University of Alabama in Huntsville

LOUIS

Honors Capstone Projects and Theses | Honors College

2-23-2019

Design and Optimization of Direct Route to Enceladus

Takuto Iriyama

Tyler J. Sholes

Akifumi Takeyama

Follow this and additional works at: https://louis.uah.edu/honors-capstones

Recommended Citation

Iriyama, Takuto; Sholes, Tyler J.; and Takeyama, Akifumi, "Design and Optimization of Direct Route to Enceladus" (2019). *Honors Capstone Projects and Theses*. 413.
https://louis.uah.edu/honors-capstones/413

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.

Design and optimization of direct route to Enceladus

by

Tyler J Sholes
Akifumi Takeyama
Takuto Iriyama

An Honors Capstone

submitted in partial fulfillment of the requirements
for the Honors Diploma or Certificate

to
The Honors College
of

The University of Alabama in Huntsville

2/23/2019

Honors Capstone Director: Dr. Matt Turner 4/23/2019

Student (signature) Date

Digitally signed by Matthew Turner
DN: cn=Matthew Turner, o=The University of Alabama in Huntsville, ou=UAH, email=matt.turner@uah.edu, c=US
Date: 2019.04.23 19:09:36 -05'00'

Director (signature) Date

Digitally signed by D. Keith Hollingsworth
DN: cn=D. Keith Hollingsworth, o=University of Alabama in Huntsville, ou=Mechanical and Aerospace Engineering, email=keith.hollingsworth@uah.edu, c=US
Date: 2019.04.24 11:45:16 -05'00'

Department Chair (signature) Date

5/1/19

Honors College Dean (signature) Date

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis. In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Tyler Sholes
Student Name (printed)

Student Signature

4/23
Date

Akifumi Takeyama
Student Name (printed)

Student Signature

4/23/2019
Date

Takuto Iriyama
Student Name (printed)

Student Signature

4/23/2019
Date

Design and optimization of direct route to Enceladus

by

Tyler J Sholes
Akifumi Takeyama
Takuto Iriyama

An Honors Capstone

submitted in partial fulfillment of the requirements
for the Honors Diploma or Certificate

to
The Honors College
of

The University of Alabama in Huntsville

2/23/2019

Honors Capstone Director: Dr. Matt Turner

4/23/2019

_____

Student (signature) Date

_____

Director (signature) Date

_____

Department Chair (signature) Date

_____

Honors College Dean (signature) Date

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis. In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Tyler Sholes
Student Name (printed)

Student Signature

4/23
Date

Akifumi Takeyama
Student Name (printed)

Student Signature

4/23/2019
Date

Takuto Iriyama
Student Name (printed)

Student Signature

4/23/2019
Date

Abstract

The class mission for witch the honors capstone was built off of was the Interior and Composition for Enceladus Exploration (ICEE). This mission focused on sending an orbiter, lander and rover to Enceladus, a moon of Saturn. For our Honors thesis we have been analyzing and optimizing the direct route from Earth to Saturn's moon Enceladus. Enceladus is an interesting moon of Saturn from a scientific point of view because it constantly ejects particulates from its inside into Saturn's orbit, partially creating the ring around Saturn. There are several scientific articles written about indirect ways via flybys and other assistance methods to get to Saturn. There are very few written about using SLS and an additional kick stage to reach Saturn. For our honors thesis we have done A) analyze the trajectory from Earth to Saturn B) Design two solid rocket kick stages to both to accelerate our spacecraft out of Earth's orbit and to slow us down once we reach Saturn so that we can stay in Saturn's orbit and C) Design a RCS to control the spacecraft during the run between Earth and Saturn. Together we will be optimizing this route to allow us to carry the most amount of mass we can from Earth to Saturn. To do the optimization we will be using information provided by our instructor and from MAE 440 to design a rocket nozzle that maximizes thrust without running into to many issues of flow separation, Plotting different mass to Enceladus vs initial C3 to fit a curve, use MATLAB to find the maximum of that function, plot that point, and then repeat until the points become close enough that they are within the margins of error of the process.

Introduction

The purpose of this research project is to design and optimized the trajectory and motor for the ICEE. The project that was undertaken was analyzing the trajectory from Earth to Saturn using custom made solid rocket motors. In the Chapter 1 and 2, it is about the design and optimization of the Solid Rocket motor. The method of the optimization of the motor is to look for the maximum point of the graph between the C3 and ballast masses. The result of the mass of propellant for kick stage was 8760 kg and for braking stage was 5680 kg. Also, the final design of the motor is shown in the Table 2. Chapter 3 and 4 are about the design and trade study of the trajectory. Matlab Simulink Simscape Multibody was chosen for the simulation program and compared with Hohmann Transfer Trajectory. Table 3 shows the comparison of the result of simulation and Hohmann Transfer Trajectory. Chapter 5 is about the Reaction Control system for the attitude control. The end product that has been arrived is much more than we were expecting, not only was designing custom built solid rocket motor that are sized for SLS done, additional work was also done towards the project. The additional work included, making a video simulation of the trajectory to Saturn and one for the class project in addition spacecraft on the way to Saturn and once it arrives at Enceladus.

Chapter 1: Design of Solid Rocket motor

The design of the solid rocket motor began with heavy modifications to the MAE 440 spreadsheet so that it could both take in the inputs, but also link the pages together the way it was needed. The motor is consisted by kick and braking stage. The delta-V required for both stages are calculated based on the trajectory analysis, then the grain was designed to perform that

delta-V. The delta-V required for kick stage is 13539.9 m/s and for braking stage is 11483 m/s. These numbers came from the fact that a C3 of 105 was needed and the incoming C3 at saturn was a C3 of 3.5. The grain was assumed as a simple cylinder, and both ends are inhibited and the nozzle can be deformed by the high temperature. All payload mass and inert mass of spacecraft was summed up to total mass. The geometry of Hydroxyl-Terminated PolyEther was used to design grain. Then, for each burning time to burn 0.1 inches (web distance), the thrust, mass, velocity, acceleration and nozzle radius were calculated sequencely. The optimization was done by changing the number, radius and length of grains. Case mass was based on this grain size, and added to the total mass. The modified 440 spreadsheet analyzed the bern, layer by layer, of the solid rocket fuel, analysed the thrust, and calculated a delta V. The inputs of the modified spreadsheet were C3 and the figures shown in Table 2.

Chapter 2: Optimization of the motor

Following the modifications to the spreadsheet, a table was created of the inputs of C3 values and outputs of ballast masses. An arbitrary 3 points were taken, within a range we assumed the maximum did not occur. With those three points a trendline was created, and then matlab was used to find the maximum of that trendline. This system was repeated until it was shown to have less than a 0.5% increase from one point to the next. This process can be seen in figures 1 through 4. As a result, the mass of propellant for kick stage was 8760 kg and for braking stage was 5680 kg.

Table 1: C3 optimization chart

C3 (km^2/s^2)	Ballast (Kg)
0	377.413
10	1208.36
20	1702.66
29.68	2078.235
40.02	2321.79
45.64	2378.19
46.97	2413.4

Table 2: Rocket final design

Radius, outter	331.2	180
Radius, inner	250	160
Length	2	4.5
# Grains	1	2
Temperature at burn	0	0

