                | 33            | 4             |
| Sweet Potato/ Yams (One medium)    | 20            | 4             |
| Beets, cooked (One cup/ 150 grams) | 12            | 4             |
| Bell peppers (One cup/149 grams)   | 6             | 3             |
| Broccoli (One cup/ 91 grams)       | 4             | 2             |
| Asparagus (One cup/ 180 grams)     | 4             | 4             |

Bell peppers can actually contribute 317% of the Reference Daily Intake (RDI) for vitamin C and 93% of the RDI for vitamin A, both of which can be lacking on low-carb diets. Similarly, broccoli was found to provide over 100% of the RDI for vitamins C and K. Asparagus, meanwhile, contributed to a solid source of vitamins A, C, and K, and lab studies have found that it can help stop the growth of several types of cancer, and protect brain health while reducing anxiety,

according to PubMed [9]. The above data shows that a diet needs to consist of the right types of vegetables, as astronauts want to limit the carbs they intake, so non-starchy and high-fiber vegetables are best suited to the job.

#### *4. Foods to avoid and their Components*

There are also many foods to avoid while creating the ideal healthy diet, both on Earth and in space. Some of these are listed below, but the key pattern to look for is a high number of carbs with not a high number of fiber, or other important nutrients found on the Reference Daily Intake.

The data for the Table 4 below was found on Healthline [3].

**Table 4: Popular food items and their respective carbohydrates and fiber components**

| Type of Food                                         | Carbs (grams) | Fiber (grams) |
|------------------------------------------------------|---------------|---------------|
| Pasta (One cup/ 250 grams)                           | 40            | 3             |
| Whole-wheat pasta (One cup/ 250 grams)               | 31            | 6             |
| Oatmeal (One cup/ 90 grams)                          | 28            | 4             |
| Steel-cut oats, cooked (One cup/ 90 grams)           | 48            | 10            |
| Whole-grain cereals and granola (One cup/ 122 grams) | 60            | 14            |
| Sweetened Yogurt (One cup/ 245 grams)                | 47            | 0             |
| Juice (355 ml)                                       | 48            | 2             |

Alternatives to the foods above can be had, though they may pose difficulties for the space portion of the space diet. Low-carb diets don't generally include spaghetti or pasta unless it's an unrealistically small portion, but spiralized vegetables or shirataki noodles are popular alternatives. While granola and other whole wheats are good sources of fiber, they normally tradeoff high carb diets for this fiber intake, so they are not a good idea. In the research from Healthline journal, it was also discovered that most liquid carbs can actually have a bigger negative impact on the body than solid carbs, so most alcohols and juices are not part of a great low-carbs diet.

### *5. NASA Nutritional Biochemistry Research*

Despite the extensive research available on healthy food diets here on Earth, there are still distinct challenges that need to be tackled when preparing for the diets aboard the ISS or the Moon. There is, however, a group of NASA scientists dedicated to food research and production. One such scientist, Dr. Scott Smith, NASA nutritionist and the manager for the nutritional biochemistry lab, provided great insight on the research it takes to keep crews healthy in space. His job is to understand what the body needs and provide this information to the food lab that creates the food the astronauts will consume and take. Current ways to monitor food and nutrition intake on the ISS include iPad logs of everything the astronauts eat, which then can inform them of whether or not they are getting enough fluid, vitamin A, etc. It's important for longer term missions like the one of a lunar base, because if the crew is missing one or two key nutrients in their diet, it could turn deadly. History shows this, because it was estimated that scurvy killed more sailors than all other causes of death together. Scurvy occurs when there is a vitamin C deficiency, and sailors out at sea would not get any of this nutrient in their diets, and almost 2 million sailors died in the time span between Columbus' trip and the industrial age. Therefore, the crew needs to follow the research conducted on current and past spaceflights to maintain their nutritional health. According to Dr. Scott Smith, some key things are that the astronauts must eat during their flights. This will directly affect how they maintain their body mass, and then this will help them maintain the daily number of calories they are consuming. This will in turn lead to the ground teams analyzing how much protein and other important nutrients they are getting and fine-tuning the diets accordingly. Some of the dangers of a long-term mission like the team is planning can include bone loss, muscle loss, and perhaps a change in how the cardiovascular system works. The immune system could function differently, or the crews' vitamin D status could rapidly change. Radiation was also an

issue that Dr. Scott Smith mentioned could be dangerous on a Moon mission or a Mars mission. Nutrition could, Dr. Smith asserts, help prevent some of the radiation damage. He claims that different foods like broccoli and other healthy foods can antagonize oxidative damage and cancer occurrence, which is hugely beneficial for the crews on the Moon. Crews' bodies also change considerably when in space. For instance, the body stores more iron during spaceflight because your blood volume contracts about 10 to 15 percent. Another change is that your body can begin to store too much sodium, which can be dangerous in the long run. High sodium can lead to bone issues as well as eye issues, which is a known danger of long-term flight. These vision issues can stem from a number of things, like diet, or perhaps CO<sub>2</sub> filtration in spaceflight. This information was all collected from [10].

#### *6. Protein Alternatives*

As Dr. Scott Smith previously mentioned, it is integral to the health of the astronauts that their nutrition is closely monitored. Some past ways the ISS teams evaluated crews' nutritional statuses included collecting blood and urine samples throughout missions to find out if their diets were keeping their muscles, bones, vitamins, and minerals within the acceptable limits deemed by mission control. This would happen over six-month intervals and is helpful for determining how the Moon crews will have to adjust their daily food intakes.

Loss of calcium is a serious concern during spaceflight, and nutrition can provide several solutions. It was hypothesized that a diet with a lower ratio of animal protein to potassium can lead to less loss of bone minerals. Interestingly, animal protein from meat and dairy foods can be rich in high-acid chemicals. Foods with plenty of potassium, meanwhile, provide the opposite, chemicals that produce more base to neutralize acid in the body. It was discovered that when a diet is more acidic

than basic, bone breakdown occurs more frequently. Therefore, potential alternatives to meats from animals were researched.

*a. Alternative Protein #1: Crickets*

It is not only recommended to lower the amount of animal red meats in the crews' diet, but it is also difficult to store and produce red meats on a lunar base. According to the Dietary Reference Intake, the average person needs about 0.8 grams of protein per kilogram of body weight. The first alternative that came up in our research that could provide dense protein at a significantly smaller mass and volume fraction were insects. They can be quite protein dense and can be raised and maintained quickly. Crickets specifically were investigated for their quick yield times and diet consisting of leafy greens that may already be growing around the habitat. Already on Earth, they are being raised to turn into a protein rich flour that can be used in lieu of regular wheat-based flour. This can also remove some of the stigma of eating live bugs. Table 5 below shows how much protein crickets can provide versus beef, although according to the Food and Agriculture Organization of the United Nations 20 percent of the world's consumers eat insects. The data was found from [11].

**Table 5: Protein density of crickets and beef.**

| Protein Source | Amount of Protein                                  |
|----------------|----------------------------------------------------|
| Crickets       | 21 grams of protein per 100 grams of cooked weight |
| Beef           | 28 grams per 100 grams                             |

Crickets provide a lean animal protein that require less space, less water, and produce far fewer greenhouse-gas emissions than beef.

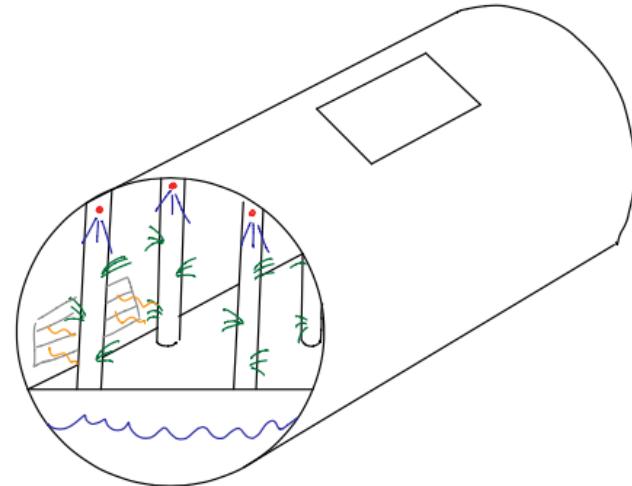
*b. Alternative Protein #2: Lab-grown Meats*

Similarly, promising meats that provide steady amounts of protein could end up being lab-grown. Cultured beef, chicken, and other meats for a fraction of the water percentage and without livestock could be available. In 2011, a study was conducted that determined that growing meat in labs would cut down on the land required to produce steaks, sausages, and bacon by 99 percent, and reduce the associated need for water by over 90 percent. This would be incredibly beneficial to a lunar base that stands to be limited on water to begin with. This data was found from [12].

### III. Structure Choice

The structure of the model is a very important aspect to the life and future of the whole mission. Starting with the type of system, there were three approaches considered: hydroponics, aeroponics, and aquaponics. With hydroponics, a nutrient water system would be ran through vertical farming tubes to supply the plant with everything it needs to survive. With aquaponics, fish are placed in the same model as the vertical plant farming. This way they can supply the plants with nutrients and water. The last and final approach is aeroponics. Aeroponics is very similar with hydroponics but instead of a nutrient water system running through the vertical farming, a mist is sprayed on the plants to keep them alive. Comparing all of the different approaches, aeroponics was the most beneficial for the team.

Designing the structure was the next step in making this system. With aeroponics being the choice for plant growth, 24 vertical towers will be placed inside of the system. Two rows will be placed inside the model with 12 in each row. This leaves a space in the middle for the crew to walk freely between the plants. In order for the plants to grow temperature has to be regulated, a nutrient spray has to be set on a timer, lights have to be controlled, and a dehumidifier to keep the humidity in order. Figure 1 shows a rough drawing of the system.



**Figure 1: A sketch of the TREES inside a habitat module.**

With the 24 towers, there are a variety of plants the astronauts can plant. Starting with the vegetables, there are black beans, green beans, lettuce, kale, chard, butternut squash, zucchini, tomatoes, and peppers. The list for fruit has much fewer choices with only strawberries and other berries. Each tower can roughly hold 28 plants each.

#### IV. Mass/Volume Calculations for the Structure

With the structure holding the pumps, towers, plants, water, lights, fertilizer, controls, and plastic sheets, the mass and volume can be broken down to multiple components. All of these aspects to the model are important to the survivability of the system as a whole, so they are all necessary.

##### *1. Pump Sizing/Power Calculations*

Starting with the pumps, a small calculation has to be made first: the mass of the towers. Knowing that there are going to be 24 vertical towers, here is the calculation of all of the towers combined:

$$1 \text{ Tower} = 35 \text{ kg}$$

$$\text{Therefore } 24 \text{ Towers} * 35 \text{ kg} = 840 \text{ kg}$$

$$\text{Height of Each Tower} = 4 \text{ meters}$$

$$\text{Diameter of Each Tower} = 0.5 \text{ meters}$$

Knowing this much information, the volume of all the towers is roughly 0.72 meters cubed.

To get the water to all of the timed sprinklers, 4 pumps are required to get to the 24 towers. The calculation for the mass of the pumps and the energy are:

$$\text{Mass of 1 pump} = 10 \text{ kg}$$

$$\text{With 4 pumps: } 4 * 10 = 40 \text{ kg}$$

$$\text{Volume of one pump} = 0.3 * 0.2 * 0.2 = 0.012 \text{ meters cubed}$$

$$\text{Volume of all the pumps: } 4 * 0.012 = 0.048 \text{ meters cubed}$$

$$\text{The flow rate of the pumps} = 0.454 \text{ meters cubed per hour}$$

$$\text{Pressure difference} = 1 * 1.62 * 4 = 6.48$$

$$\text{Pump Power for 1 tower} = 6.48 * .454 / 3.6e3 = 8.1765e-4 \text{ Watts}$$

*With a total Pump Power for all Towers = 1.7337 Watts*

Knowing all of this information, the total power for the pump/tower system is about 1.7337 Watts.

With this knowledge, the habitats team and the astronauts know the amount of power needed for the pumps.

## *2. Light Selection and Power Calculations*

Using the resource,” Best Grow Lights For Growing Vegetables Indoor” by Benjamine Kilbride [13] he states that the LEDs should be about 18 inches away from the plants. This means that the most vertical height one row of LEDS can cover is 28 inches or 0.7112 meters. Knowing this information, the calculations involving the lights (including power) are below. We start with the basic electronic properties of a row of LEDs and some other basic things:

$$V_{LED} = 3.5 \text{ volts}$$

$$I_{LED} = 20 \text{ mA}$$

$$L_{Habitat} = 12 \text{ meters}$$

$$L_{LED} = 2.13 \text{ meters}$$

$$S_{vertical\ spacing} = .7112 \text{ meters}$$

Now we can determine how many “rows” equivalent we would need to illuminate the length of the habitat:

$$N_{rows} = L_{Habitat} / L_{LED} = 5.6243$$

$$T_{on\ per\ day} = 16 \text{ hours}$$

*Rough Estimate Based on a Grow Light = 120 meters*

$$P_{total, curtain} = 566.9291 \text{ Watts}$$

$$N_{curtains} = 4$$

$$P_{total} = P_{total, curtain} * N_{curtains} = 2.2677e3 \text{ Watts}$$

$$m_{light} = 0.0272 \text{ mg (about 3 lbs per 23" of LEDS)}$$

$$\text{Volume of Lights} = 0.2430 \text{ meters cubed}$$

With all of this information, the astronauts know the total power for the lights and can plan around to make sure each plant reaches the correct amount of light for optimal growth.

## V. Control of the Systems

For the controls aspect of the ecosystem, there are three control systems to consider: the watering cycle for the plants, the humidity of the ecosystem, and the regulation of temperature.

### 2. Watering

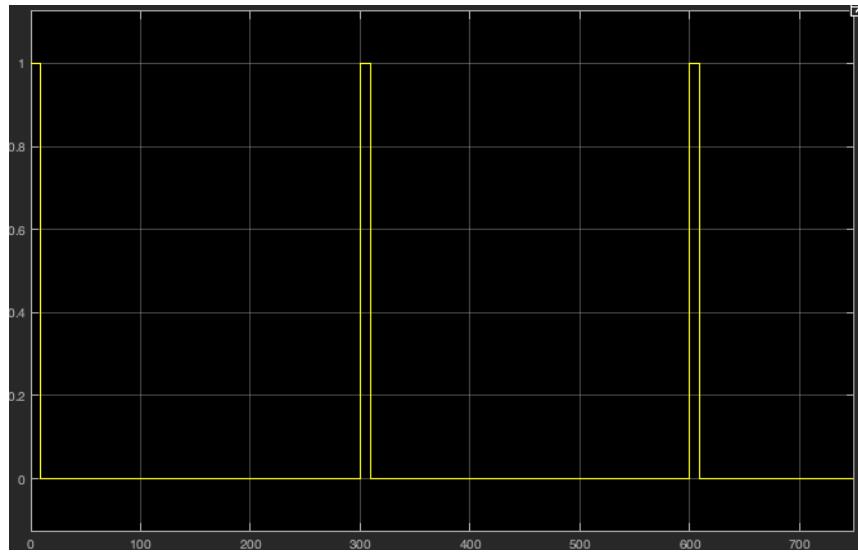
For the Agriculture System, each plant needs water and nutrients to survive. Two sprinklers are attached through the piping of the aeroponic system to spray nutrients and water equally amongst the plants. For optimization for plant growth, the plants need to be watered for 15 seconds every 5 min. A Simulink model was made to demonstrate a spraying pattern to match that time restriction.

Figure 2 shows the Simulink model with a graph showing the watering cycle in action.



**Figure 2: Basic Simulink for a Watering Cycle**

Reece Hansen



**Figure 3: Graph of Simulink to Demonstrate Watering Cycle**

This simple design demonstrates the idea needed for a plant watering cycle. For the mission, going with a third-party design will save time, money, and resources for project. Table 6 shows a short list of examples of third-party parts:

**Table 6: List of Third-Party Parts for the Watering System**

| Timed Sprinkler                 | Cost      |
|---------------------------------|-----------|
| Orbit 1 Output Port Digital 1x  | 29.98     |
| Orbit Mechanical Water 1x       | 13.98     |
| Melnor Hydrologic 4-zone        | 49.98     |
| Garden Accessory Irrigation 2x  | 39.97     |
| Melnor 4-Zone Water Timer       | 54.97     |
| Orbit Digital 2 Outlet          | 36.21     |
| Melnor Raincloud Smart Water    | 112.99    |
| Orbit In-Ground Blu-Lock Tubing | 131.72    |
|                                 |           |
|                                 |           |
|                                 |           |
|                                 |           |
|                                 |           |
| Average                         | 58.725    |
| Ranging                         | 1-4 Tubes |

As demonstrated in Table 6, the amount of watering sprinklers can range from one to four tubes and can range in price from 14 dollars to 132 U.S. dollars based on which design is best for our mission.

### *3. Humidity*

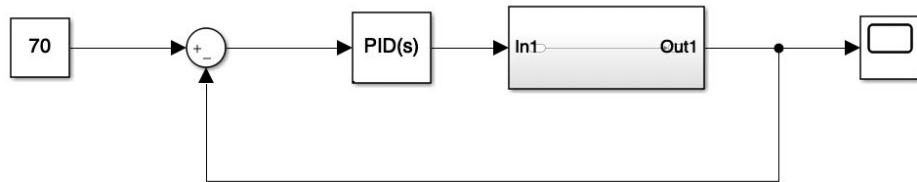
The humidity of the agriculture module also needs to be addressed. To optimize plant growth, the chamber humidity must lie between 50-70%. If the humidity of the model goes above 90%, the plant's leaves could mold and result in the death of the plant. In order to reduce humidity, a dehumidifier is needed. Table 7 lists the third party made dehumidifiers:

**Table 7: List of Third Party Dehumidifiers**

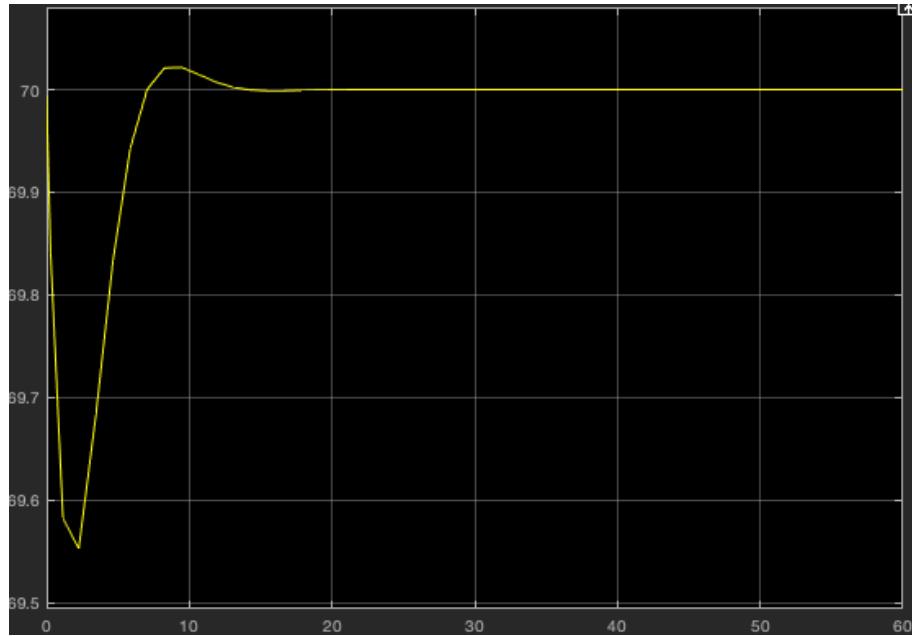
| Dehumidifier                         | Cost         |
|--------------------------------------|--------------|
| 1.5 Pints Lightweight Portable       | 19.08        |
| 32 Pint Magic Chef MC4PDH            | 51.99        |
| 20 Ounce PureDry Deluxe              | 99.99        |
| 50 Pint Toshiba 115 Volt Energy Star | 229          |
| 45 Pint Portable w/Drain             | 139.99       |
| Hyrdorwave 4.22 pint                 | 79.95        |
| 50 Pint GE Portable                  | 229.99       |
| Crane EE-100 Portable 4 Pint         | 76.3         |
| 35 Pint Hisense                      | 199          |
| 70 Pint Portable GE                  | 179.99       |
| Frigidair 70 Pints                   | 259          |
| Frigidair 50 Pints                   | 238          |
| Colzer 30 Pint                       | 139          |
| 30 Pint hOmelabs Portable            | 199.97       |
| <hr/>                                |              |
| Average                              | 152.946429   |
| Ranging                              | 1.5-70 Pints |

As demonstrated in Table 7, the size for a dehumidifier can range from 1.5 pints to 70 pints and can range in price from 19 dollars to 230 dollars based on which design is best for our mission.

Each of the dehumidifiers needs to keep the humidity between 50-70% but for optimum plant growth, 70% humidity is needed. Using close to the same PID controller and Simulink as the temperature regulator, a constant 70% is doable. Below is the Simulink and the graph corresponding to the Simulink:



**Figure 4: Simulink of a Dehumidifier**

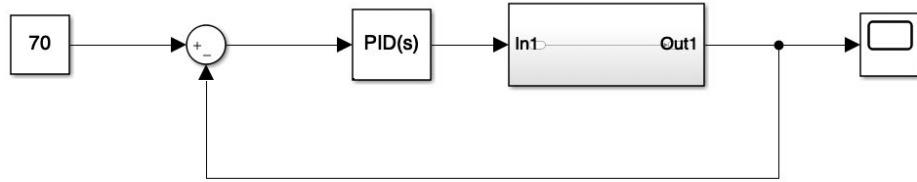


**Figure 5: Graph of Simulink Showing Regulation of Humidity**

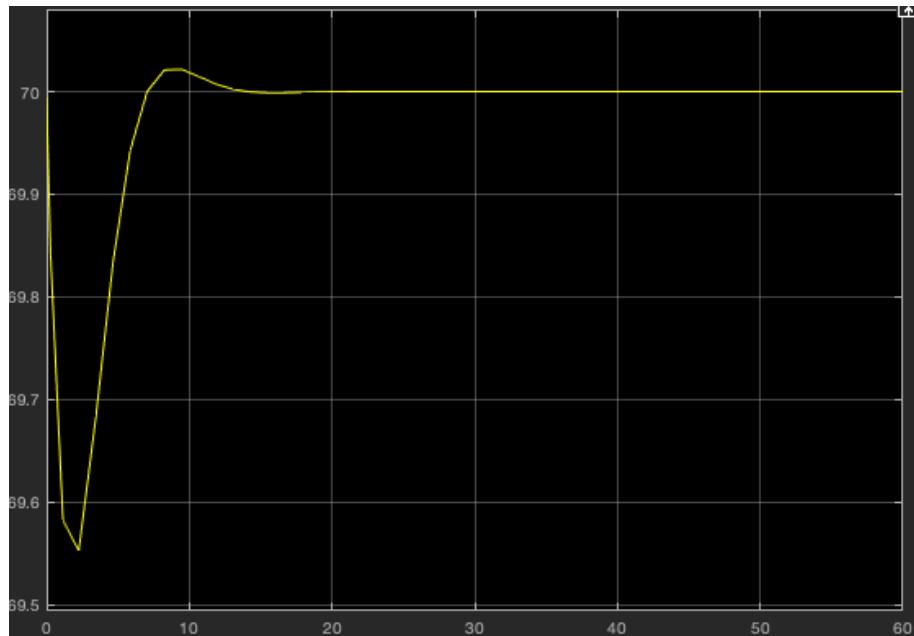
The two figures above (Figure 4 and 5) show how the humidity can maintain a constant 70% humidity for optimum plant growth. This will help the model and plants survive longer, recycle water, and make sure the plants don't grow plant mold.

#### 4. Temperature

Temperature is the last aspect that needs to be controlled. For optimal plant growth, the temperature needs to be around 70 degrees Fahrenheit or 21.1 degrees Celsius. A thermostat must be created and implemented in the module to keep temperature constant. A Simulink model was created to demonstrate the use of a thermostat. This Simulink model contains a PID controller to maintain the 21.1 degrees Celsius needed for the plants. Figure 6 shows the Simulink model with its corresponding graph of the temperature over time in Figure 7:



**Figure 6: Simulink of a Temperature Regulator**



**Figure 7: Graph of Simulink Showing Regulation of Temperature**

As shown in Figure 7 above, the temperature maintains a 70 degrees F. While it is possible to create one of these, going to a third party will benefit the team more by saving time, money, and resources. Table 9 lists different products made by third party companies:

**Table 9: Excel Sheet of Different Thermostats**

| Thermostat                       | Cost       |
|----------------------------------|------------|
| Nest-Learning Thermostat         | 249        |
| Ecobee Alexa Built-in Thermostat | 249.99     |
| Google Nest Snow Thermostat      | 129.99     |
| Ecobee 3 Lite 2.0 Thermostat     | 169        |
| Nest Learning Thermostat E       | 209.99     |
| Honeywell Lyric T5+              | 94         |
| Honeywell Lyric T9               | 169.98     |
| Honeywell Lyric T6+              | 159.71     |
| Sensi Smart Programmable         | 54.5       |
| Lux Products GEO-WH              | 99.91      |
| Honeywell Rth6580wf              | 24         |
| Honeywell T5                     | 89         |
|                                  |            |
|                                  |            |
|                                  |            |
| Average                          | 141.589167 |

As demonstrated in Table 9, the thermostats can range in price from 24 dollars to 250 dollars based on which design is best for our mission.

## VI. Power Calculation Program

```
%{
AAE 450 Aeroponics Space Req'd Script
Variety of Plants:
Vegetables:
-Legumes: Black Beans, green beans
-Leafy: Lettuce, kale, Chard
-Squash: Butternut, zucchini
-Vine: Tomatoes, peppers
Fruits:
-Berries: Strawberries, currents
%
num_crew = 19; %initial assumption
num_plants = 11; %number of plants the crew should be able to consume
see image above}
```

```

req_plants = 2; %required plants for successful harvesting of each
growing_per = 36; %days
harv_period = 3; %days
total_rotating = growing_per/harv_period+1; %total rotating growth
cycles for constant production
total_plants_pp = num_plants*req_plants; %total required plants per
person
tower_cap = 60; %aeroponics tower that can hold 28 plants
req_towers = total_plants_pp/tower_cap*num_crew*total_rotating
%required towers
req_towers = 90.5667
plants_per_week = num_plants*req_plants*7/harv_period*num_crew %number
of plants harvested every week
plants_per_week = 975.3333
tower_height = 4; %m
tower_base_d = 0.5; %diameter in m
tot_tower_vol = tower_height*tower_base_d^2/4*pi*req_towers %total vol
req in m^3
tot_tower_vol = 71.1309
total_floor_space = tower_base_d^2/4*pi*req_towers*3 %add in 20%
morespace to account for the passage ways, etc. in m^2
total_floor_space = 53.3482
num_tower_p_module = 23; %current best estimate
num_Mg_p_tower = 25/264.17205; %in Mg, needs 25 gallons per tower
Mg_water_p_module = req_towers/num_tower_p_module*num_Mg_p_tower
Mg_water_p_module = 0.3726
% For potatoes:
num_crew = 19; %initial assumption
num_plants = 1; %this is for potatoes specifically
req_plants = 9; %required plants for successful harvesting of each
growing_per = 90; %days
harv_period = 3; %days
total_rotating = growing_per/harv_period+1 %total rotating growth cycles
for constant production
total_rotating = 31
total_plants_pp = num_plants*req_plants; %total required plants per
person
tower_cap = 60; %aeroponics tower that can hold 28 plants
req_towers = total_plants_pp/tower_cap*num_crew*total_rotating
%required towers
req_towers = 88.3500
plants_per_week = num_plants*req_plants*7/harv_period*num_crew %number
of plants harvested every week
plants_per_week = 399
tower_height = 4; %m
tower_base_d = 0.5; %diameter in m
tot_tower_vol = tower_height*tower_base_d^2/4*pi*req_towers %total vol
req in m^3
tot_tower_vol = 69.3899
total_floor_space = tower_base_d^2/4*pi*req_towers*3 %add in 20%
morespace to account for the passage
total_floor_space = 52.0424
% Light CALCULATIONS

```

```
% This page [ref] says LEDs should be about 18 inches away from the
plants meaning that the most vertical height one row of LEDS can
cover is 28in or 0.7112m
LED_voltage = 3.5; %Volts
LED_current = 20; %mA
length_hab = 12; %meters
rec_LED_vert_spacing = 0.7112; %meters
num_LED_rows = tower_height/rec_LED_vert_spacing
num_LED_rows = 5.6243
num_hours_light = 16; %hrs
LED_p_meter = 120; %a rough estimate based on a grow light on amazon
[ref]
total_LEDs = num_LED_rows*length_hab*LED_p_meter;
total_power_per_LED_curtain = total_LEDs*LED_voltage*LED_current/1000
%Watts
total_power_per_LED_curtain = 566.9291
num_curtains = 4;
total_power_for_light = num_curtains*total_power_per_LED_curtain %Watts
total_power_for_light = 2.2677e+03
mass_p_light = 0.00136078*(length_hab/0.6) %Mg, about 3 lbs per 23" of
LEDs see this link [ref]
mass_p_light = 0.0272
total_mass_lighting = mass_p_light*num_curtains*num_LED_rows %in Mg
total_mass_lighting = 0.6123
vol_lights = .03*length_hab*0.03*num_LED_rows*num_curtains %Vol in m^3
and guestimating 3cm by 3cm profile for lights
vol_lights = 0.2430
Pump Power Req's
Flow_rate = 0.454249414; %in m^3/hr, equal to 2 gal/min
press_diff = 1*1.62*tower_height;
pump_power_tower_p_hour = press_diff*Flow_rate/3.6e3 %in W
pump_power_tower_p_hour = 8.1765e-04
tot_pump_tower = pump_power_tower_p_hour*req_towers*24 % total pump
power for towers
tot_pump_tower = 1.7337
```

## References

- [1] NASA, “HUMAN INTEGRATION DESIGN HANDBOOK,” NASA Available: [https://www.nasa.gov/sites/default/files/atoms/files/human\\_integration\\_design\\_handbook\\_revision\\_1.pdf](https://www.nasa.gov/sites/default/files/atoms/files/human_integration_design_handbook_revision_1.pdf).
- [2] CDC, “FastStats - Body Measurements,” CDC Available: <https://www.cdc.gov/nchs/fastats/body-measurements.htm>.
- [3] Gunnars, K., “A Low-Carb Meal Plan and Menu to Improve Your Health,” Healthline Available: <https://www.healthline.com/nutrition/low-carb-diet-meal-plan-and-menu#foods-to-eat>.
- [4] Spritzler, F., “14 Foods to Avoid (Or Limit) on a Low-Carb Diet,” Healthline Available: <https://www.healthline.com/nutrition/14-foods-to-avoid-on-low-carb>.
- [5] J.; D. L. A. P. H. S. D., “Fruit and vegetable consumption and risk of coronary heart disease: a meta-analysis of cohort studies,” The Journal of nutrition Available: <https://pubmed.ncbi.nlm.nih.gov/16988131/>.
- [6] Robertson, R., “The 8 Healthiest Berries You Can Eat,” Healthline Available: <https://www.healthline.com/nutrition/8-healthy-berries>.
- [7] Stull, A. J., Cash, K. C., Johnson, W. D., Champagne, C. M., and Cefalu, W. T., “Bioactives in blueberries improve insulin sensitivity in obese, insulin-resistant men and women,” The Journal of nutrition Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3139238/>.
- [8] Spritzler, F., “The 21 Best Low-Carb Vegetables,” Healthline Available: <https://www.healthline.com/nutrition/21-best-low-carb-vegetables>.
- [9] Xiang J;Xiang Y;Lin S;Xin D;Liu X;Weng L;Chen T;Zhang M; “Anticancer effects of deproteinized asparagus polysaccharide on hepatocellular carcinoma in vitro and in vivo,” *Tumour biology : the journal of the International Society for Oncodevelopmental Biology and Medicine* Available: <https://pubmed.ncbi.nlm.nih.gov/24310501/>.
- [10] Stafford, W., “Ep 43: Diet Like an Astronaut,” NASA Available: <https://www.nasa.gov/johnson/HWHAP/diet-like-an-astronaut>.

- [11] McCausland, P., “Meet the Bug Farmers Who Want to Feed You Crickets,” *The Atlantic* Available: <https://www.theatlantic.com/health/archive/2015/09/americas-cricket-farmers/406843/>.
- [12] Zaraska, M., “Lab-grown meat is in your future, and it may be healthier than the real stuff,” *The Washington Post* Available: [https://www.washingtonpost.com/national/health-science/lab-grown-meat-is-in-your-future-and-it-may-be-healthier-than-the-real-stuff/2016/05/02/aa893f34-e630-11e5-a6f3-21ccdbc5f74e\\_story.html](https://www.washingtonpost.com/national/health-science/lab-grown-meat-is-in-your-future-and-it-may-be-healthier-than-the-real-stuff/2016/05/02/aa893f34-e630-11e5-a6f3-21ccdbc5f74e_story.html).
- [13] Kilbride, B. “Best Grow Lights for Growing Vegetables Indoor,” *The Old Farmer’s Almanac*. 2019. Available: <https://www.almanac.com/best-grow-lights-growing-vegetables-indoor>

## **Vehicles: Rovers**

## I. Code

### A. Trajectory Control Parameter Specification

```
% Rover Trajectory Control Testing
% The goal of this program is to formulate basic dynamics and kinematics
% governing the motion of a rover on the lunar surface, as well as to test
% basic control strategies for commanding desired motion
clc;
% Vehicle Parameters
% (Assuming every wheel is stiff, and the rover itself is a rigid body
% these parameters will specify the physical characteristics of the rover)
m = 500; % total mass of body (kg)
m0 = m / 2;
n = 6; % number of wheels
l_r = 9.4; % rover length (m)
w_r = 5.5; % rover width (m)
h_r = 4.3; % rover height (m)
l_w = 6; % rover wheel base (m)
% Mars Spring Tire 2 parameters
r_t1 = 0.2667; % outer tire radius (m)
r_t2 = 0.2159; % inner tire radius (m)
w_t = 0.2032; % tire width (m)
m_t = 5.41; % tire mass (kg)
I_t = 0.5 * m_t * (r_t1^2 + r_t2^2); % tire moment of inertia (kg m^2)
p = 0.25 * 745.7; % individual drive motor power (Nm/s)
vmax = 13 * 1000 / 3600; % maximum forward velocity
Tmax = p / (vmax / r_t1); % maximum available applied torque (Nm)
% Lunar Parameters
g_m = 1.6; % Moon gravitational acceleration (m/s^2)
mu0 = 0.60; % Lunar regolith static friction coefficient
mu = 0.55; % Lunar regolith dynamic friction coefficient
R = 360 / 2; % desired circular path radius (m)
Fr = mu * m_t * g_m; % wheel motion resistance force (N)
Tr = r_t1 * Fr; % rolling resistance torque (Nm)
% desired wheel angle
%del = atan((l_w/2) - R*cos(alpha.data)) ./ (R*sin(alpha.data) + (w_r/2));
```

## B. Solar Array Sizing

### %% EOL Calculation

```
EOL_STAR = 783.6; % End of Life power requirement for STAR
EOL_CAP = 1000; % End of Life power requirement for CAP
EOL_SWAG = 1000; % End of Life power requirement for SWAG
EOL_BTB = 25620; % End of Life power requirement for BTB
```

### %% BOL Calculation

```
BOL_STAR = EOL_STAR / ((1-0.005)^15); % Beginning of Life power requirement for STAR
BOL_CAP = EOL_CAP / ((1-0.005)^15); % Beginning of Life power requirement for CAP
BOL_SWAG = EOL_SWAG / ((1-0.005)^15); % Beginning of Life power requirement for SWAG
BOL_BTB = EOL_BTB / ((1-0.005)^15); % Beginning of Life power requirement for BTB
```

### %% Length and Width Calculation

```
total_area_STAR = BOL_STAR/300; % Total solar array area for STAR
total_area_CAP = BOL_CAP/300; % Total solar array area for CAP
total_area_SWAG = BOL_SWAG/300; % Total solar array area for SWAG
total_area_BTB = BOL_BTB/300; % Total solar array area for collection rover
```

```
STAR_per_array = total_area_STAR / 2; % Area of each array for STAR
CAP_per_array = total_area_CAP / 2; % Area of each array for the CAP
SWAG_per_array = total_area_SWAG / 2; % Area of each array for SWAG
BTB_per_array = total_area_BTB / 2; % Area of each array for BTB
```

```
width_STAR = sqrt(STAR_per_array/3); % Width of each array for STAR
width_CAP = sqrt(CAP_per_array/3); % Width of each array for CAP
width_SWAG = sqrt(SWAG_per_array/3); % Width of each array for SWAG
width_BTB = sqrt(BTB_per_array/3); % Width of each array for BTB
```

```
length_STAR = width_STAR * 3; % Length of each array for STAR
length_CAP = width_CAP * 3; % Length of each array for CAP
length_SWAG = width_SWAG * 3; % Length of each array for SWAG
length_BTB = width_BTB * 3; % Length of each array for the collection rover
```

### %% Output

```
fprintf('\nSTAR will need a BOL power generation of %f W\n', BOL_STAR)
fprintf('CAP will need a BOL power generation of %f W\n', BOL_CAP)
fprintf('SWAG will need a BOL power generation of %f W\n', BOL_SWAG)
fprintf('BTB will need a BOL power generation of %f W\n', BOL_BTB)
fprintf('Assuming all solar arrays are the same size, STAR will need 3 solar arrays at %f x %f meters\n', width_STAR, length_STAR)
fprintf('Assuming all solar arrays are the same size, CAP will need 3 solar arrays at %f x %f meters\n', width_CAP, length_CAP)
fprintf('Assuming all solar arrays are the same size, SWAG will need 3 solar arrays at %f x %f meters\n', width_SWAG, length_SWAG)
```

```
fprintf('Assuming all solar arrays are the same size, BTB will need 3 solar arrays at %f x %f  
meters\n\n', width_BTB, length_BTB)
```

### C. Edge Detection Algorithm

The following code performs canny edge detection on an image. It was adapted from Ref [8].

```
#Import necessary libraries
from PIL import Image
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

#Import image from link
im = get_image_from_url('image_link','test_img.png')

#Return just sobel filter of image
def sobel_filter(im):
    G_x = np.array([[1,0,-1],[2,0,-2],[1,0,-1]])#G matrix x
    G_y = np.array([[1,2,1],[0,0,0],[-1,-2,-1]]) #G matrix y
    im_proc_x = convolve_image(im,G_x) #convolution image x
    im_proc_y = convolve_image(im,G_y) #convolution image y
    # Initialize arrays
    mag = np.zeros(im_proc_x.shape,dtype=np.uint8)
    theta = np.zeros(im_proc_x.shape,dtype=np.float32)
    mag = np.sqrt(im_proc_x**2+im_proc_y**2)
    theta = np.arctan2(im_proc_y,im_proc_x)
    im_gb5x5 = blur_box_filter_5x5(np_im_gray)
    theta,im_proc_x, im_proc_y, sobel = sobel_filter(im_gb5x5) #get sobel filter image
    return sobel

#Final canny edge detection function
def canny(image):
    bw_image = cv2.cvtColor(lane_image, cv2.COLOR_RGB2GRAY) # Convert to grayscale
    blur_image = cv2.GaussianBlur(bw_image,(7,7),0) #Blur
    canny_detection = cv2.Canny(blur_image,1,200) # use openCV canny to return parameters
    return canny_detection

im_5x5 = blur_box_filter(np_im_gray)
sobel = sobel_filter(im_5x5) #get sobel filter image
canny_image = canny(lane_image) #Get canny image

cv2_imshow(canny_image) # Show Canny Edge Detection Image
cv2_imshow(sobel) # show Sobel Edge Detection Image
```

## D. Clearing Rover Power and Sizing

### Power\_analysis.m

```

rot = 50:10:200; v = 0:0.005:0.06; m = 585; V = 0.1; Rot = 120;
speed = zeros(length(v),length(rot)); A = 90; L = 1.5;
for i = 1:length(rot)
    speed(:,i) = rover_speed(v,rot(i)); %columns = varying v, rows = varying rot
end
unrestricted_speed = rover_speed(0.05,rot);
time = ((A/L)./unrestricted_speed).*1.2; %1.2 for turning losses
[power,auger_pwr] = clearing_power(m,rot);
varying_rot_P = power./1000;
figure(1)
for i = 1:length(rot)
    plot(v,speed(:,i),'DisplayName',num2str(rot(i)))
    hold on
end
title('Rover Clearing Travel Speed vs. Top Speed');
xlabel('Top Speed (m/s)'); ylabel('Clearing Travel Speed (m/s)')
grid on; legend;
%Lines of varying auger rotation.
figure(2)
plot(rot,varying_rot_P,'DisplayName','Total Rover Power'); hold on;
plot(rot,auger_pwr./1000,'DisplayName','Auger Power');
grid on; legend; xlabel('Auger Rotation (RPM)');
ylabel('Total Rover Power (kW)'); title('Power vs. Auger Speed');
% using speed allowed by auger, power requirements are already high.
% Thus, more evidence to limit top speed below what auger
% allows.
figure(3)
for i = 1:length(rot)
    plot(rot,time,'DisplayName',num2str(rot(i)))
    hold on
end
grid on; title('Clearing Time vs. Auger Rotation'); xlabel('Auger Rotation (RPM)');
ylabel('Clearing Time (s)');
% This shows that auger speed has diminishing returns thus, limiting auger
% speed will not significantly impact clearing efficiency but will help
% more with power.
figure(4)
plot(rot,auger_pwr./1000); grid on; title('Auger Power vs. Auger Rotation')
xlabel('Auger Rotation (RPM)'); ylabel('Auger Power');

```

### clearing\_power.m

```

function [power,aug_P] = clearing_power(m,rot)
hp2W = 746; % horsepower to watts
L = 3; % 3m auger
%opt_rot = 120; %rev per min, optimal via src

```

```

rho = 1500; % kg/m^3 lunar regolith density
bay_vol = (1.5*1.75*2)/2; % m^3 half rover bay volume
g = 9.81/6;
c = 0.4; % rolling resistance coeff from engineering toolbox

v = rover_speed(1000,rot);
aug_P = 0.45*(L/3.048)*(rot/100)*hp2W;
plow_P = bay_vol*rho*g*v;
roll_P = m*g*c*v;
power = aug_P+plow_P+roll_P;
end

```

**clearing\_aug\_rate.m**

```

function consumption = clearing_aug_rate(rot)
Bu2m = 0.03539606; %cubic meters per bushel
consumption_data = 1700; %bushels per hour
rpm_data = 1750; % rev per min
%opt_rot = 120; %rev per min, optimal via src
aug_flux = (consumption_data*Bu2m)/(rpm_data*60); %cubic meters per rev
consumption = aug_flux*rot; %cubic meters per min
end

```

**rover\_speed.m**

```

function speed = rover_speed(v,rot)
L = 3; r = 0.254/2;
aug_rate = clearing_aug_rate(rot);
aug_vol = pi*r^2*L;
augShaft_clear_t = (aug_vol./aug_rate).*60;
fwd_speed_aug = r./augShaft_clear_t;

speed = min(fwd_speed_aug,v);
end

```

## E. Scouting rover range, endurance and power estimation (Ajay Chandra)

```

%% Ajay Chandra, scouting rover range, endurance and power estimation
%% define constants
m = 551.47; %kg, total scouting rover mass
g = 1.62; %m/sec2, gravitational acceleration on lunar surface
c = 0.4; % rolling resistance of tires on lunar dust
batt_capacity = 6; %kWh, battery capacity
r_wheel = 0.29; %m, radius of wheel

spd = [0.02:0.001:0.8]; %m/sec rover speed
resistive_pwr = 4.*((m .*g .*c ./r_wheel) .* spd)./(1000.*0.84); %rover s

%% Instrument Power Consumption Value in kW
p_spectrometer = 0.75; %spectrometer
p_LiDAR = 0.0179; %LiDAR imager
p_GPR = 0.01; %Ground Penetrating Radar
p_cam = 0.0179;
% Commented out since these will not be used while the rover is at higher
% speeds. Uncomment each line as required to view impact of each equipment
%p_drill = 0.01;
%p_arm = 1;
%p_batt_cool = 0.1;

endurance = batt_capacity./(resistive_pwr + p_spectrometer + p_LiDAR...
    + p_GPR + p_cam + p_drill + p_arm + p_batt_cool); % hours, endurance
R = (endurance).* (spd.*0.001.*3600).*0.5; %km, range

figure(1);
plot(spd,resistive_pwr);
xlabel('Top speed (m/sec)')
ylabel('Power Required (kW)');
title('Power Consumption as a function of Rover Speed - Ajay Chandra');
xlim([spd(1) spd(end)])
grid on;

figure(2);
plot(spd,endurance);
xlabel('Top speed (m/sec)')
ylabel('Battery Life (h)');
title('Endurance as a function of Rover Speed - Ajay Chandra');
grid on;
xlim([spd(1) spd(end)])
figure(3);
plot(spd,R);
xlabel('Top Speed (m/sec)')
ylabel('Range (m)');
title('Maximum Distance Traversable from base as a function of top speed - Ajay Chandra');
grid on;
xlim([spd(1) spd(end)])

```

## II. Initial Design Logic/Concepts

### A. Clearing Rover

Initial research to determine the best suited vehicle brought forward many options seen in industrial mining operations for materials like sand and gravel. However, the properties and characteristics of materials on earth are not perfectly similar to those of Lunar regolith so the properties and behavior of regolith must be researched.