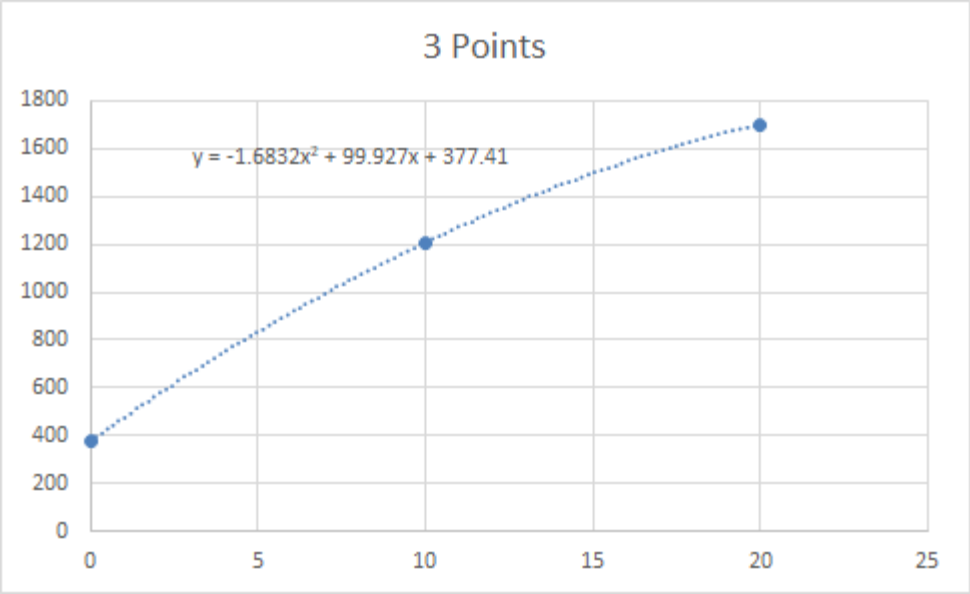


Figure 1: 3 point system

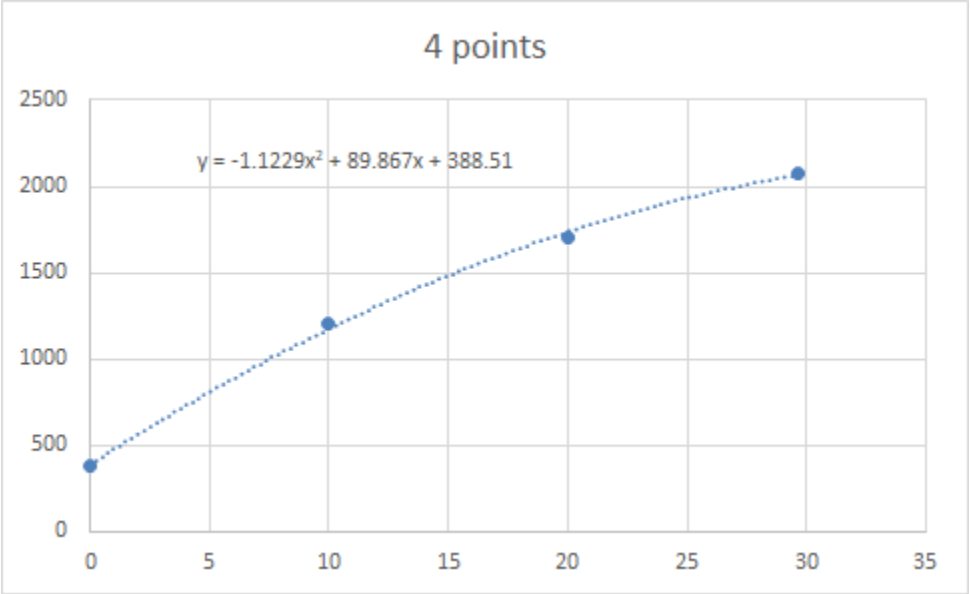


Figure 2: 4 point system

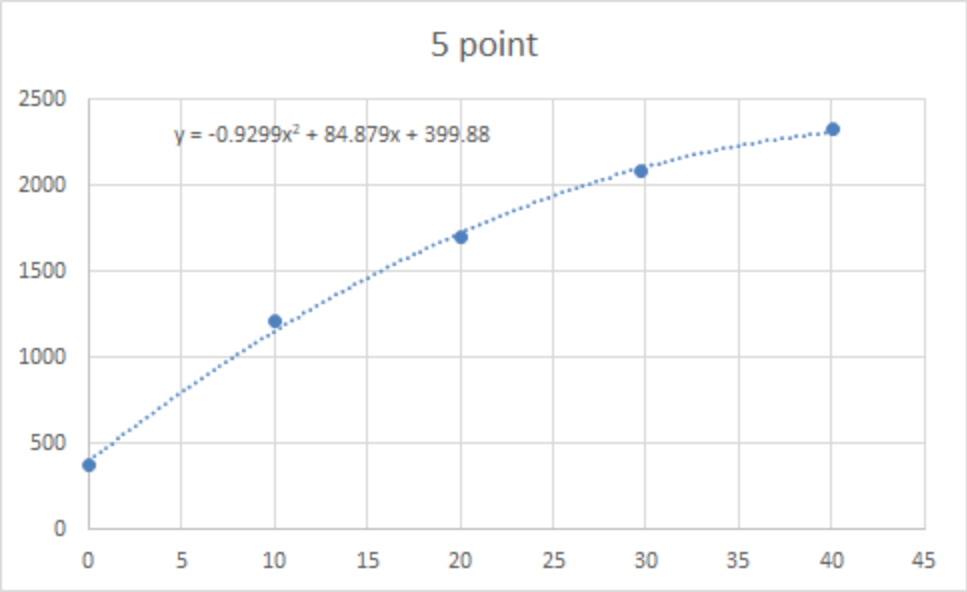


Figure 3: 5 point system

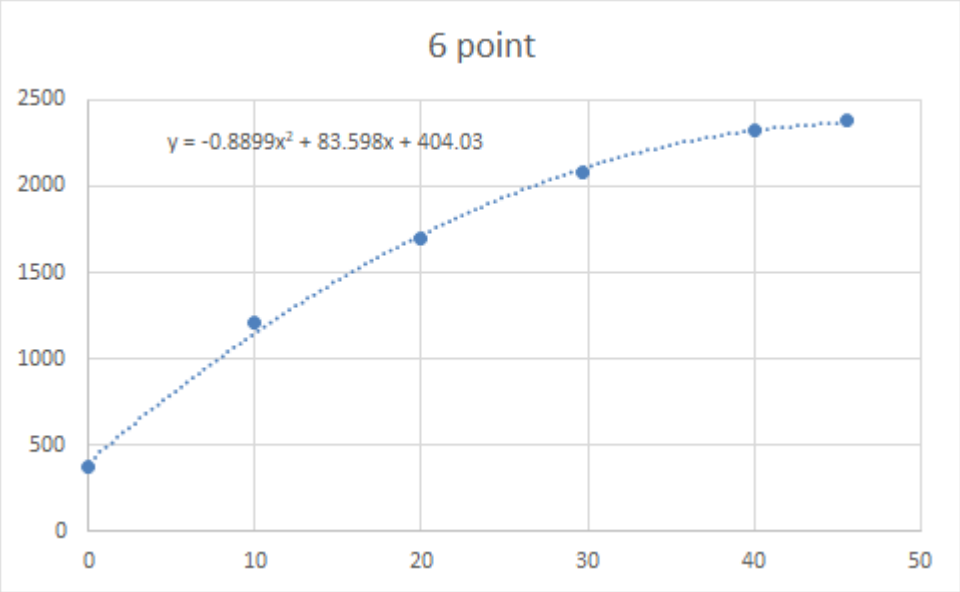


Figure 4: 6 point system

Chapter 3: Design of Trajectory

After optimizing the rocket motor, it was decided that additional work was to be done for the honors capstone. This extra

work was first designing a simulation of the route for the honors spacecraft. After much research Matlab Simulink Simscape Multibody was chosen for the simulation program. Figure 5 shows the starting positions of all of the planets and spacecraft for the Saturn route. The ground rules for this model are that all of the planets have their own gravity that follows an inverse square pattern, all planets had to be scaled up to be visible from the default isometric view without changing their gravitational effects, the planets were free to move on their own in a full 6 degree of freedom (doff) setup.

Figure 5: 6 doff solar system model

The other option of the trajectory is the Hohmann transfer trajectory. It is the most efficient trajectory for direct route to the destination if the semi-major axis ratio of the final to initial orbit is less than 11.94. For the ratio of the semi-major axis of Saturn to

Earth is 9.6, so Hohmann transfer orbit is the most efficient for direct trajectory.

Chapter 4: Result and Discussion of Trajectory Simulation

The result of simulation for the total delta V between Earth to Enceladus is 10.18 km/s and total trip time is 3.5 years. The launch date window is June 2-8th 2038. The spacecraft flybys Jupiter for the gravity assist and accelerates its velocity there, then it is expected around beginning of 2042 at Saturn. This result of the simulation is under the assumption that engine has unlimited thrust.

The total delta V for Hohmann transfer trajectory from low Earth orbit to Enceladus orbit is 16.19 km/s. This value is calculated under the some of the assumptions that are following.

- Earth, Saturn, and Enceladus are circular orbits
- Initial position is the 400 km altitude of the low Earth orbit
- Arrival position is 84 km altitude circular orbit of Enceladus
- Eccentricity is 0.9 for the Saturn orbit insertion
- Saturn insertion orbit periapsis is 238037 km from the center of the Saturn which is the semi-major axis of the Enceladus

The velocities of departure at Earth, Saturn orbit insertion, and Enceladus orbit insertion are 10.29 km/s, 1.26 km/s and 4.63 km/s respectively. The characteristic energy called C3 become 105.8 km^2/s^2. The travel time of the Hohmann transfer between Earth to Saturn is 10 years.

Table 1 is the summary of the performance of the trajectory of the both case. From the all of the perspective, the simulation result is better than the result of the Hohmann Transfer Orbit. For

this mission, the trajectory of the result of simulation was selected for the ICEE Mission.

Table 3: Comparison of Result Between Simulation and Hohmann Transfer Trajectory

	Simulation Result	Hohmann Transfer
Total delta V (km/s)	10.18	16.19
C3 (km^2/s^2)	26.7	105.8
Trip time (years)	3.5	10

Chapter 5: Reaction Control System

For the attitude control momentum wheels and thrusters are used as actuators. The momentum wheels are mainly used for the angular momentum control and thrusters are used for the momentum of longitudinal direction. Communication and science purpose, the pointing control accuracy and slew rate have requirements. For satisfy the requirements, momentum wheels are used because it is possible more accurate attitude control than used by thrusters. Thrusters are used for the deployment accommodation, adjustment for the disturbance of navigation, and cancellation of the saturated momentum wheel storage. The spacecraft deploy Lander, Rover and Sondes, so it is require the accommodation for the change of momentum when the deployment occurs. The error of the navigation is expected from the gravity of the small bodies and solar pressure, so thrusters can adjust its position and velocities from the disturbance. The momentum wheel have a limitation of the amount that can store and it is needed for the cancellation of the stored momentum using

the thrusters. This thruster is not working as a main engine and the 1N level of the thrust of the monopropellant liquid thruster is enough for the ICEE spacecraft. This decision is considered from the experiential data of the former spacecraft missions.

Conclusion

Our research calculated analysis about the route from the Earth to Enceladus. The trajectory was analysed using MATLAB and Simulink and the result of the simulation was better than the Hohmann Transfer Trajectory. The comparison of the result of both trajectory is in the Table 3. Also, the motor used in the mission was designed using a method provided in MAE 440 class at UAH in fall 2018. For required thrust and delta-V, both a kickstage and braking stage had to be designed. The motors were designed by hand at first, then it was optimized using arbitrary three points and then maximizing the trend line until it has less than a 0.5 % increase from the last optimization. From the analysis, the mass of propellant for kick stage was 8760 kg and for braking stage was 5680 kg. The final design of the motor is shown is the Table 2. The trajectory using two solid rocket motor was possible, and it performs greatly especially about its volume. The attitude control accuracy using RCS can be improved by combining liquid thrusters and momentum wheels.