One of the most important factors is particle size. The average lunar regolith particle is approximately 0.07mm, which is described as sandy-silt or silty-sand, somewhere between the two. Next, in terms of density and distribution it is described as being “amazingly consistent to a depth of at least several meters” with probability of 0.05 for hitting a rock at 1m depth [35]. Additionally, the low lunar gravity allows for 3m deep trenches to be dug without the walls collapsing [35][36]. For consistency, the regolith is described as behaving like wet beach sand due to mechanical interlocking of irregular particle surfaces. This same property means that packed regolith has rather high shear strength. At the time of the referenced material, exact effects of ice at the poles were unknown but speculated to include diminishing the shear strength property and increasing the coarseness of particles.

#### *1. Initial Design Ideas*

Early designs of the clearing rover mimicked what is now known as the collecting rover. As aforementioned, the initial inspiration was derived from modern mining operations. Systems like slushers, drag buckets, and simple excavators were considered. However, it was determined that collecting excess mass and requiring emptying of the systems consumed too much excess

power for the purpose of clearing an area. Rather, it is more efficient to just push the looser regolith on the surface away from the designated area, creating berms around the cleared area. The berms could later be collected for other purposes like shielding the habitat from radiation, but strictly for clearing, the collection of the surface regolith is not necessary.

Even with these requirements, a system like a slusher could still be feasible, however, it is limited in the shapes and sizes of areas that can be cleared. So, a new requirement is that the system must be versatile and capable of clearing various sized and shaped areas. Due to the other rovers being created on small, standardized platforms, it was determined that using the same platform with specialized attachments is most effective and efficient for the clearing process.

With the basic process and conceptual design of the rover established, the actual attributes that determine its clearing abilities must be designed. Specifically, understanding how efficiently the auger will clear regolith is essential. A very similar process to moving dry, loose regolith on

Earth is seen in the process of emptying agricultural grain bins. This process uses a sweeping, horizontal auger to drive grain towards the center of the bin where an opening leads to another auger that sends the grain up a chute and out of the bin. There is a lot of data published regarding the consumption specifications for these augers at given operating conditions, so, these specifications were used to create a mathematical model of the auger. With this mathematical model, we can get an informative rate of volume consumed per revolution. This rate is helpful because it is independent of the rotation speed, which will likely be different on the Moon than it is on Earth due to power requirements and the different materials being cleared. It is assumed that extrapolating from this model is reasonable given the simplistic relationships between the various metrics. Additionally, certain specifications like the auger diameter and twist are designed to be

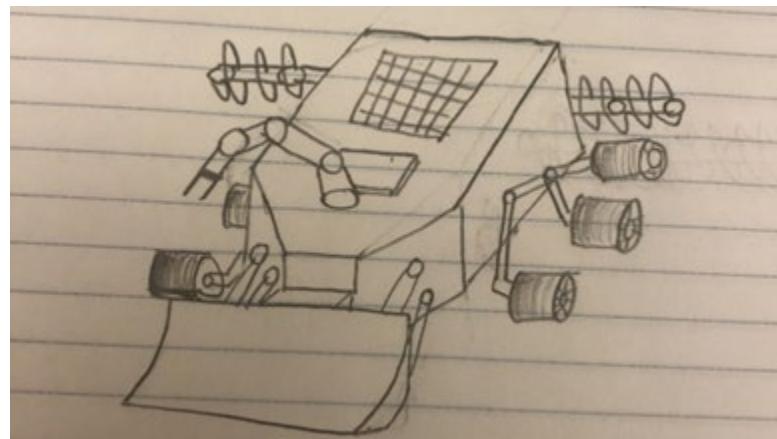
exactly the same on the rover as on the auger used to develop the model, which is 10 inches in diameter.

The conversion used to convert the given consumption rate in bushels per hour to cubic meters per revolution is shown in the dimensional analysis of the equation below:

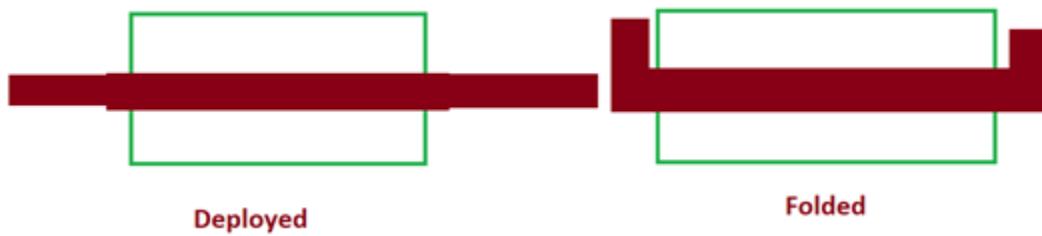
$$\left[ \frac{m^3}{rev} \right] = \frac{Bu}{hr} * \frac{1hr}{60min} * \frac{1}{rev/min} * \frac{m^3}{Bu}$$

In this formula, values for bushels per hour and revolutions per minute are taken from the auger specifications and the other terms are known conversions.

## 2. Initial Design Concepts



**Fig 1: General Clearing Rover Design**



**Fig 2: Deployable Auger Concept to fit into 1U Space**

## B. Collection Rover

### 1. Initial Design Pictures

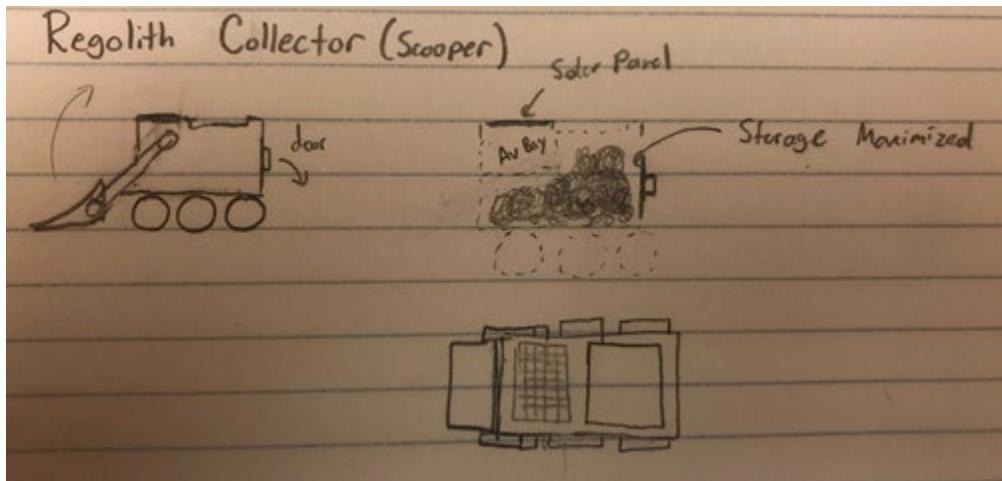


Fig 3: Scooper Rover Initial Concept Design

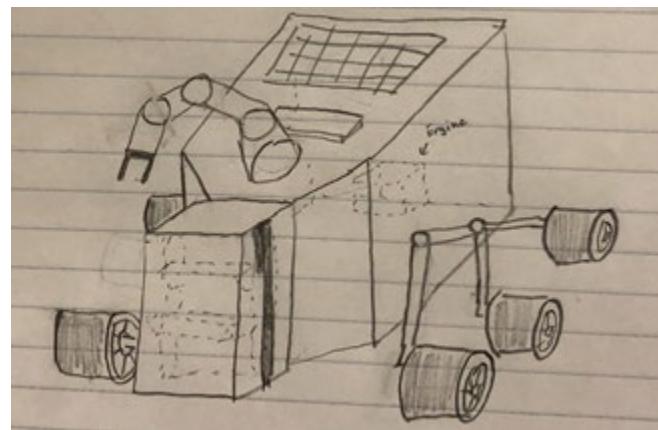
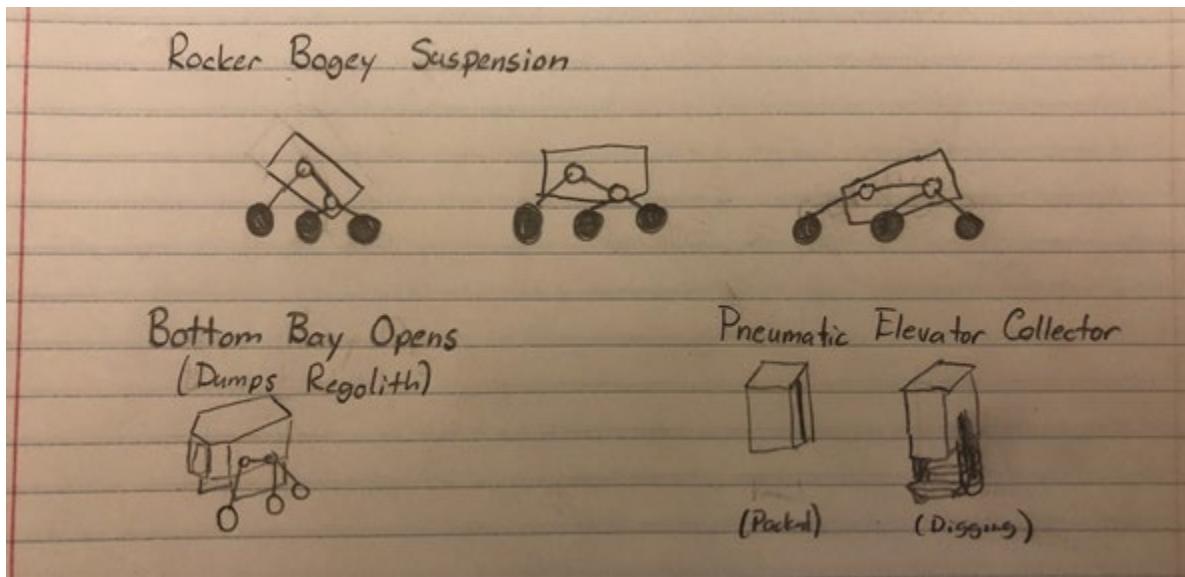


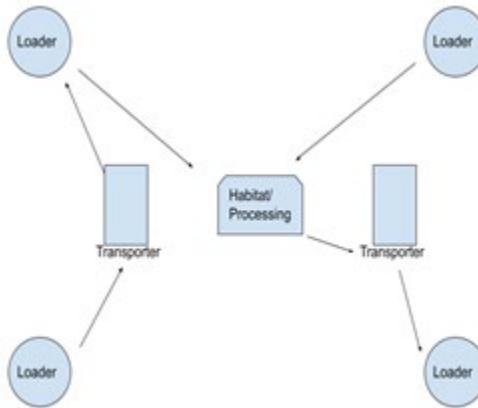
Fig 4: General Collection Rover Design



**Fig 5: Collection Rover Design Implementations**

## 2. Collection Process

For the collection, we are proposing a 3 or 4 step process which includes excavation, transportation, processing, and fragmentation if the regolith is too densely packed on the surface [37]. There are three styles of proposed systems to execute this process. Firstly, a standard loader/hauler system where there are separate vehicles for excavation and transportation. Excavation vehicles or machines would operate at off-site locations while transporters collect the excavated material and bring it to base to be processed. A basic layout of this is seen in the figure below:

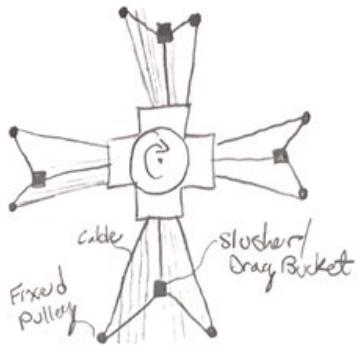


**Fig. 6: Loader-hauler mining layout**

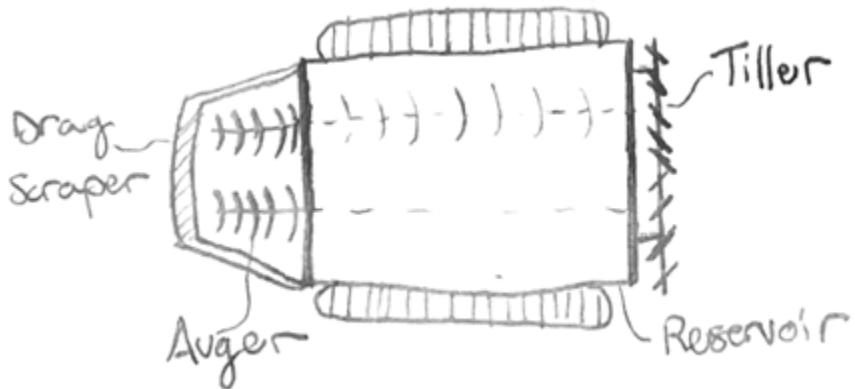
Next is an integrated loader/hauler system where there is one vehicle for excavation and transportation, and finally a conveyor system would cut out the transporter vehicles and continuously transport material from excavation sites.

The most refined designs of vehicles to accomplish these systems are shown in two drawings here. In Fig. 7, we see a loader machine where slushing buckets are dragged using pulleys, cables, and winches to excavate strips of regolith which is collected in the center hub.

Transporter vehicles would empty the hubs and the systems would have to be relocated once the entire area is excavated. In Fig. 8 we see a rover-style loader vehicle, which could operate as a loader or integrated loader hauler. Here, an optional tiller would loosen regolith before being concentrated by the drag bucket and loaded onto the vehicle by a series of augers.



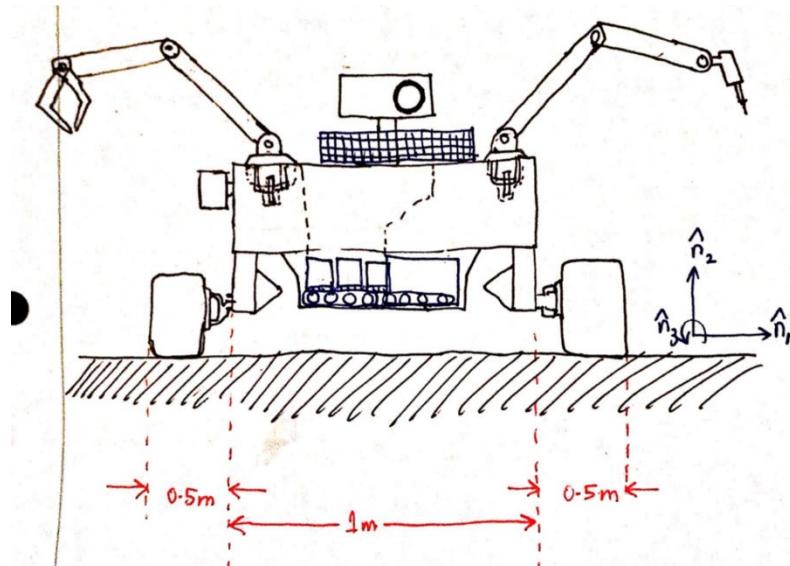
**Fig. 7: Slusher mining system**



**Fig. 8: Integrated loader-hauler vehicle**

### C. Scouting Rover

Shown below in Fig. 9 is a preliminary design of the scouting rover. Note that although the arms are designed to have multiple tools attached to them at once, only one is drawn on each for representation purposes. The tools featured on this design are a LiDAR imager, two ERAs, a claw and fluted sampler drill, sample canisters, a spectrometer, a beacon storage bay, rocker bogie suspension, and solar panels.



**Fig. 9: Scouting Rover Preliminary Sketch (Front View)**

This preliminary design provides a ground clearance of 0.30 m, inclusive of the sample canisters and spectrometer in the floor section.

A subsequent design iteration concluded the need for more surface area on the upper surface of the rover to fit the required size of solar panels and provide enough room for sample placement.

Also, to conserve the ground clearance and add the ground penetration radar, we decide to create a ‘false floor’ for the sample canisters and spectrometer to rest on. The ground penetration radar is fitted below this false floor.

## D. Mining Rover

### 1. Initial Design and Concepts

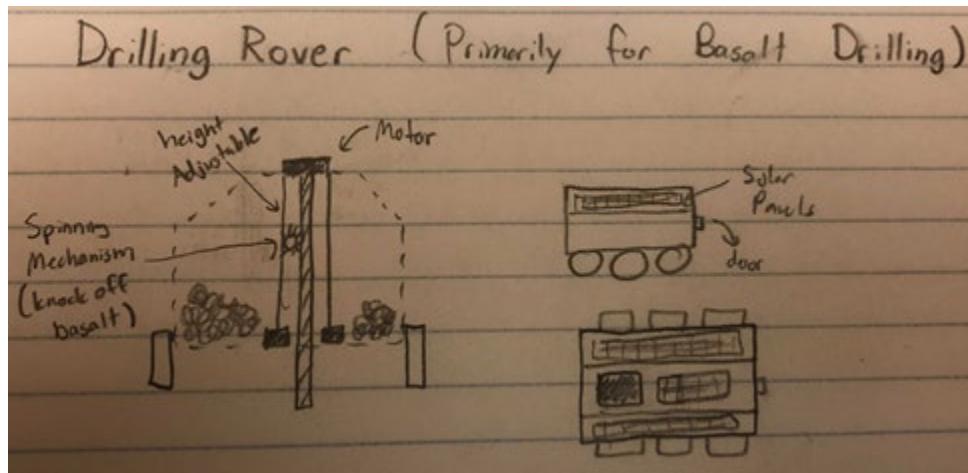


Fig 10: Mining Rover Initial Concept Design

| Drill       | Apollo Drill                   | Mining Rover | Reasoning                                                    |
|-------------|--------------------------------|--------------|--------------------------------------------------------------|
| Drill Bit   | Steel                          | Carbide      | Apollo Missions had trouble drilling through rocks           |
| Drill Stem  | Titanium                       | Titanium     | No Issues found with Drill Stems                             |
| Fluting     | Not continuous until Apollo 16 | Continuous   | Continuous fluting prevent bit binding from regolith jamming |
| Augur Angle | Low                            | Low          | Enables Self Propelled Drilling, Prevents choking            |
| Spin Speed  | 280 rpm                        | >160 rpm     | Prevents Choking, High speeds are safer for control          |

Fig 11: Drill Design Improvement over Apollo Missions Chart

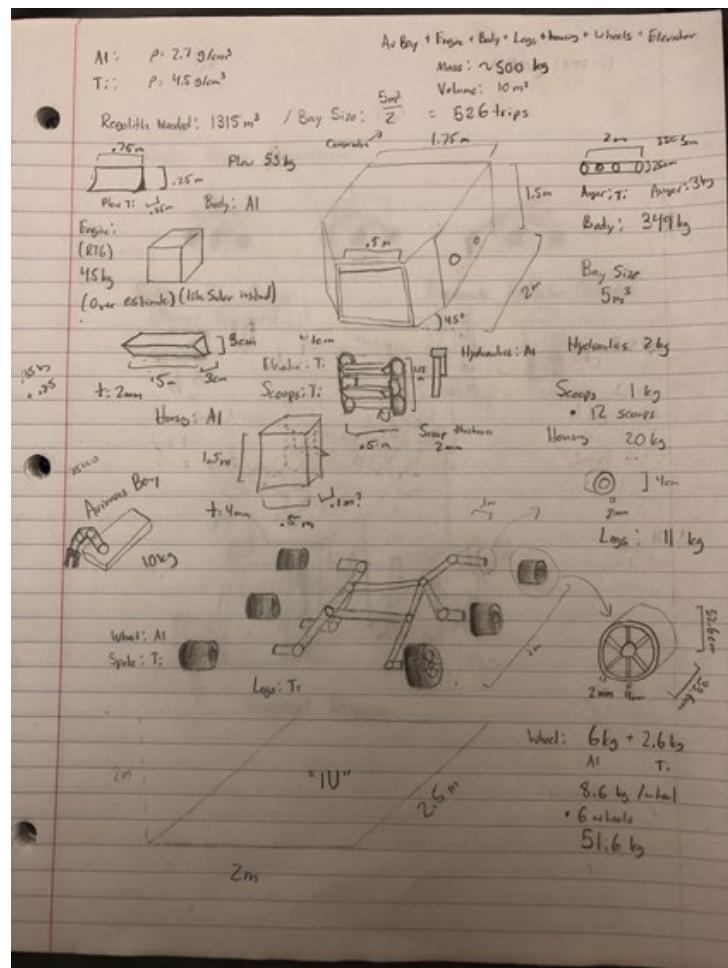
| Soil Composition (%)           | Lunar Highlands (Apollo 16) | Hawaiian Basalt (1981 Study) |
|--------------------------------|-----------------------------|------------------------------|
| SiO <sub>2</sub>               | 45                          | 46.4                         |
| TiO <sub>2</sub>               | 0.5                         | 2.4                          |
| Al <sub>2</sub> O <sub>3</sub> | 27.2                        | 14.2                         |
| Fe <sub>2</sub> O <sub>3</sub> | 0                           | 4.1                          |
| FeO                            | 5.2                         | 8.9                          |
| MgO                            | 5.7                         | 9.5                          |
| CaO                            | 15.7                        | 10.3                         |
| Total                          | 99.3                        | 95.8                         |

Fig 12: Lunar Soil Composition vs Hawaiian Basalt

| Soil Composition (%)           | Low Ti Mare Soil (Apollo 12) | High Ti Mare Soil (Apollo 11) |
|--------------------------------|------------------------------|-------------------------------|
| SiO <sub>2</sub>               | 46.4                         | 42                            |
| TiO <sub>2</sub>               | 2.7                          | 7.5                           |
| Al <sub>2</sub> O <sub>3</sub> | 13.5                         | 13.9                          |
| Fe <sub>2</sub> O <sub>3</sub> | 0                            | 0                             |
| FeO                            | 15.5                         | 15.7                          |
| MgO                            | 9.7                          | 7.9                           |
| CaO                            | 10.5                         | 12                            |
| Total                          | 98.3                         | 99                            |

**Fig 13: Mare Soil Composition**

### E. General Rover Design



**Fig 14: Volume Analysis Drawing**

## F. External Robotic Appendage (ERA) (Ajay Chandra)

The need for a robotic arm in each of our rover systems has been elaborated on under the Vehicles: Rovers section in the Final Report document, in addition to this appendix. Therefore, we institute the External Robotic Appendage (ERA). Regardless of whether we use it for sample collection or rover re-stabilization, the following design requirements hold:

### 1. Adaptability:

Due to the various requirements of each rover, the ‘hand’ end of the arm must be able to accommodate multiple tools. In addition, the arms must be replaceable by humans when they arrive at the colony.

### 2. Performance:

Lunar dust is to be kept out of the joints, as this will hinder performance.

### 3. Precision:

The arm must be able to pick up and place objects 5 meters across.

### 4. Autonomy:

The arm must be able to accept input from control systems on all rovers and be able to carry out pre-programmed routines. If an anomaly occurs, an error report is to be generated and the user (at the lunar base or on Earth) is to be notified.

## 5. Power:

The arm must route power from the main battery to the individual tools at the end of the arm. In addition, the arm itself must draw power from the rover's battery.

# III. Design Decisions

## A. Motor Choice

Due to the year-long period between commencement of rover operations and the arrival of humans, reliability is a high priority. In addition, precision is required to accurately maneuver and place navigation beacons and core samples. Therefore, the motor of choice for is the RE 25 Ø25 mm brushed DC motor obtained from the Maxon Group, Switzerland.

## B. Sizing

The rover arm spans 1.54 meters when fully extended, and is capable of recovering a rover from a tipped state provided that the arm itself has not been damaged. In addition, this span is sufficient to reach the ground from the top of the rover, and therefore allows for placement of objects on the ground as well as sample collection.

## C. Sensors

To allow for autonomy, infrared sensors are required to judge the distance to the target object. Also, a gyroscopic sensor is required to maintain attitude knowledge and determine future movements. In case manual operation is required, a LiDAR camera is present on the end of the

robotic arm. This camera, in tandem with the camera mounted to the rover itself will be used to aid the operator.

#### D. Tools

The scouting and mining rovers each require the ability to pick up and place objects on the ground. Object sizes range from as narrow as 5 cm to as wide as 40 cm. For this purpose, a claw is designed, shown in the figure below:



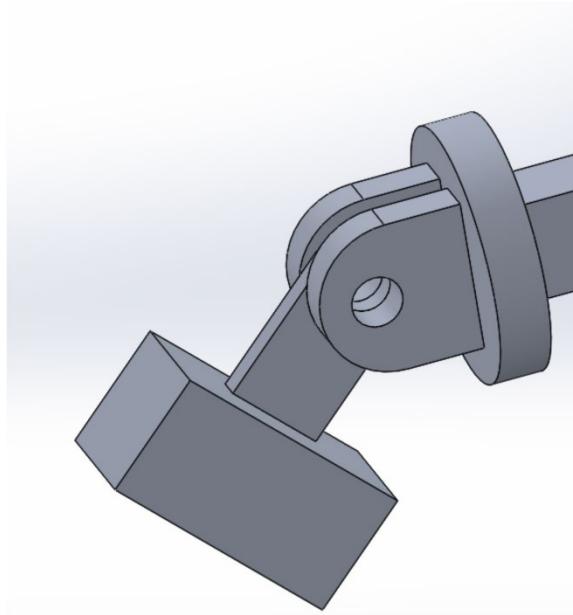
**Fig. 15 Robotic claw (created by Vishank Battar)**

In addition, a fluted sampling drill is required for the scouting rover to collect samples. This is pictured below in figure below:



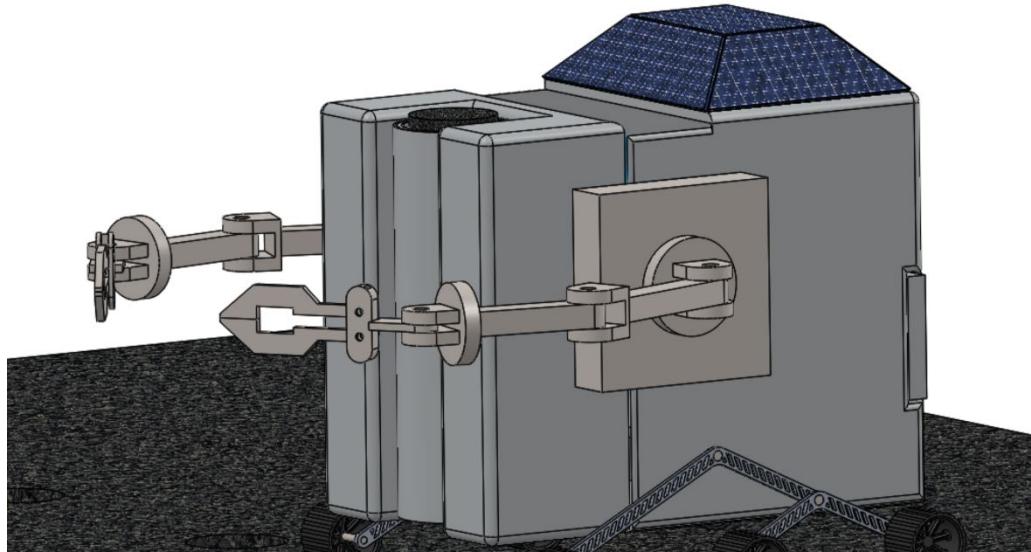
**Fig. 16 Fluted Sampling Drill (Created by Vishank Battar)**

To assist in recovering a rover from a tipped state without risking any other tools on the arm, a stabilization block is designed, pictured in the figure below:



**Fig. 17 Stabilization block (created by Vishank Battar)**

Shown below in the figures below are the ERA's mounted to the mining and scouting rover respectively.



**Fig. 18 ERA used on the mining rover (created by Vishank Battar)**



**Fig. 19 ERA used on the scouting rover (Vishank Battar)**

The table below summarizes the mass, power, volume and cost of the ERA. We note that the total mass comes up to 409.039 kg, consumes 205.8 W of power, and has a net monetary value of 25874.05 USD.

**Table 1: Mass, Power, Volume, and Cost summary of ERA**

| Component | Mass (kg) | Power (kW) | Volume (m <sup>3</sup> ) | Quantity | Cost (\$) |
|-----------|-----------|------------|--------------------------|----------|-----------|
| DC motor  | 0.115     | 0.02       | 2.11E-05                 | 7        | 239.15    |
| IR sensor | 0.117     | 0.006      | 9.72E-06                 | 2        | 200       |

|                      |                   |                  |          |   |                    |
|----------------------|-------------------|------------------|----------|---|--------------------|
| Gyroscopic Sensor    | 117               | 0.006            | 9.72E-06 | 3 | 700                |
| Camera               | 10.4              | 0.0179           | 0.017328 | 2 | 4850               |
| Mid-section Assembly | 12.9              | 0                | 0.013    | 1 | 6000               |
| Base Assembly        | 23.3              | 0                | 0.02     | 1 | 6000               |
| <b>Total</b>         | <b>409.039 kg</b> | <b>0.2058 kW</b> |          |   | <b>25874.05 \$</b> |

The flowchart in the figure below demonstrates an overview of the ERA's control software. Given a pre-programmed routine (from either the lunar colony or the ground station on Earth), the arm will operate while checking periodically for anomalies. If an anomaly is detected, an error report is sent to the operator with a timestamp and the exact sensors detecting the anomaly. The operator can then choose to override and continue operations, or assume manual control of the arm as the situation demands.

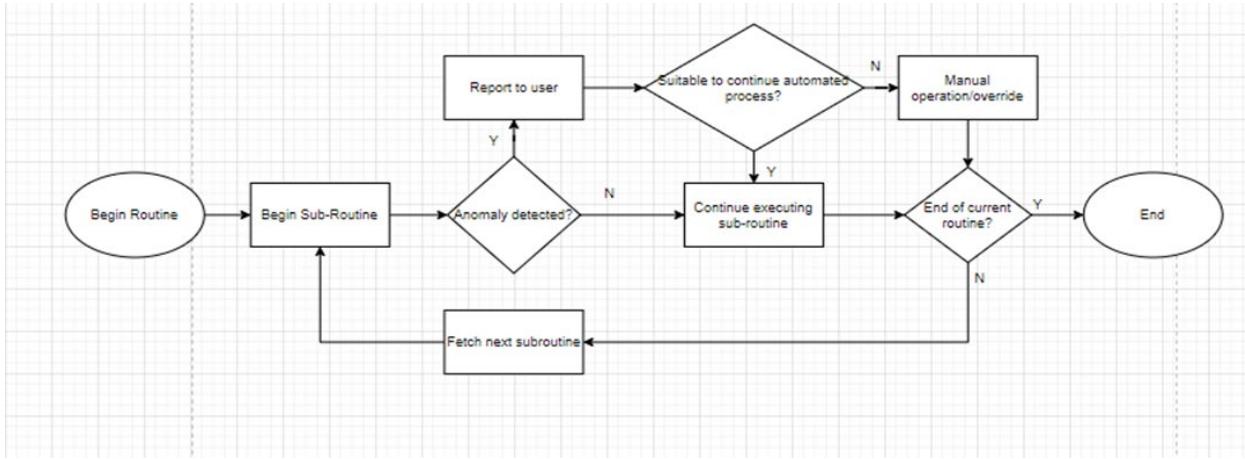


Fig. 20 Autonomy overview for ERA

## IV. Rover Risk/Fault Analysis

Risk analysis was performed on all of the rovers to the following likelihood x consequence risk assessment system. The scale of overall risk severity goes from 1 to 25, with 1 being the least serious and 25 being the most serious. This analysis was performed by the entire rover team.

### A. Mining Rover

**Table 2: Risk/Fault Analysis for Mining Rover Pre-mitigation**

| Ref ID       | Intermediate Event      | Ref ID  | Basic Event                                      | Likelihood | Consequence | Overall |
|--------------|-------------------------|---------|--------------------------------------------------|------------|-------------|---------|
| MINING ROVER |                         |         |                                                  |            |             |         |
| VRM-1        | Drive-train Failure     | VRM-1.1 | broken steering linkage                          | 3          | 5           | 15      |
|              |                         | VRM-1.2 | dust in motor                                    | 1          | 5           | 5       |
|              |                         | VRM-1.3 | loss of motor power                              | 2          | 5           | 10      |
| VRM-2        | Drill failure           | VRM-2.1 | DC motor voltage spike                           | 1          | 1           | 1       |
|              |                         | VRM-2.2 | Dust jams drill drive gears                      | 2          | 5           | 10      |
|              |                         | VRM-2.3 | Drill breaks from overuse                        | 3          | 3.5         | 10.5    |
|              |                         | VRM-2.4 | Drill overheats and melts                        | 1          | 3.5         | 3.5     |
| VRM-3        | Storage failure         | VRM-3.1 | Blockage in storage bay prevents further loading | 3          | 2           | 6       |
|              |                         | VRM-3.2 | ERAs fail to maneuver core into bay              | 2          | 1           | 2       |
|              |                         | VRM-3.3 | storage bay is overfilled                        | 2          | 1           | 2       |
|              |                         | VRM-3.4 | storage bay is compromised by impact             | 1          | 4           | 4       |
| VRM-4        | Instrumentation failure | VRM-4.1 | Imaging/camera failure                           | 1          | 2.5         | 2.5     |
|              |                         | VRM-4.2 | Loss of attitude knowledge                       | 1          | 2           | 2       |
|              |                         | VRM-4.4 | Loss of position knowledge                       | 2          | 2           | 4       |
| VRM-5        | Battery Failure         | VRM-5.1 | Voltage surge due to solar flare                 | 2          | 3.5         | 7       |
|              |                         | VRM-5.2 | Battery overheats                                | 1          | 2           | 2       |
|              |                         | VRM-5.3 | Battery discharges completely                    | 2          | 2.8         | 5.6     |
|              |                         | VRM-5.4 | Electrical connection loss                       | 1          | 3           | 3       |
|              |                         | VRM-5.5 |                                                  |            |             | 0       |
| VRM-6        | Structural Failure      | VRM-6.1 | Failure of floor of storage bay                  | 1          | 3.8         | 3.8     |
|              |                         | VRM-6.2 | Suspension failure due to overload               | 2          | 2.6         | 5.2     |
|              |                         | VRM-6.3 | Wheel failure due to overload                    | 2          | 2.2         | 4.4     |
|              |                         | VRM-6.4 | ERA failure due to overload                      | 2          | 3.4         | 6.8     |
| VRM-7        | Suspension Failure      | VRM-7.1 | Jammed suspension                                | 4          | 2           | 8       |
| VRM-8        | Loss of Stability       | VRM-8.1 | Tipping/falling                                  | 3          | 4           | 12      |
|              |                         | VRM-8.2 | Loss of ERA attitude knowledge                   | 2          | 1.2         | 2.4     |
|              |                         | VRM-8.3 | Loss of vehicle attitude knowledge               | 2          | 1.2         | 2.4     |

Of note in this original analysis is that the areas of drive train and drill operation pose a significant risk, due to the sensitivity of some of their components, the difficult conditions, and the possible difficulty of repairing any issues. There is also some concern over MARCY's stability, as the storage of heavy basalt cores has to be performed carefully to avoid an unstable vehicle configuration.

**Table 3: Risk/Fault Analysis for Mining Rover Post-Mitigation**

| Ref ID              | Intermediate Event      | Ref ID  | Mitigation/Warning                                                                    | Post-Mitigation |             |     |
|---------------------|-------------------------|---------|---------------------------------------------------------------------------------------|-----------------|-------------|-----|
|                     |                         |         |                                                                                       | Likelihood      | Consequence |     |
| <b>MINING ROVER</b> |                         |         |                                                                                       |                 |             |     |
| VRM-1               | Drive-train Failure     | VRM-1.1 | Better surface surveillance                                                           | 1               | 5           | 5   |
|                     |                         | VRM-1.2 | Cover motor                                                                           | 1               | 5           | 5   |
|                     |                         | VRM-1.3 | backup power                                                                          | 1               | 5           | 5   |
| VRM-2               | Drill failure           | VRM-2.1 | surge protector                                                                       | 1               | 1           | 1   |
|                     |                         | VRM-2.2 | Cover gears                                                                           | 1               | 5           | 5   |
|                     |                         | VRM-2.3 | Extra drill tips/ reinforce drills                                                    | 1               | 3.5         | 3.5 |
|                     |                         | VRM-2.4 | Temperature sensors, stop drilling when too hot                                       | 1               | 3.5         | 3.5 |
| VRM-3               | Storage failure         | VRM-3.1 | Use ERAs to unblock storage bay, don't load when storage bay reaches maximum capacity | 2               | 2           | 4   |
|                     |                         | VRM-3.2 | Better controls code                                                                  | 1               | 1           | 1   |
|                     |                         | VRM-3.3 | Prevent this from happening in code with simple if statement                          | 1               | 1           | 1   |
|                     |                         | VRM-3.4 | reinforce storage bay with titanium lining                                            | 1               | 3           | 3   |
| VRM-4               | Instrumentation failure | VRM-4.1 | Replacement camera parts kept at base                                                 | 1               | 2           | 2   |
|                     |                         | VRM-4.2 | Backup information based on last known attitude                                       | 1               | 2           | 2   |
|                     |                         | VRM-4.4 | Backup information based on last known location                                       | 2               | 2           | 4   |
| VRM-5               | Battery Failure         | VRM-5.1 | Solar ray protection for electronics                                                  | 1               | 3.5         | 3.5 |
|                     |                         | VRM-5.2 | Temperature warning for high heat in battery area                                     | 1               | 2           | 2   |
|                     |                         | VRM-5.3 | Warning for irregular battery discharge rates                                         | 1               | 2.8         | 2.8 |
|                     |                         | VRM-5.4 | Warning when power lost                                                               | 1               | 3           | 3   |
|                     |                         | VRM-5.5 |                                                                                       |                 |             | 0   |
| VRM-6               | Structural Failure      | VRM-6.1 | Reinforce storage bay floor/have replacement floor on hand                            | 1               | 2.5         | 2.5 |
|                     |                         | VRM-6.2 | Clean out storage bay material accumulation regularly                                 | 1               | 2.6         | 2.6 |
|                     |                         | VRM-6.3 | Clean out storage bay material accumulation regularly                                 | 1               | 2.2         | 2.2 |
|                     |                         | VRM-6.4 | Clean out storage bay material accumulation regularly                                 | 1               | 3.4         | 3.4 |
| VRM-7               | Suspension Failure      | VRM-7.1 | Object detection for suspension clearing issues                                       | 2               | 2           | 4   |
| VRM-8               | Loss of Stability       | VRM-8.1 | Onboard emergency programming for arm usage to assist rover                           | 2               | 2           | 4   |
|                     |                         | VRM-8.2 | Backup information based on last known attitude                                       | 2               | 1.2         | 2.4 |
|                     |                         | VRM-8.3 | Backup information based on last known attitude                                       | 2               | 1.2         | 2.4 |

## B. Clearing Rover

**Table 4: Risk/Fault Analysis for Clearing Rover Pre-Mitigation**

| Ref ID         | Intermediate Event  | Ref ID   | Basic Event                   | Likelihood | Consequence | Overall |
|----------------|---------------------|----------|-------------------------------|------------|-------------|---------|
| CLEARING ROVER |                     |          |                               |            |             |         |
| VRC-1          | Drive Train Failure | VRC-1.1  | Broken Steering Linkage       | 3          | 4           | 12      |
|                |                     | VRC-1.2  | Dust in Motor                 | 1          | 1           | 1       |
|                |                     | VRC-1.3  | Loss of Motor Power           | 2          | 5           | 10      |
| VRC-2          | Body Failure        | VRC-2.1  | Body Buckles                  | 1          | 4           | 4       |
|                |                     | VRC-2.2  | Micrometeoroid Impacts        | 4          | 2           | 8       |
| VRC-3          | Instrument Failure  | VRC-3.1  | GPS Position Loss             | 2          | 2           | 4       |
|                |                     | VRC-3.2  | Solar Flare Interference      | 1          | 3           | 3       |
|                |                     | VRC-3.3  | Solar Wind Circuit Ionization | 2          | 3           | 6       |
| VRC-4          | Battery Failure     | VRC-4.1  | Battery Explosion             | 2          | 5           | 10      |
|                |                     | VRC-4.2  | Battery Recharge Failure      | 2          | 4           | 8       |
| VRC-5          | Suspension Failure  | VRC-5.1  | Fatigue Failure               | 2          | 4           | 8       |
| VRC-6          | Robotic Arm Failure | VRC-6.1  | Robotic Arm Jam               | 2          | 4           | 8       |
|                |                     | VRC-6.2  | Robotic Arm Claw Failure      | 1          | 3           | 3       |
|                |                     | VRC-7.1  | Wheel Bearing Jam             | 2          | 3           | 6       |
| VRC-7          | Wheel Failure       | VRC-7.2  | Wheel Fatigue Failure         | 1          | 4           | 4       |
|                |                     | VRC-7.3  | Spoke Failure                 | 1          | 4           | 4       |
|                |                     | VRC-8.1  | Fatigue Failure               | 1          | 5           | 5       |
| VRC-9          | Plow Failure        | VRC-9.1  | Plow Hydraulics Failure       | 2          | 5           | 10      |
| VRC-10         | Auger Failure       | VRC-10.1 | Auger Jam                     | 3          | 3           | 9       |

**Table 5: Risk/Fault Analysis for Clearing Rover Post-Mitigation**

| Ref ID         | Intermediate Event  | Ref ID   | Mitigation/Warning                                              | Post-Mitigation |   |     |
|----------------|---------------------|----------|-----------------------------------------------------------------|-----------------|---|-----|
| CLEARING ROVER |                     |          |                                                                 |                 |   |     |
| VRC-1          | Drive Train Failure | VRC-1.1  | Better surface surveillance                                     | 1               | 5 | 5   |
|                |                     | VRC-1.2  | Cover motor                                                     | 1               | 5 | 5   |
|                |                     | VRC-1.3  | backup power                                                    | 1               | 5 | 5   |
| VRC-2          | Body Failure        | VRC-2.1  | Stress testing, reinforce with titanium                         | 0.5             | 4 | 2   |
|                |                     | VRC-2.2  | Impact Testing, reinforce with titanium                         | 4               | 1 | 4   |
| VRC-3          | Instrument Failure  | VRC-3.1  | Warning sent through beacon system                              | 1               | 2 | 2   |
|                |                     | VRC-3.2  | NASA Solar Flare detection warning                              | 1               | 2 | 2   |
|                |                     | VRC-3.3  | Radiation shield around sensitive electronics                   | 1               | 3 | 3   |
| VRC-4          | Battery Failure     | VRC-4.1  | Thermally stable environment in rover and temp warning          | 1               | 4 | 4   |
|                |                     | VRC-4.2  | Warning on low power, backup circuit                            | 2               | 2 | 4   |
| VRC-5          | Suspension Failure  | VRC-5.1  | stress testing to know when to replace                          | 1               | 3 | 3   |
| VRC-6          | Robotic Arm Failure | VRC-6.1  | Frequent Lubrication and angle limits to prevent jamming        | 1               | 3 | 3   |
|                |                     | VRC-6.2  | Better control code                                             | 0.5             | 3 | 1.5 |
|                |                     | VRC-7.1  | Frequent Lubrication                                            | 1               | 3 | 3   |
| VRC-7          | Wheel Failure       | VRC-7.2  | stress testing to know when to replace                          | 1               | 3 | 3   |
|                |                     | VRC-7.3  | reinforce spokes with titanium                                  | 0.5             | 4 | 2   |
| VRC-8          | Axle Failure        | VRC-8.1  | stress testing to know when to replace                          | 1               | 3 | 3   |
| VRC-9          | Plow Failure        | VRC-9.1  | Low pressure sensor warning                                     | 1               | 5 | 5   |
| VRC-10         | Auger Failure       | VRC-10.1 | Reverse auger/kill power if jammed both ways, alert ground crew | 2               | 1 | 2   |

## C. Construction Rover

**Table 6: Risk/Fault Analysis for Construction Rover Pre-Mitigation**

| Ref ID                     | Intermediate Event      | Ref ID  | Basic Event                   | Likelihood | Consequence | Overall |
|----------------------------|-------------------------|---------|-------------------------------|------------|-------------|---------|
| CONSTRUCTION/MODULAR ROVER |                         |         |                               |            |             |         |
| VRD-1                      | Drivetrain failure      | VRD-1.1 | Dust in DC motor              | 1          | 4           | 4       |
|                            |                         | VRD-1.2 | Broken Steering Linkage       | 3          | 1           | 3       |
|                            |                         | VRD-1.3 | Loss of motor power           | 2          | 5           | 10      |
| VRD-2                      | Body Failure            | VRD-2.1 | Body Buckles                  | 1          | 4           | 4       |
|                            |                         | VRD-2.2 | Micrometeoroid impacts        | 2          | 2           | 4       |
| VRD-3                      | Instrumentation Failure | VRD-3.1 | GPS Position Loss             | 2          | 2           | 4       |
|                            |                         | VRD-3.2 | Solar Flare Interference      | 2          | 3.5         | 7       |
|                            |                         | VRD-3.3 | Solar Wind Circuit Ionization | 2          | 2           | 4       |
| VRD-4                      | Battery Failure         | VRD-4.1 | Battery Explosion             | 1          | 4           | 4       |
|                            |                         | VRD-4.2 | Battery Recharge Failure      | 1          | 3           | 3       |
| VRD-5                      | Suspension Failure      | VRD-5.1 | Fatigue Failure               | 2          | 3           | 6       |
|                            |                         | VRD-5.2 | Shock/Impact Failure          | 1          | 2           | 2       |
|                            |                         | VRD-5.3 | Detachment                    | 1          | 1           | 1       |
| VRD-6                      | Wheel Failure           | VRD-6.1 | Impact Damage                 | 1          | 5           | 5       |

**Table 7: Risk/Fault Analysis for Clearing Rover Post-Mitigation**

| Ref ID                     | Intermediate Event      | Ref ID  | Mitigation/Warning                                                                                | Post-Mitigation |
|----------------------------|-------------------------|---------|---------------------------------------------------------------------------------------------------|-----------------|
| CONSTRUCTION/MODULAR ROVER |                         |         |                                                                                                   |                 |
| VRD-1                      | Drivetrain failure      | VRD-1.1 | Cover motor                                                                                       | 1               |
|                            |                         | VRD-1.2 | Better surface surveillance                                                                       | 1               |
|                            |                         | VRD-1.3 | backup power                                                                                      | 1               |
| VRD-2                      | Body Failure            | VRD-2.1 | reinforce body with titanium lining                                                               | 1               |
|                            |                         | VRD-2.2 | extra protection of key areas                                                                     | 2               |
| VRD-3                      | Instrumentation Failure | VRD-3.1 | Backup location data based on last position/use instruments                                       | 2               |
|                            |                         | VRD-3.2 | Solar ray protection for electronics                                                              | 1               |
|                            |                         | VRD-3.3 | Radiation shielding                                                                               | 1               |
| VRD-4                      | Battery Failure         | VRD-4.1 | Shielding around battery, replacement battery on site                                             | 1               |
|                            |                         | VRD-4.2 | have light or sensor to indicate when battery is not charging so problem can be fixed immediately | 0.5             |
| VRD-5                      | Suspension Failure      | VRD-5.1 | cycle testing prior to launch to know when to replace suspension                                  | 1               |
|                            |                         | VRD-5.2 | reinforce with titanium                                                                           | 1               |
|                            |                         | VRD-5.3 | reinforce attachment points                                                                       | 0.5             |
| VRD-6                      | Wheel Failure           | VRD-6.1 | reinforce body with titanium lining                                                               | 1               |

## D. Scouting Rover

**Table 8: Risk/Fault Analysis for Scouting Rover Pre-Mitigation**

| Ref ID         | Intermediate Event      | Ref ID  | Basic Event                | Likelihood | Consequence | Overall |
|----------------|-------------------------|---------|----------------------------|------------|-------------|---------|
| SCOUTING ROVER |                         |         |                            |            |             |         |
| VRS-1          | Drive-train Failure     | VRS-1.1 | Dust jams DC motor         | 2          | 5           | 10      |
|                |                         | VRS-1.2 | DC motor overheat          | 1          | 3           | 3       |
| VRS-2          | Instrumentation Failure | VRS-2.1 | Loss of position knowledge | 2          | 2           | 4       |
|                |                         | VRS-2.2 | Imaging/camera failure     | 2          | 2.5         | 5       |
|                |                         | VRS-2.3 | Radiation damage           | 2          | 3           | 6       |
| VRS-3          | Battery failure         | VRS-3.1 | Battery death              | 2          | 5           | 10      |
|                |                         | VRS-3.2 | Battery Overheat           | 1          | 3           | 3       |
| VRS-4          | Structural failure      | VRS-4.4 | Impact damage              | 2          | 3           | 6       |
|                |                         | VRS-4.5 | Wheel nut failure          | 1          | 3           | 3       |
| VRS-5          | Suspension failure      | VRS-5.1 | Tipping/falling            | 2          | 3           | 6       |
|                |                         | VRS-5.2 | Suspension breakage        | 1          | 5           | 5       |

**Table 9: Risk/Fault Analysis for Scouting Rover Post-Mitigation**

| Ref ID         | Intermediate Event      | Ref ID  | Mitigation/Warning                                                                     | Post-Mitigation |     |
|----------------|-------------------------|---------|----------------------------------------------------------------------------------------|-----------------|-----|
| SCOUTING ROVER |                         |         |                                                                                        |                 |     |
| VRS-1          | Drive-train Failure     | VRS-1.1 | Cover motor                                                                            | 1               | 5   |
|                |                         | VRS-1.2 | Temperature sensors, stop usage when too hot                                           | 1               | 3   |
| VRS-2          | Instrumentation Failure | VRS-2.1 | Redundant Star Trackers/ Reposition using RF beacons                                   | 1               | 2   |
|                |                         | VRS-2.2 | Redundant camera Reposition using RF beacons                                           | 1               | 2.5 |
|                |                         | VRS-2.3 | Warning: Bit-flips, computing glitches. Mitigation: shielding                          | 1               | 3   |
| VRS-3          | Battery failure         | VRS-3.1 | Warning: Notify base when at 20%. Mitigation: charge battery                           | 1               | 5   |
|                |                         | VRS-3.2 | TCS, temperature sensors, stop usage when too hot                                      | 0.5             | 3   |
| VRS-4          | Structural failure      | VRS-4.4 | Impact testing                                                                         | 2               | 3   |
|                |                         | VRS-4.5 | Warning: Strain gage reading                                                           | 0.5             | 3   |
| VRS-5          | Suspension failure      | VRS-5.1 | Vertical axis of vehicle becomes parallel to surface, Mitigation: use arm to stabilize | 1               | 2   |
|                |                         | VRS-5.2 | Warning: Strain gage reading                                                           | 0.5             | 5   |

## E. Collection Rover

**Table 10: Risk/Fault Analysis for Collection Rover Pre-Mitigation**

| Ref ID | Intermediate Event  | Ref ID  | Basic Event                   | Likelihood       | Consequence | Overall |
|--------|---------------------|---------|-------------------------------|------------------|-------------|---------|
|        |                     |         |                               | COLLECTION ROVER |             |         |
| VRL-1  | Drive Train Failure | VRL-1.1 | Broken Steering Linkage       | 3                | 4           | 12      |
|        |                     | VRL-1.2 | Dust in Motor                 | 1                | 1           | 1       |
|        |                     | VRL-1.3 | Loss of Motor Power           | 2                | 5           | 10      |
| VRL-2  | Body Failure        | VRL-2.1 | Body Buckles                  | 1                | 4           | 4       |
|        |                     | VRL-2.2 | Micrometeoroid Impacts        | 2                | 2           | 4       |
| VRL-3  | Instrument Failure  | VRL-3.1 | GPS Position Loss             | 2                | 2           | 4       |
|        |                     | VRL-3.2 | Solar Flare Interference      | 1                | 3           | 3       |
|        |                     | VRL-3.3 | Solar Wind Circuit Ionization | 2                | 3           | 6       |
| VRL-4  | Battery Failure     | VRL-4.1 | Battery Explosion             | 1                | 5           | 5       |
|        |                     | VRL-4.2 | Battery Recharge Failure      | 2                | 4           | 8       |
| VRL-5  | Suspension Failure  | VRL-5.1 | Fatigue Failure               | 2                | 4           | 8       |
| VRL-6  | Robotic Arm Failure | VRL-6.1 | Robotic Arm Jam               | 2                | 4           | 8       |
|        |                     | VRL-6.2 | Robotic Arm Claw Failure      | 1                | 3           | 3       |
| VRL-7  | Wheel Failure       | VRL-7.1 | Wheel Bearing Jam             | 2                | 3           | 6       |
|        |                     | VRL-7.2 | Wheel Fatigue Failure         | 1                | 4           | 4       |
|        |                     | VRL-7.3 | Spoke Failure                 | 1                | 4           | 4       |
| VRL-8  | Axle Failure        | VRL-8.1 | Fatigue Failure               | 1                | 5           | 5       |
| VRL-9  | Storage Failure     | VRL-9.1 | Bay Door Jamming              | 2                | 5           | 10      |
|        |                     | VRL-9.2 | Storage Entrance Blockage     | 2                | 4           | 8       |

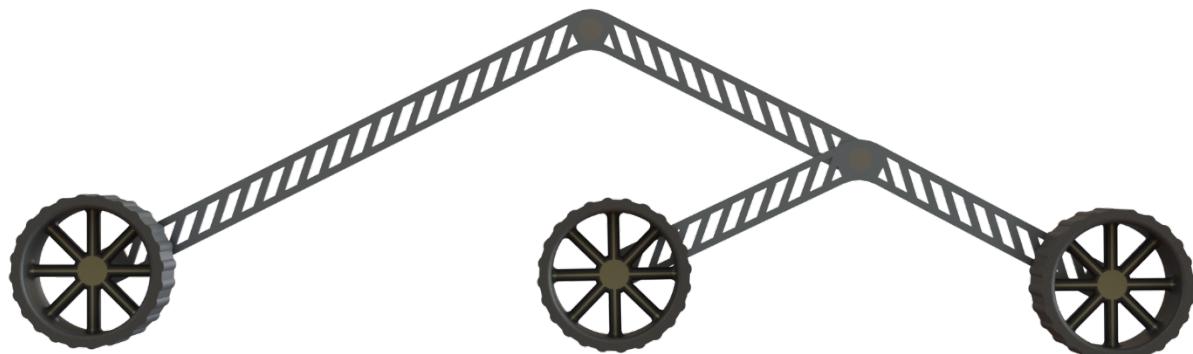
**Table 11: Risk/Fault Analysis for Collection Rover Post-Mitigation**

| Ref ID                  | Intermediate Event  | Ref ID  | Mitigation/Warning                                     | Post-Mitigation |     |     |
|-------------------------|---------------------|---------|--------------------------------------------------------|-----------------|-----|-----|
|                         |                     |         |                                                        | 1               | 2   | 3   |
| <b>COLLECTION ROVER</b> |                     |         |                                                        |                 |     |     |
| VRL-1                   | Drive Train Failure | VRL-1.1 | Better surface surveillance                            | 1               | 5   | 5   |
|                         |                     | VRL-1.2 | Cover motor                                            | 1               | 5   | 5   |
|                         |                     | VRL-1.3 | Backup power                                           | 1               | 5   | 5   |
| VRL-2                   | Body Failure        | VRL-2.1 | Stress testing                                         | 1               | 4   | 4   |
|                         |                     | VRL-2.2 | Impact Testing                                         | 1               | 2   | 2   |
| VRL-3                   | Instrument Failure  | VRL-3.1 | Reposition using RF beacons                            | 1               | 1.5 | 1.5 |
|                         |                     | VRL-3.2 | NASA Solar Flare detection warning                     | 1               | 3   | 3   |
|                         |                     | VRL-3.3 | Radiation shield around sensitive electronics          | 1               | 4   | 4   |
| VRL-4                   | Battery Failure     | VRL-4.1 | Thermally stable environment in rover and temp warning | 1               | 2   | 2   |
|                         |                     | VRL-4.2 | Warning on low power, backup circuit                   | 1               | 5   | 5   |
| VRL-5                   | Suspension Failure  | VRL-5.1 | Fatigue Testing                                        | 1               | 4   | 4   |
| VRL-6                   | Robotic Arm Failure | VRL-6.1 | Torque Sensor                                          | 2               | 2   | 4   |
|                         |                     | VRL-6.2 | Effective Materials                                    | 1               | 2   | 2   |
| VRL-7                   | Wheel Failure       | VRL-7.1 | Torque Sensor                                          | 1               | 2   | 2   |
|                         |                     | VRL-7.2 | Fatigue Testing                                        | 1               | 4   | 4   |
|                         |                     | VRL-7.3 | Stress testing                                         | 1               | 4   | 4   |
| VRL-8                   | Axle Failure        | VRL-8.1 | Fatigue Testing                                        | 1               | 5   | 5   |
| VRL-9                   | Storage Failure     | VRL-9.1 | Dust Blocker                                           | 1.5             | 4   | 6   |
|                         |                     | VRL-9.2 | Locking sensor on storage bay entrance                 | 2               | 4   | 8   |

## V. CAD Models



**Fig. 21: Render of Clearing Rover in Stowed Configuration (created by Chase Neff)**



**Fig. 22: Rocker Bogie Suspension Assembly (created by Chase Neff)**



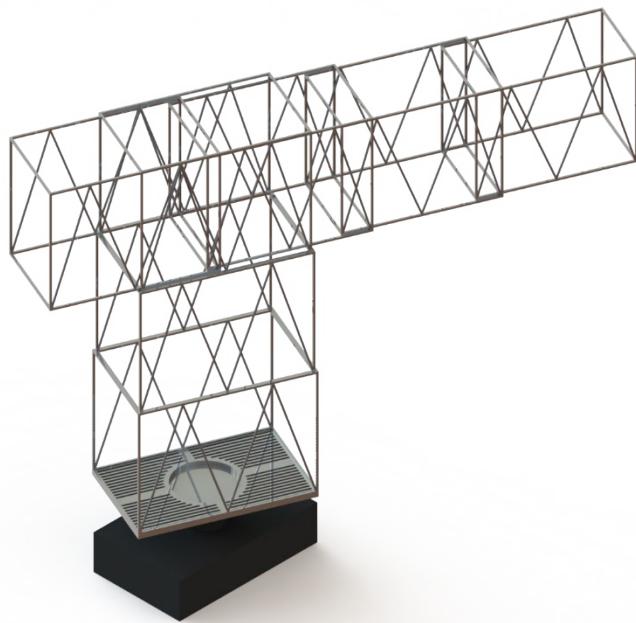
**Fig. 23: Rocker Bogie Upper/Lower Linkages w/ Wheel Axles (created by Chase Neff)**



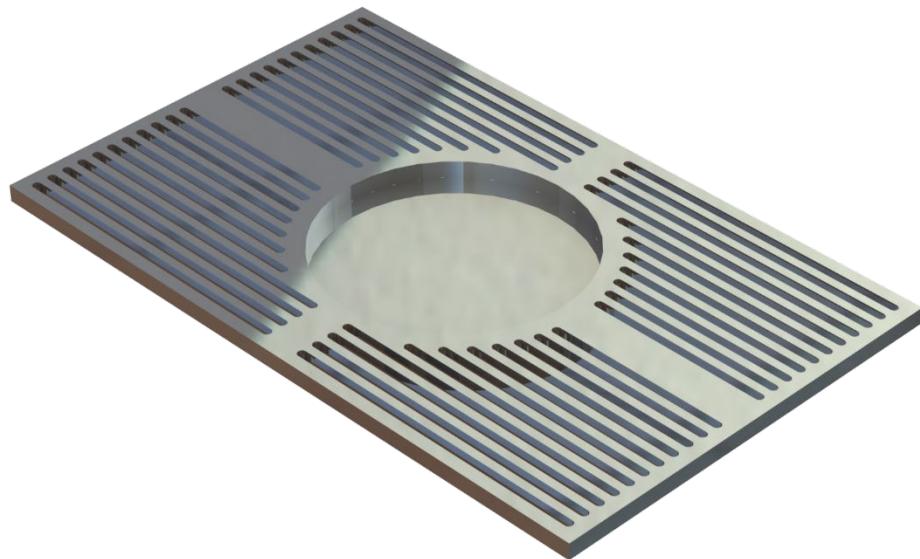
**Fig. 24: Autonomous Rover Wheel (Solid) (created by Chase Neff)**



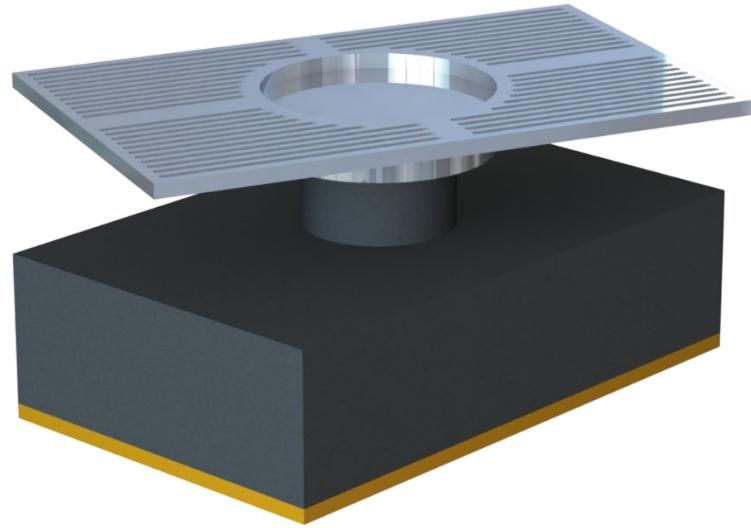
**Fig. 25: HDA (Heavy Duty Arm) Module (created by Chase Neff)**



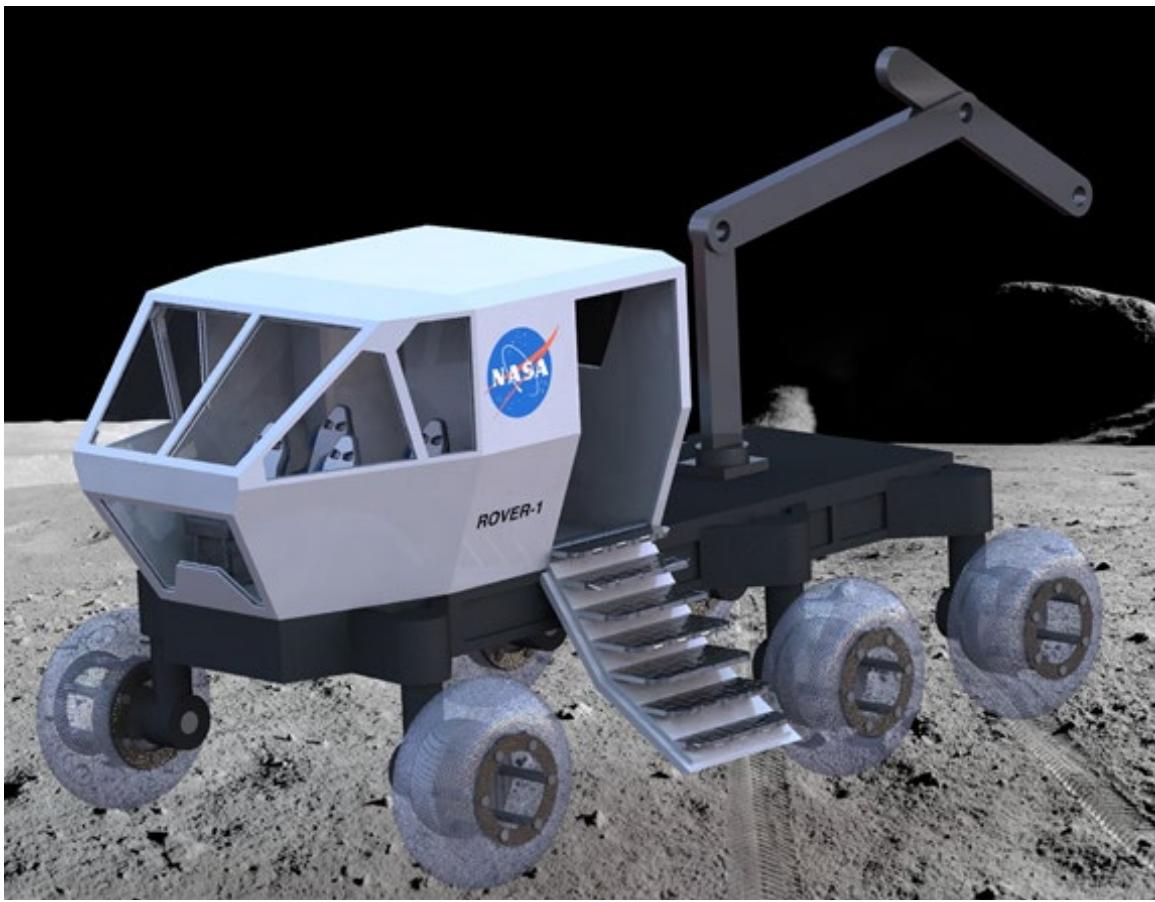
**Fig. 26: Crane Module (created by Vishank Battar)**



**Fig. 27: Crane Module Swivel Base (created by Chase Neff)**



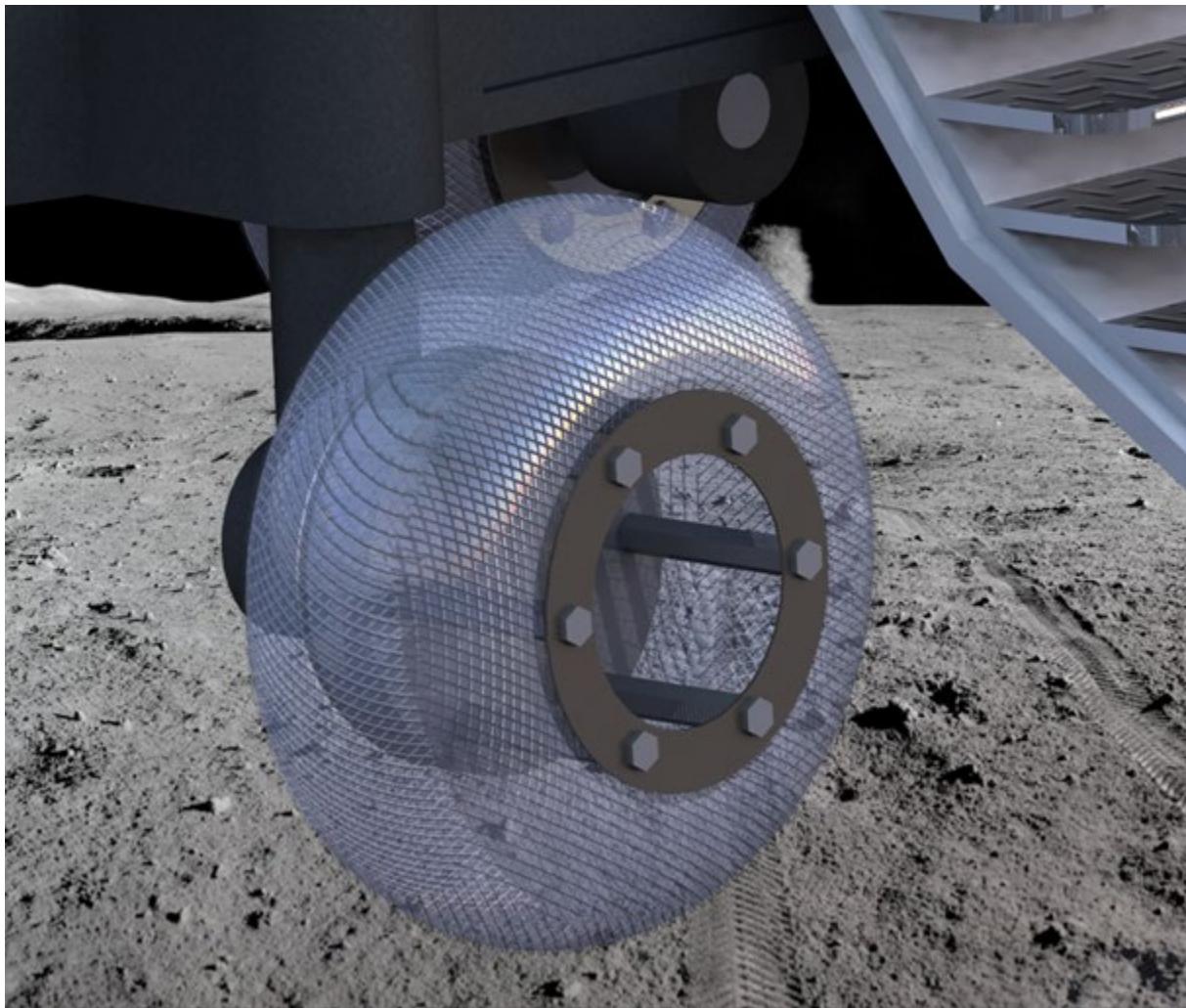
**Fig. 28: Crane Module Swivel Base and Flatbed Mount (created by Chase Neff)**



**Fig. 29: Render of Construction Rover w/ HDA Module (created by Chase Neff)**



**Fig. 30: Render of Construction Rover w/ Crane Module (created by Chase Neff)**



**Fig. 31: Large Rover Wheel (Spring Mesh) (created by Chase Neff)**



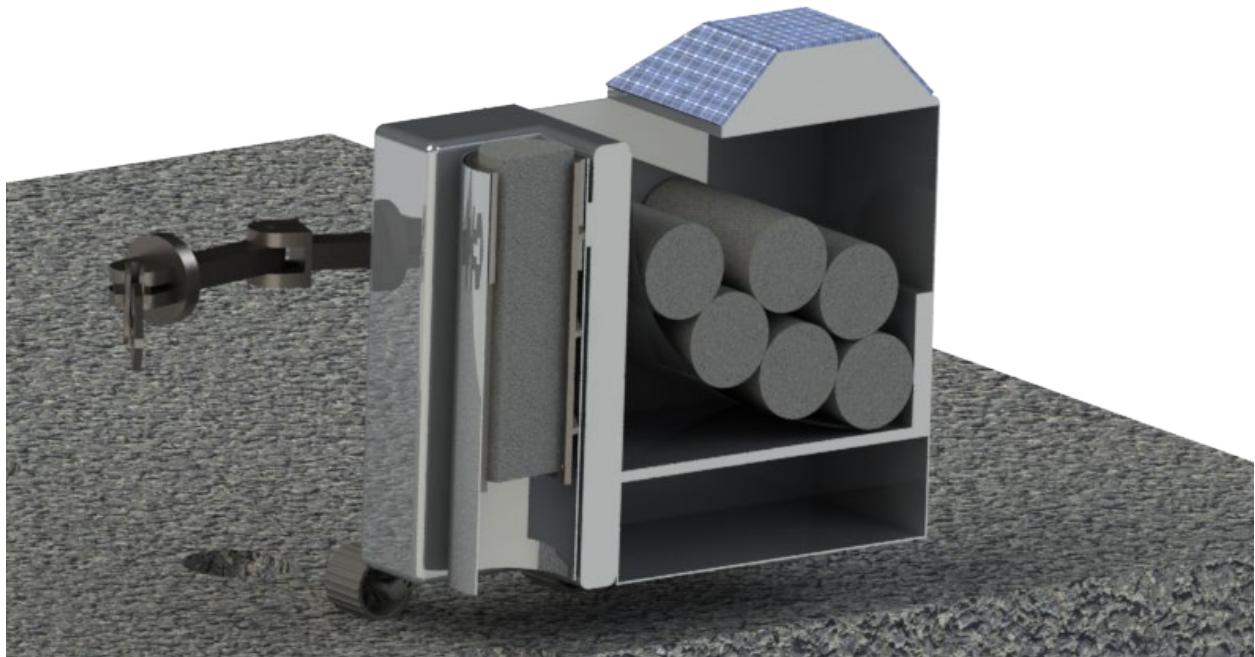
**Fig. 32: Isometric View of Scouting Rover (Created by Vishank Battar)**



**Fig. 33: Cross Sectional View of Scouting Rover (Created by Vishank Battar)**



**Fig. 34: Isometric View of Mining Rover (Created by Vishank Battar)**



**Fig. 35: Cross Sectional View of Mining Rover (Created by Vishank Battar)**

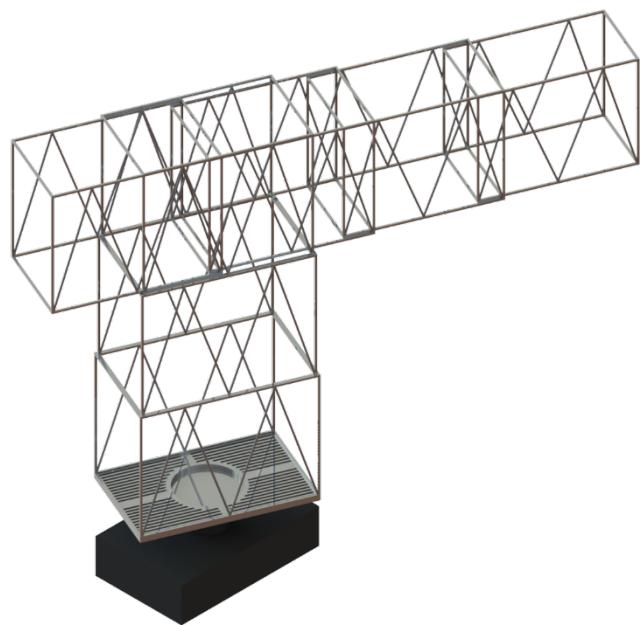


**Fig. 36: Isometric View of Core Drill (Created by Vishank Battar)**

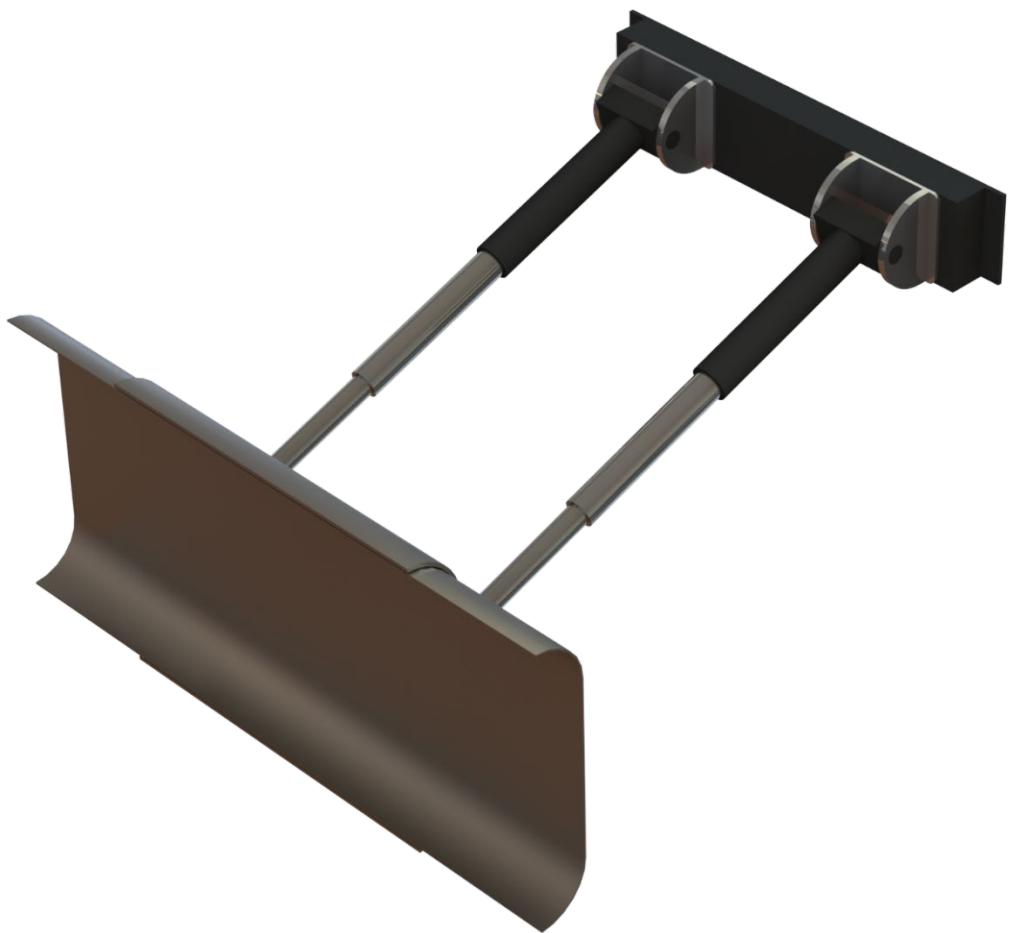


**Fig. 37: Close-up View of Core Drill Dogs (Created by Vishank Battar)**

Brian Jeffers



**Fig. 38: Crane Module Isometric**



**Fig. 39** Regolith Plow Assembly (created by Chase Neff)



**Fig. 121 Render of Clearing Rover in Operational Configuration (created by Chase Neff)**

## References

- [1] Qian, X., Yu, H., and Chen, S. *A Global-Shutter Centroiding Measurement CMOS Image Sensor with Star Region SNR Improvement for Star Trackers*. IEEE Transaction on Circuits and Systems for Video Technology, Vol. 26, No. 8. 2016.
- [2] Gammell, J. D., Tong, C. H., Berczi, P., Anderson, S., Barfoot, T. D., and Enright, J. *Rover Odometry Aided by a Star Tracker*. 2013 IEEE Aerospace Conference. 2013.
- [3] Durrant-Whyte, H. and Bailey, T. *Simultaneous Localization and Mapping: Part 1*. IEEE Robotics and Automation Magazine, Vol. 13, No. 2. 2006.
- [4] Julier, S. J. and Uhlmann, J. K. *Building a Million Beacon Map*. Sensor Fusion and Decentralized Control in Robotic Systems IV, Vol. 4571. October 4, 2001.
- [5] Stone, J. M., LeMaster, E. A., Powell, J. D., and Rock, S. *GPS Pseudolite Transceivers and their Applications*. ION National Technical Meeting 99. January 25, 1999.
- [6] Locata Corporation. *Locata at Spatial@Gov Conference*. Retrieve from <http://www.locata.com/technology/faqs/>
- [7] NASA Mars Exploration Rovers Team. *Moving Around Mars*. Retrieved from <https://mars.nasa.gov/mer/mission/timeline/surfaceops/navigation/#how-far-traveled>
- [8] Mahmoudian, N. (2021). *Lecture\_10 Perception*. Brightspace.
- [9] Mahmoudian, N. (2021). *ME597\_Perception*. Brightspace.
- [10] “IMU Sensors,” *sensonor.no* Available: <https://www.sensonor.com/applications/space/>.
- [11] Maki, J. N., Bell, J. F., Herkenhoff, K. E., Squyres, S. W., Kiely, A., Klimesh, M., . . . Lorre, J. (2003). Mars Exploration Rover Engineering Cameras. *Journal of Geophysical Research: Planets*, 108(E12). doi:10.1029/2003je002077

[12] “RPLidar A1M8 ,” *RobotShop* Available: <https://www.robotshop.com/en/rplidar-a1m8-360-degree-laser-scanner-development-kit.html>.