Appendices

Appendix 1: Matlab code

```
function [ Madd, MFT1, MLO, eko, ebo, wmbo, idxo, BrakeMotorNameOutput,KickMotorNameOutput, ...
    CS, mlfb] = Senior_Design_Motors( C3in, MD, DVB, DVK, C1, C2, C3, C4, g0 )
 %motors calculates total mass from dry mass, Total impulse of the breaking
```

```
 %motors, total impulse of the kickstage motors, the deltav of the break
 %stage, the delta v of the kick stage, breaking stage motor mass, kick
 %stage motor mass
MLE=200;
n=1; nmax=6;
ISPLB=320;
ISPLK=320;

idxs=zeros(nmax,1);
CheckStore=zeros(nmax,1);
MFS=zeros(nmax,1);
MLS=zeros(nmax,1);
eks=zeros(nmax,1);
ebs=zeros(nmax,1);
wmbs=zeros(nmax,1);
MaxThrustBrakeStore=zeros(nmax,1);
MaxThrustKickStore=zeros(nmax,1);
BrakeMotorNameStore=zeros(nmax,13);
KickMotorNameStore=zeros(nmax,13);
CS=zeros(5,length(C3in));
while 1
   if n==1
   TotalImpulseBrake=2521900*C1; MassBrakingEngine=9494*C2; ...
      PropellantMassBrake=8631*C2;     MaxThrustBrake=0*C1;BrakeMotorName=
'Orion 50XL   ';
    elseif n==2
   TotalImpulseBrake=1949000*C1; MassBrakingEngine=7395*C2; ...
      PropellantMassBrake=6669*C2;     MaxThrustBrake=0*C1;BrakeMotorName=
'Orion 50     ';
    elseif n==3
   TotalImpulseBrake=9472400*C1;MassBrakingEngine=35763*C2; ...
      PropellantMassBrake=33145*C2;
MaxThrustBrake=156823*C1;BrakeMotorName= 'Orion 50S XLT';
    elseif n==4
   TotalImpulseBrake=1186000*C1; MassBrakingEngine=4280*C2;...
      PropellantMassBrake=4280*C2;                    MaxThrustBrake=28800*C1;
BrakeMotorName= 'Orion 32     ';
    elseif n==5
   TotalImpulseBrake=491140; MassBrakingEngine=1924*C2; ...
      PropellantMassBrake=1698;   MaxThrustBrake=8303*C1;   BrakeMotorName=
'Orion 38     ';
   else
```

```
        break
    end

   if n==1
   TotalImpulseKick=2521900*C1; MassKickEngine=9494*C2; ...
       PropellantMassKick=8631*C2; MaxThrustKick=0*C1; KickMotorName= 'Orion
50XL   ';
    elseif n==2
   TotalImpulseKick=1949000*C1; MassKickEngine=7395*C2; ...
       PropellantMassKick=6669*C2; MaxThrustKick=0*C1; KickMotorName= 'Orion
50     ';
    elseif n==3
   TotalImpulseKick=9472400*C1;MassKickEngine=35763*C2; ...
       PropellantMassKick=33145*C2; MaxThrustKick=156823*C1; KickMotorName=
'Orion 50S XLT';
    elseif n==4
   TotalImpulseKick=1186000*C1; MassKickEngine=4280*C2;...
       PropellantMassKick=4280*C2;  MaxThrustKick=28800*C1;  KickMotorName=
'Orion 32     ';
    elseif n==5
   TotalImpulseKick=491140*C1; MassKickEngine=1924*C2; ...
       PropellantMassKick=1698*C2;  MaxThrustKick=8303*C1;  KickMotorName  =
'Orion 38     ';
   end


   MassKickEngine=MassKickEngine-PropellantMassKick;
   MassBrakingEngine=MassBrakingEngine-PropellantMassBrake;

   ispb=TotalImpulseBrake/PropellantMassBrake;
   if ispb>ISPLB
       c1=ispb*g0;
       wmb=1;
       err=1;
       while err>.001
           wmb0=wmb;
           MR1=(MassBrakingEngine+MLE+MD)/(wmb);
           dvb=c1*log(1/MR1);
           eb=floor(DVB./dvb);
           msb=MassBrakingEngine*eb;
           dvsb=eb*dvb;
           dvlb=DVB-dvsb;
```

```
        mlfb=log(dvlb/ISPLB*g0)*MD-
PropellantMassBrake+MassBrakingEngine+MD;
        mleb=mlfb+MLE;
        wmb=mleb+msb+MD;
        err=abs((wmb0-wmb)/wmb);
      end
   else
      mlfb=log(DVB/ISPLB*g0)*MD-MD;
      mleb=mlfb+MLE;
      wmb=mleb+MD;
      eb=0;
      P=1;
   end

   ispk=TotalImpulseKick/PropellantMassKick;
   if ispk>ISPLK
      c2=ispk*g0;
      MR2=(MassKickEngine+wmb)/(PropellantMassKick+MassKickEngine+wmb);
      dvk=c2*log(1/MR2);
      ek=floor(DVK./dvk);
      msk=MassKickEngine*ek;
      dvsk=ek*dvk;
      dvlk=abs(DVK-dvsk);
      for P=1:length(C3in)
        mlfk=log(dvlk(P)/ISPLK*g0)*wmb-
PropellantMassKick+MassKickEngine+wmb;
        mlek(P)=abs(mlfk+MLE);
      end
      MF=mlek+msk+wmb;
      ML=mlek+mleb;
   else
      mlfk=log(DVK/ISPLK*g0)*wmb-wmb;
      mlek=mlfk+MLE;
      ek=zeros(1,length(C3in));
      MF=mlek+wmb;
      ML=mlfk+mlfb;
      P2=2;

   end


   MT=0.0022*C3in.^2-0.5923*C3in+41.168;
```

```
MFT1=(MF)*C4;
Check=MT./MFT1;

U=1;
len=length(C3in);
cin=length(C3in);
while U<cin
   if ML(U)<mleb
      ML(U)=[];
      Check(U)=[];
      MF(U)=[];
      ek(U)=[];
      cin=cin-1;
      len=len-1;
      U=U-1;
   end
   U=U+1;
end
U=1;
while U<len
   if Check(U)>1.1
      Check(U)=[];
      ML(U)=[];
      MF(U)=[];
      ek(U)=[];
      len=len-1;
      U=U-1;
   end
   U=U+1;
end
U=1;
while U<len
   if MF(U)<0
      Check(U)=[];
      ML(U)=[];
      MF(U)=[];
      ek(U)=[];
      len=len-1;
      U=U-1;
   end
   U=U+1;
end
```

```
U=1;
while U<len
   if ML(U)<0
      Check(U)=[];
      ML(U)=[];
      MF(U)=[];
      ek(U)=[];
      len=len-1;
      U=U-1;
   end
   U=U+1;
end

[CM,idx]=max(Check);
   if(CM>1)
      idxs(n)=idx;
      CS(n,1:len)=Check;
      CheckStore(n)=CM;
      MFS(n)=MF(idx);
      MLS(n)=ML(idx);
      eks(n)=ek(idx);
      ebs(n)=eb;
      wmbs(n)=wmb;
      MaxThrustBrakeStore(n)=MaxThrustBrake;
      MaxThrustKickStore(n)=MaxThrustKick;
      BrakeMotorNameStore(n,:)=uint16(BrakeMotorName);
      KickMotorNameStore(n,:)=uint16(KickMotorName);
   else
      BrakeMotorNameStore(n,:)=uint16('Does Not Work');
      KickMotorNameStore(n,:)=uint16('Does Not Work');
      CS(n,1:len)=Check;
      CheckStore(n)=CM;
      idxs(n)=idx;
      MFS(n)=MF(idx);
      MLS(n)=ML(idx);
      eks(n)=ek(idx);
      ebs(n)=eb;
      wmbs(n)=wmb;
      MaxThrustBrakeStore(n)=MaxThrustBrake;
      MaxThrustKickStore(n)=MaxThrustKick;
   end
   if P==1 && P2==2
```

```
            break
        end
        n=n+1;
end
idxo=1;
Madd(idxo)=-10;
while idxo<len
    [~, nout]=max(CheckStore);
    %nout=5;
%     idxo=30377;
    MFO=MF(nout);
    MFT1=(MFO)*C4;
    MT=0.0022*C3in.^2-0.5923*C3in+41.168;
    Madd=MT-MFT1;
    if Madd<1
    Check(nout)=[];
    ML(nout)=[];
    MF(nout)=[];
    Madd(nout)=[];
    ek(nout)=[];
    len=len-1;
    idxo=idxo-1;
    end
    idxo=idxo+1;
end


    [CM,idx]=max(Check);
        if(CM>1)
            idxs(n)=idx;
            CS(n,1:len)=Check;
            CheckStore(n)=CM;
            MFS(n)=MF(idx);
            MLS(n)=ML(idx);
            eks(n)=ek(idx);