[13] Eisenman, Allan R., et al. “Mars Exploration Rover Engineering Cameras.” *Sensors, Systems, and Next-Generation Satellites V*, 2001, doi:10.1117/12.450671.

[14] *Chang'e 4 first scientific data is released* Available: <https://moon.bao.ac.cn/pubMsg/detail-CE4EN.jsp#>.

[15] Kakehashi, Y., and Takahashi, M., “Terrain Features Estimation for the Lunar/Planetary Rover Control,” *AIAA Guidance, Navigation, and Control Conference*, 2010.

[16] Auralius Manurung (2021). Finding optimal path on a terrain, MATLAB Central File Exchange. <https://www.mathworks.com/matlabcentral/fileexchange/39034-finding-optimal-path-on-a-terrain>,

[17] NASA, Sibille, L., Carpenter, P., Schlagheck, R., & French, R. A. (2006, September). Lunar Regolith Simulant Materials: Recommendations for Standardization, Production, and Usage. [https://www.nasa.gov/sites/default/files/atoms/files/nasa\\_tp\\_2006\\_214605.pdf](https://www.nasa.gov/sites/default/files/atoms/files/nasa_tp_2006_214605.pdf)

[18] Li, L.H., Lian, J., Chen, B.C., Chang, J., and Huang, H.Y. *Trajectory tracking and traction coordinating controller design for lunar rover based on dynamics and kinematics analysis*. School of Automotive Engineering, Faculty of Vehicle Engineering and Mechanics, State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian, P.R. China. *Journal of Vibroengineering*, Vol. 16, Issue 6, 2014.

- [19] Zuber, M., Head, J., Smith, D., et al. *Constraints on the volatile distribution within Shackleton crater at the lunar south pole*. Nature, 2012. <https://doi.org/10.1038/nature1121>
- [20] Wagner, R.V., Robinson, M.S., Speyerer, E.J., and Mahanti, P. *Topography of 20-km Diameter Craters on the Moon*. Arizona State University, Tempe, AZ.
- [21] Sinha, A., and Sinha, R. *Design of Stair-Climbing Rocker-Bogie Mechanism*. International Journal of Innovative Research in Science, Engineering and Technology, July 2018.
- [22] Miller, D.P., and Lee, T.L. *High-Speed Traversal of Rough Terrain Using a Rocker-Bogie Mobility System*. Conference paper. Proceedings of Robotics 2002: The 5th International Conference and Exposition on Robotics for Challenging Situations and Environments, June 2002. [https://doi.org/10.1061/40625\(203\)54](https://doi.org/10.1061/40625(203)54)
- [23] “Multi-Layer Insulation,” Meyer Tool & MFG, Oak Lawn, IL
- [24] Plachta, D. W., Feller, J. R., Guzik, M. C., “In-Space Cryogenic Propellant Storage Applications for a 20 W at 20 K Cryocooler”, NASA Glenn Research Center, Cleveland, OH, NASA Ames Research Center, Moffett Field, CA
- [25] Anvari, A., Farhani, F., Niaki, K., “Comparative Study on Space Qualified Paints Used for Thermal Control of a Small Satellite,” Iranian Research Organization for Science and Technology, Department of Mechanical Engineering, Jan. 2009.
- [26] Kerslake, T., Gustafson, E., “On-Orbit Performance Degradation of the International Space Station P6 Photovoltaic Arrays,” Glenn Research Center, Cleveland, Ohio, Jul. 2003
- [27] Jones, E.M. *Drilling Troubles*. Apollo 15 Lunar Surface Journal. 1996. <https://www.hq.nasa.gov/alsj/a15/a15.alsepdep.html>

[28] NASA Content Administrator. *Apollo 16*. NASA, July 8, 2009.

[https://www.nasa.gov/mission\\_pages/apollo/missions/apollo16.html](https://www.nasa.gov/mission_pages/apollo/missions/apollo16.html)

[29] *Drill, Apollo Lunar Surface (ALSD)*. Smithsonian National Air and Space Museum.

[https://airandspace.si.edu/collection-objects/drill-apollo-lunar-surface-alsd/nasm\\_A1976109500](https://airandspace.si.edu/collection-objects/drill-apollo-lunar-surface-alsd/nasm_A1976109500).

[30] Zhang, T. and Ding, X. *Drilling Forces Model for Lunar Regolith Exploration and Experimental Validation*. Acta Astronautica, Vol. 131, p. 190-203. 2017.

[31] Lewis. *Lunar Bases and Space Activities of the 21st Century*. Lunar and Planetary Institute,

NASA Astrophysics Data System, 1985.

[https://www.lpi.usra.edu/publications/books/lunar\\_bases/LSBchapter07.pdf](https://www.lpi.usra.edu/publications/books/lunar_bases/LSBchapter07.pdf)

[32] “Car - Required Power and Torque,” *Engineering ToolBox*. [Online]. Available: [https://www.engineeringtoolbox.com/cars-power-torque-d\\_1784.html](https://www.engineeringtoolbox.com/cars-power-torque-d_1784.html). [Accessed: Mar-2021].

[33] “Rolling Resistance,” *Engineering ToolBox*. [Online]. Available: [https://www.engineeringtoolbox.com/rolling-friction-resistance-d\\_1303.html](https://www.engineeringtoolbox.com/rolling-friction-resistance-d_1303.html). [Accessed: Mar-2021].

[34] “Custom Augers-Technical Data,” *Premier Components, Inc.* [Online]. Available: <https://premier-com.com/wp-content/uploads/2017/02/83-86.pdf>.

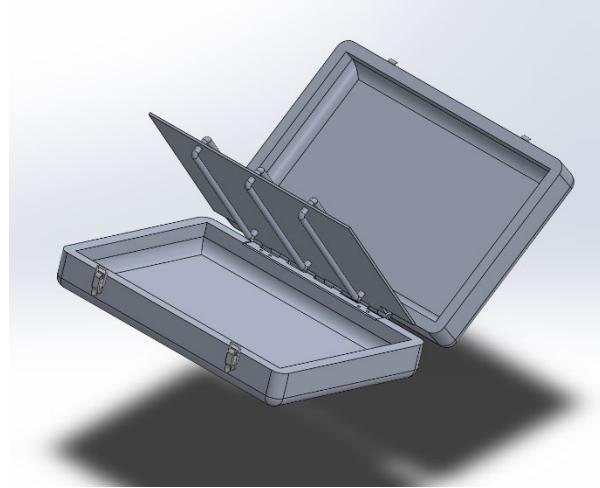
[35] Carrier, III, W. D. (2005, September). The Four Things You Need to Know About the Geotechnical Properties of Lunar Soil. Lunar Geotechnical Institute.

- [36] NASA Science Mars Exploration Team. *Communications with Earth*. August 8, 2019. Retrieved from <https://mars.nasa.gov/msl/mission/communications/>
- [37] Gertsch, L. S. (2003). Surface Mine Design and Planning for Lunar Regolith Production. AIP Conference Proceedings, 1–9. <https://doi.org/10.1063/1.1541408>
- [38] Asnani, V., Delap, D., and Creager, C. *The development of wheels for the Lunar Roving Vehicle*. Journal of Terramechanics, June 2009. <https://www.sciencedirect.com/science/article/abs/pii/S0022489809000263>.
- [39] Kilkenny, N.S. *Reinventing the Wheel*. NASA Imaging Technology Center. October 26, 2017. <https://www.nasa.gov/specials/wheels/>
- [40] Lockney, D. *Superelastic Tire*. NASA Technology Transfer Program. <https://technology.nasa.gov/patent/LEW-TOPS-99>
- [41] Olson, C. F., Matthies, L. H., Wright, J. R., Li, R., and Di, K., “Visual terrain mapping for Mars exploration,” *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*.
- [42] Ranganath, N., Panchakshari, H.V., Ramesh, A., and Hareesh, A. *Rover Wheel*. International Journal of Automobile Engineering (IJAUE), July-December, 2019. <https://www.iaeme.com/ijaue/issues.asp?JType=IJAUE&VType=1&IType=2>
- [43] “The Avenue to Quality Enhanced Bin Clean-out,” *Illinois Grain & Seed Equipment*. [Online]. Available: [https://e1374e44-8a1f-4fec-b500-0ef1ceeed927.filesusr.com/ugd/70fdd8\\_bd795720e70d4d96b0987256e0c3dbd7.pdf](https://e1374e44-8a1f-4fec-b500-0ef1ceeed927.filesusr.com/ugd/70fdd8_bd795720e70d4d96b0987256e0c3dbd7.pdf).

[44] “The Mars Rovers,” *NASA*, 23-Mar-2021. [Online]. Available: <https://spaceplace.nasa.gov/mars-rovers/en/>. [Accessed: Feb-2021].

# CAD

## I. Food Warmer Briefcase - Entire System



**Fig. 9 Open view of the Food Warmer Briefcase, showing the heating element supported by the hinge in the back.**

Through the Google Survey that was provided to the AAE 450 group by the CAD team, we received a request for a Food Warming device from the Human Factors team. The requester asked for a Food Warmer based off of the one used on the International Space Station which resembles a suitcase.

Our design of the Food Warmer Briefcase can be seen in Fig. #q which shows the isometric view of the open configuration of the Food Warmer, along with the heating element, latches, and hinge in the back. As can be seen, we are modeling the Food Warmer Briefcase off of the one on the International Space Station, but also are modeling it after a common toaster oven. We are modeling some of the briefcase after a toaster oven in order to estimate the amount of power that would be used. In doing this, we find that the Food Warmer Briefcase utilizes 1-3 kW of Power during its heating phase.

In modeling this briefcase after something that is fairly common on Earth, it is fairly simple to complete this design. We are able to utilize our knowledge of briefcases to aid with the dimensions and ergonomic usability of this Food Warmer. Firstly, we are deciding to make this briefcase 0.5m x 0.35m x 0.13m since that is close to the average size of a briefcase and looks similar in dimensions to what the Food Warmer on the International Space Station is like. This makes the briefcase take up a total overall volume of about  $0.022 \text{ m}^3$ . The briefcase, due to the utilization of draw latches, is able to open and close securely and stay closed if not desired to be open. The briefcase also uses a hinge system in the back which allows for the briefcase to open and close in one piece. Once the briefcase is opened, the heating element is exposed and can be seen as a solid sheet of metal with pipes that are used to hold the food in place. In order to use the Food Warmer, a person would need to open the briefcase, insert the packet of food under the pipe, turn on the heating element, and close the briefcase. Then once the food is finished heating, the person may retrieve the food packet and begin consuming.

Although this model is fairly simple, it did present one major difficulty in trying to model the electronics. Since electronic parts, like wires, are usually not well represented in Computer-Aided Design, it was our choice to exclude them from the final assembly. It is important to note, then, that the final assembly will increase in mass and volume values with the addition of the electronic system that will power the heating element.

There were two main iterations for this design. In the first iteration, we used velcro in order to seal the briefcase closed instead of using draw latches. In using velcro, it allowed for the design to be more compact (since draw latches were not necessary to include in the design). However, the use of velcro brought up one large disadvantage: the briefcase would not be entirely sealed when closed. Since velcro is made of many small loops and hooks, there is space for particles to travel

through it. Therefore, any food or spill from the inside of the food warmer had the possibility of not being contained inside the food warmer and could instead exit into the surrounding area. Since this can be dangerous, we decided on using draw latches instead of velcro in the second iteration since they can tighten the briefcase closed and provide a seal so that no unwanted particles can escape.

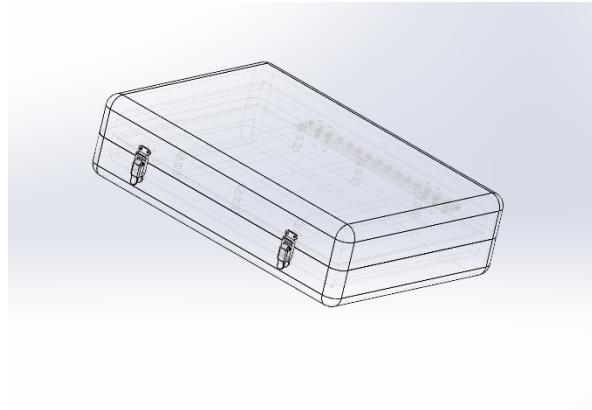
For this briefcase, we are using both aluminum and steel. Aluminum is being used in places that do not require as much strength, such as the briefcase walls and the food warmer. For parts that require more strength, such as the draw latches and the hinge in the back of the assembly, steel is being used. In using aluminum for the majority of the assembly, this helps with our desire to decrease weight as much as possible. With all these parts, the summed mass of the assembly equates to around 28.53kg, but this value will increase with the addition of the electronic parts.

The Food Warmer Briefcase contributes to the habitat system, as it allows for the people in the lunar habitat to heat their food anywhere they need to be since it is portable. This is necessary as it provides people in the lunar habitat with the ability to have heated food, as well as a reminder of their normal food routine that they used to have on Earth. Not only in helping heat food, this Food Warmer Briefcase also can help in providing a reminder of what being at home is like which can help combat homesickness.

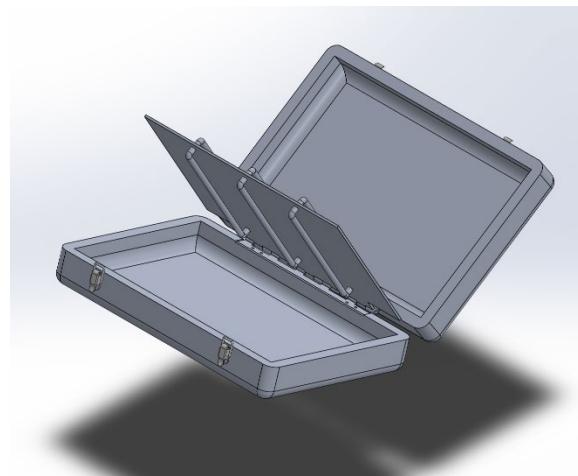
#### A. Food Warmer Briefcase - Heating Element

This Food Warmer, along with the one on the International Space Station, utilizes a conduction heating design to heat up food packets to be eaten by astronauts on board. My design, the Food Warmer Briefcase, uses the same heating element and has the same overall goal of heating food for the lunar station's inhabitants. Fig. #a shows the wireframe view of the Food

Warmer Briefcase and how the heating element is situated, whereas Fig. #b shows the open configuration of the briefcase with the heating element up and ready to be used.



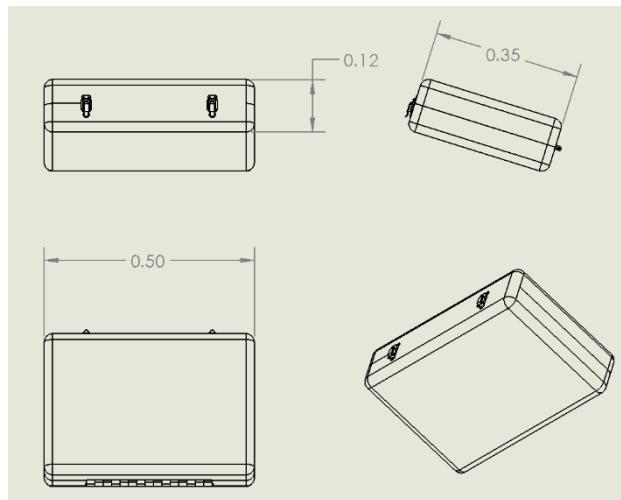
**Fig. 10 Wireframe view of the Food Warmer Briefcase, showing how the heating element is arranged when in the closed configuration.**



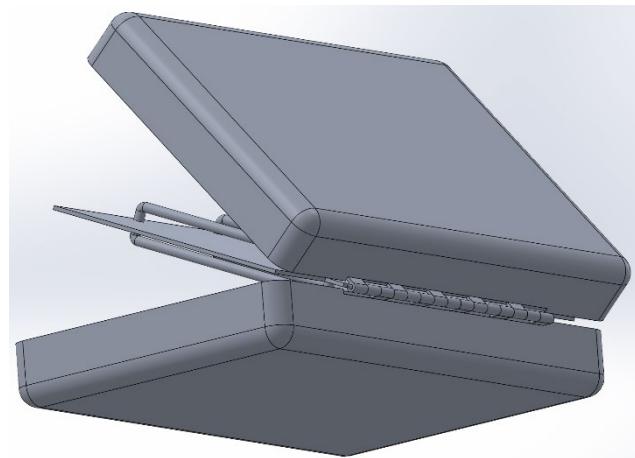
**Fig. 11 Open view of the Food Warmer Briefcase, detailing the heating element.**

## B. Food Warmer Briefcase - Briefcase & Hinge

This design is a Food Warmer that is based off of the food warming device on the International Space Station. My design uses the same overall look in being a briefcase and has the same overall goal of housing the heating element to heat the food packets for astronauts. The briefcase also utilizes a hinge mechanism in the back that allows for it to swing open and keep its top and bottom attached at all times. Fig. #c is a drawing of the Food Warmer Briefcase in its closed position with dimensions of the overall length, width, and height of the briefcase, whereas Fig. #d shows the back view of the Food Warmer Briefcase in its open configuration, detailing the hinge in the back.

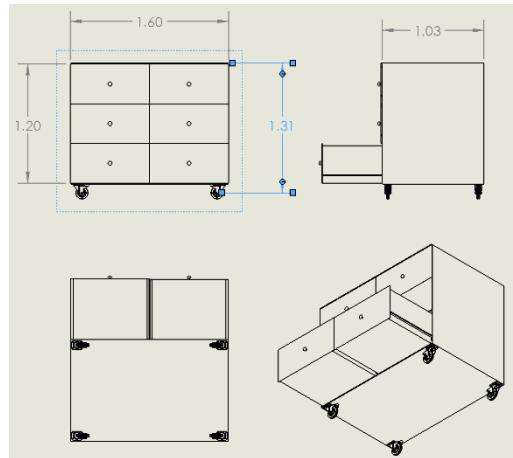


**Fig. 12 Drawing with overall dimensions of the Food Warmer Briefcase.**



**Fig. 13 Open view of the briefcase, showing the hinge in the back.**

### C. Medicine Cabinet - Entire System



**Fig. 14 Drawing of the Medicine Cabinet with 2 of its 6 drawers open, and dimensions of overall length, width, and height.**

Through the Google Survey that was provided to the AAE 450 group by the CAD team, we received a request for a medicinal cabinet from the Human Factors team. The requester asked for a Medicine Cabinet based on normal medicine cabinets that are used on Earth in order to hold medicine for each person living in the lunar habitat.

As was requested by the Human Factors team, Fig. #r shows the drawing of our design of a medicine cabinet. The cabinet is comprised of two main components: the housing and the drawers & dividers. The housing is the outer assembly which houses the drawers & dividers. This part allows for 6 drawers to be placed inside, and also contains sliding mechanisms which allow for the drawers to slide back and forth for easy usage. The housing also consists of the 4 wheels on the bottom which can swivel in 360 degrees, giving the ability for the medicine cabinet to maneuver around the habitat. This is important since it allows for the medicine cabinet to be moved to anywhere in the lunar habitat to allow for easier and possibly faster usage of the medicine stored

inside. As for the drawers & dividers component, there can be a maximum of 6 drawers used in the cabinet, and each can have a maximum of 31 dividers. An even arranging of the 31 dividers in a drawer allows for 20 even compartments for medicine to be housed in that drawer. These dividers are also rearrangeable as they utilize velcro on their top, bottom, left, and right sides to statically be arranged in whatever shapes and compartments that are needed. This can allow for multiple different ways to organize the insides of the drawers which can be helpful in organizing different sizes and amounts of medicine for each inhabitant. Moreover, the drawers also have sliding mechanisms which allow for them to slide in and out of the housing. This allows for easy access to the medicine in the drawers, just like a regular medicine cabinet on the Earth.

There were a couple of iterations that we decided to change during the design process of the Medicine Cabinet. Firstly, it became apparent that if all the dividers had the same side of velcro on them, then none would stick to each other. The dividers need to be able to stick to each other as they also provide the function of giving more structural support to one another. Therefore, we decided on the next iteration of using the hooked end of velcro on the tops, bottoms, and left sides, and the looped end of the velcro on the right sides. Since the inside of each drawer was lined with the looped end of the velcro, each divider would be able to attach to the bottom of each drawer, and now each divider can attach to each other as long as their opposing sides match. Secondly, we wanted for the drawers to be bolted while not in use so that it was impossible for the drawers to open when not in use (so that no accidental spills could occur). We decided to move to our current iteration of utilizing sliding mechanisms instead, since it was impractical to have to unscrew multiple bolts each time medicine was needed. It was also found that this could be a hazard since there may be situations where medicine is needed immediately but it would take too long for the medicine cabinets to be opened. Furthermore, since there is less gravity on the moon, it would be

more difficult for medicine to break and spill on the floor, therefore there is less risk in what we were originally thinking.

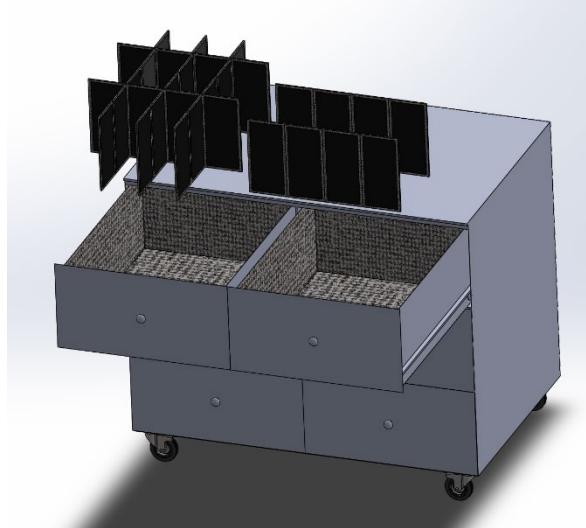
For this assembly, we also made a small animation to show how each part would interact with each other and how the drawers would be opened and closed. Furthermore in the animation, it shows a few different configurations of the dividers inside the drawers and shows how to move them from one drawer to another. This animation helped us realize one design flaw which was that the back 3 dividers need to be twisted in order to come out of the drawer instead of just being brought straight upward. This is only a small flaw though, as the back 3 dividers can easily be taken out still. Therefore, we decided to not change this part of the design.

The Medicine Cabinet is made of steel, aluminum, and plastic. Just like with the Food Warmer Briefcase, areas that need less strength use aluminum or plastic such as the housing, drawers, and dividers, whereas areas such as the wheels need to be made of steel since they need more strength. The full assembly sums up to be about 59kg total with a 93-divider and 6-drawer configuration, but this mass will increase with the addition of medicine. The material volume is  $0.09\text{m}^3$ , the overall structure takes up about  $2.27\text{m}^3$  of space, and there is  $1.70\text{m}^3$  of drawer space for storing the medicine. Finally, this medicine cabinet uses 0kW of power since there are no electronic parts being used.

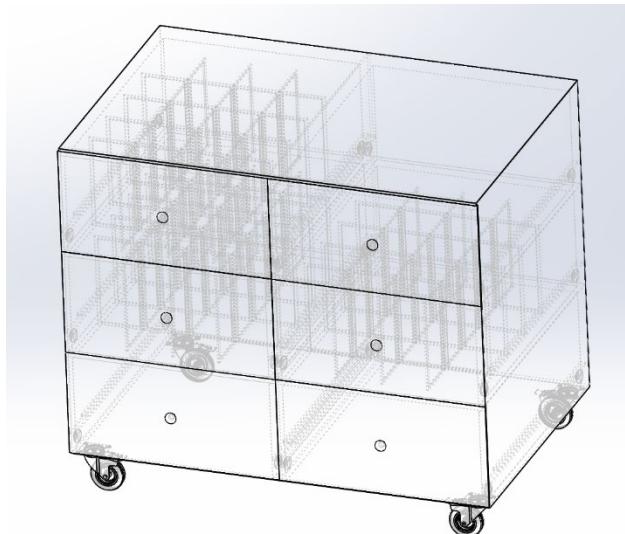
#### **D. Medicine Cabinet - Drawers & Dividers**

The Medicine Cabinet was designed to simply store medicine in the lunar habitat, but to also have the ability to reorganize its compartments. The cabinet can use its plastic dividers which are lined on its edges with velcro to compartmentalize the drawers to any configuration that is needed. This can allow for larger or smaller compartments as needed for different types of

medicine. Fig. #e shows two separate configurations for the dividers, and Fig. #f shows the wireframe view of the cabinet with a configuration of 93 dividers in three of its six cabinets, making 31 dividers to a drawer, in turn producing 20 different compartments per drawer.



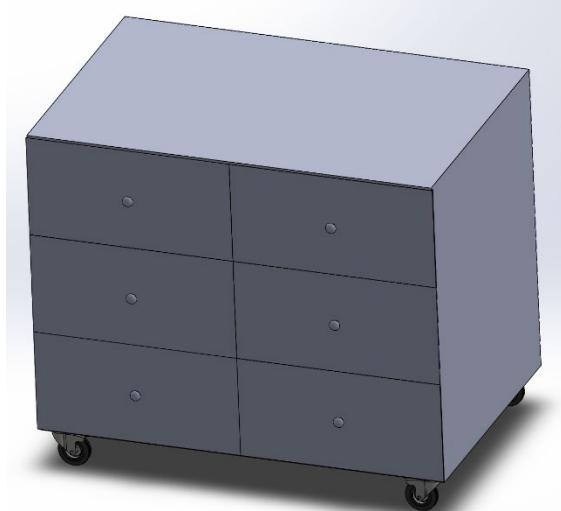
**Fig. 15** View of the medicine cabinet with drawers open and dividers in different configurations.



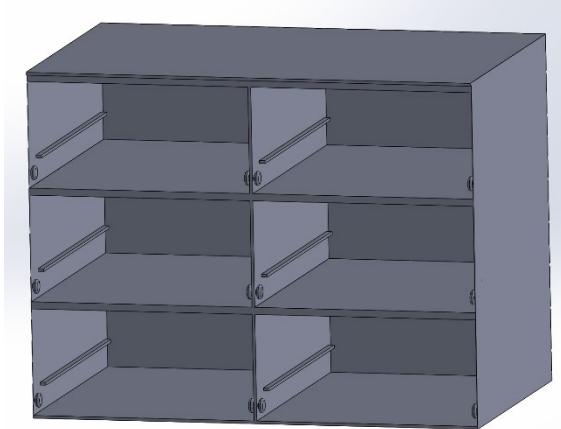
**2. Fig. 16** Wireframe view of the medicine cabinet with 93 dividers in 3 of its 6 drawers.

#### E. Medicine Cabinet - Housing

As stated previously, the Medicine Cabinet was designed to simply store medicine in the lunar habitat, but to also have the ability to move around. The cabinet can use its 4 wheels on the bottom which can swivel 360 degrees in order to maneuver around the habitat. The housing also allows for 6 drawers to fit inside the medicine cabinet, allowing for a maximum of  $1.70\text{m}^3$  of total space for medicine. [Fig. #g](#) shows the medicine cabinet housing with its 6 drawers and 4 wheels, and [Fig. #h](#) shows the wireframe view of the cabinet housing. This figure also shows the mechanisms on each side of the drawers that act like sliders to allow for the drawers to slide into place.

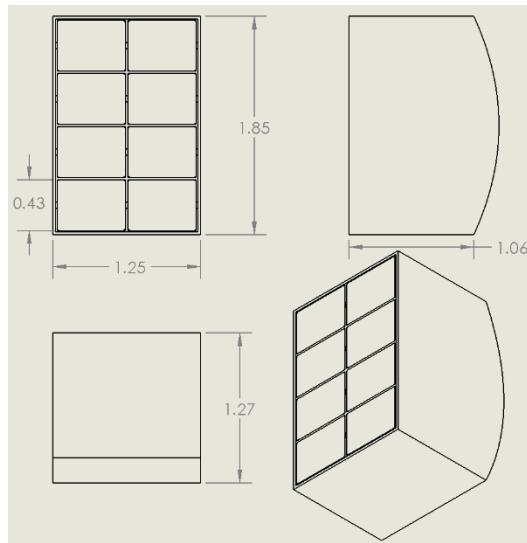


**Fig. 17 Isometric view of the medicine cabinet, detailing the housing, its 6 drawers, and its 4 wheels.**



**3. Fig. 18 Isometric view of the medicine cabinet without drawers and wheels. This helps to show the inner mechanisms that allow for the drawers to slide into place.**

#### **4. Oxygen Regeneration System**



**Fig. 19 Drawing of the International Standard Payload Rack designed by our team, along with the payload rack and 8 payload drawers.**

Through the Google Survey that was provided to the AAE 450 group by the CAD team, we received a request for certain parts of an Oxygen Regeneration System from the Human Factors team. The requester asked for two main things: the designs of Adsorbent and Desiccant Beds, and a design based off of the International Standard Payload Rack that is used on the International Space Station.

For this request, like all others, we adhere to the modeling standards that were put forth within the CAD team. These standards include the documentation's units, software, optimization, desired characteristics, and environmental concerns. For the document's units, we adhere to Mass,

Kilogram, Second (MKS) which are the normal SI units that are used throughout the AAE 450 group. As a team, we also decided on the utilization of one software: SolidWorks. This one software allows for all of us the ability to access each others' designs and edit them as needed, as well as increase our collaboration on designs. For optimization, we want to minimize both mass and volume values on every design so that fewer trips would be needed to the moon and therefore less money would need to be allocated for transportation costs. Desired Characteristics for each design is to have maneuverability if possible so that it could be used all throughout the habitat, as well as have the ability to be disassembled into smaller parts which can make it easier for it to be transported if the need arises. These two main assemblies, although not having the ability to be maneuverable, do have the exact characteristics of disassembly. Lastly, there are environmental concerns of the low-gravity environment that came into effect when designing these assemblies. Although low-gravity is a concern, this worry went away immediately since these two assemblies are modeled after the exact International Standard Payload Rack that is used on the International Space Station.

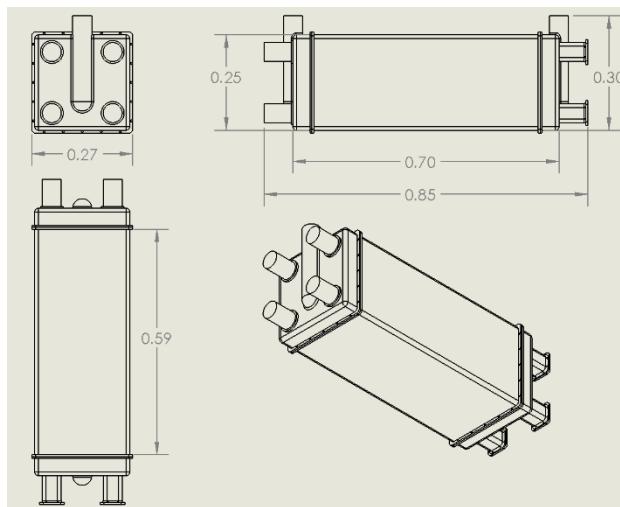
As can be seen in Fig. #s, we designed this drawing based off of the International Standard Payload Rack on the International Space Station. This assembly contains 8 payload drawers and the payload housing which houses those drawers. Inside each payload drawer is enough space to fit one Adsorbent Bed and one Desiccant Bed each, allowing for ample space for the Oxygen Regeneration System. The Adsorbent Bed and Desiccant Bed are designed out of an aluminum housing with different mounting assemblies on the front and back to allow for the beds to be mounted to the payload drawers and for airflow to travel through the beds. The beds contain three main layers on the inside: two layers of glass beads and one layer of silica gel. The silica gel is laid in between the two glass beads layers which act as the sorbead necessary in this system. The

Adsorbent and Desiccant Beds are fairly similar in the design, but have different mounting assemblies on either side and have different lengths of housing assemblies. Each drawer also contains a latch which can be used to keep the Adsorbent and Desiccant Beds from falling or sliding out of the payload drawers. The back of the payload rack matches the cutouts that are present in the latches, allowing for electronic wires to pass through the payload drawers. Moreover, the payload rack fits inside the overall housing which is curved in the back to provide space for electronics systems. The overall system allows for 8 drawers to fit into one overall housing, and with the possibility of a maximum 2 beds (one Adsorbent and one Desiccant) per drawer, the International Standard Payload Rack designed by our team can hold up to 16 beds for aiding in the Oxygen Regeneration System.

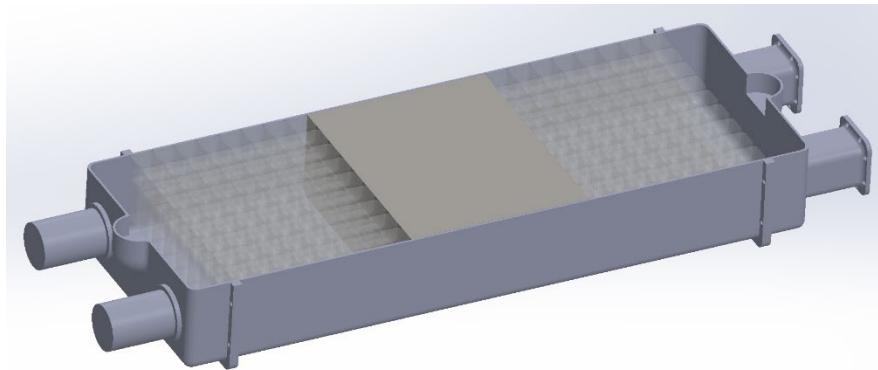
The overall assembly has materials including aluminum, glass, silica gel, and plastic. The Adsorbent and Desiccant Beds are made of aluminum and contain glass and silica gel, whereas the payload rack is made of plastic and the overall housing is made of aluminum. Each Adsorbent Bed (including the layers of silica gel and glass beads) is about 72kg and contains about  $0.03\text{m}^3$  of material. On the other hand, each Desiccant Bed (also including the layers of silica gel and glass beads) is about 90kg and contains about  $0.03892\text{m}^3$  of material. Furthermore, everything else (including the overall housing, payload rack, 8 drawers, and 8 latches) combines to a mass of about 669kg, contains about  $0.53\text{m}^3$  of material, and takes up about  $2.451\text{m}^3$  of space. These components will be utilized by the Life Supports Systems to aid in the process of regenerating breathable oxygen for the people who inhabit the lunar colony.

## F. Oxygen Regeneration System - Adsorbent Bed

The Adsorbent Bed design is to be used in order to help in the regeneration process for oxygen while in the lunar habitat. Fig. 20 shows a drawing of the Adsorbent Bed. This bed, combined with the Desiccant bed, is narrow enough to fit within one drawer of the International Standard Payload Rack. Fig. 21 shows a cross-section view of the Adsorbent Bed. This view shows 3 main layers: a silica gel layer in the middle which is surrounded by two layers of glass beads. These layers aid in the process of the overall oxygen regeneration system.



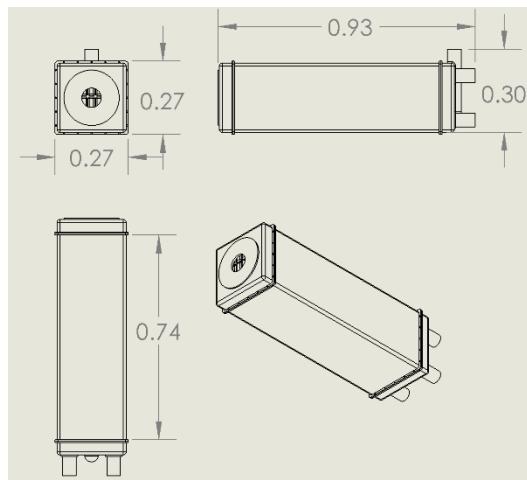
**Fig. 20 Drawing of the Adsorbent Bed with both mounting assemblies on either end, and with dimensions of length, width, and height.**



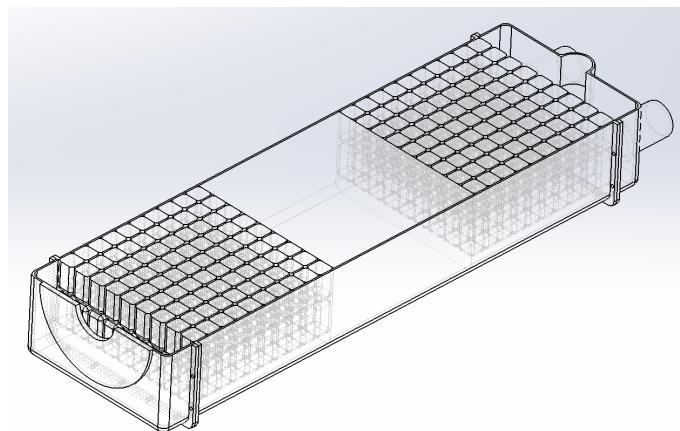
**Fig. 21 Cross-Section view of the Adsorbent Bed with a silica gel layer surrounded by layers of glass beads.**

## G. Oxygen Regeneration System - Desiccant Bed

The Desiccant Bed design is used in order to help in the regeneration process for oxygen while on the moon. Fig. 22 shows a drawing of the Desiccant Bed. This bed, combined with the Adsorbent bed, is narrow enough to fit within one drawer of the International Standard Payload Rack. Fig. 23 shows a wireframe cross-section view of the Desiccant Bed. This view shows 3 main layers: a silica gel layer in the middle which is surrounded by two layers of glass beads. These layers aid in the process of the overall oxygen regeneration system.



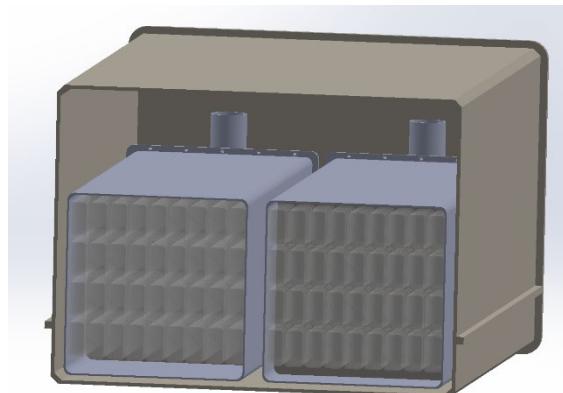
**Fig. 22 Drawing of the Desiccant Bed with the mounting assemblies on each end, and with dimensions of length, width, and height.**



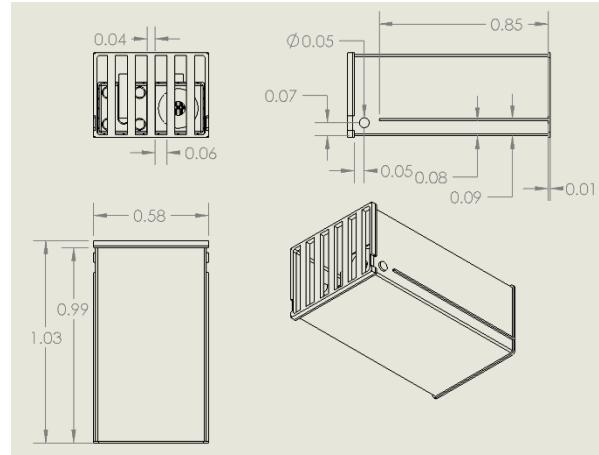
**Fig. 23 Wireframe cross-section view of the Desiccant Bed with a silica gel layer surrounded by layers of glass beads.**

## H. Oxygen Regeneration System - Payload Drawer and Latch

The International Standard Payload Rack contains drawers in order to hold its payloads. We designed a carrier for payloads specifically for the Oxygen Regeneration System for our Lunar Colony that is modeled after the International Standard Payload Rack. As can be seen in Fig. 24, this is a cross-section view of one of the payload drawers which is housing one Adsorbent Bed and one Desiccant Bed. This payload drawer is able to easily slide into the payload rack with the help of its sliding mechanisms as can be seen in Fig. 25. Also in Fig. 25, the latch can be seen which is bolted onto the payload drawer and has openings for electronics to pass outside the drawer from the inside.



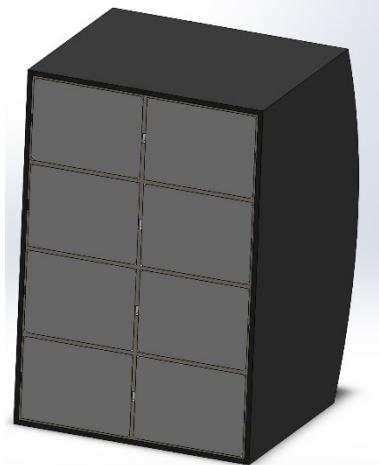
**Fig. 24 Cross-section view of the payload drawer with one Adsorbent Bed and one Desiccant Bed.**



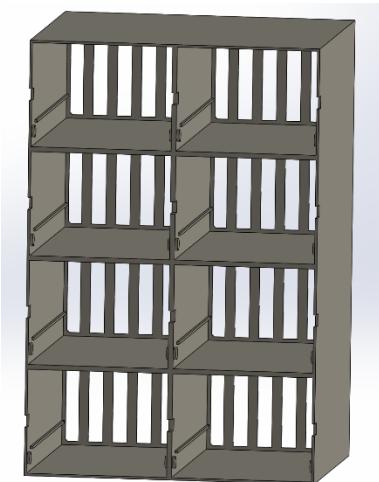
**Fig. 25 Drawing of a payload drawer with one Adsorbent Bed (left) and one Desiccant Bed (right), with a latch bolted to the back.**

## I. Oxygen Regeneration System - International Standard Payload Rack

As stated above, we modeled our lunar habitat's payload rack after the International Standard Payload Rack. This ISPR, as shown in Fig. 26, looks like and serves the same purpose as the ISPR used in the International Space Station. In Fig. 27, the payload rack can be seen. This payload rack contains sliding mechanisms which allow for the payload drawer to slide into place. The back of the payload rack is also slotted to match up with the payload drawer's latch to allow for electronics to come out from the payload boxes.



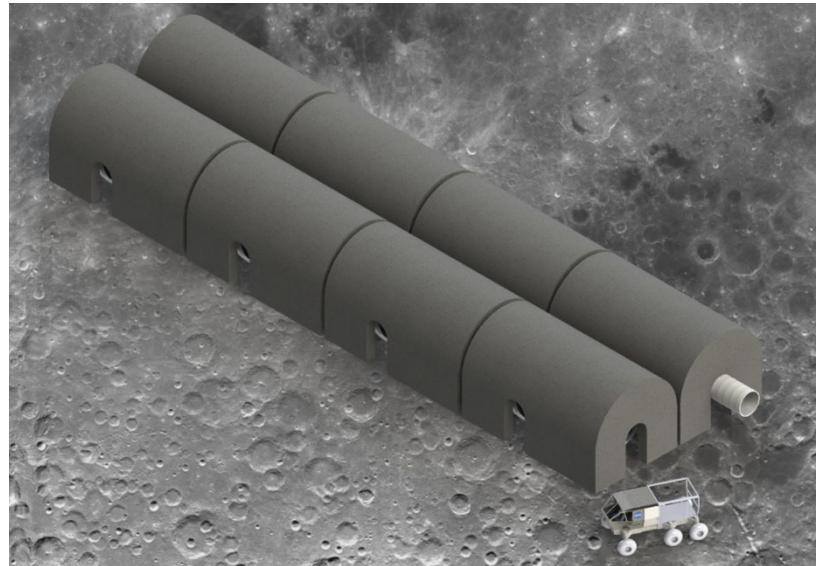
**Fig. 26** Front view of the International Standard Payload Rack's full assembly, including 8 payload drawers and the payload rack.



**Fig. 27** Front view of the payload rack which contains areas for 8 potential payload drawers and cutouts in the back for electronics systems.

## J. Full Habitat Design

The habitat was designed to be fully modular, with two different types of module (inhabited modules and airlocks). The modularity of the habitat enables modules sent during different launches to be easily integrated into the existing structure. Fig. 28 shows the full external habitat on the lunar surface shielded with a layer of regolith, and Fig. 29 shows a top section view of the habitat with the various module layouts.



**Fig. 28** Exterior view of the full modular habitat design.



**Fig. 29** Top section view of the full modular habitat design.

## K. Collins Dining & Food Preparation Module

The Collins Dining and Food Preparation module is designed to be a communal space where crew can relax and eat. The design of the room was optimized to convey a comfortable place. The colors, appearances, and furniture were curated by the Human Factors team to produce the module design shown in Fig. 30. Fig. 31 shows a top section view of the Collins Dining and Food Preparation Module. Fig. 32 and Fig. 33 show the layout of the dining and living space within the module.



**Fig. 30 Cutaway view of the Collins Dining & Food Preparation Module.**



**Fig. 31** Top section view of the Collins Dining & Food Preparation Module.



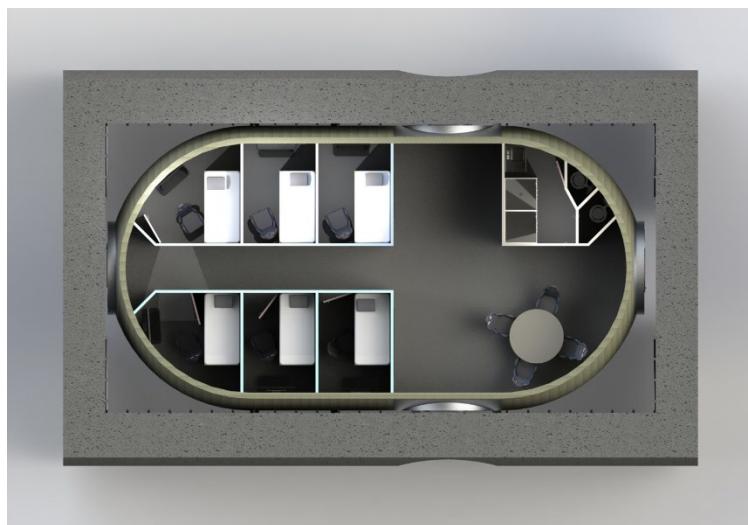
**Fig. 32** Interior view of the Collins Dining & Food Preparation Module.



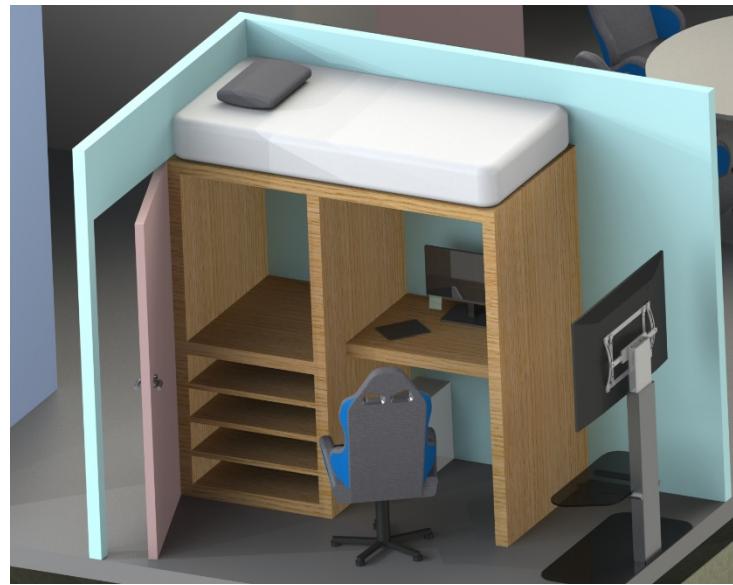
**Fig. 33 Interior view of the Collins Dining & Food Preparation Module.**

## L. Crew Module

The Crew Module is designed to house 6 crew members, have a bathroom for 2 crew members, and have a flexible corner space. Fig. 34 shows a top section view of the Crew Module and floor layout. An example bedroom is shown in Fig. 35, and the “flex corner” space is shown in Fig. 36.



**Fig. 34 Top section view of the Crew Module design**



**Fig. 35 Interior view of a bedroom in the Crew Module.**

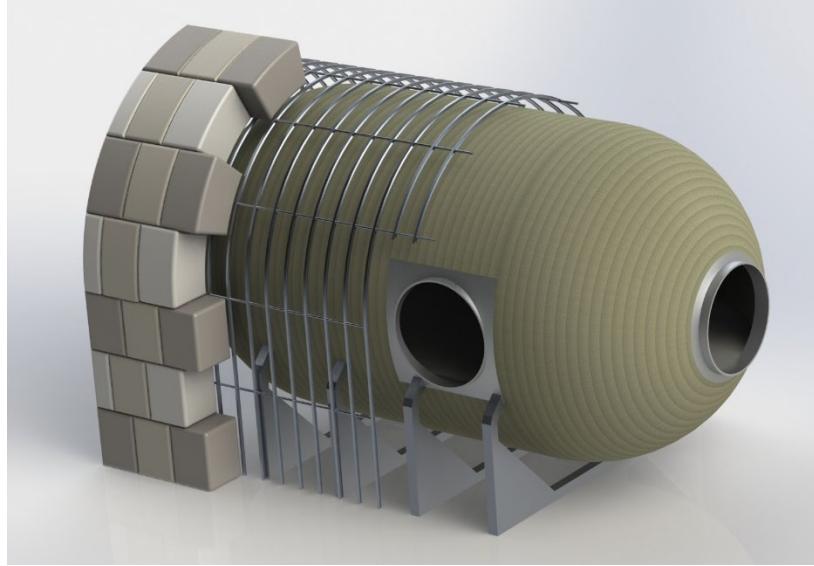


**Fig. 36 Section view of the Crew Module showing the “flex corner”.**

### M. Regolith Shielding Cage

The habitat modules require radiation shielding which is accomplished using in-situ lunar regolith. The regolith is bagged to the required shielding thickness specification, and fixed to a

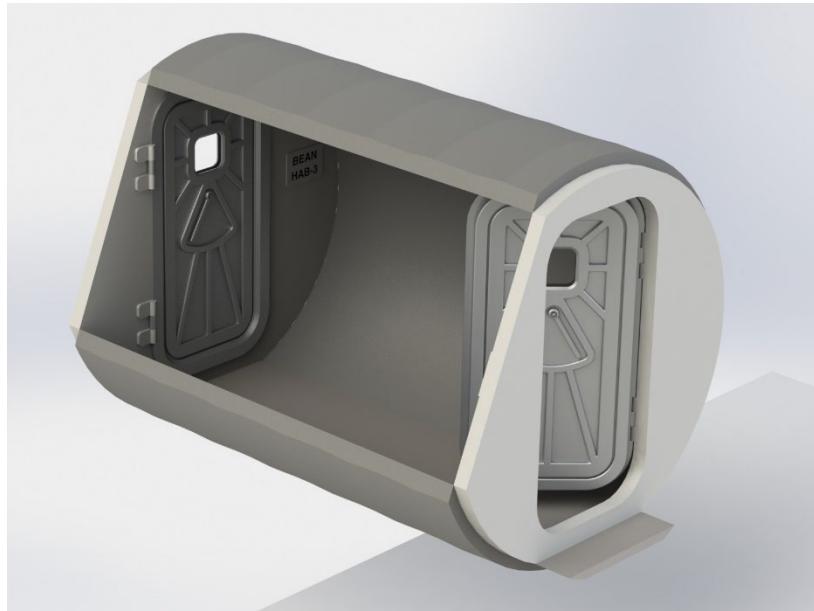
freestanding rigid I-beam cage for support. Fig. 37 shows an exterior layered view of the bagged regolith, regolith cage, and inflatable habitat structure.



**Fig. 37 Exterior view of the habitat with regolith shielding cage and regolith bags.**

#### N. Airlock Module

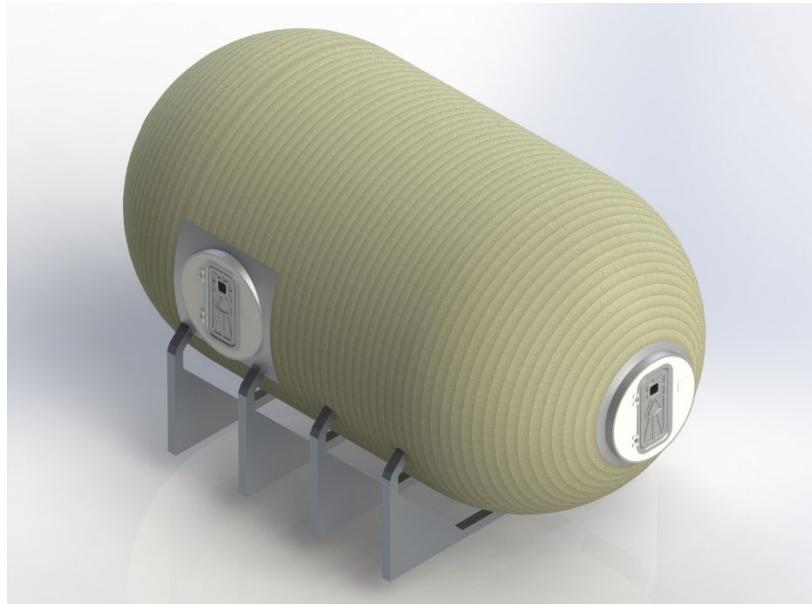
The 8 inflatable habitat modules are connected using modules that serve as airlocks. While every connected airlock is capable of serving as an airlock with exterior lunar access, only 2 airlocks are designated for exterior access. The remaining interior airlocks serve as emergency backup should one of the habitat modules have a leak or become depressurized. Fig. 38 shows a section view of an interior airlock with no exterior access.



**Fig. 38 Section view of an airlock module connecting two habitat modules.**

## O. Inflatable Habitat Design

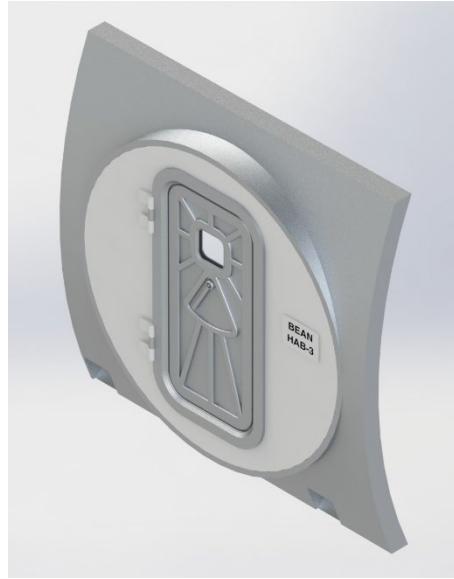
The inflatable habitat design serves as the main module type for the full habitat. The inflatable bladder is a woven barrier designed to be transported to the moon uninflated and tightly packed. Once on the lunar surface, the bladder is inflated, and the floor and hardpoint support structure is installed. In order for a single module to connect to other modules, rigid hard ports are woven into the inflatable material to allow for rigid doorways and connections between modules. Fig. 39 shows the exterior view of an inflatable habitat module with hard ports and a hardpoint support structure.



**Fig. 39** Exterior view of the inflatable habitat design.

#### P. Hard Port Design

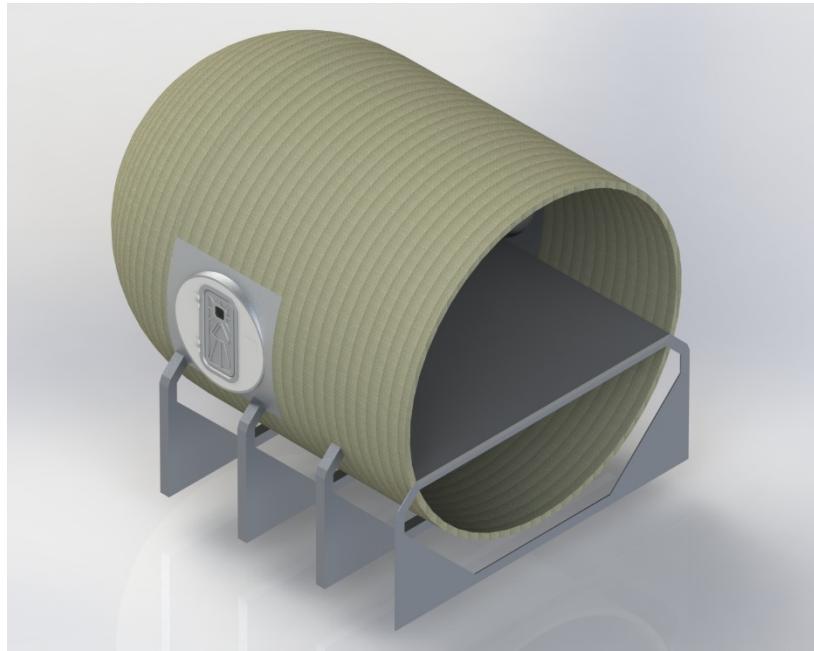
The hard ports for the inflatable habitat we designed to allow modules to connect to each other with full modularity. The rigid ports are woven into the inflatable habitat bladder to allow the modules to stay pressurized when inflated. Fig. 40 shows the hard port design for the side of the inflatable habitat module.



**Fig. 40 Hard port design for the side of the inflatable habitat module.**

#### **Q. Hardpoint Supports**

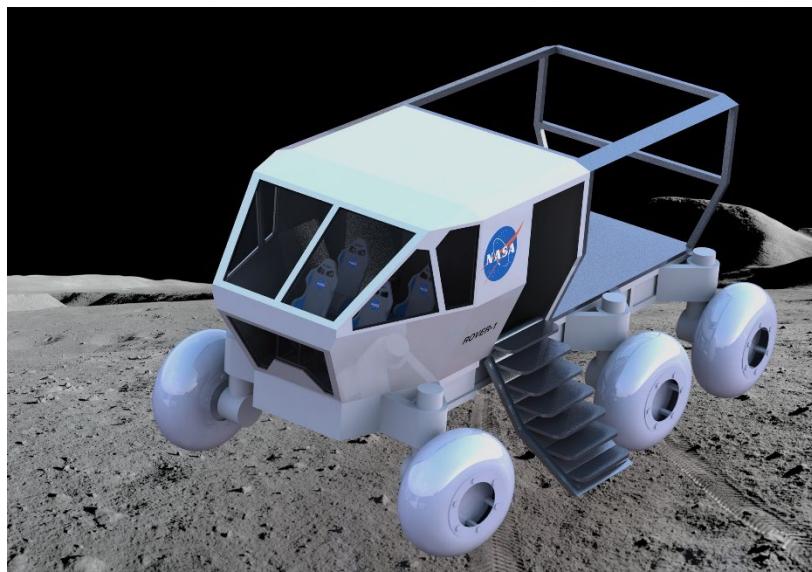
Since the habitat is inflatable and circular, the module needs to have a support structure that anchors the floor and prevents the module from rolling. The hardpoint supports shown in Fig. 41 are passed through the inflatable habitat layer with sealing gaskets and support the habitat floor from underneath. These hardpoint ribs are spaced along the long axis of the habitat to ensure adequate support along the entire habitat.



**Fig. 41 Section view of the hardpoint supports integrated into the inflatable habitat module.**

## R. Modular Construction Rover

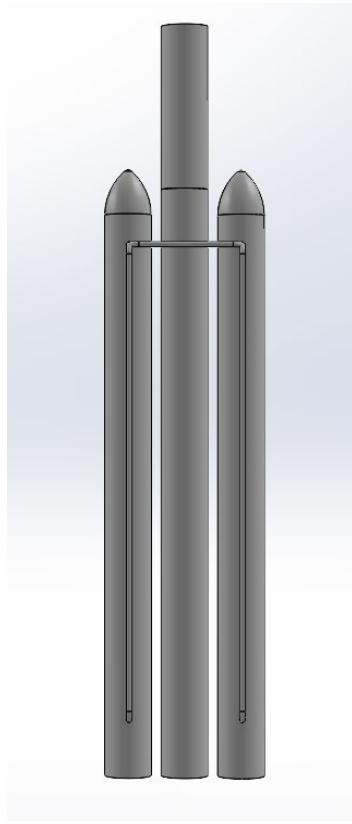
The overall habitat system on the lunar surface needs a rover to assist with the construction and positioning of the inflated habitat modules and airlock modules. The rover vehicle shown in Fig. 42 is designed to have a modular rear deck that accommodates different equipment like extra storage or construction cranes. The cabin is designed to be pressurized so the crew members can remain inside the rover while operating it and avoid dealing with EVA suits. The rover cabin sits 4 crew members, and is connected to a rear airlock module with a flip down door to reach the lunar surface.



**Fig. 42 Exterior view of the Modular Construction Rover.**

## S. Falcon Heavy

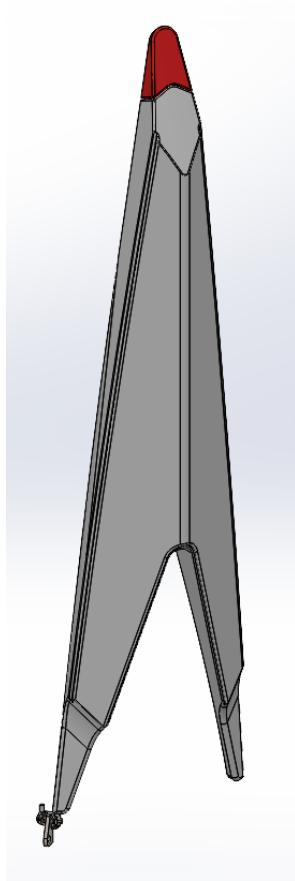
SpaceX's Falcon Heavy was selected as the payload delivery vehicle for the initial large cargo launches in the first phase of the mission. A CAD model was not requested, but the CAD team is including a launch animation in the video overview of the mission. Several subsystems are modeled independently to create the assembly. Those subsystems are the (1) base structural stages, (2) landing legs, (3) gridfins, and (4) payload module and fairing. A model of the base structural stages, which include the Falcon Heavy's two core stages and two boosters is shown in Figure 43.



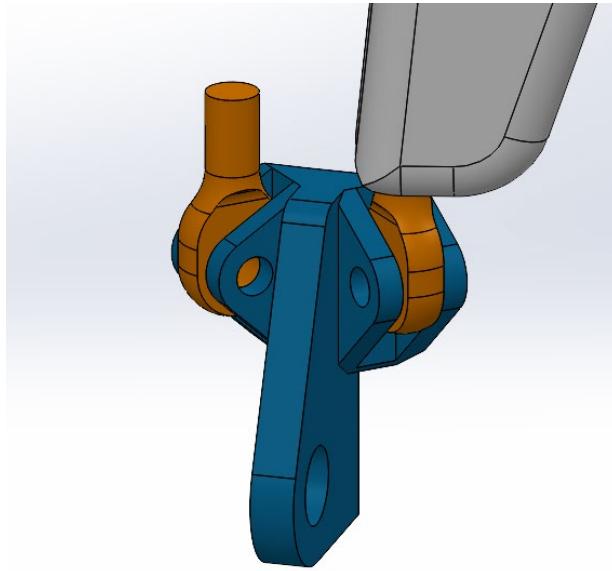
**Fig. 43 Falcon Heavy Core Stages and Boosters**

At the base of the boosters are the deployable landing legs, which are composed of the legs, the leg shroud, and the hardware that allows for deployment. The geometry of the legs is rather

complicated and there is no published material in which exact sizing can be found, so a number of educated guesses have to be made. A view of the landing leg assembly is shown in Figure 44. The legs are depicted in white and the leg shroud in red. A close-up view of the deployment mechanism is shown in Figure 45 with the mount depicted in blue and the rotating bolts in orange.

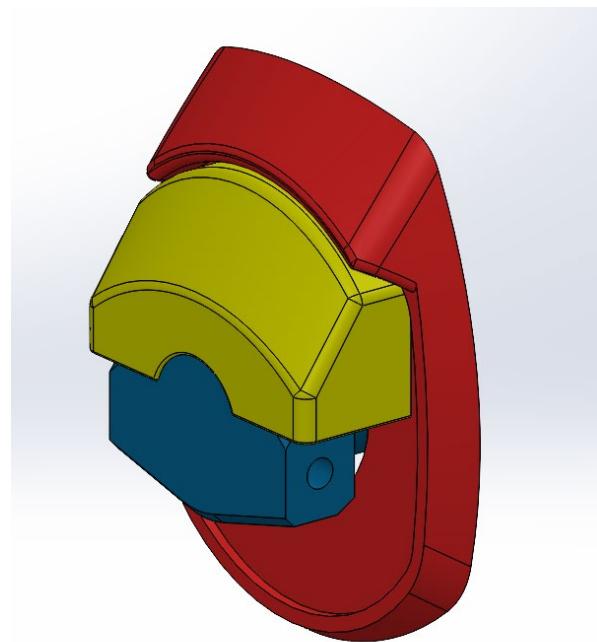


**Fig. 44 Falcon Heavy Landing Leg Assembly**

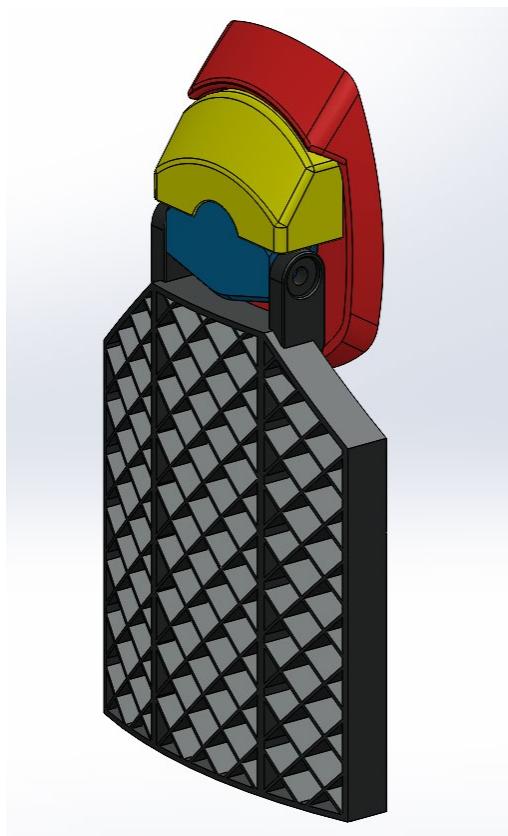


**Fig. 45 Falcon Heavy Landing Leg Rotational Deployment Mechanism**

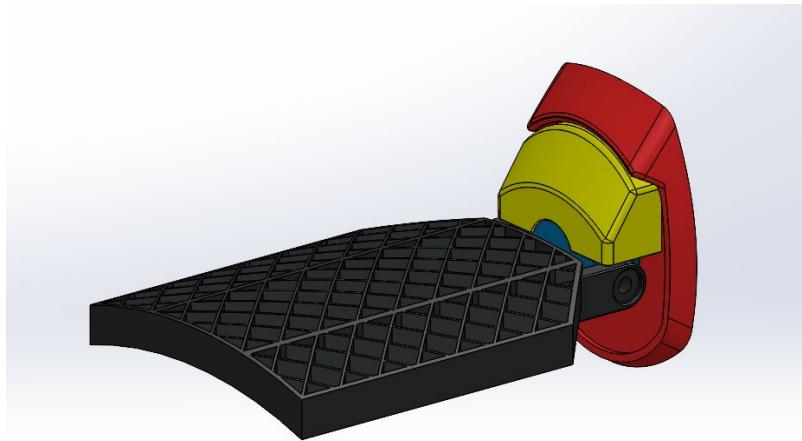
The gridfin assemblies are located at the top of the Falcon Heavy boosters. As with the landing legs, the geometry is complicated and there is little to no reference material that lists exact dimensions or geometry, so educational guesses are made once again. The gridfin assembly is composed of the external mounting bracket, control mount, mount shroud, and gridfin. Figure 46 depicts the mounting hardware with the bracket shown in red, the mount shroud in yellow, and the control mount in blue. The entire gridfin assembly is shown with gridfins stowed in Figure 47 and deployed in Figure 48.



**Fig. 46 Falcon Heavy Gridfin Mounting Hardware**



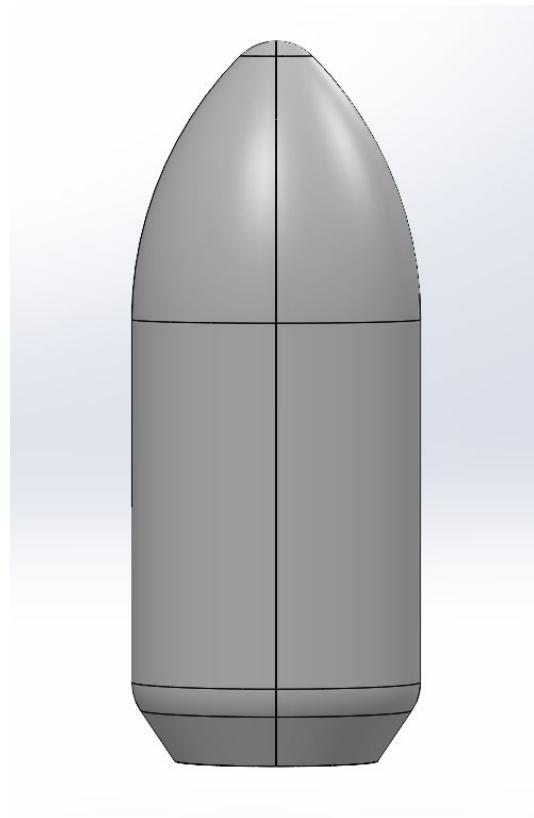
Blake Gossage

**Fig. 47 Falcon Heavy Gridfin Assembly (Stowed)****Fig. 48 Falcon Heavy Gridfin Assembly (Deployed)**

The payload assembly sits on top of the Falcon Heavy's second stage and is composed of the payload module, module adapter, and fairing. The payload module and adapter are shown together in Figure 49 with the module in blue and adapter in black. Exact dimensions are available in the Falcon Heavy's payload user guide published by SpaceX, so the model is geometrically accurate. The fairing, which was constructed in two symmetrical parts, is shown in Figure 50. A rendered close-up of the payload assembly (module highlighted in gold) in Figure 51 and the entire rendered launch vehicle model is shown in Figure 52.



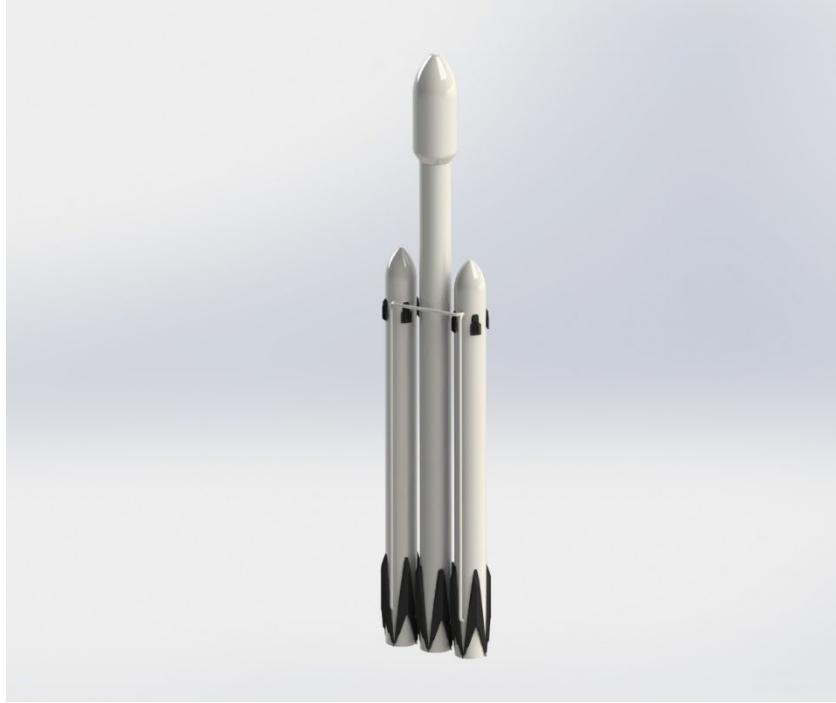
**Fig. 49 Falcon Heavy Payload Module and  
Module Adapter**



**Fig. 50 Falcon Heavy Fairing**



**Fig. 51 Falcon Heavy Payload Assembly (Cutaway)**

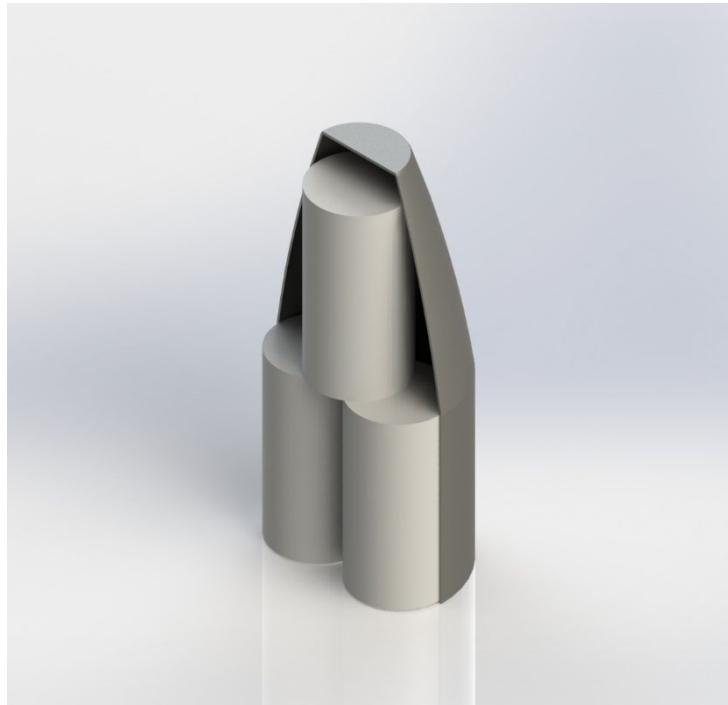


**Fig. 52 Falcon Heavy Launch Vehicle**

#### T. Starship Payload Module

SpaceX's Starship was selected as the launch vehicle for the later stages of the mission. The Structures team requested that the Starship payload be modeled for demonstration of configuring

three collapsed habitats inside the module. As with the Falcon Heavy payload module, exact dimensions are given in the payload user guide, so the model is geometrically accurate. Figure 53 shows the payload module with three habitats inside, the volume of each being represented by a single cylinder.



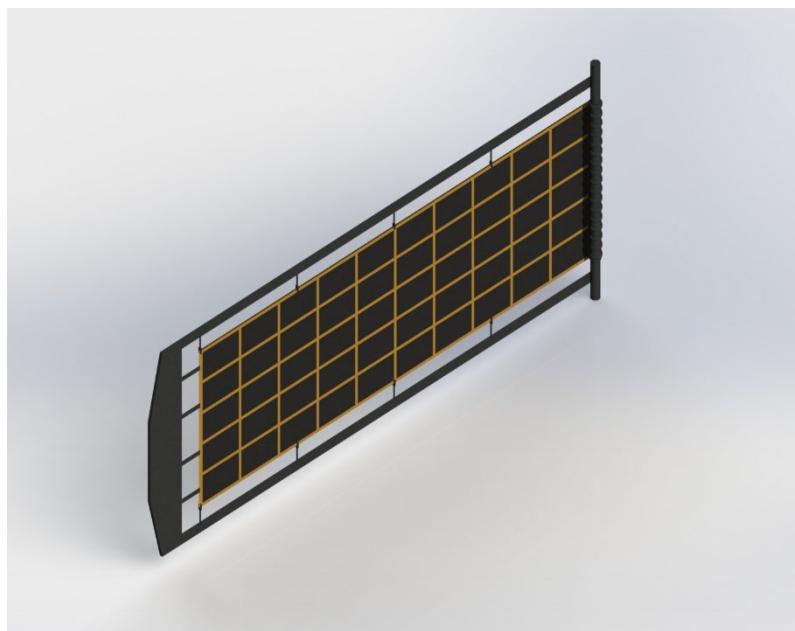
**Fig. 53 Starship Payload Module with Three Compressed Inflatable Habitats**

## U. Deployable Solar Panels

The Power and Thermals team requested a model of the lander-class deployable/retractable solar panels made by Deployable Space Systems. Very little information exists on the geometric or material specifications of the solar array, so the model is constructed based on pictures of the CAD models on the Deployable Space Systems website. Figure 54 shows the stowed system and Figure 55 shows the deployed system.