            ebs(n)=eb(idx);
            wmbs(n)=wmb;
            MaxThrustBrakeStore(n)=MaxThrustBrake;
            MaxThrustKickStore(n)=MaxThrustKick;
            BrakeMotorNameStore(n,:)=uint16(BrakeMotorName);
            KickMotorNameStore(n,:)=uint16(KickMotorName);
        else
```

```
            BrakeMotorNameStore(n,:)=uint16('Does Not Work');
            KickMotorNameStore(n,:)=uint16('Does Not Work');
            CS(n,1:len)=Check;
            CheckStore(n)=CM;
            idxs(n)=idx;
            MFS(n)=MF(idx);
            MLS(n)=ML(idx);
            eks(n)=ek(idx);
            ebs(n)=eb;
            wmbs(n)=wmb;
            MaxThrustBrakeStore(n)=MaxThrustBrake;
            MaxThrustKickStore(n)=MaxThrustKick;
        end
%     if Madd(idxo)>0
%         break
%     end
[~,idxo]= max(CheckStore);
% idxo=1754;
MLO=ML(idxo)*C4;
MFO=MF(idxo)*C4;
eko=eks(idxo);
ebo=ebs;
wmbo=wmb;
Madd=Madd(idxo)
        if P==1 && P2==2
            BrakeMotorNameOutput='No Solid';
            KickMotorNameOutput='No Solid';
        end
end

%clear,clc
warning off;



f=30.038E9; %Frequancy in Hz
Pt=33.7; %power to transmit in watts
DE=1.2E12; %distance to transmition source
L0=306+21; %free space path loss
Lp=20; %internal losses
k=1.38E-23; %boltsmans constant
D=4; %diameter of transmitting antena
```

```
Dr=70; %diameter of recieving antenna
B=30*10^6; %ka-band bandwidth
Ts=306+21; %noise temperature
nu=.50; %transmission error rate
Gt=nu*((pi*D*f*10^6)/3E8)^2;
Gr=nu*((pi*Dr*f*10^6)/3E8)^2;
Pdb=10*log10(Pt/1);
R=3*10^6; %rate at witch you send data (needs to be 1-9E#)
r=10^floor(log10(R));
t=60;

Lpe=-147+10*log10(DE)+20*log10(f);
Gtdb=-159.6+10*log10(nu)+20*log10(D)+20*log10(f);
Grdb=-159.6+10*log10(nu)+20*log10(Dr)+20*log10(f);
EIRP=Pt*Gt;
EIRPdb=Pdb+Gtdb;
DEG=13;

Pr=(EIRPdb)*Grdb/(Lp*L0);
Pn=k*Ts*B;
SNR=Pr/Pn;
N0=k*Ts;
C=B*log2(1+SNR)/10^6;

Eb_N0=Pdb+Grdb+Grdb-L0-Lp-10*log10(R)-(-228.6+10*log10(Ts));
Margin=Eb_N0-5;

ParamRFSatLink.Altitude=DE;
ParamRFSatLink.Frequency= f;
ParamRFSatLink.SamplesPerSymbol= t*R/r;
ParamRFSatLink.FilterSpan= 6;
ParamRFSatLink.preDistortion= 0;
ParamRFSatLink.sourceSampleTime= 1/(2*r);
ParamRFSatLink.sourceSamplesPerSymbol= (t/2)*R/(r/100);
ParamRFSatLink.resetBER= 0;
ParamRFSatLink.GindB= -30.6266;
ParamRFSatLink.GoutdB= 32.9118;
ParamRFSatLink.RXTemp= Ts;
ParamRFSatLink.DoppOffset= 0.7;
ParamRFSatLink.IQImbal= [0 0 0 0];
ParamRFSatLink.PhaseNoise= -100;
ParamRFSatLink.Phaseoff=0;
```

```
ParamRFSatLink.DCBlock= 0;
ParamRFSatLink.CarrierSync= 1;
ParamRFSatLink.IQComp= 0;
ParamRFSatLink.TXAntGain= Gtdb;
ParamRFSatLink.RXAntGain= Grdb;

%%
open_system('Coms_Try_2')
set_param('Coms_Try_2', 'StopTime', '.1')
sim('Coms_Try_2')
ber=BER.Data;
idx=length(ber);
Dr=ber(idx,3);
err=ber(idx,1);
time=BER.time;
simtime=time(idx);
fprintf('Data rate is %f MB/s\n',Dr*err/(simtime*1E6));
fprintf('Eb/N0 margin is %s db\n',Margin);
fprintf('Allowable jitter is %f deg \n',DEG/2);



%% Tyler Sholes
% MAE 491-02
%Orbiter team

clear, clc, close all
format compact
%% Constants

g0=32.2; %ft/s^2
g0m=9.81; %m/s^2
C1=1; %lbm to lbm
C2=1; %lbf to lbf
C3=2.20462; % kg to lbm
C4=1/2203.62; %metrick tons to lbm
C5=62.428; %g/cm^3 to lbm/ft^3
C6=3.53147e-8; %mm^3 to ft^3
C7=0.224809; %lbf to N
C8=0.0283168; %ft^3 to m^3
C9=3280.84; %Km/s to ft/s
%% Variables
```

```
DryMass=3924.228; %Actual Dry Mass
%DryMass=5920.4*1.13 %Mission growth margin
%DryMass=3000; %expiremental values

C3in=0:.001:85; %Input
%C3in=12:.001:13;
%C3in=12.3; %My optimal

%% Functions
DeltaVBraking=2808*C9;
DeltaVKickStage=(85)^(1/2)*3280.84-(C3in).^(1/2)*3280.84; %Jupiter Gravity Assist
%DeltaVKickStage=10.3*3280.84-(C3in).^(1/2)*3280.84; %Saturn Direct
[Madd, MFT1, ML, ek, eb, wmb, idx, BrakeMotorName, KickMotorName, Check, mlfb]=...

Senior_Design_Motors(C3in,DryMass,DeltaVBraking,DeltaVKickStage,C1,C2,C3,C4,
g0);

MT=0.0022*C3in.^2-0.5923*C3in+41.168;
C3out=-21.3201*((MFT1-1.30217).^(1/2)-6.31394);
VL=(ML)/(1.021);
VLE=(766010698.704-236877010.714)*C6;
VT=0;
ELT=VL/(VLE+VT);
el=ceil(ELT);
MFT2=MFT1+Madd;
C3outn=-21.3201*((MFT2-1.30217).^(1/2)-6.31394);
C3total=85-C3in+C3out;

Whole=zeros(12,length(C3in));
Whole(1,:)=zeros(1,length(C3in));
Whole(2,:)=C3in;
Whole(3,:)=C3out;
Whole(4,:)=C3outn;
Whole(5,:)=Madd;
Whole(6,:)=MT;
Whole(7,:)=MFT1;
Whole(8,:)=MFT2;
% Whole(9,:)=ek;
% Whole(10,:)=eb;
```

```
Whole(11,:)=ML;
Whole(12,:)=(wmb)*C4;
MAX=Whole(:,idx);
%Tmax=max(MAX(9)*MaxThrustKick*C7,MAX(10)*MaxThrustBrake*C7);

%% plotting
% figure(1)
% scatter(C3in,Madd);
% hold on
% fplot(@(x) 0*x);
% hold off
% title('Additional mass needed for each C3');
% xlim([0 85])
% ylabel('Additional mass nessisary');
% xlabel('C3');
% figure(2)
% hold on
% scatter(C3in,Check(1,:),1,[0, 0, 1])
% scatter(C3in,Check(2,:),1,[.5, .5, 0])
% scatter(C3in,Check(3,:),1,     [1, 0, 0])
% scatter(C3in,Check(4,:),1,[0, 1, 1])
% scatter(C3in,Check(5,:),1,[0, 1, 0])
% fplot(@(x) 1+0*x,'linewidth',1,'color','k');
% title('Velocity ratio for each motor');
% xlim([0 85])
% ylabel('Velocity ratio');
% xlabel('C3');
% legend({'Orion 50XL','Orion 50','Orion 50S XLT','Orion 32','Orion
38'},'Location','northeast')
% hold off

%% Outputs
fprintf('C3 input = %f\n',Whole(2,idx))
fprintf('C3 output without added mass = %f\n',MAX(3))
fprintf('C3 out with added mass = %f\n',MAX(4))
fprintf('Added mass nessisary = %f metric Tons\n',MAX(5))
fprintf('Mass SLS can take to C3 input = %f metric Tons\n',MAX(6))
fprintf('Mass of Payload to take us to Saturn = %f metric Tons\n',MAX(7))
fprintf('Number of kick stage solid rocket motors = %d %s
engines\n',MAX(9),BrakeMotorName)
fprintf('Number of brake stage solid rocket motors = %d %s
engines\n',MAX(10),KickMotorName)
```

```
fprintf('Volume nessisary to store the liquid fuel is %f cubic m\n',VL)
% fprintf('Number of liquid engines is %d with %d cubit meters of extra fuel tank volume \n',el, VT*el)
% fprintf('Total number of engines = %d \n',MAX(9)+MAX(10)+el)
% fprintf('Maximum force on spacecraft is %f N\n',Tmax)
% fprintf('Maximum acceleration on spacecraft is %f m/s^2\n',Tmax/(MFT1*1000))
% fprintf('Maximum acceleration on spacecraft is %f Gs\n',(Tmax/(MFT1*1000))/g0m)