**Fig. 54 Lander-Class Solar Panel (Stowed)**



**Fig. 55 Lander-Class Solar Panel (Deployed)**

## V. Kitchen Counter

As part of a request by the Habitat Systems team for a living and dining habitat model, the kitchen counter is modeled with standard dimensions. It features two sliding drawers and a wooden countertop. The model is shown in Figure 56. No specific reference was used.



**Fig. 56 Living/Dining Habitat Kitchen Counter**

### **W. Refrigerator**

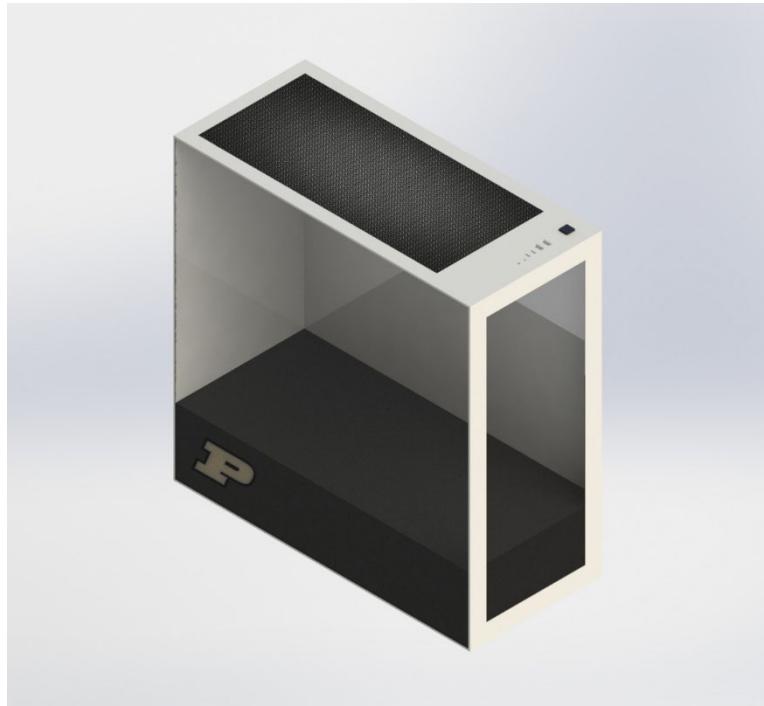
Also part of a request from the Habitat Systems team, the refrigerator is modeled using standard dimensions. It features separate refrigerator and freezer sections with rotating and sliding doors. The refrigerator section also has several shelves and storage space built into the door. The model is shown in Figure 57. Like the kitchen counter, no specific reference model was used.



**Fig. 57 Living/Dining Habitat Refrigerator Model**

## X. Computer Case

The Human Factors team requested models of several commonly used items including small personal electronics. The computer case is modeled to reflect the design principles of several popular enthusiast-grade computer cases. A mesh filter is placed on top of the case to prevent dust buildup inside the case. A glass side panel and front panel are used to provide easy visual access to the components. No specific computer hardware was requested, so the case model is empty. The case features a suite of front panel input and output interfaces including two USB type A, two USB type C, and 3.5 millimeter microphone/headphone audio jacks. The model is shown in Figure 58.



**Fig. 58 Computer Case**

## **Y. Computer Keyboard**

Accompanying the model of the computer case is a keyboard model. This model, requested by the Human Factors team, is designed with reference to the Apple magic keyboard because of its small, simple form factor and lack of wires. It features a miniaturized 60% layout and an aluminum chassis. The model is shown in Figure 59.



**Fig. 59 Computer Keyboard**

## Z. Personal Tablet

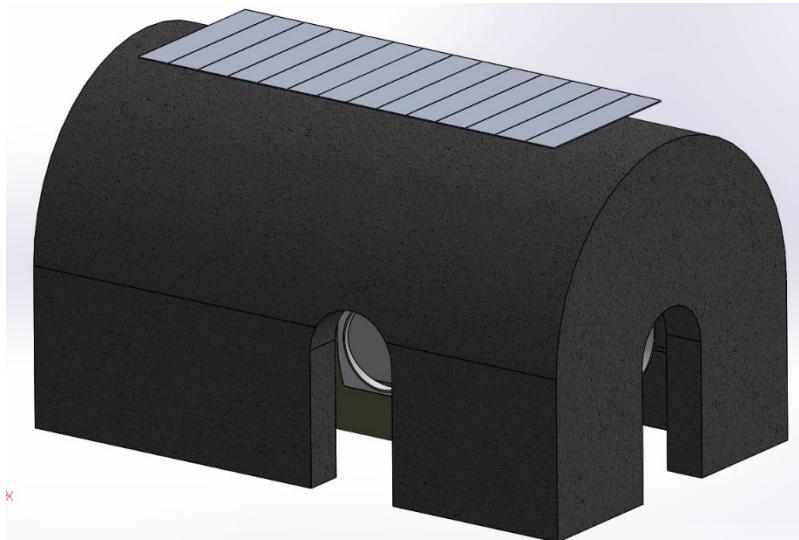
The personal tablet model is designed with reference to the Apple iPad. It was requested by the Human Factors team. The model features thin bezels, rounded corners, a glossy black chassis, and a large screen. A power button is located on the right side and a USB type C charging/data port is found on the bottom, alongside a 3.5 millimeter combination headphone/microphone jack. The model is shown in Figure 60.



**Fig. 60 Personal Tablet**

## AA. ATCS

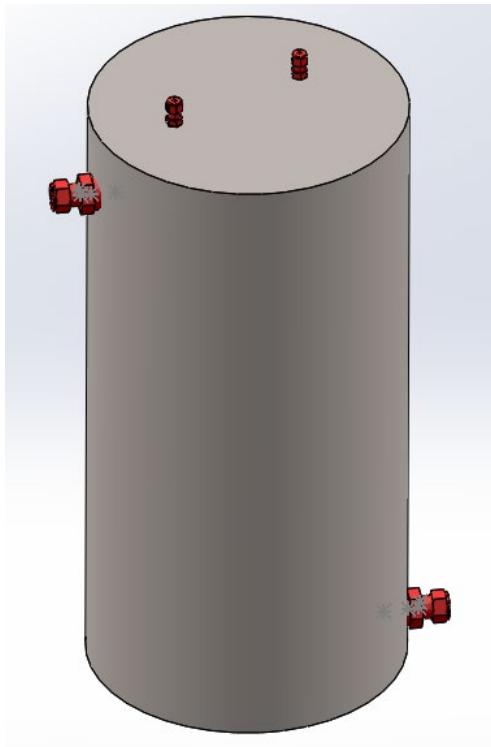
Power and thermal requested a model of the Active Thermal Control System (ATCS) to visualize its placement on top of a habitat module, and determine dimensions of the radiators. In order to complete this it was determined that the radiators should consist of a set of 1 meter by 5 meter sections that would allow it to fold up in order to occupy less volume for the trip to the moon. The area specified by power and thermal was 70 meters squared which required 14 sections to fill. These sections were attached with hinges so they could accordion out much like the ATCS on the International Space Station. A view of the deployed radiators on the habitat module is pictured below. In order to attach to the habitat module a bracket was modeled which could be bolted into the regolith using lag bolts, and then the radiators could be bolted to the bracket. The radiators are mounted horizontal on the top of the module as they are required to have a 90 degree incidence angle to the sun for maximum efficiency.



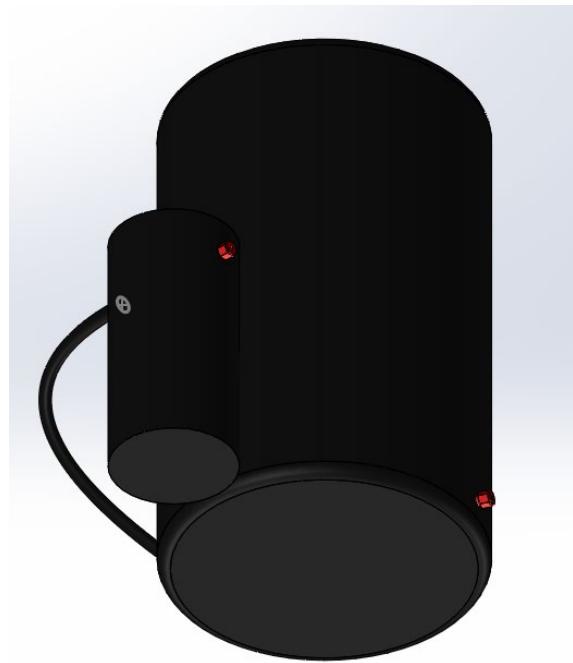
**Fig 61: ATCS deployed on habitat module.**

In addition to the radiators the ATCS consists of a few other components which were also modeled. The components modeled include: heat exchanger, pump and filter assembly, cold

plates, and tubing. The tubing proved difficult to model, and due to the model of the entire flow loop not being required the components were determined to be sufficient. The heat exchanger has four ports on it. Two for the internal loop and two for the external loop. This can be seen below. The pump and filter assembly is a convenient package that combines these two essential components. This assembly can also be seen below. The cold plates are essentially the same model as the radiator at a smaller scale, and inside the habitat instead of external.



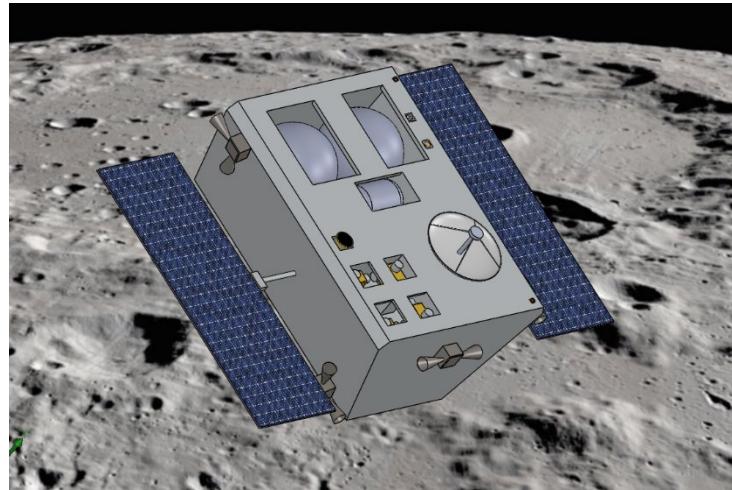
**Figure 62: Heat Exchanger Assembly**



**Figure 63: Pump and Filter Assembly**

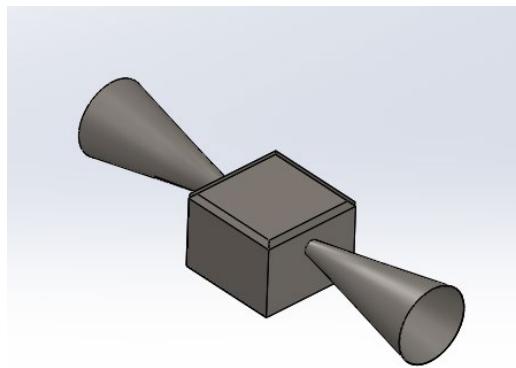
## BB. Communication Satellite

The satellites vehicles team requested a model for the communications satellites. The initial direction was to use the Iridium NEXT satellites as a basis for the approximate size and weight of the satellites while the actual values were figured out. The initial goal was to provide some baseline moment of inertia values for the GNC team to use for designing the reaction wheel assembly. In order to get these values a basic model was created using the dimensions provided. This allowed for using the Solidworks mass properties tool to quickly and easily obtain moment of inertia values as well as surface areas. This model only included the satellite bus and solar panels. Once the dimensions of the actual satellite were determined the model was updated to have the appropriate dimensions. In addition to updating the dimensions of the model the full sensor suite and propulsion system was added. The thrusters were placed as defined by the satellites team. The propulsion system also includes the propellant tanks as well as the helium pressurant tank. These tank dimensions were determined to fit the ideal propellant volume. The sensors included sun sensors and star trackers. In addition to these sensors there are three types of antennas on the satellite. In order to approximate the mass of the designed communications satellite the material of each component was defined in Solidworks. After defining all the materials the dry mass was within 50kg of the mass defined by the design team. Below is the assembly of the satellite.



**Figure 64: Communications Satellite Assembly**

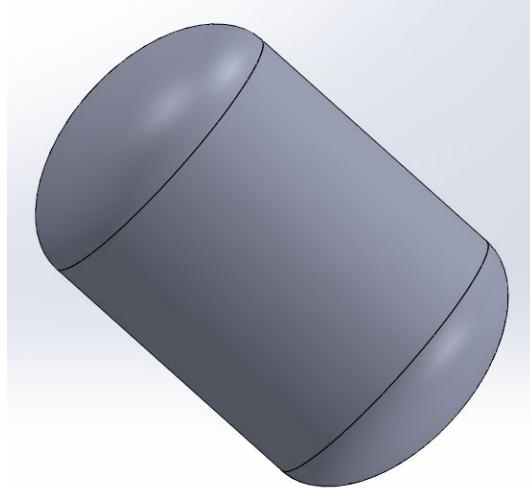
Below is a view of the thrust pack which can be seen on various parts of the satellite. This consists of the nozzle, and the pack itself which would contain the combustion chambers for each nozzle and the various tubing to provide propellants to each combustion chamber. The nozzle areas were defined by the propulsion team.



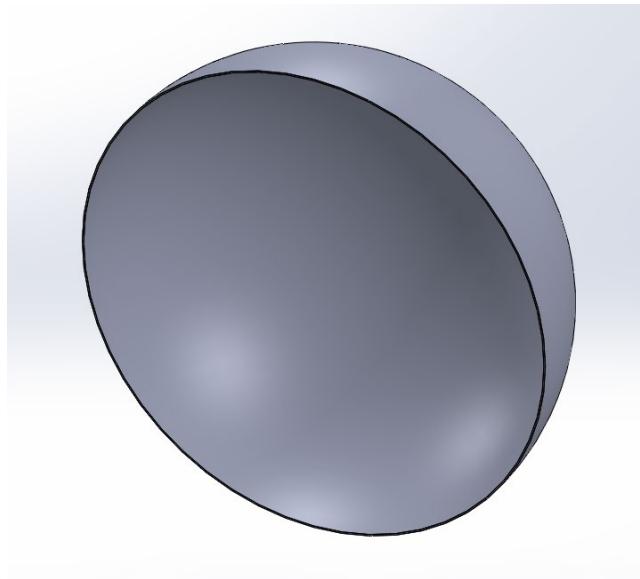
**Figure 65: Communication Sat Thrust Pack**

The other part of the propulsion system that was modeled was the tanks. These can be seen below, and were created using the revolve command after the desired outline of the tank was drawn. Then to hollow out the tanks and provide a more accurate mass the shell command was

used. This process yielded the propellant and pressurant tanks with accurate volumes and masses.



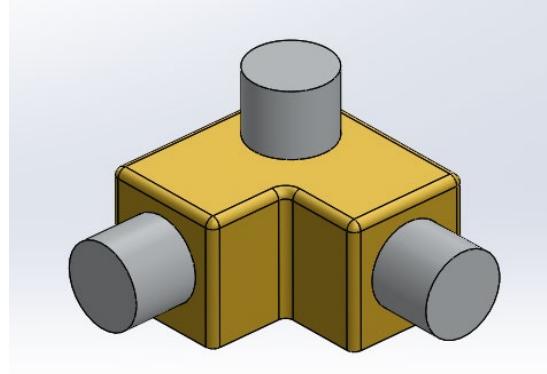
**Figure 66: Pressurant Tank**



**Figure 67: Propellant Tanks Section View**

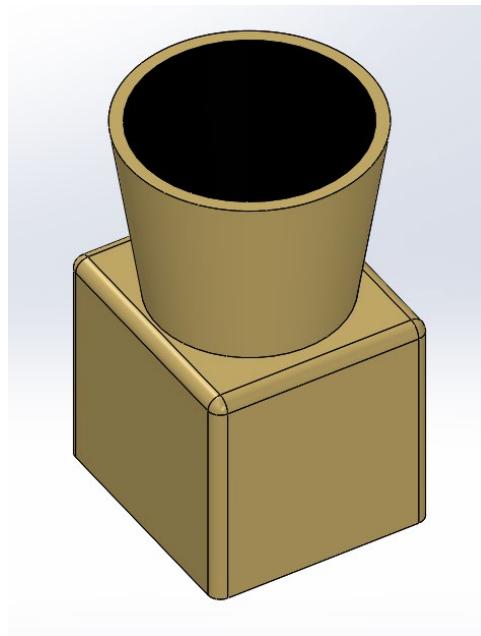
In order to maintain orientation of the satellite a reaction wheel assembly is included. As defined by the satellites design team the communications satellite includes four of these assemblies. The size was determined based on the control requirements given by the GNC team.

The general design was based on reference material found online, but not on any one model in particular. The mass of the wheels is accurate, and the materials are true to the real assembly.



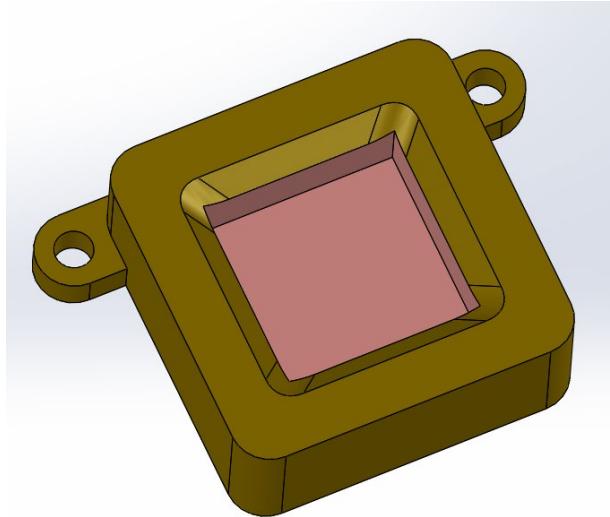
**Figure 68: Reaction Wheel Assembly**

The sensors on the satellite firstly include star trackers. These were fairly simple models to create, as they are essentially just cameras. The dimensions were based on a sample provided by the design team. To give a more realistic view color was added to give the appearance of depth in the lens.



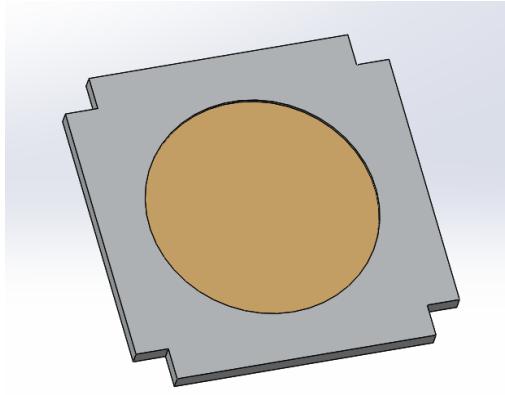
**Figure 69: Star Tracker**

Another sensor that was included on the sat was a set of sun sensors. These allow the satellite to determine its position with respect to the sun. They were also pretty easy to model, and can be seen below. The outer part of the sensor is brass, and is given that correct material in Solidworks. The interior is more complex, but was just approximated as a copper sensor inside the brass body.



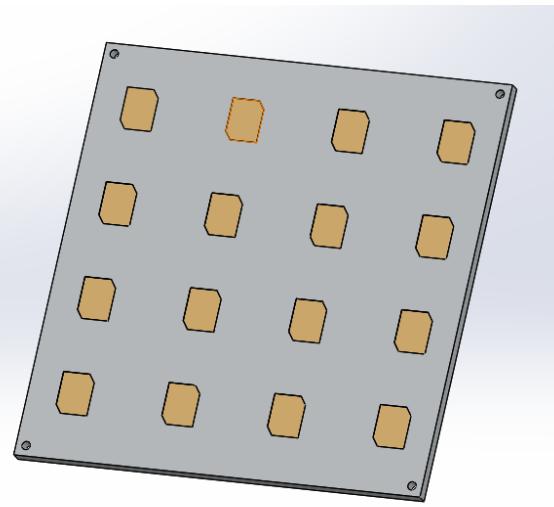
**Figure 70: Sun Sensor**

The most important parts of the communications satellite are no doubt the several antennas that are included. Firstly, and most simply, is the S-band antenna. This consists of a nickel alloy frame around a brass antenna cover. The model was fairly easy to create using only the extrude, cut, and fillet tools in Solidworks.



**Figure 71: S-Band Antenna**

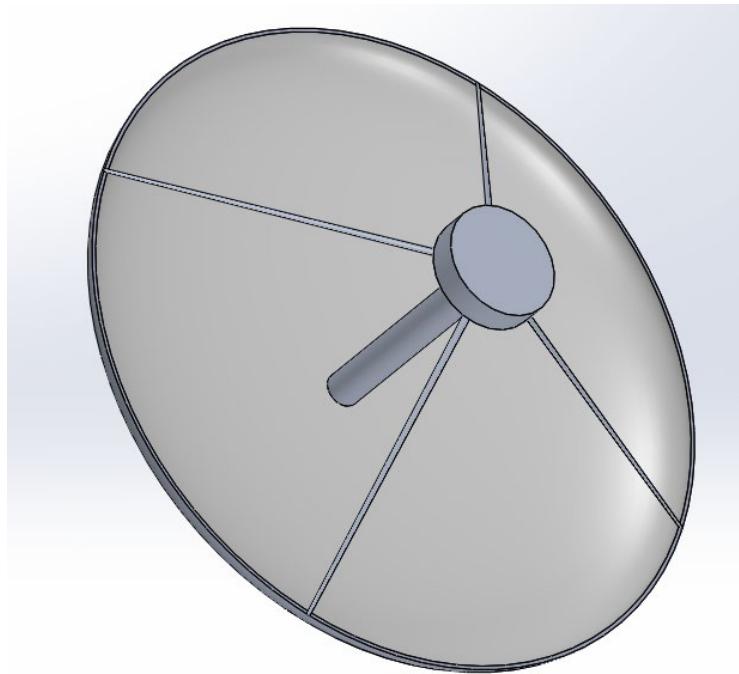
The second antenna on the satellite is the X-Band Antenna. The process for modeling this antenna was largely the same as the S-Band Antenna, except for the shape of the antenna elements.



**Figure 72: X-Band Antenna**

The largest and most difficult antenna to model was the KA-Band Antenna. It is easy to tell just from the picture of it below that this was the most difficult antenna to model. In order to model this antenna the revolve command was used to create the dish. Then the central post was extruded out. The most difficult part was the spars that connect the central tower to the dish.

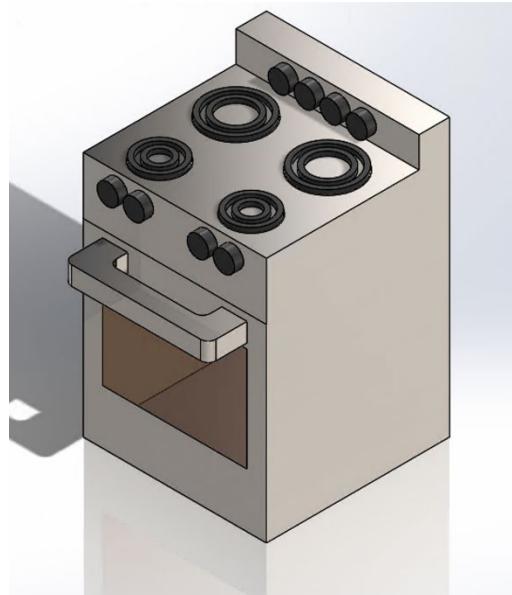
These were created with the loft command and making use of reference planes to provide end points.



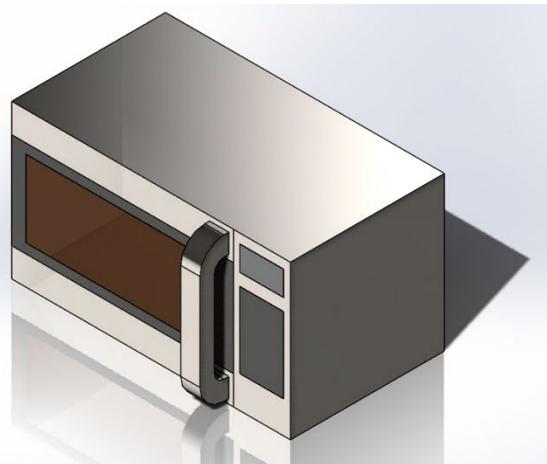
**Figure 73: KA-Band Antenna**

### CC. Interior Pieces of Living Space - Microwave and Oven/Stovetop

The microwave and oven/stovetop were modeled in order to be included in the layout of the interior of the living and dining modules designed by Lara Cackovich of the Human Factors team. The model assembly of these modules was done by John Matuszewski. The two models can be seen below in Figures 74 and 75.



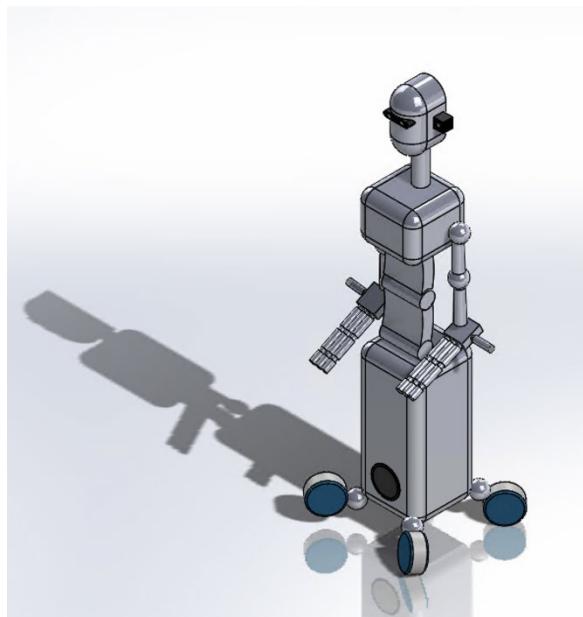
**Figure 74: Oven/Stovetop for Interior Living and Dining Modules**



**Figure 75: Microwave for Interior Living and Dining Modules**

## DD. Assistant Robot

The Assistant Robot was an idea created by the Interior Vehicles subteam. Its purpose is to assist in tasks in the habitat. These tasks include things such as habitat maintenance and autonomous experiments. The team also decided that the Assistant Robot should be able to perform exterior maintenance on the habitat. This would mean the vehicle would need to be equipped with the proper material to protect it from the conditions of the Moon. Tasks it would perform on the exterior of the habitat would be things such as solar panel and radio tower repairs. Using the Assistant Robot to handle menial tasks in the habitat not only lightens the workload of the humans living in the colony, but also takes away the risk of injury from performing those tasks. The vehicle has a volume of 1.3 cubic meters, a mass of 204 kilograms, can travel at 1.5 meters per second, and requires 120 kilowatts of power to operate, according to the Interior Vehicles subteam. The Assistant Robot can be seen below in Figure 76. It has cameras in its eyes and



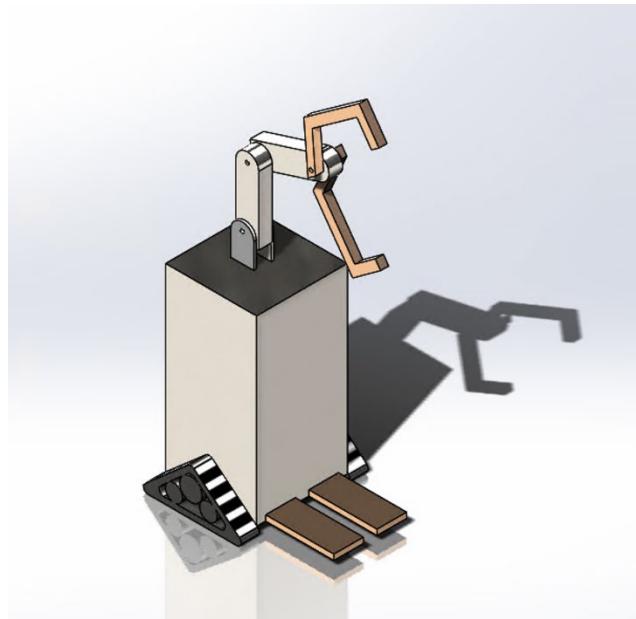
**Figure 76: Assistant Robot**

ears as well as on the front and back of the torso. The legs and arms rotate about the spheres. The torso rotates about the cylinders, allowing the vehicle to get taller and shorter. The hands work like human hands with full wrist and finger mobility. The head also rotates about the neck piece.

### **EE. Heavyload Robotic Arm**

The Heavyload Robotic Arm was created by the Interior Vehicles subteam in order to lift heavy loads (about 2.5 to 3 times the body weight of an average human being), clean the habitat, assist with autonomous experiments, and help with precision control through things such as replacing components and assisting with electrical work. According to the Interior Vehicles subteam, the volume of the Heavyload Robotic Arm body is two cubic meters and the volume of its arm is 0.5 cubic meters. It takes three kilowatts of power to operate, weighs 350 kilograms, and can move at a speed of up to 0.8 meters per second. It has about eight degrees of freedom in its arm, three degrees of freedom in the wheels, and two degrees of freedom on the forklift features. The Heavyload Robotic Arm can also lift up to 230 kilograms and be able to reach a

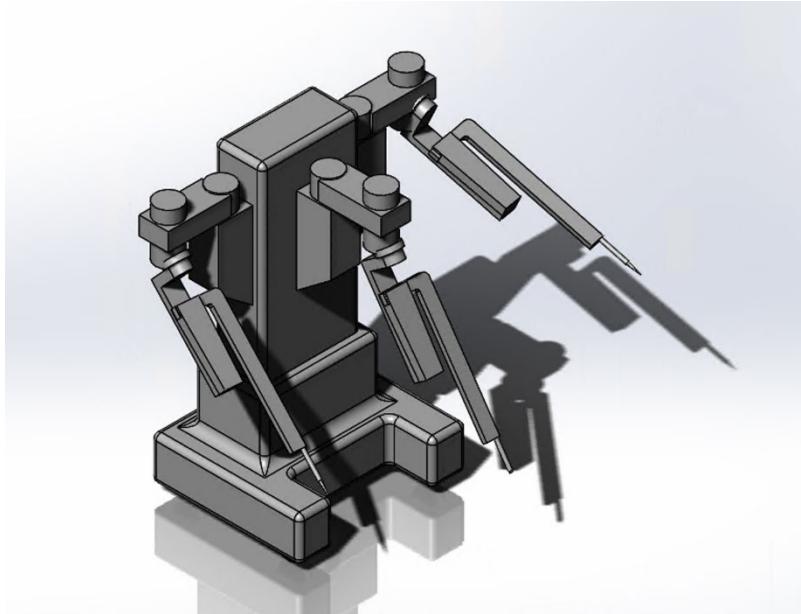
maximum height of 3.5 meters. The model of the Heavyload Robotic Arm can be seen below in Figure 77.



**Figure 77: Heavyload Robotic Arm**

#### FF. Surgical Mechanical Arm

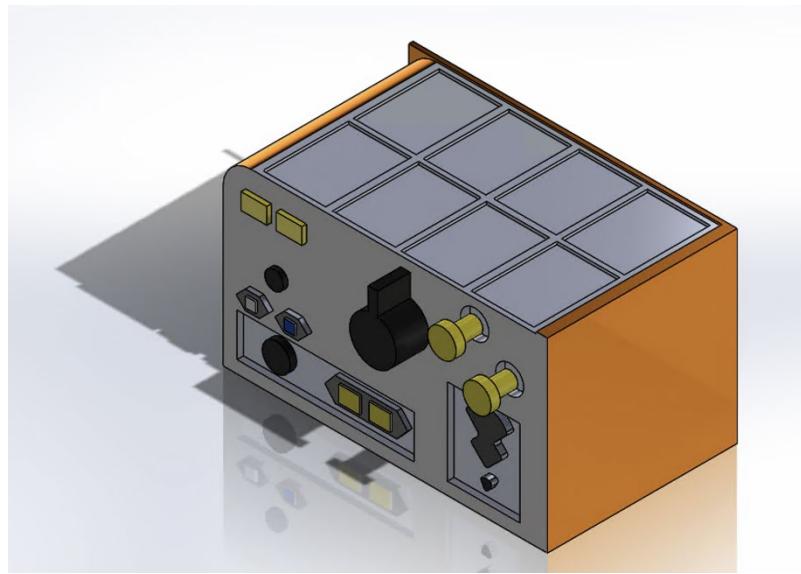
The Surgical Mechanical Arm was developed by the Interior Vehicles subteam to assist with major medical operations in the habitat. It will be able to handle things such as cardiac surgery, general surgery, and urology surgery. The total volume of the vehicle is 3.5 cubic meters, the total mass is 294.3 kilograms, and it takes 105 Watts for the vehicle to operate, according to the Interior Vehicles subteam. Also from the subteam's specifications, the vehicle is 1.78 meters tall and can reach 1.5 meters. The two arms on either side of the vehicle have claws attached to them that will be able to hold and use tools. The middle arm has a camera attached so that a screen can display the place on which the arm is operating and what it is doing. The model of the Surgical Mechanical Arm can be seen on the next page in Figure 78.



**Figure 78: Surgical Mechanical Arm**

#### **GG. Potable Water Dispenser**

The Potable Water Dispenser was designed by Kyle Alvarez of the Human Factors team to provide drinkable water to the inhabitants of the Moon colony. The system used on the habitat is based on the potable water dispenser that is on the International Space Station. The Potable Water Dispenser is 0.506 meters tall, 0.881 meters wide, and 0.516 meters deep. The model for the dispenser can be seen below in Figure 79.



**Figure 79: Potable Water Dispenser**

## HH. Human Research Lab Rack (HRL)

The HRL was designed by the Research Facilities team to handle and store the station's research experiment equipment as well as test the physiological and chemical changes in the human body when exposed to lower lunar gravity for extended periods of time. The HRL also includes an ultrasound with an ultrasound computer to measure the heart health and effects of the gravity on the astronauts. It has a volume of  $0.865 \text{ m}^3$  and a mass of 400 kg. As seen below, the HRL was modeled to have moving handles and drawers to demonstrate the storage of research instruments.



**Figure 80. Human Research Lab Rack with Ultrasonic Computer**

The ultrasonic computer is a modified laptop with a special keyboard to interface with the machine. The computer plugs directly into the ultrasound machine and is mounted on the side of the HRL for easy access.



**Figure 81. Ultrasonic Computer**

## **II. Mass Spectrometer (Research Facilities)**

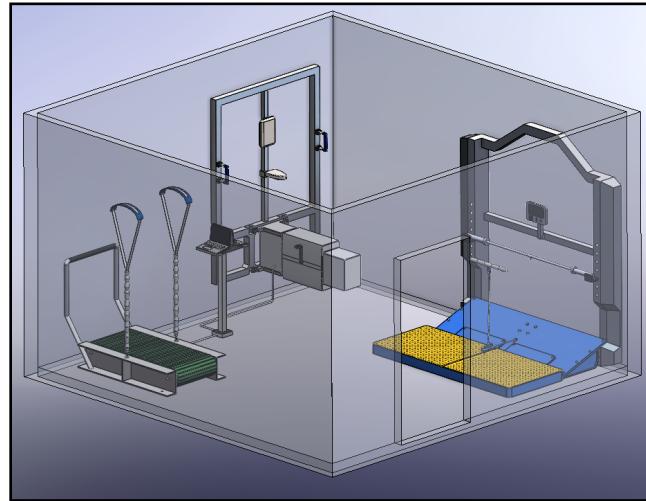
A Mass spectrometer was designed for the interior habitat for Research Facilities. The mass spectrometer is useful in measuring the elemental makeup of lunar samples brought back to the habitat. There are a multitude of mass spectrometers in the project with others being found on satellites and the scouting rover. The model was made to show its full range of options on its control panels for accurate measurements. It has a volume of  $.11 \text{ m}^3$  and a mass of 50 kg.



**Figure 82. Mass Spectrometer**

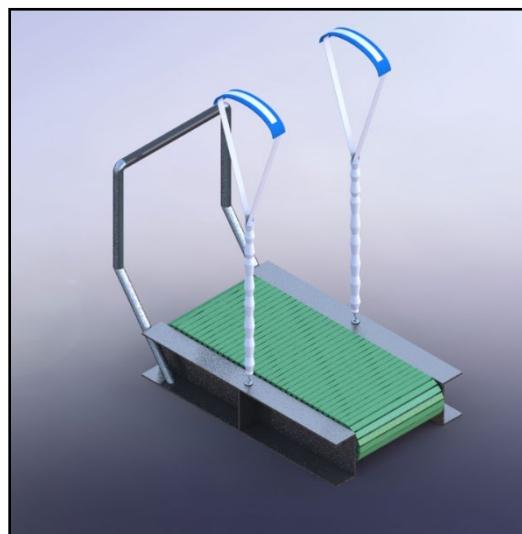
### JJ. Gym Module

The Gym Module is an assembly of 3 different exercise machines modeled for the Human Factors team. The purpose of the Gym Module is to combat the muscle atrophy and bone density loss that colonists will experience when in low lunar gravity. The combination of all three machines gives colonists a full body workout to ensure they stay healthy during their mission. The three machines are all modeled after their ISS counterparts: the treadmill machine is modeled after the COLBERT, the cycling machine is modeled after the CEVIS, and the resistance machine is modeled after the ARED. The Gym module itself was originally designed to fit in a  $50 \text{ m}^3$  room, but was later spread out in one of the habitat modules.



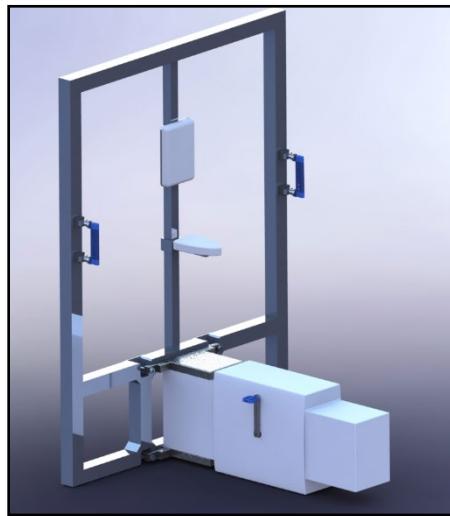
**Figure 83. Gym Module overview**

The first machine modeled for the gym module was the COLBERT. I used reference images from COLBERT to model the treadmill. However, the COLBERT was created to operate in the microgravity on the ISS, but on the moon, the colonists will be dealing with slightly more gravity. Because of this, I modified the restraints into elastic shoulder straps that would help hold the colonist down while running, but also allow them to slide in and out of the machine easily. I also added a safety bar to help in starting and stopping on the machine. It has a mass of 167.75 kg and an operational volume of  $5.5 \text{ m}^3$ .

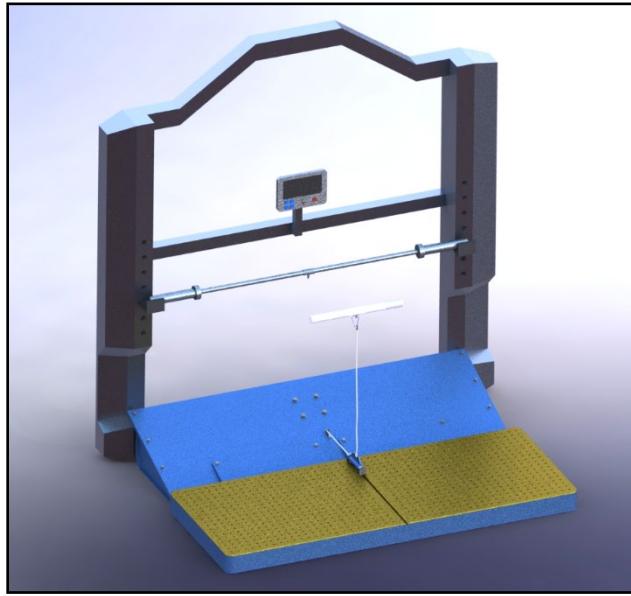


**Figure 84. COLBERT machine**

Next, I modeled the CEVIS. This model was also edited slightly from the original machine. I included some handles on the frame of the CEVIS to help in mounting and dismounting the machine. I also modeled the machine with the intention of it being bolted to a wall, since the gravity will be low enough to enable this configuration. It has a mass of 193.68 kg and an operational volume of  $4.4 \text{ m}^3$ .

**Figure 85. CEVIS machine**

The final machine I modeled was the ARED. Based on reference images, this machine had many complex moving parts for it to operate normally in microgravity. Since our model will not be in microgravity, I simplified it to better suit the volume requirements on our station. It has a mass of 490.9 kg and an operational volume of  $11.6 \text{ m}^3$ .



**Figure 86. ARED machine**

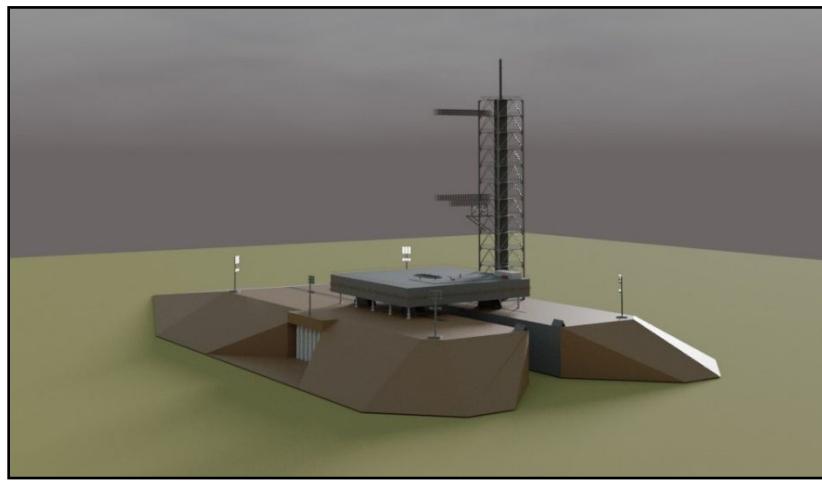
In order to control the settings and workouts on the cycling machine and the treadmill, I created a computer station with two laptops that connect to each machine. This model has a volume of  $.16 \text{ m}^3$  and a mass of 10 kg.



**Figure 87. Computer Control Station**

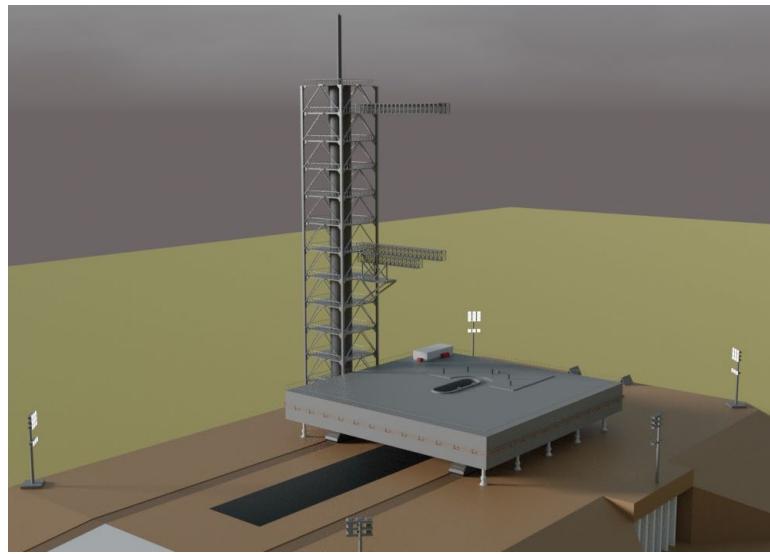
## KK. Kennedy Space Center Launch Pad (KSC)

The KSC Pad was modeled for the final video, as animations are needed showing the Falcon Heavy and Starship taking off. Since we are launching from KSC, I decided to model the launch pad off the launch pad 39a, used to take the first men to the moon on Apollo 11, as well as a host of other historic launches. I modified the mobile launcher to handle the exhaust of the Falcon Heavy and also remade the launch tower to better fit the height of the Falcon Heavy.



**Figure 88. KSC Launch Pad**

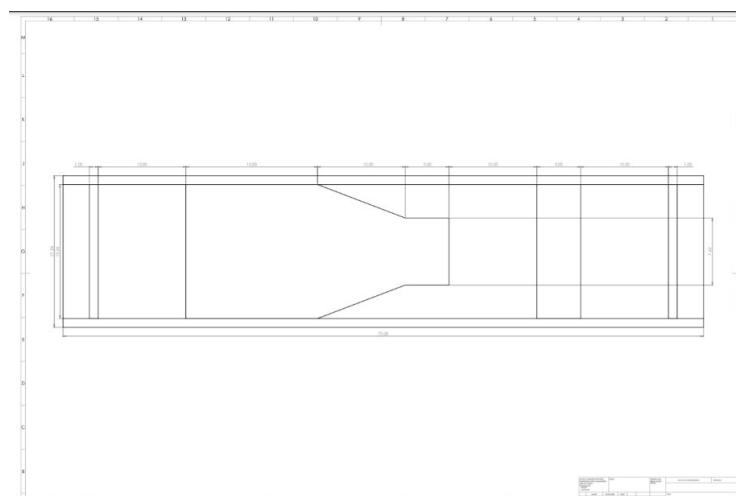
I simplified the Launch tower to have two cargo/maintenance tunnels leading towards the rocket. I tried to model the original design as closely as possible and used repeating tower “blocks” around a solid core elevator. I also tried to realistically reinforce the shuttle tunnels to ensure they wouldn’t collapse under the weight of cargo and personnel. As discussed earlier, I also changed the exhaust port of the mobile launcher to fit the Falcon Heavy, as well as modeling the maintenance pipes and buildings on top of the pad.



**Figure 89. Launch Tower & Mobile Launcher Platform**

### LL.Air Filter Drawing

An air filter drawing designed for the Habitats team. It contains 4 different filters, a screen, cyclone, and sintered metal filter to help in prefiltration and a fine filter for post filtration. This filter is important for the Habitat modules to help limit exposure to the fine aerosol of regolith that could be tracked into the habitats from outside, which could prove lethal to colonists if not properly mitigated. It has a final volume of  $.017 \text{ m}^3$ .



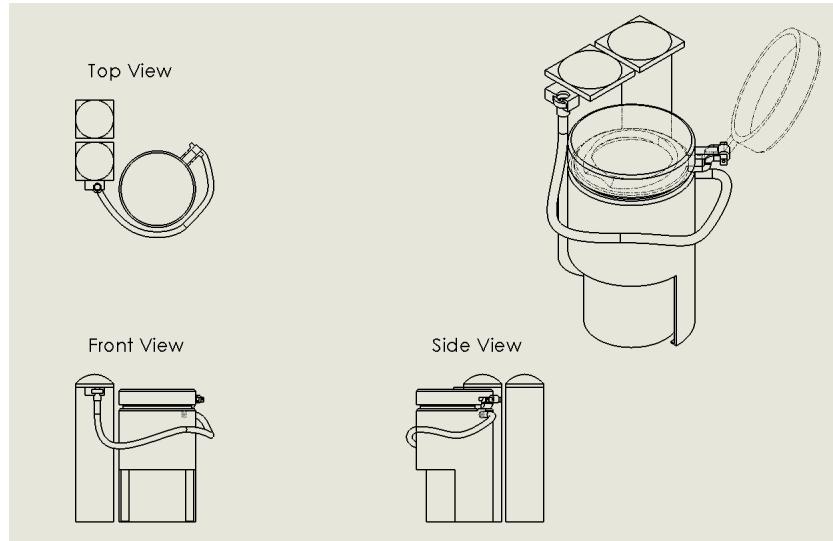
**Figure 90. Air Filter CAD drawing**

## MM. Universal Waste Management System

The Universal Waste Management System or UWMS for short is a toilet that is used onboard the ISS. It was commissioned by the Human Factors Team as a piece of necessary equipment for the everyday lives of the colonists. The UWMS was chosen due to the reduced gravity of the moon, as it has a vacuum function that reduces the risk of waste particles missing or splashing from the toilet, as well as waste storage tanks in lieu of a plumbing system (which would require water to operate). It has a volume of .34 cubic meters and a mass of 45 kg.



**Figure 91. Universal Waste Management System**



**Figure 92. UWMS CAD Drawing**

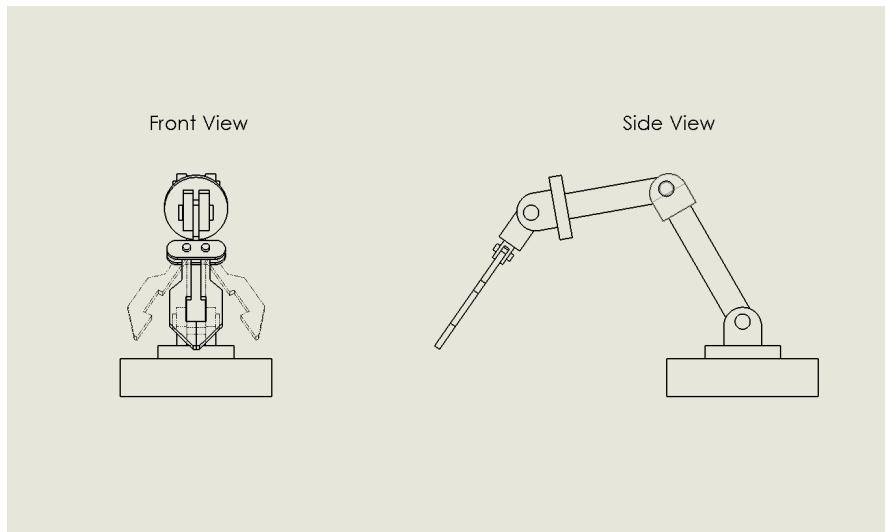
#### NN. Modular Robotic Arm

The Modular Robotic Arm is a robot arm that can be fitted with multiple attachments to serve various purposes, and is primarily intended to be mounted on the STAR (Scouting Traveler Autonomous Rover). It was commissioned by the Controls team as a piece of necessary equipment for the rover.

The first attachment is a claw that can pick up and manipulate objects, and would be used for placing beacons on the surface of the moon, among other tasks. The arm has a maximum reach of 2.88 m and the base is .96 by .96 m.



**Figure 93. Robot Arm with Claw Attachment**



**Figure 94. Robot Arm CAD Drawing**

The second attachment is a drill that can be used for collecting samples from the surface of the moon.



**Figure 95. Robot Arm with Drill Attachment**

The third attachment is a welding apparatus that can be used for various repair tasks.



**Figure 96. Robot Arm with Welding Attachment**

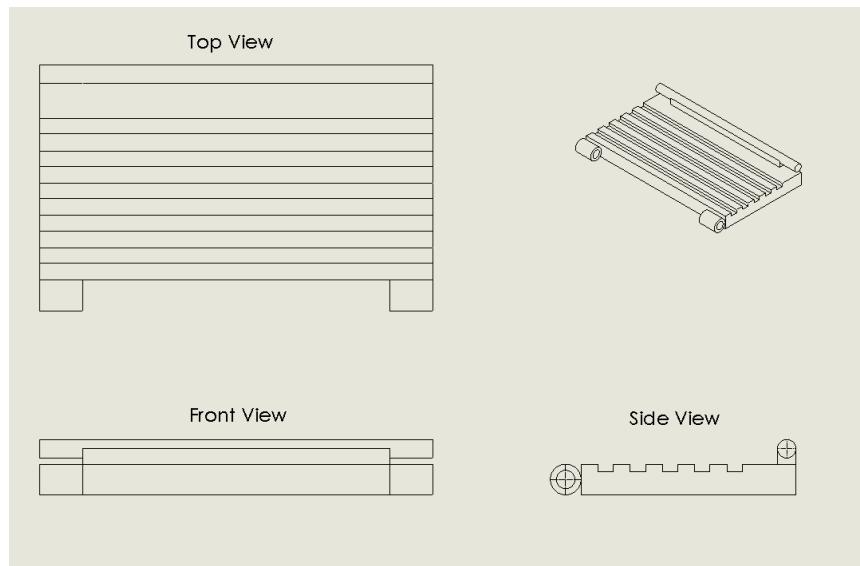
More modules may be designed and used by the arm in the future if the situation calls for it.

## OO. Lunar Lander Cargo Variant

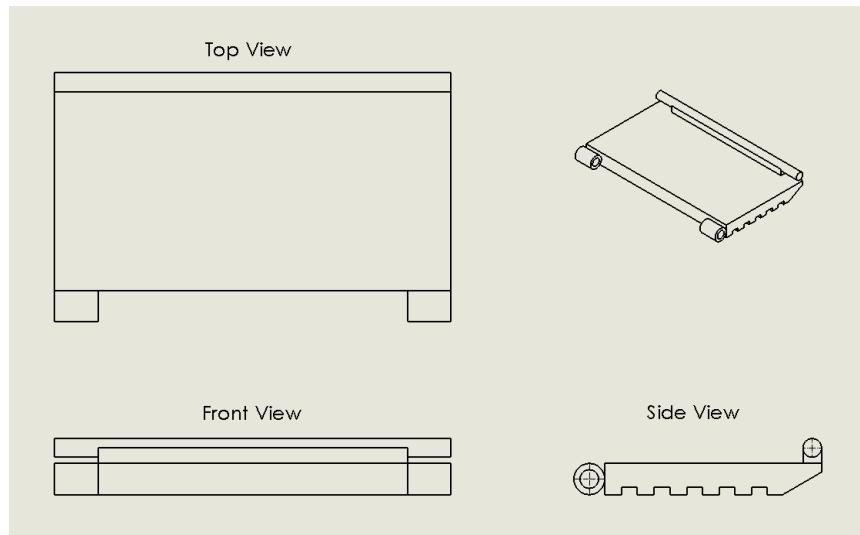
The Lunar Lander Cargo Variant is an uncrewed variant of the Apollo lander module commissioned by the Propulsion team that will be used to transport rovers to the colony ahead of colonists in order to begin activities such as scouting and mining. The lander is equipped with a foldable ramp to deploy the rovers once it has reached the moon's surface. As a side note, I did not model the Apollo lander itself, I modeled the ramp and modified the lander to deploy with it. The modeler of the lander is Dano Vinson, and I downloaded their work from GradCAD.



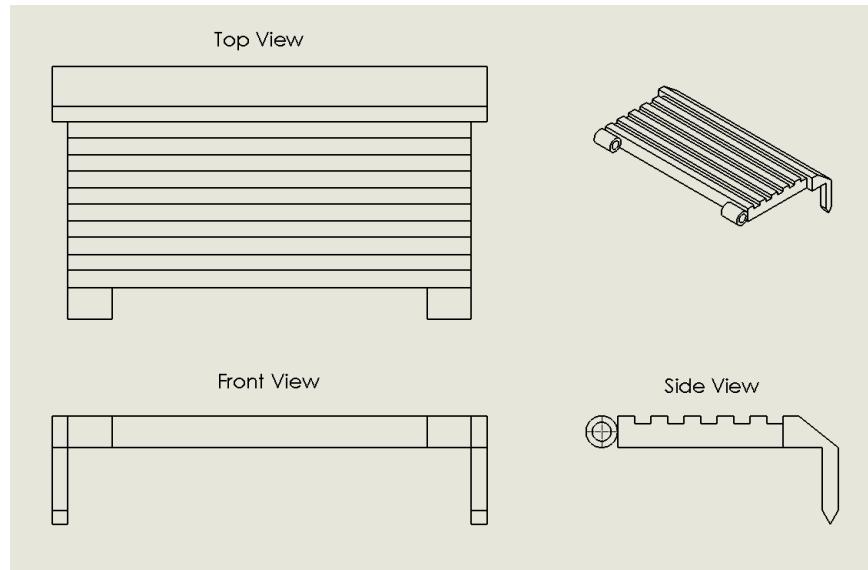
**Figure 97. Cargo Lander with Ramp Deployed**



**Figure 98. CAD Drawing of Ramp Component**



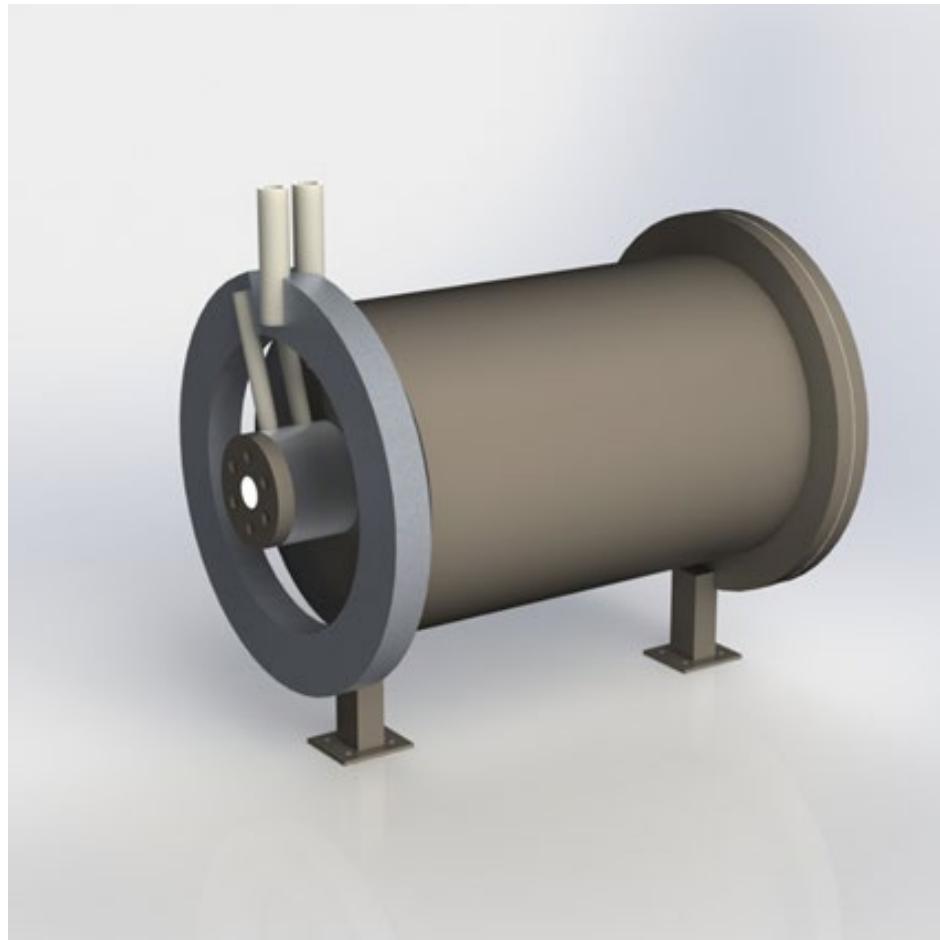
**Figure 99. CAD Drawing of Ramp Component (2)**



**Figure 100. CAD Drawing of Final Ramp Component with Anchor Spikes**

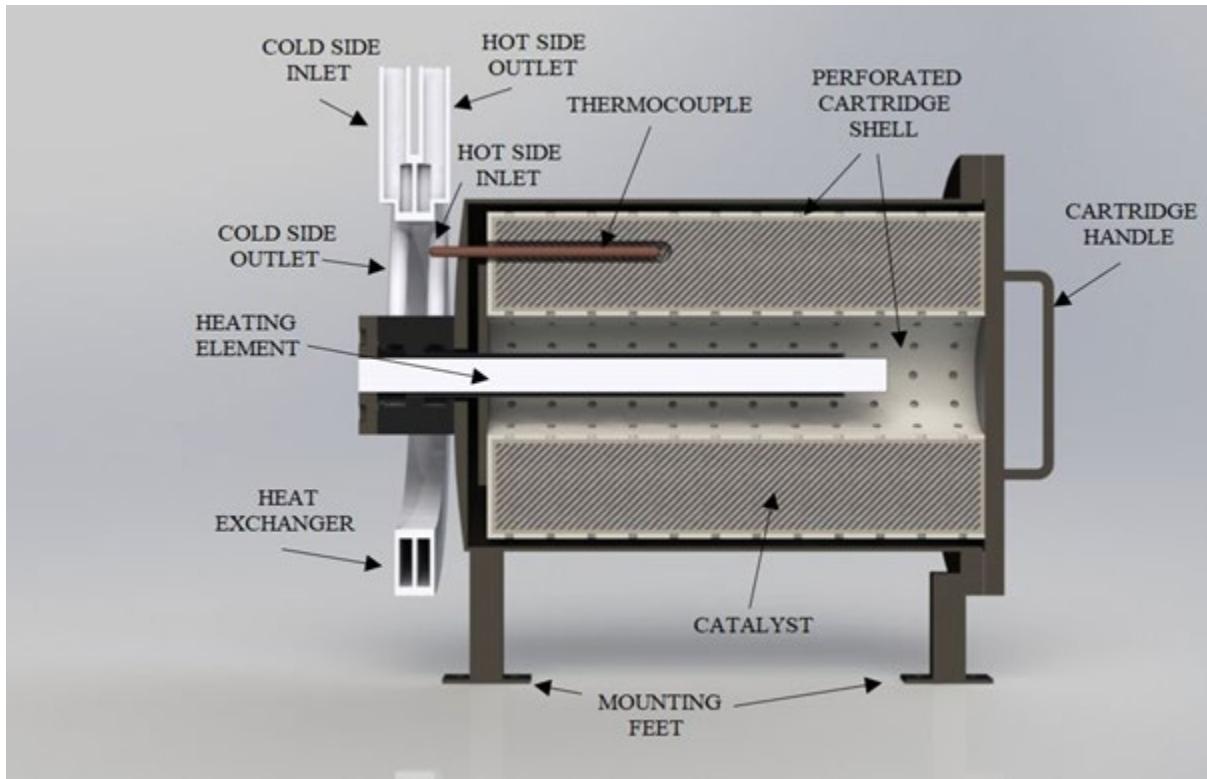
## PP. Bosch Reactor

The Bosch Reactor is an oxygen regeneration system for use in each crew module. The design was loosely based off the reactor described in “A Carbon Dioxide Reduction Unit Using Bosch Reaction and Expendable Catalyst Cartridges” by Holmes, et al. An isometric render of the Bosch Reactor can be found below in Fig. #aa.



**Fig. 101: Isometric view of Bosch Reactor with cartridge.**

The Bosch Reactor intakes CO<sub>2</sub> and removes the carbon via a catalyst. The catalyst is contained within a removable cartridge for easy replacement after the catalyst reaches the end of its life cycle. A sectioned view of the reactor with the catalyst cartridge is shown below in Fig. #ab. This reactor is sized for the capability to process the CO<sub>2</sub> output from 4 adult men. Additionally, Fig. #ab features the thermocouple placement, the heating element necessary for the reaction, the inlet/outlet ring, and other key geometry for the Bosch Reactor.

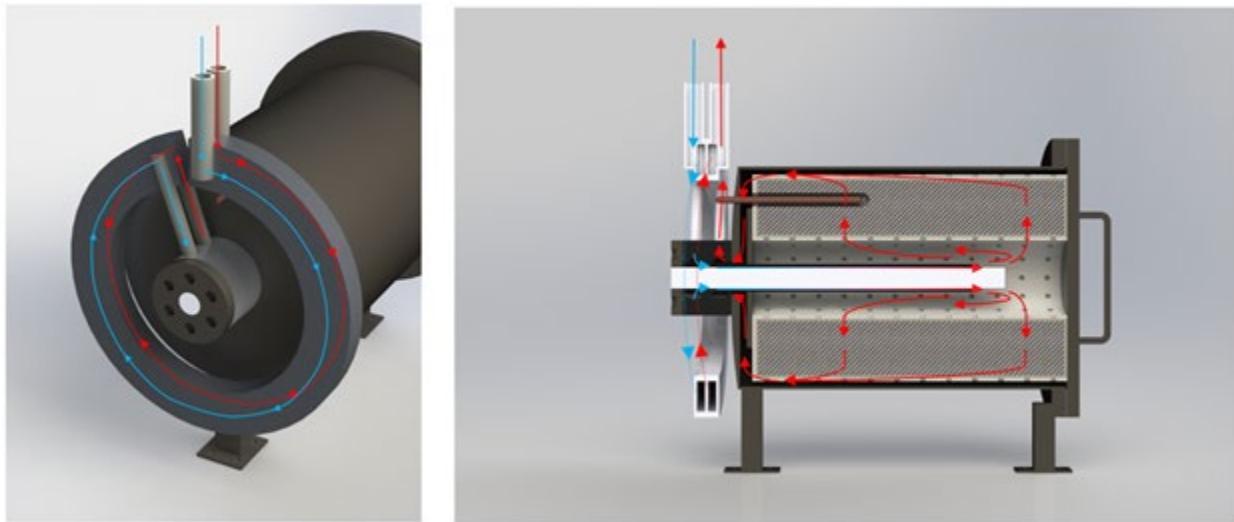


**Fig 102: Sectioned right plane view of the Bosch Reactor**

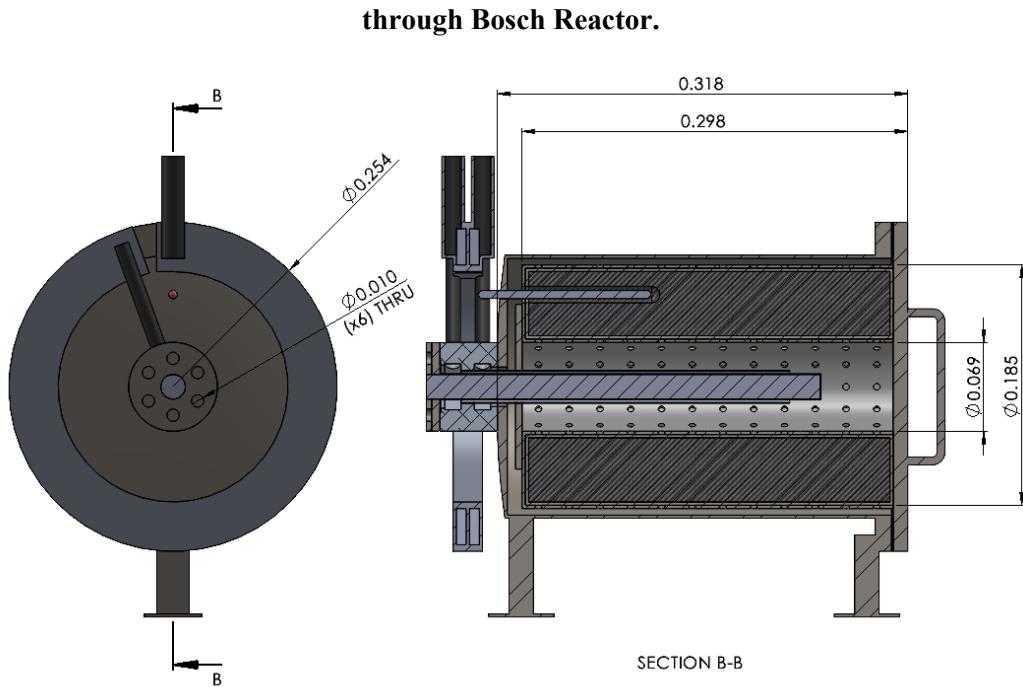
**with a catalyst cartridge.**

The inlet and outlet ring is an annular heat exchanger with a cold side and hot side. Each side is a closed channel that carries the flow around a  $345^\circ$  turn across a heat exchanger which uses the heat from the exhaust to warm the intake. The cold side is used for  $\text{CO}_2$  intake and the hot side is used to exhaust the produced oxygen. Inside the reactor, the  $\text{CO}_2$  is passed over the central heating element before reaching the catalyst. After exiting the heating element, the warm  $\text{CO}_2$  passes through the perforated cartridge shell to the catalyst where it reacts to produce oxygen. The hot oxygen then enters the heat exchanger where it is expelled to the habitat. This process can be seen illustrated below in Fig #ac.

Additionally, critical dimensions of the Bosch Reactor and the catalyst cartridge is provided below in Fig. ad illustrating the scale of the reactor.



**Fig 103: (left) flow through the annular intake ring. (right) flow**



**Fig: 104: Critical dimensions (meters) of the Bosch Reactor and catalyst cartridge.**

## QQ. Blood Centrifuge

A blood centrifuge was modeled for use in the laboratory habitats. This centrifuge was designed to hold eight 10 ml test tubes. The chassis geometry was very loosely based off the Sprint<sup>TM</sup> 8 Clinical Centrifuge. The centrifuge has configurable settings and an LED screen. The chassis dimensions are 32.5 x 29.3 x 23 cm and the whole system weighs 8 kg. An isometric render is provided below in Fig #ad.



**Fig 105: Blood Centrifuge for use in the laboratory module.**

## RR. Crew Sinks

A simple commercial sink was modeled for use in the crew habitats. The material used for the sink is aluminum. Each leg is a 5.08 cm outer diameter with a 0.635 cm thickness and a length of 91.44 cm. There are two configurations available for this sink with different sizes to accommodate different habitat layouts. Figure #af provides two configurations of the habitat sink.



**Fig 106: Two configurations for crew module sinks.**

#### SS. Office Monitors

As seen in other parts of the report, dozens of common objects have been modeled for the habitat modules. The office monitor in Fig. #ag is an example of such common objects. This particular monitor has a 23 inch screen which is sufficient for a wide range of work and research related tasks.



**Fig 107: Habitat desk monitor, an example of common objects modeled by the CAD team for the habitats.**

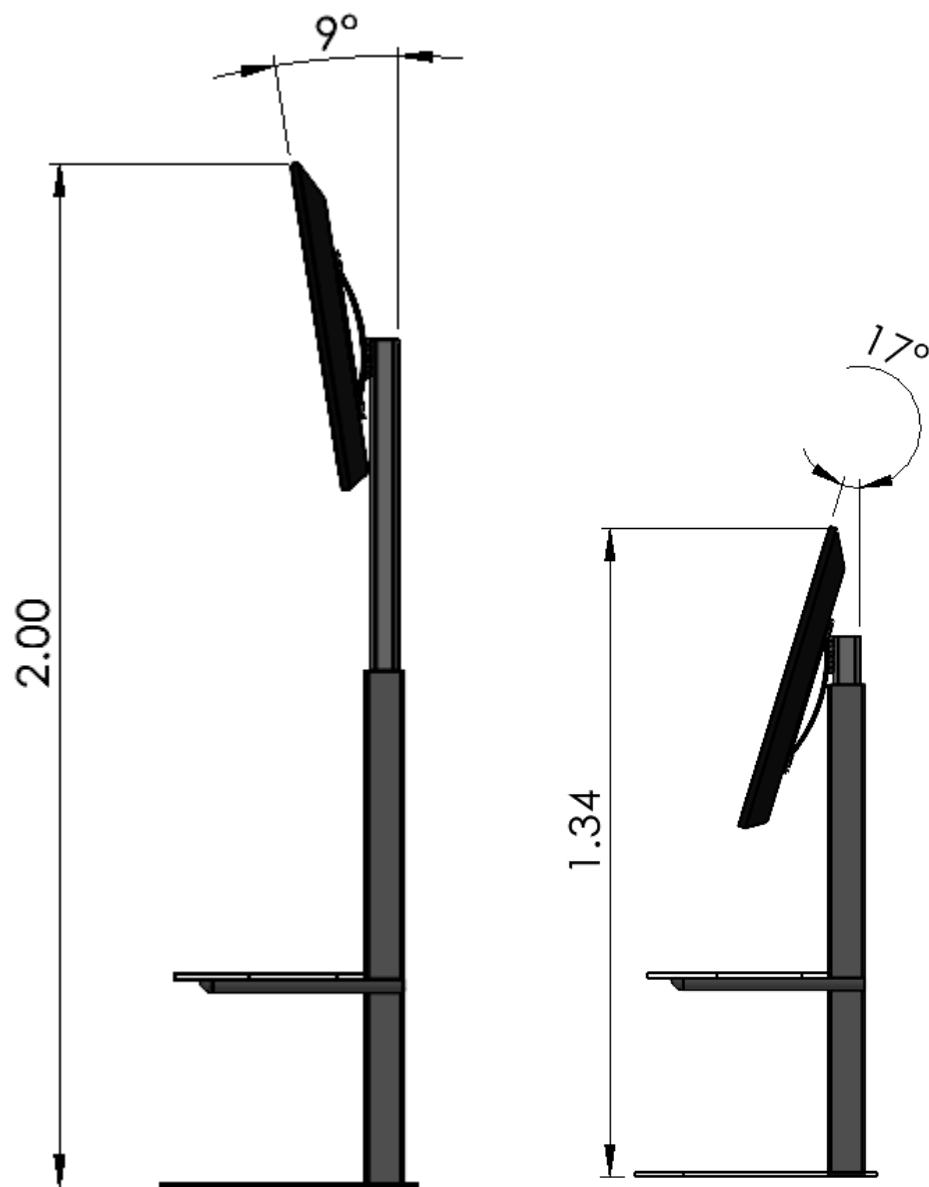
#### TT. Adjustable TV Mount

Another requested item that was considered necessary for the colonists was a TV mount that was compatible with the habitats. Traditional TV mounts are typically fixed to walls, however the radial geometry of the habitat walls make this complicated, so the obvious choice was a design that could stand from the ground. This also gives an opportunity to incorporate a storage/shelving system below the TV to hold additional items. Additionally, the team that requested this model specified that the mount should be adjustable and compatible for a range of

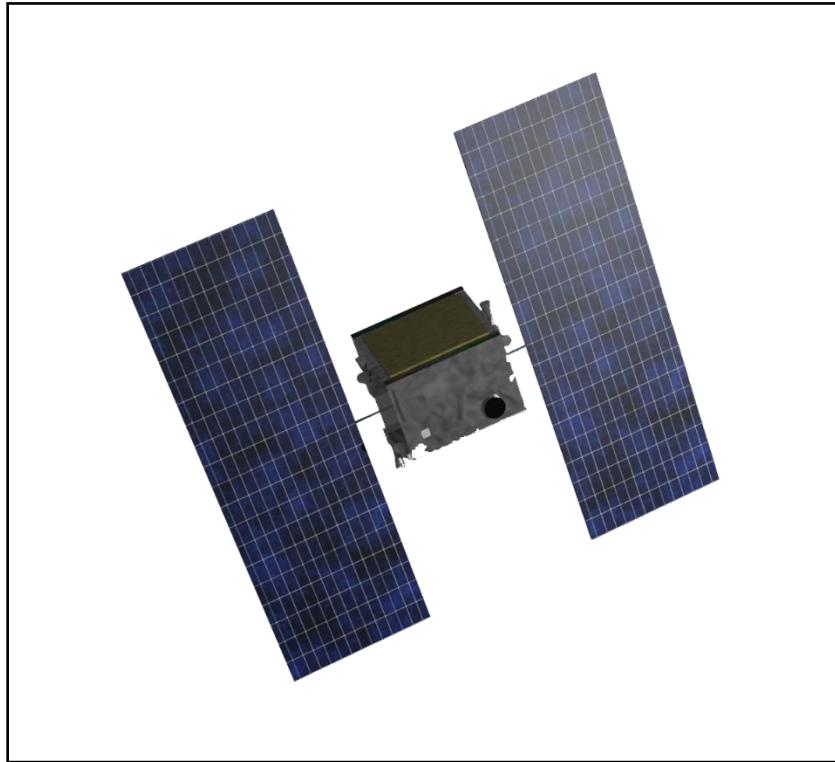
TV screen sizes. Fig. #ah below shows the TV mount with a 48" TV. Additionally, Fig. #ai provides dimensions showing the range of pitch adjustability, and height adjustability. The mount has a vertical range of 66 cm and a pitch range of 27°.



**Fig. 108:** Render of adjustable TV mount.



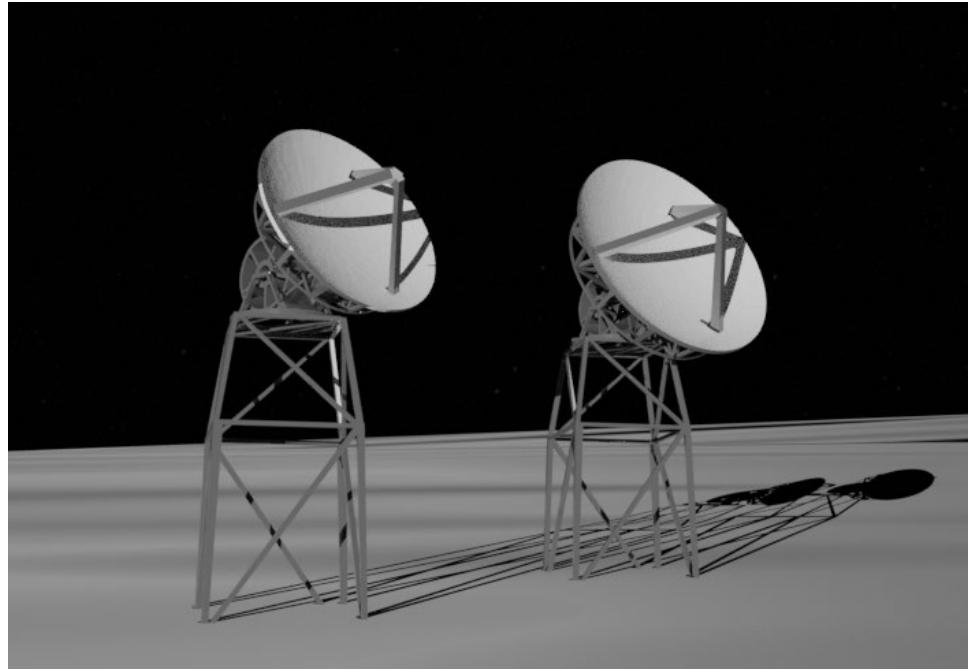
**Fig 109. Habitat TV mount configurable range.**

**UU. Observation Satellite FISTO**

**Fig. 110: Render of FISTO fully deployed.**

This model is the observation satellite FISTO, for the purpose of giving high definition images spectroscopy, and radar analysis near the south pole with a short period between passes. It is based on the Lockheed LM400 series [1], with alternate solar panel and sensors. The enclosed bus is large enough to house the primary science instruments and associated electronics. This satellite is detailed further in the Satellites section.