fprintf('Mass of liquid propelent needed to take us to Saturn = %f metric Tons\n',MAX(11))
fprintf('Mass arriving at Saturn = %f metric Tons\n',MAX(12))
%C3total(idx)



clear,clc
TerrestrialPlanetScaling=1.2e3;
format longE
n=2; C1=-1; C2=1; C4=-5100; C3=85; C5=1; %play with this line
jd=juliandate('30-apr-2020','dd-mmm-yyyy');
[posEarth,velEarth]=planetEphemeris(jd,'SolarSystem','Earth','405','km');
[posSaturn,velSaturn]=planetEphemeris(jd+1560,'SolarSystem','Saturn','432t','km'
); %13761  14073.5
[posJupiter,velJupiter]=planetEphemeris(jd+220,'SolarSystem','Jupiter','405','km');
[posSun,velSun]=planetEphemeris(jd,'SolarSystem','Sun','405','km');
[posMars,velMars]=planetEphemeris(jd,'SolarSystem','Mars','405','km');
[posMercury,velMercury]=planetEphemeris(jd,'SolarSystem','Mercury','405','km');
[posNeptune,velNeptune]=planetEphemeris(jd,'SolarSystem','Neptune','405','km');
[posUranus,velUranus]=planetEphemeris(jd,'SolarSystem','Uranus','405','km');
[posVenus,velVenus]=planetEphemeris(jd,'SolarSystem','Venus','405','km');
Spacecraft.Px = -posEarth(1)*1000+C2*6.05e6*TerrestrialPlanetScaling; ...
   Spacecraft.Py = -posEarth(2)*1000; Spacecraft.Pz = -(posEarth(3)*1000);
Spacecraft.Vx = -C5*(velEarth(1)*1000+n*1000); ...
   Spacecraft.Vy = -C5*(velEarth(2)*1000+1000*C1*(C3-n^2-
(2800/1000)^2)^(1/2)); ...
   Spacecraft.Vz = -C5*(velEarth(3)*1000+C4);


PE=(posEarth(1)^2+posEarth(2)^2+posEarth(3)^2)^(1/2);
TE=atan(posEarth(2)/posEarth(1));
PhiE=atan((posEarth(1)^2+posEarth(2)^2)^(1/2)/posEarth(3));
```

```
PJ=(posJupiter(1)^2+posJupiter(2)^2+posJupiter(3)^2)^(1/2);
TJ=atan(posJupiter(2)/posJupiter(1));
PhiJ=atan((posJupiter(1)^2+posJupiter(2)^2)^(1/2)/posJupiter(3));
PobjJ=PJ-PE;
TobjJ=TJ-TE;
PhiobjJ=PhiJ-PhiE;

PS=(posSaturn(1)^2+posSaturn(2)^2+posSaturn(3)^2)^(1/2);
TS=atan(posSaturn(2)/posSaturn(1));
PhiS=atan((posSaturn(1)^2+posSaturn(2)^2)^(1/2)/posSaturn(3));
PobjS=PS-PE;
TobjS=TS-TE;
PhiobjS=PhiS-PhiE;

jdstart=juliandate('31-jan-2000','dd-mmm-yyyy');
jdend=juliandate('31-dec-2020','dd-mmm-yyyy');

for t=jdstart:(1):jdend
   [posEarth,velEarth]=planetEphemeris(t,'SolarSystem','Earth','405','km');
   [posSaturn,velSaturn]=planetEphemeris(t,'SolarSystem','Saturn','432t','km');
   [posJupiter,velJupiter]=planetEphemeris(t,'SolarSystem','Jupiter','405','km');

   PE=(posEarth(1)^2+posEarth(2)^2+posEarth(3)^2)^(1/2);
   TE=atan(posEarth(2)/posEarth(1));
   PhiE=atan((posEarth(1)^2+posEarth(2)^2)^(1/2)/posEarth(3));

   PJ=(posJupiter(1)^2+posJupiter(2)^2+posJupiter(3)^2)^(1/2);
   TJ=atan(posJupiter(2)/posJupiter(1));
   PhiJ=atan((posJupiter(1)^2+posJupiter(2)^2)^(1/2)/posJupiter(3));
   PtJ=PJ-PE;
   TtJ=TJ-TE;
   PhitJ=PhiJ-PhiE;

   PS=(posSaturn(1)^2+posSaturn(2)^2+posSaturn(3)^2)^(1/2);
   TS=atan(posSaturn(2)/posSaturn(1));
   PhiS=atan((posSaturn(1)^2+posSaturn(2)^2)^(1/2)/posSaturn(3));
   PtS=PS-PE;
   TtS=TS-TE;
   PhitS=PhiS-PhiE;

%     if PtJ==PobjJ || PtJ==-PobjJ
      if TobjJ+.1>=TtJ && TobjJ-.1<=TtJ
```

```
%            if PhitJ<=PhiobjJ+.1  && PhitJ>=PhiobjJ-.1
%               if PtS==PobjS || PtJ==-PobjS
                if TtS>=TobjS-.1 && TobjS+.1>=TtS
%                     if PhitS>=PhiobjS-.1  && PhitS<=PhiobjS+.1
                      t
                      launch_date=datetime(t,'convertfrom','juliandate')

%                     end
                   fprintf('N=5')
                end
%                  fprintf('N=4')
%               end
%               fprintf('N=3')
%            end
          fprintf('N=2')
       end
       if TtS>=TobjS-.1 && TobjS+.1>=TtS
          fprintf('N=5')
       end
%         fprintf('N=1')
%     end



   disp(((t-jdstart)/(jdend-jdstart))*100)
end



%-------------------------------------------------------
% All values are in SI units.
% Dimensions are scaled for visualization purposes.
% Scaling has no impact on model dynamics.
format longE


S=csvread('State_Vectors_to_S.csv');
idx=length(S);


% Model-Wide Parameters
G = 6.67e-11;
```

```
% Scaling factors
SunScaling = 0.5e2;
TerrestrialPlanetScaling = 1.2e3;
GasGiantScaling = 1; %2.5e2;

n=2; C1=-1; C2=1; C4=-5100; C3=85; C5=1; %play with this line
jd=juliandate('30-apr-2020','dd-mmm-yyyy')+1.0715E8*0.000011;
[posEarth,velEarth]=planetEphemeris(jd,'SolarSystem','Earth','405','km');
[posSaturn,velSaturn]=planetEphemeris(jd+1560,'SolarSystem','Saturn','432t','km'
); %13761  14073.5
[posJupiter,velJupiter]=planetEphemeris(jd+220,'SolarSystem','Jupiter','405','km');
[posSun,velSun]=planetEphemeris(jd,'SolarSystem','Sun','405','km');
[posMars,velMars]=planetEphemeris(jd,'SolarSystem','Mars','405','km');
[posMercury,velMercury]=planetEphemeris(jd,'SolarSystem','Mercury','405','km');
[posNeptune,velNeptune]=planetEphemeris(jd,'SolarSystem','Neptune','405','km');
[posUranus,velUranus]=planetEphemeris(jd,'SolarSystem','Uranus','405','km');
[posVenus,velVenus]=planetEphemeris(jd,'SolarSystem','Venus','405','km');


posEnceladus.Px=-posSaturn(1)*1000+3000000; posEnceladus.Py=-
posSaturn(2)*1000; posEnceladus.Pz=-posSaturn(3)*1000;
velEnceladus.Vx=-velSaturn(1)*1000; velEnceladus.Vy=-velSaturn(2)*1000;
velEnceladus.Vz=-velSaturn(3)*1000;

% Spacecraft
Spacecraft.M=2684;
% Spacecraft.Px = -posEarth(1)*1000+C2*6.05e6*TerrestrialPlanetScaling;
Spacecraft.Py = -posEarth(2)*1000; Spacecraft.Pz = -(posEarth(3)*1000);
% Spacecraft.Vx = -C5*(velEarth(1)*1000+n*1000); Spacecraft.Vy = -
C5*(velEarth(2)*1000+1000*C1*(C3-n^2-(2800/1000)^2)^(1/2)); Spacecraft.Vz
= -C5*(velEarth(3)*1000+C4);
Spacecraft.Px = S(idx,2); Spacecraft.Py = S(idx,3); Spacecraft.Pz = S(idx,4);
Spacecraft.Vx = S(idx,5); Spacecraft.Vy = S(idx,6); Spacecraft.Vz = S(idx,7);



% Sun Parameters
Sun.M = 1.99e30;Sun.R = 6.96e8*SunScaling;
Sun.Px = posSun(1)*1000; Sun.Py = posSun(2)*1000;Sun.Pz = posSun(3)*1000;
Sun.Vx = velSun(1)*1000; Sun.Vy = velSun(2)*1000;Sun.Vz = velSun(3)*1000;
```

```
% Mercury Parameters
Mercury.M =3.30e23; Mercury.R = 2.44e6*TerrestrialPlanetScaling;
Mercury.Px = posMercury(1)*1000; Mercury.Py = posMercury(2)*1000; Mercury.Pz = posMercury(3)*1000;
Mercury.Vx = velMercury(1)*1000; Mercury.Vy = velMercury(2)*1000; Mercury.Vz = velMercury(3)*1000;