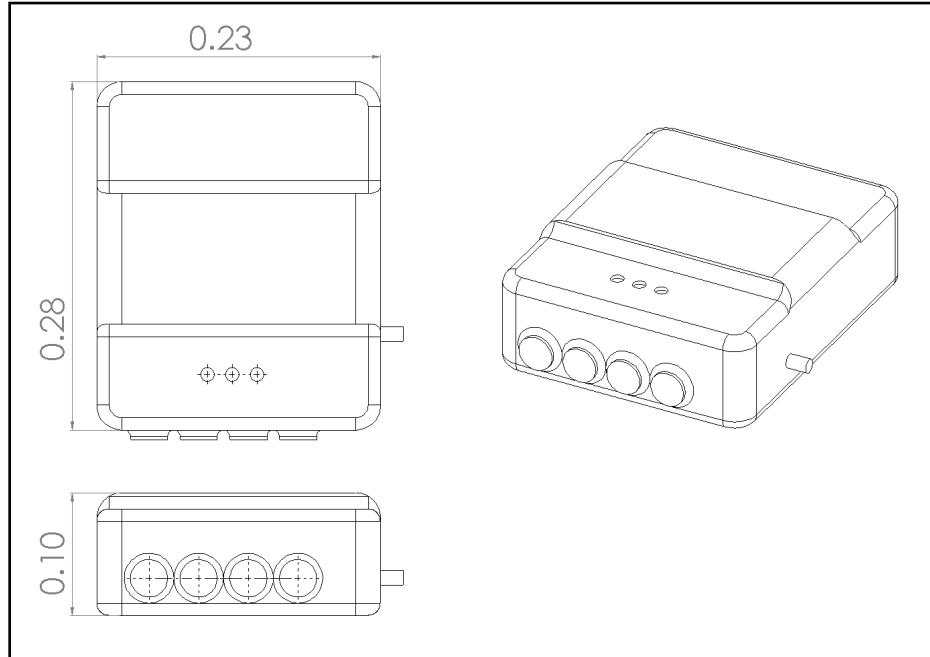
## VV. Wheatley Ground Station



**Fig. 111: Both the 7.6m (left) and 10m (right) dishes sitting on “Mount Malapert”**

The Wheatley Ground station has two dishes for communications with earth. In an effort to save mass, they are constructed out of a tiered scaffold structure to raise them off the surface. They can rotate about two axes, pointing in any direction above the horizon. The design and sizing are detailed in the Communications section.

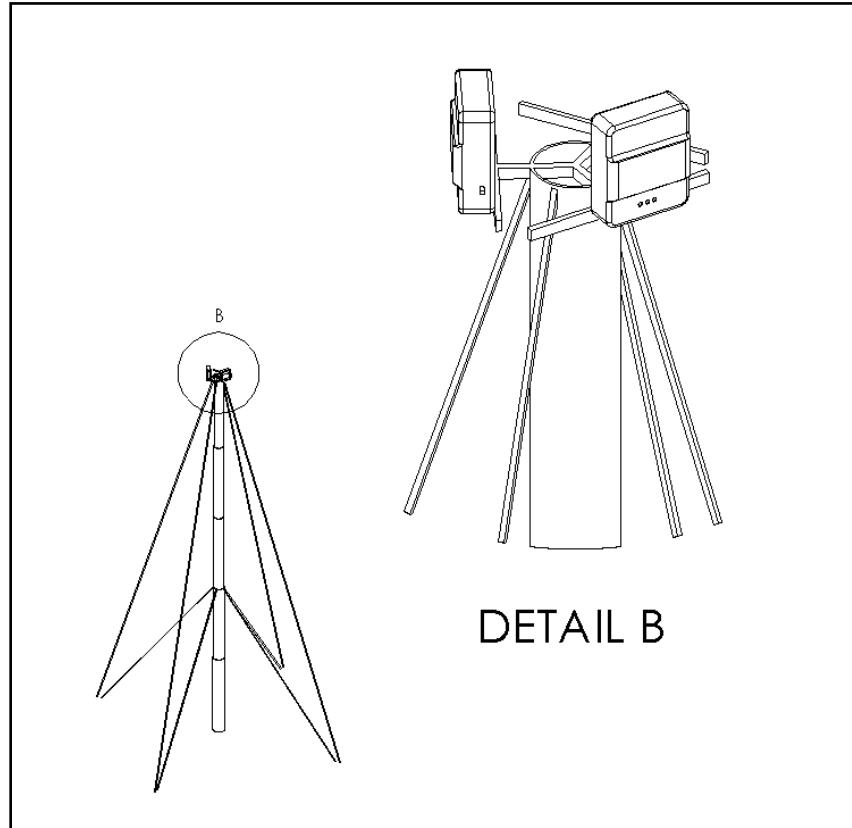
## WW. Repeater Module



**Fig. 112:** The UHF repeater module. Dimensions are in meters.

In addition to the ground station, the Communications team designed a set of relay towers to connect devices over long distances. These act similar to cell towers, and use repeaters similar to those developed for use on Earth.

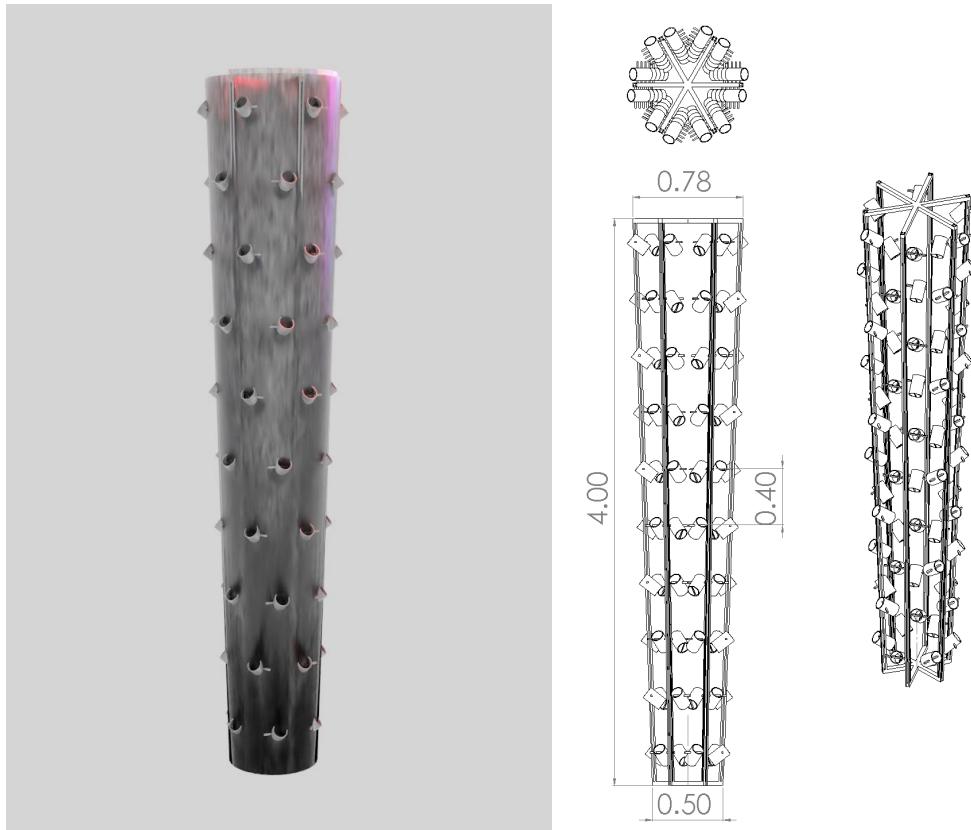
## XX. Communications Relay Tower



**Fig. 113: Full view and closeup of the top of the relay tower. Notice how in Detail B, the UHF Repeater is visible.**

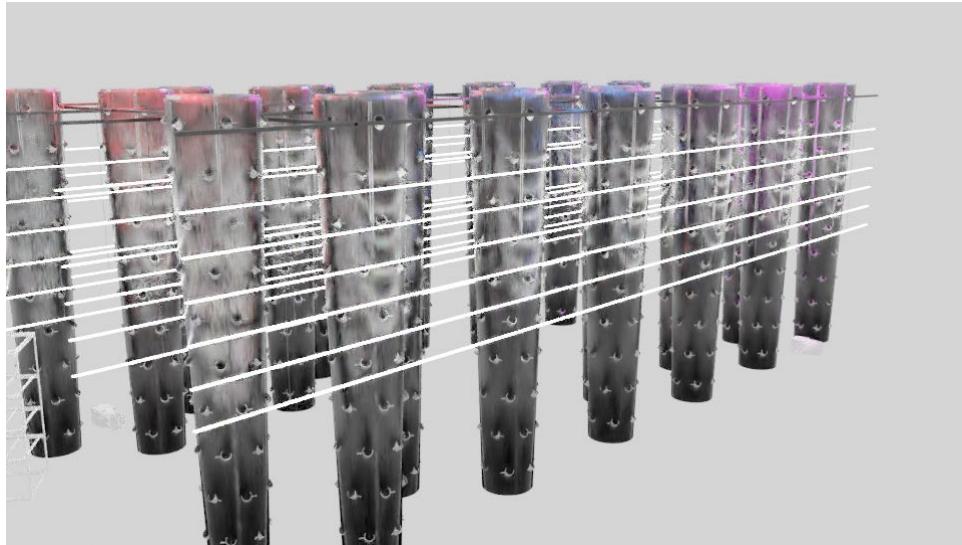
To raise the repeaters above the surface for longer communications, they are mounted on a collapsible, lightweight tower. This tower is ten meters tall, and has mounting points for several pieces of hardware at the top. Further details on its development are listed in the communications section.

## YY. TREES Agriculture Aeroponics Tower



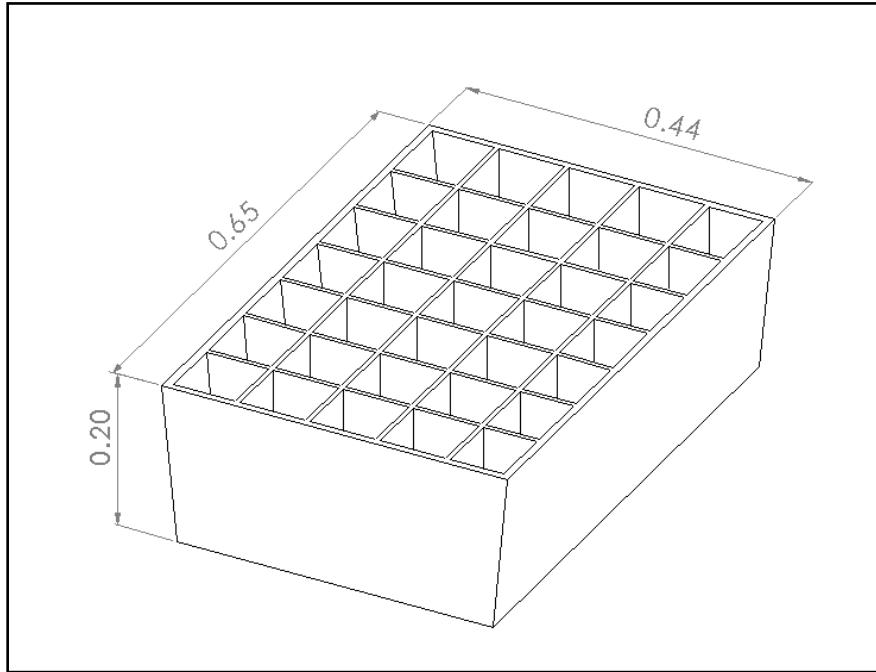
**Fig. 114: A render of the tower covered in its plastic wrap, as well as a dimensioned view of the structural components.**

This is the primary piece of the agriculture system, TREES. It is a lightweight, collapsible structure made of aluminum extrusion that can support a massive number of plants in a small footprint in the inflatable habitats that will be set up on the lunar surface. Without wall or ceiling mounts, it was a difficult challenge to figure out the optimal combination of surface area for plants, and good root spacing, and low mass. The use of plastic wrap to contain the nutrient spray reduces mass significantly compared to other plans established on Earth. The development is detailed in the Agriculture section.

**ZZ.TREES Agriculture Lighting system**

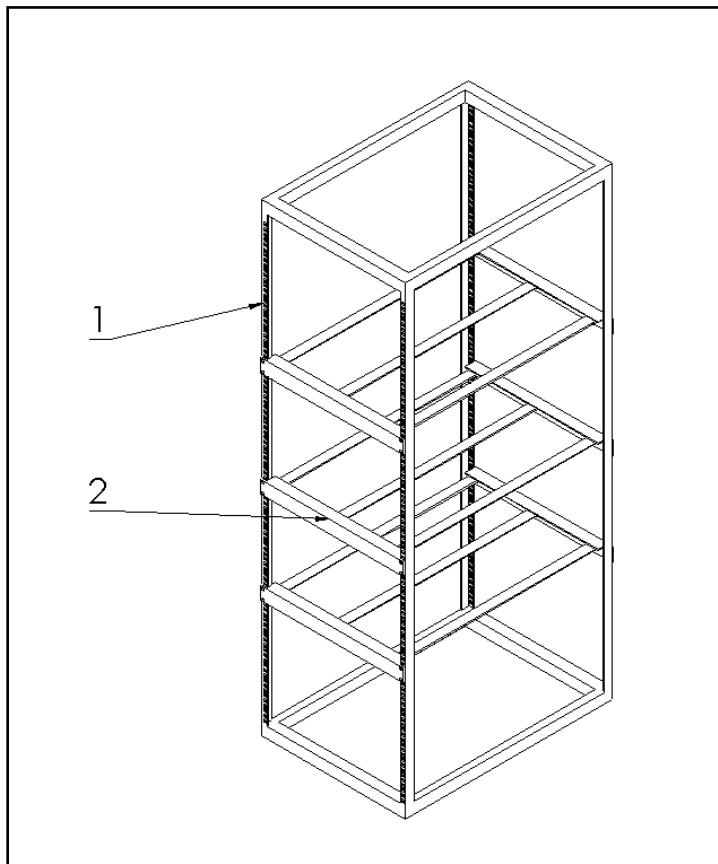
**Fig 115:** A render of several towers, and their accompanying lighting strips.

As the habitats are completely opaque to protect the inhabitants from dangerous radiation, the risks associated with a translucent segment are too large to consider. This necessitates bringing light with sufficient intensity to grow plants within the confines of the hab. These low-mass hanging LED lights mount on the tops of the grow towers, and emit specific wavelengths of light for growing plants without excess energy.

**AAA. TREES Seedling Tray**

**Fig. 116: Simple dimensioned view of the seedling tray.**

A simple partitioned tray for starting plants before transferring them to the aeroponic setup. Plastic and collapsible, they are a minor part of the design. They are designed to fit on the 19-inch standard NASA rack shelves.

**BBB. Standard Modular Rack**

**Fig. 117: Isometric view of the standard rack. Notice (1) the standard mounting holes, and (2) the moveable shelf units.**

In addition to the wall-integrated shelving that John Matuszewski has developed, there was also a need for portable and modular shelving for inside and outside the habitat. These were built to the NASA 19-inch rack standard, which is also used for Earth-bound server racks. Front and back mounting rails ensure strength for heavy components without adding mass to the base of the structure, or requiring mounting points to walls or ceilings. The development of this rack is detailed in the Food Storage section.

### CCC. The Lunar Surface



**Fig. 118t: Earthrise over Shackleton Crater.**

A critical part of modeling activities in the NextStep mission is an accurate model of the surface itself. The most accurate data for terrain height on the Moon has been collected by the Lunar Reconnaissance Orbiter (LRO)'s laser altimeter [2]. Accurate to half a meter of elevation and with up to a data point every 5 meters, this is more than enough to give a surface that reflects the general elevation. The roughness of the surface at a smaller scale for the rocks and boulders

### DDD. The Earth-Moon System



**Fig. 119: An image of the moon from several hundred kilometers of altitude. Notice the overall surface albedo, and the detailed night/day shadow.**

Also, for posing various spacecraft in an accurate Earth-Moon system, models have been generated at true scale in Blender and textured with spherical projections from NASA's global surveying initiatives [3]. Distances and radii were gathered from NASA's planetary factsheets [4]. The Earth is a smooth sphere, but the Moon additionally has a low resolution bump-map to provide accurate shadows for large geography on the surface while it revolves around the Earth also given by NASA [3]. These pieces together will make more appearances in the video, where spacecraft modeled will be in true-scale orbit around these bodies.

## References

[1] Lockheed Martin Corporation, "LM400 Product Card Fact Sheet", 2020.

[https://www.lockheedmartin.com/content/dam/lockheed-martin/space/documents/satellite/LM400\\_Product\\_Card\\_Fact\\_Sheet.pdf](https://www.lockheedmartin.com/content/dam/lockheed-martin/space/documents/satellite/LM400_Product_Card_Fact_Sheet.pdf)

[2] Smith, D., "Lunar Reconnaissance Orbiter Lunar Orbiter Laser Altimeter, data set LRO-LOLA-4-GDR-V1.0", NASA Planetary Data System, 2017. [https://pds-geosciences.wustl.edu/lro/lro-l-lola-3-rdr-v1/lrolol\\_1xxx/browse/LOLA\\_GDR/South\\_pole.html](https://pds-geosciences.wustl.edu/lro/lro-l-lola-3-rdr-v1/lrolol_1xxx/browse/LOLA_GDR/South_pole.html)

[3] Wright, E., Petro, N., "CGI Moon Kit", NASA Scientific Visualization Studio, 2019.

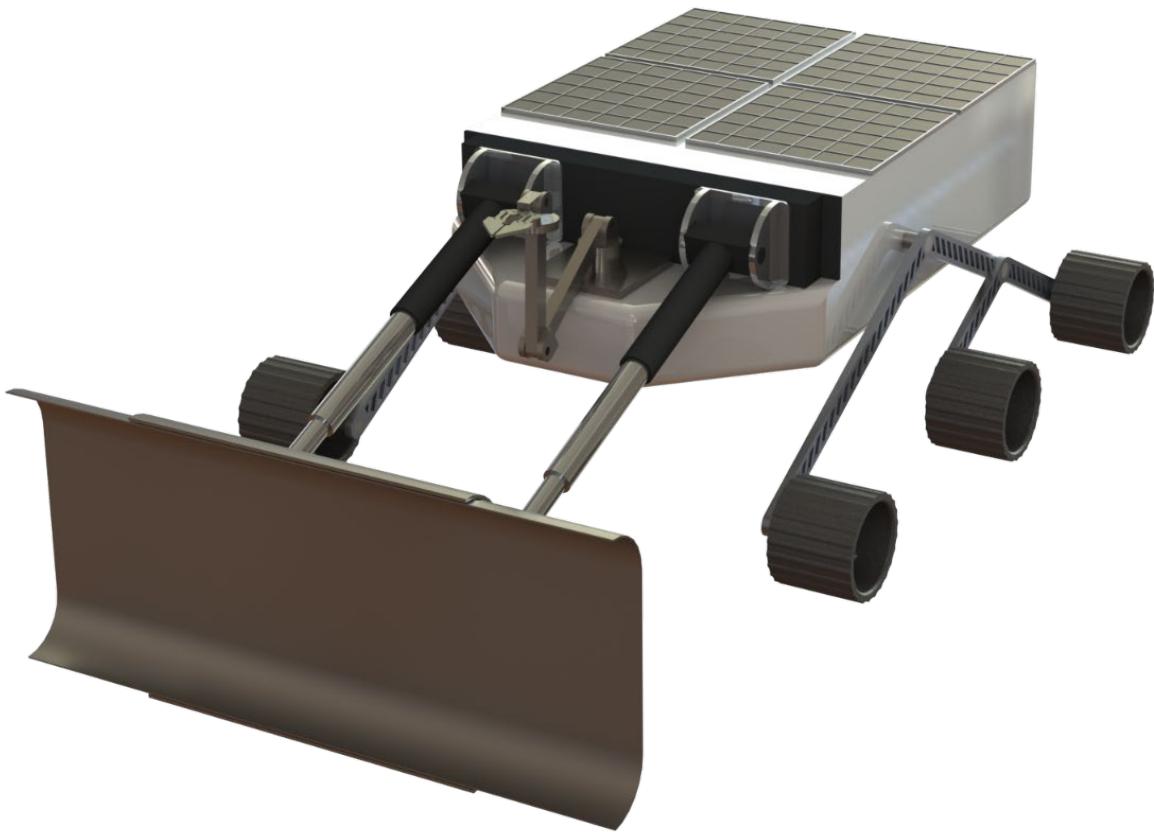
<https://svs.gsfc.nasa.gov/4720>

[4] Williams, D., "Planetary Fact Sheet - Metric", 21 October, 2019.

<https://nssdc.gsfc.nasa.gov/planetary/factsheet/index.html>

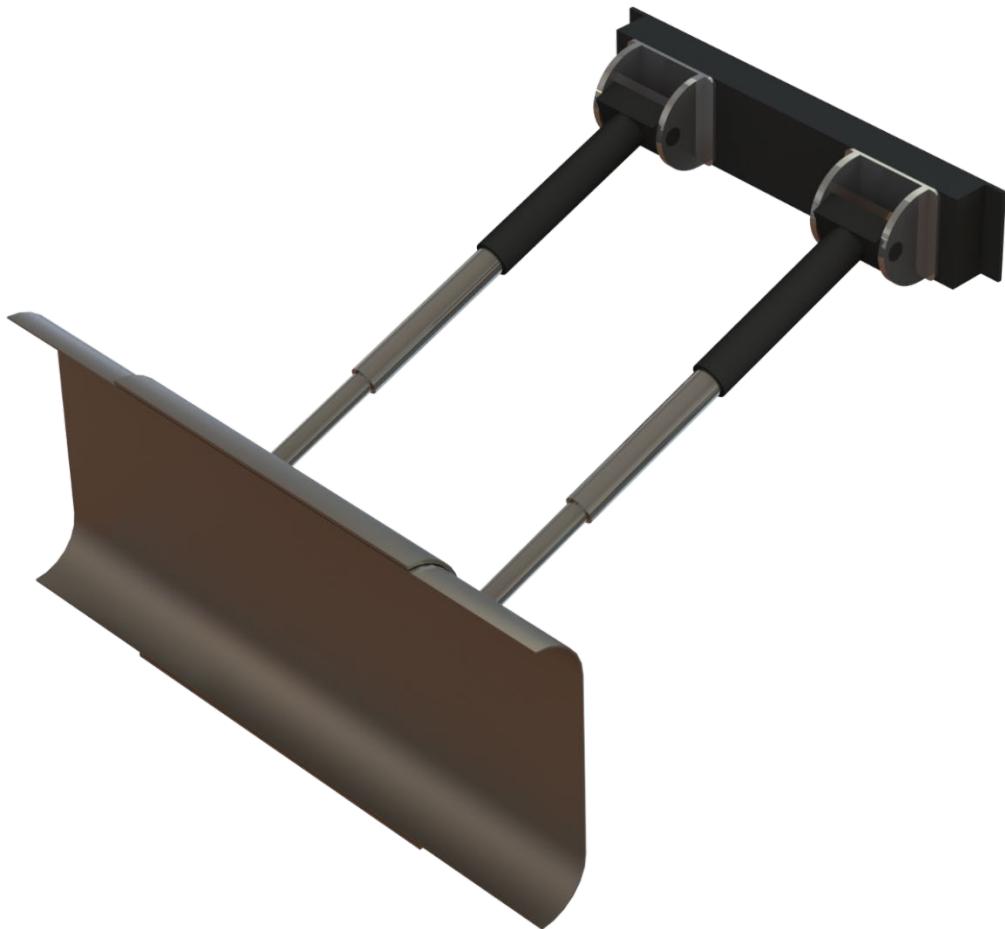
**EEE. Clearing Rover**

**Fig. 120: Render of Clearing Rover in Stowed Configuration**

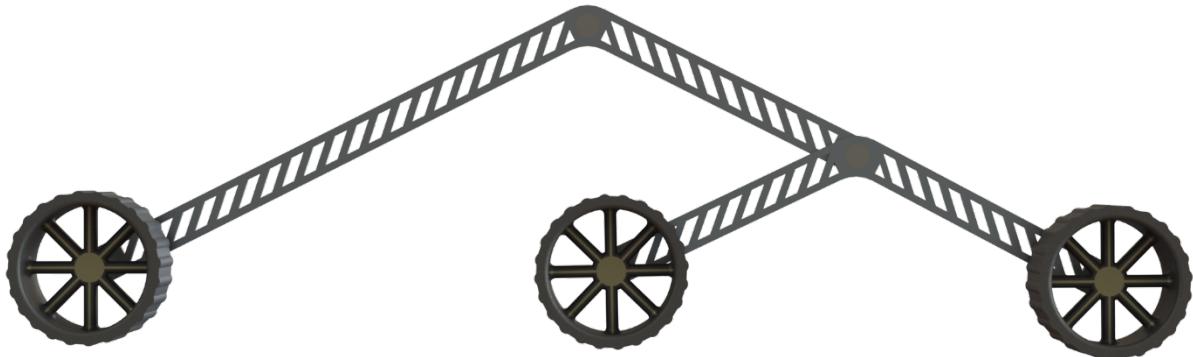


**Fig. 121 Render of Clearing Rover in Operational Configuration**

**FFF. Regolith Plow**



**Fig. 123: Regolith Plow Assembly**

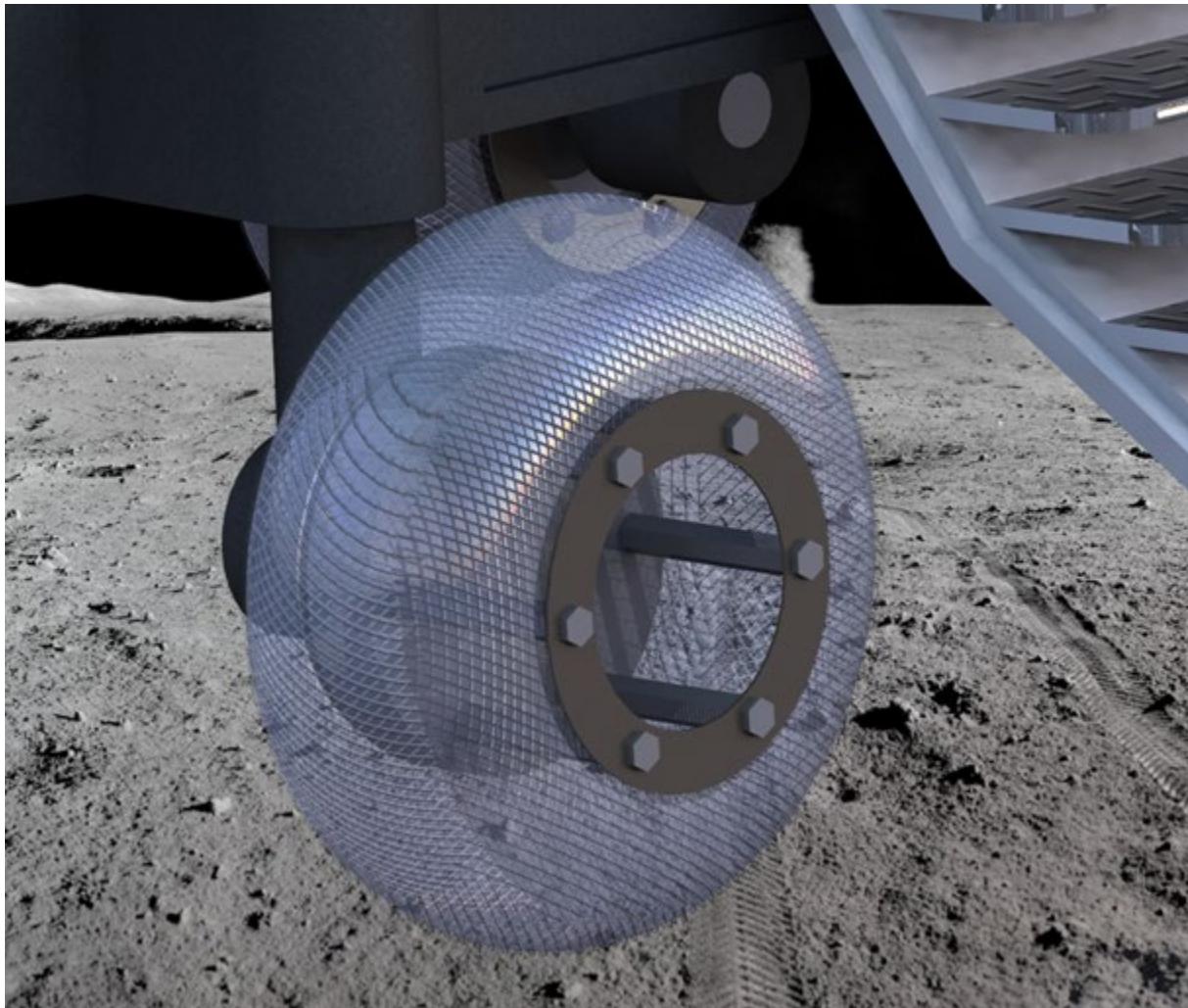
**GGG. Rocker Bogie****Fig. 124 Rocker Bogie Suspension Assembly****Fig. 125 Rocker Bogie Upper/Lower Linkages w/ Wheel Axles**

**HHH. Autonomous Rover Wheel (Solid)**



**Fig. 126 Autonomous Rover Wheel (Solid)**

### III. Large Rover Wheel (Spring Mesh)

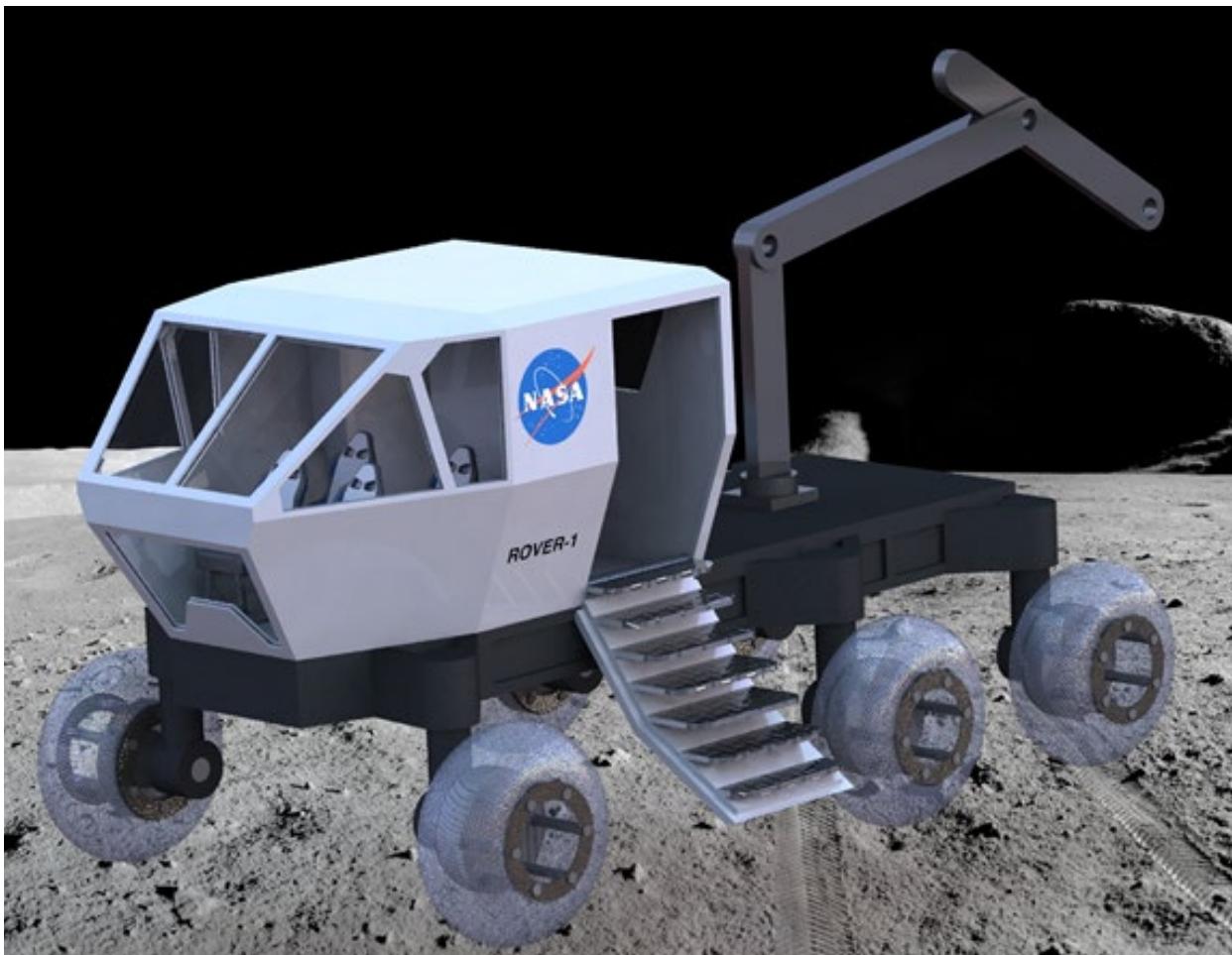


**Fig. 127 Large Rover Wheel (Spring Mesh)**

**JJJ.HDA (Heavy-Duty Arm) Module**

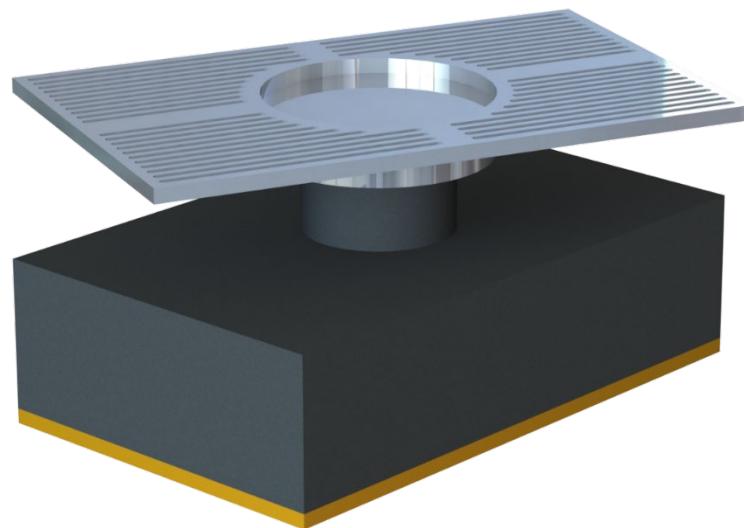


**Fig. 128: HDA (Heavy-Duty Arm) Module for Construction Rover**

**KKK. Construction Rover with HDA (Heavy-Duty Arm) Module**

**Fig. 129: Render of Construction Rover with HDA Module**

**LLL. Crane Module Swivel Base and Modular Flatbed Mount**



**Fig. 130: Crane Module Swivel Base and Modular Flatbed Mount**

### MMM. Scouting Rover (STAR)

The Rovers Team required a scouting rover with the ability to:

1. retrieve rock samples from the ground
2. deposit the samples inside the rover
3. undergo mass spectrometry
4. traverse rough terrain
5. fit within 1U
6. have ground penetrating radar
7. prevent dust from entering the storage bay
8. Harvest energy

Given these requirements, I created a model of STAR shown below. The first of the four images shows an isometric view of the scouting rover with claw attachments at the end of the robotic arm.

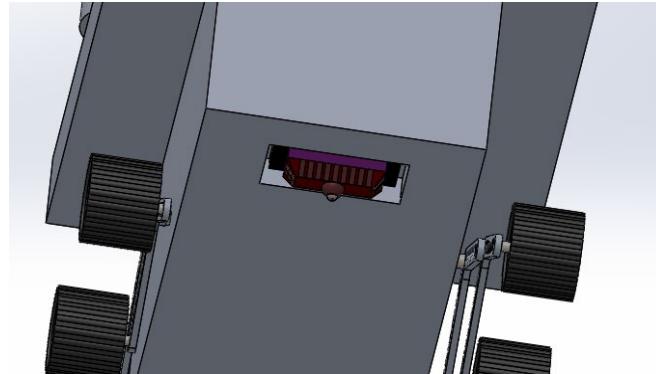


**Fig. 131: Isometric View of STAR (Created by Vishank Battar)**



**Fig. 132 Cross-Sectional View of STAR (Created by Vishank Battar)**

STAR has a conveyor belt that is able to maneuver canisters full of sample from the entry past the mass spectrometer



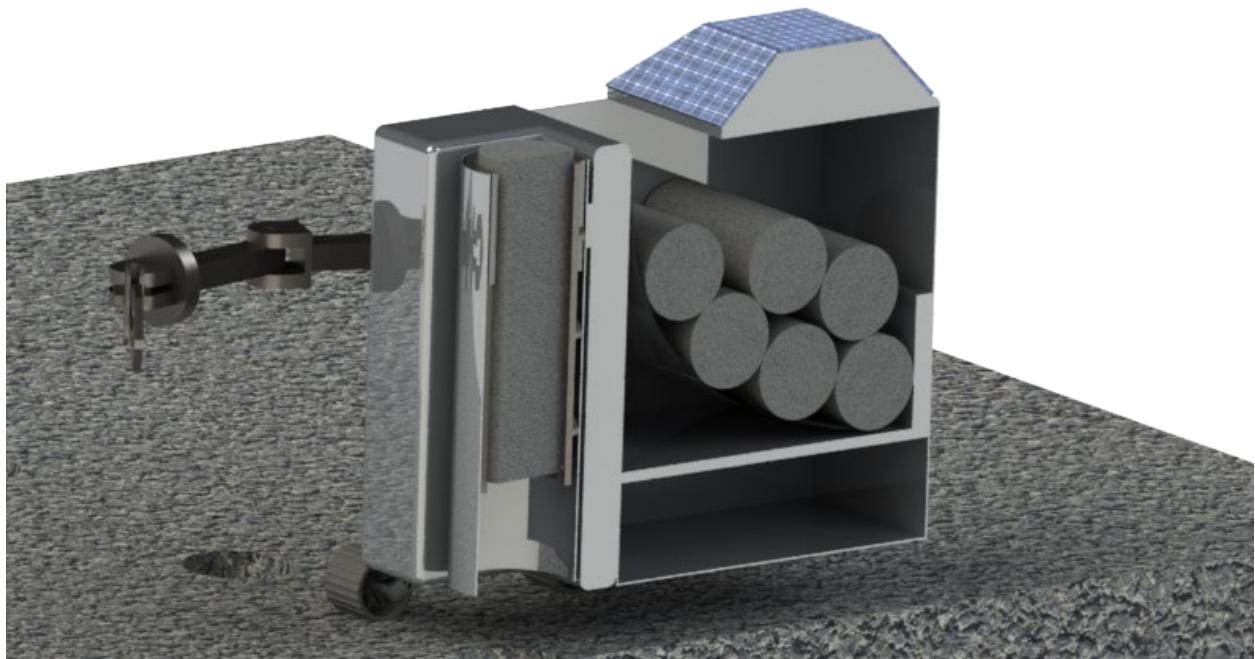
**Fig. 133 Ground Penetratign Radar (Created by Vishank Battar)**

#### NNN. Mining Rover (MARCY)

The rovers subteam requests an autonomous mining rover system with a core drill, a storage bay for basalt cores, a rocker-bogie system, solar panels for energy creation, and a hinged door in the back of the rover that allows the storage bay to be emptied. Given these requirements and the fact that the rover must fit within 1U (2x2.5m), I create the following design for the mining rover.



**Fig. 134 Isometric View of MARCY (Created by Vishank Battar)**



**Fig. 135 Cross Sectional View of MARCY (Created by Vishank Battar)**



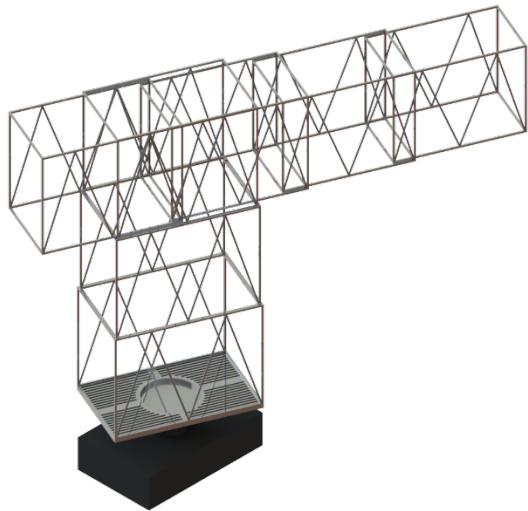
**Fig. 136 Isometric View of Core Drill (Model by Vishank Battar)**



**Fig. 137 Close-up View of Core Drill (Model by Vishank Battar)**

**OOO. Construction Rover with Crane Module**

**Fig. 138 Render of BTB with Crane Module (Model by Vishank Battar)**



**Fig. 139 Crane Module Swivel (Model by Vishank Battar)**