% Venus Parameters
Venus.M = 4.87e24;Venus.R = 6.05e6*TerrestrialPlanetScaling;
Venus.Px = posVenus(1)*1000; Venus.Py = posVenus(2)*1000;Venus.Pz = posVenus(3)*1000;
Venus.Vx = velVenus(1)*1000; Venus.Vy = velVenus(2)*1000;Venus.Vz = velVenus(3)*1000;

% Earth Parameters
Earth.M = 5.97e24;Earth.R = 6.05e6*TerrestrialPlanetScaling;
Earth.Px = posEarth(1)*1000; Earth.Py = posEarth(2)*1000;Earth.Pz = posEarth(3)*1000;
Earth.Vx = velEarth(1)*1000; Earth.Vy = velEarth(2)*1000;Earth.Vz = velEarth(3)*1000;

% Mars Parameters
Mars.M = 6.42e23;Mars.R = 3.39e6*TerrestrialPlanetScaling;
Mars.Px = posMars(1)*1000; Mars.Py = posMars(2)*1000;Mars.Pz = posMars(3)*1000;
Mars.Vx = velMars(1)*1000; Mars.Vy = velMars(2)*1000;Mars.Vz = velMars(3)*1000;

% Jupiter Parameters
Jupiter.M = 1.90e27;Jupiter.R = 6.99e7*GasGiantScaling;Jupiter.RGB = [0.8 0.8 0.8];
Jupiter.Px = posJupiter(1)*1000; Jupiter.Py = posJupiter(2)*1000;Jupiter.Pz = posJupiter(3)*1000;
Jupiter.Vx = velJupiter(1)*1000; Jupiter.Vy = velJupiter(2)*1000;Jupiter.Vz = velJupiter(3)*1000;

% Saturn Parameters
Saturn.M = 5.68e26;Saturn. R = 5.82e7*GasGiantScaling; Saturn.RGB = [0.8 0.8 0.8];
Saturn.Px = S(idx,8); Saturn.Py = S(idx,9); Saturn.Pz =S(idx,10);
Saturn.Vx = S(idx,11); Saturn.Vy = S(idx,12); Saturn.Vz = S(idx,13);
```

```
% Saturn.Px = posSaturn(1)*1000; Saturn.Py = posSaturn(2)*1000; Saturn.Pz =
posSaturn(3)*1000;
% Saturn.Vx = velSaturn(1)*1000; Saturn.Vy = velSaturn(2)*1000; Saturn.Vz =
velSaturn(3)*1000;

% Enceladus
Enceladus.M = 1.08E+20; Enceladus.R = 2.52E+5*GasGiantScaling; Enceladus.RGB
= [0 0 0];
Enceladus.Px=S(idx,14); Enceladus.Py=S(idx,15); Enceladus.Pz=S(idx,16);
Enceladus.Vx=S(idx,17); Enceladus.Vy=S(idx,18); Enceladus.Vz=S(idx,19);

% Uranus Parameters
Uranus.M = 8.68e25;Uranus.R = 2.54e7*GasGiantScaling;Uranus.RGB = [0.8 0.8
0.8];
Uranus.Px = posUranus(1)*1000; Uranus.Py = posUranus(2)*1000;Uranus.Pz =
posUranus(3)*1000;
Uranus.Vx = velUranus(1)*1000; Uranus.Vy = velUranus(2)*1000;Uranus.Vz =
velUranus(3)*1000;

% Neptune Parameters
Neptune.M = 1.02e26;Neptune.R = 2.46e7*GasGiantScaling;Neptune.RGB = [0.8
0.8 0.8];
Neptune.Px = posNeptune(1)*1000; Neptune.Py = posNeptune(2)*1000;Neptune.Pz
= posNeptune(3)*1000;
Neptune.Vx = velNeptune(1)*1000; Neptune.Vy = velNeptune(2)*1000;Neptune.Vz
= velNeptune(3)*1000;

%% Deltas
DVx= Spacecraft.Vx-Enceladus.Vx
DVy= Spacecraft.Vy-Enceladus.Vy
DVz= Spacecraft.Vz-Enceladus.Vz

Fx= Spacecraft.M*DVx/100
Fy= Spacecraft.M*DVy/100
Fz= Spacecraft.M*DVz/100



% ------------------------------------------------------------
%  Generated by MATLAB on 30-Mar-2019 13:10:38
%  MATLAB version: 9.6.0.1072779 (R2019a)
% ------------------------------------------------------------
```

```
paramRFSatLink = struct;
paramRFSatLink.Altitude = 1.2E+9;
paramRFSatLink.Frequency = 38036;
paramRFSatLink.SamplesPerSymbol = 8;
paramRFSatLink.FilterSpan = 6;
paramRFSatLink.preDistortion = 0;
paramRFSatLink.sourceSampleTime = 1E-5;
paramRFSatLink.sourceSamplesPerSymbol = 400;
paramRFSatLink.resetBER = 0;
paramRFSatLink.GindB = -30.62657180225564;
paramRFSatLink.GoutdB = 32.911826101838727;
paramRFSatLink.RXTemp = 20;
paramRFSatLink.DoppOffset = 0.7;
paramRFSatLink.IQImbal = [0 0 0 0];
paramRFSatLink.PhaseNoise = -100;
paramRFSatLink.DCBlock = false;
paramRFSatLink.CarrierSync = true;
paramRFSatLink.IQComp = false;
paramRFSatLink.TXAntGain = 1181.5845204057846;
paramRFSatLink.RXAntGain = 10043.468423449169;



sim('Basic_solar_system_W_launch_date');
State_Vectors(:,1)=Time.signals.values;
State_Vectors(:,2)=X.signals.values;
State_Vectors(:,3)=Y.signals.values;
State_Vectors(:,4)=Z.signals.values;
State_Vectors(:,5)=Vx.signals.values;
State_Vectors(:,6)=Vy.signals.values;
State_Vectors(:,7)=Vz.signals.values;
State_Vectors(:,8)=SX.signals.values;
State_Vectors(:,9)=SY.signals.values;
State_Vectors(:,10)=SZ.signals.values;
State_Vectors(:,11)=SVx.signals.values;
State_Vectors(:,12)=SVy.signals.values;
State_Vectors(:,13)=SVz.signals.values;
State_Vectors(:,14)=EX.signals.values;
State_Vectors(:,15)=EY.signals.values;
State_Vectors(:,16)=EZ.signals.values;
```

```
State_Vectors(:,17)=EVx.signals.values;
State_Vectors(:,18)=EVy.signals.values;
State_Vectors(:,19)=EVz.signals.values;
dlmwrite('State_Vectors_to_S.csv',State_Vectors,'delimiter', ',', 'precision', 15);


%-------------------------------------------------------
% All values are in SI units.
% Dimensions are scaled for visualization purposes.
% Scaling has no impact on model dynamics.
clear,clc

% Model-Wide Parameters
G = 6.67e-11; % Universal gravitational constant

% Scaling factors
SunScaling = 0.5e2;
TerrestrialPlanetScaling = 1.2e3;
GasGiantScaling = 1; %2.5e2;

n=2; C1=-1; C2=1; C4=-5100; C3=85; C5=1; %play with this line
jd=juliandate('30-apr-2020','dd-mmm-yyyy');
[posEarth,velEarth]=planetEphemeris(jd,'SolarSystem','Earth','405','km');
[posSaturn,velSaturn]=planetEphemeris(jd+1560,'SolarSystem','Saturn','432t','km'
); %13761 14073.5
[posJupiter,velJupiter]=planetEphemeris(jd+220,'SolarSystem','Jupiter','405','km');
[posSun,velSun]=planetEphemeris(jd,'SolarSystem','Sun','405','km');
[posMars,velMars]=planetEphemeris(jd,'SolarSystem','Mars','405','km');
[posMercury,velMercury]=planetEphemeris(jd,'SolarSystem','Mercury','405','km');
[posNeptune,velNeptune]=planetEphemeris(jd,'SolarSystem','Neptune','405','km');
[posUranus,velUranus]=planetEphemeris(jd,'SolarSystem','Uranus','405','km');
[posVenus,velVenus]=planetEphemeris(jd,'SolarSystem','Venus','405','km');


posEnceladus.Px=-posSaturn(1)*1000+3000000;                posEnceladus.Py=-
posSaturn(2)*1000; posEnceladus.Pz=-posSaturn(3)*1000;
velEnceladus.Vx=-velSaturn(1)*1000;          velEnceladus.Vy=-velSaturn(2)*1000;
velEnceladus.Vz=-velSaturn(3)*1000;

% Spacecraft
Spacecraft.M=2684;
```

```
Spacecraft.Px = -posEarth(1)*1000+C2*6.05e6*TerrestrialPlanetScaling;
Spacecraft.Py = -posEarth(2)*1000; Spacecraft.Pz = -(posEarth(3)*1000);
Spacecraft.Vx = -C5*(velEarth(1)*1000+n*1000); Spacecraft.Vy = -
C5*(velEarth(2)*1000+1000*C1*(C3-n^2-(2800/1000)^2)^(1/2)); Spacecraft.Vz
= -C5*(velEarth(3)*1000+C4);

% Sun Parameters
Sun.M = 1.99e30;Sun.R = 6.96e8*SunScaling;
Sun.Px = posSun(1)*1000; Sun.Py = posSun(2)*1000;Sun.Pz = posSun(3)*1000;
Sun.Vx = velSun(1)*1000; Sun.Vy = velSun(2)*1000;Sun.Vz = velSun(3)*1000;

% Mercury Parameters
Mercury.M =3.30e23; Mercury.R = 2.44e6*TerrestrialPlanetScaling;
Mercury.Px = posMercury(1)*1000; Mercury.Py = posMercury(2)*1000; Mercury.Pz
= posMercury(3)*1000;
Mercury.Vx = velMercury(1)*1000; Mercury.Vy = velMercury(2)*1000; Mercury.Vz
= velMercury(3)*1000;

% Venus Parameters
Venus.M = 4.87e24;Venus.R = 6.05e6*TerrestrialPlanetScaling;
Venus.Px = posVenus(1)*1000; Venus.Py = posVenus(2)*1000;Venus.Pz =
posVenus(3)*1000;
Venus.Vx = velVenus(1)*1000; Venus.Vy = velVenus(2)*1000;Venus.Vz =
velVenus(3)*1000;

% Earth Parameters
Earth.M = 5.97e24;Earth.R = 6.05e6*TerrestrialPlanetScaling;
Earth.Px = posEarth(1)*1000; Earth.Py = posEarth(2)*1000;Earth.Pz =
posEarth(3)*1000;
Earth.Vx = velEarth(1)*1000; Earth.Vy = velEarth(2)*1000;Earth.Vz =
velEarth(3)*1000;

% Mars Parameters
Mars.M = 6.42e23;Mars.R = 3.39e6*TerrestrialPlanetScaling;
Mars.Px = posMars(1)*1000; Mars.Py = posMars(2)*1000;Mars.Pz =
posMars(3)*1000;
Mars.Vx = velMars(1)*1000; Mars.Vy = velMars(2)*1000;Mars.Vz =
velMars(3)*1000;

% Jupiter Parameters
Jupiter.M = 1.90e27;Jupiter.R = 6.99e7*GasGiantScaling;Jupiter.RGB = [0.8 0.8
0.8];
```

```
Jupiter.Px = posJupiter(1)*1000; Jupiter.Py = posJupiter(2)*1000;Jupiter.Pz = posJupiter(3)*1000;
Jupiter.Vx = velJupiter(1)*1000; Jupiter.Vy = velJupiter(2)*1000;Jupiter.Vz = velJupiter(3)*1000;

% Saturn Parameters
Saturn.M = 5.68e26;Saturn. R = 5.82e7*GasGiantScaling; Saturn.RGB = [0.8 0.8 0.8];
Saturn.Px = posSaturn(1)*1000; Saturn.Py = posSaturn(2)*1000; Saturn.Pz = posSaturn(3)*1000;
Saturn.Vx = velSaturn(1)*1000; Saturn.Vy = velSaturn(2)*1000; Saturn.Vz = velSaturn(3)*1000;

% Enceladus
Enceladus.M = 1.08E+20; Enceladus.R = 2.52E+5*GasGiantScaling; Enceladus.RGB = [0 0 0];
Enceladus.Px=posSaturn(1)*1000+238020*1000+Saturn.R;
Enceladus.Py=posSaturn(2)*1000; Enceladus.Pz=posSaturn(3)*1000;
Enceladus.Vx=velSaturn(1)*1000; Enceladus.Vy=velSaturn(2)*1000+12.64*1000;
Enceladus.Vz=velSaturn(3)*1000;

% Uranus Parameters
Uranus.M = 8.68e25;Uranus.R = 2.54e7*GasGiantScaling;Uranus.RGB = [0.8 0.8 0.8];
Uranus.Px = posUranus(1)*1000; Uranus.Py = posUranus(2)*1000;Uranus.Pz = posUranus(3)*1000;
Uranus.Vx = velUranus(1)*1000; Uranus.Vy = velUranus(2)*1000;Uranus.Vz = velUranus(3)*1000;

% Neptune Parameters
Neptune.M = 1.02e26;Neptune.R = 2.46e7*GasGiantScaling;Neptune.RGB = [0.8 0.8 0.8];
Neptune.Px = posNeptune(1)*1000; Neptune.Py = posNeptune(2)*1000;Neptune.Pz = posNeptune(3)*1000;
Neptune.Vx = velNeptune(1)*1000; Neptune.Vy = velNeptune(2)*1000;Neptune.Vz = velNeptune(3)*1000;



%-------------------------------------------------------
% All values are in SI units.
% Dimensions are scaled for visualization purposes.
```

```
% Scaling has no impact on model dynamics.
clear,clc

% Model-Wide Parameters
G = 6.67e-11; % Universal gravitational constant

% Scaling factors
SunScaling = 5e1;
TerrestrialPlanetScaling = 1.2e3;
GasGiantScaling = 2.5e2;
SC=1.2e9;

n=2; C1=-1; C2=1; C4=-5100; C3=85; C5=1; %play with this line
jd=juliandate('30-apr-2020','dd-mmm-yyyy'); %30-apr-2020
[posEarth,velEarth]=planetEphemeris(jd,'SolarSystem','Earth','405','km');
[posSaturn,velSaturn]=planetEphemeris(jd+1560,'SolarSystem','Saturn','432t','km'
); %13761 14073.5
[posJupiter,velJupiter]=planetEphemeris(jd+220,'SolarSystem','Jupiter','405','km');
[posSun,velSun]=planetEphemeris(jd,'SolarSystem','Sun','405','km');
[posMars,velMars]=planetEphemeris(jd,'SolarSystem','Mars','405','km');
[posMercury,velMercury]=planetEphemeris(jd,'SolarSystem','Mercury','405','km');
[posNeptune,velNeptune]=planetEphemeris(jd,'SolarSystem','Neptune','405','km');
[posUranus,velUranus]=planetEphemeris(jd,'SolarSystem','Uranus','405','km');
[posVenus,velVenus]=planetEphemeris(jd,'SolarSystem','Venus','405','km');


posEnceladus.Px=-posSaturn(1)*1000+3000000; posEnceladus.Py=-
posSaturn(2)*1000; posEnceladus.Pz=-posSaturn(3)*1000;
velEnceladus.Vx=-velSaturn(1)*1000; velEnceladus.Vy=-velSaturn(2)*1000;
velEnceladus.Vz=-velSaturn(3)*1000;

% Spacecraft
Spacecraft.M=2684;
Spacecraft.Px = -posEarth(1)*1000+C2*6.05e6*TerrestrialPlanetScaling;
Spacecraft.Py = -posEarth(2)*1000; Spacecraft.Pz = -(posEarth(3)*1000);
Spacecraft.Vx = -C5*(velEarth(1)*1000+n*1000); Spacecraft.Vy = -
C5*(velEarth(2)*1000+1000*C1*(C3-n^2-(2800/1000)^2)^(1/2)); Spacecraft.Vz
= -C5*(velEarth(3)*1000+C4);

% Sun Parameters
Sun.M = 1.99e30;Sun.R = 6.96e8*SunScaling;
Sun.Px = posSun(1)*1000; Sun.Py = posSun(2)*1000;Sun.Pz = posSun(3)*1000;
```

```
Sun.Vx = velSun(1)*1000; Sun.Vy = velSun(2)*1000;Sun.Vz = velSun(3)*1000;

% Mercury Parameters
Mercury.M =3.30e23; Mercury.R = 2.44e6*TerrestrialPlanetScaling;
Mercury.Px = posMercury(1)*1000; Mercury.Py = posMercury(2)*1000; Mercury.Pz
= posMercury(3)*1000;
Mercury.Vx = velMercury(1)*1000; Mercury.Vy = velMercury(2)*1000; Mercury.Vz
= velMercury(3)*1000;

% Venus Parameters
Venus.M = 4.87e24;Venus.R = 6.05e6*TerrestrialPlanetScaling;
Venus.Px = posVenus(1)*1000; Venus.Py = posVenus(2)*1000;Venus.Pz =
posVenus(3)*1000;
Venus.Vx = velVenus(1)*1000; Venus.Vy = velVenus(2)*1000;Venus.Vz =
velVenus(3)*1000;

% Earth Parameters
Earth.M = 5.97e24;Earth.R = 6.05e6*TerrestrialPlanetScaling;
Earth.Px = posEarth(1)*1000; Earth.Py = posEarth(2)*1000;Earth.Pz =
posEarth(3)*1000;
Earth.Vx = velEarth(1)*1000; Earth.Vy = velEarth(2)*1000;Earth.Vz =
velEarth(3)*1000;

% Mars Parameters
Mars.M = 6.42e23;Mars.R = 3.39e6*TerrestrialPlanetScaling;
Mars.Px = posMars(1)*1000; Mars.Py = posMars(2)*1000;Mars.Pz =
posMars(3)*1000;
Mars.Vx = velMars(1)*1000; Mars.Vy = velMars(2)*1000;Mars.Vz =
velMars(3)*1000;

% Jupiter Parameters
Jupiter.M = 1.90e27;Jupiter.R = 6.99e7*GasGiantScaling;Jupiter.RGB = [0.8 0.8
0.8];
Jupiter.Px = posJupiter(1)*1000; Jupiter.Py = posJupiter(2)*1000;Jupiter.Pz =
posJupiter(3)*1000;
Jupiter.Vx = velJupiter(1)*1000; Jupiter.Vy = velJupiter(2)*1000;Jupiter.Vz =
velJupiter(3)*1000;

% Saturn Parameters
Saturn.M = 5.68e26;Saturn. R = 5.82e7*GasGiantScaling; Saturn.RGB = [0.8 0.8
0.8];
```

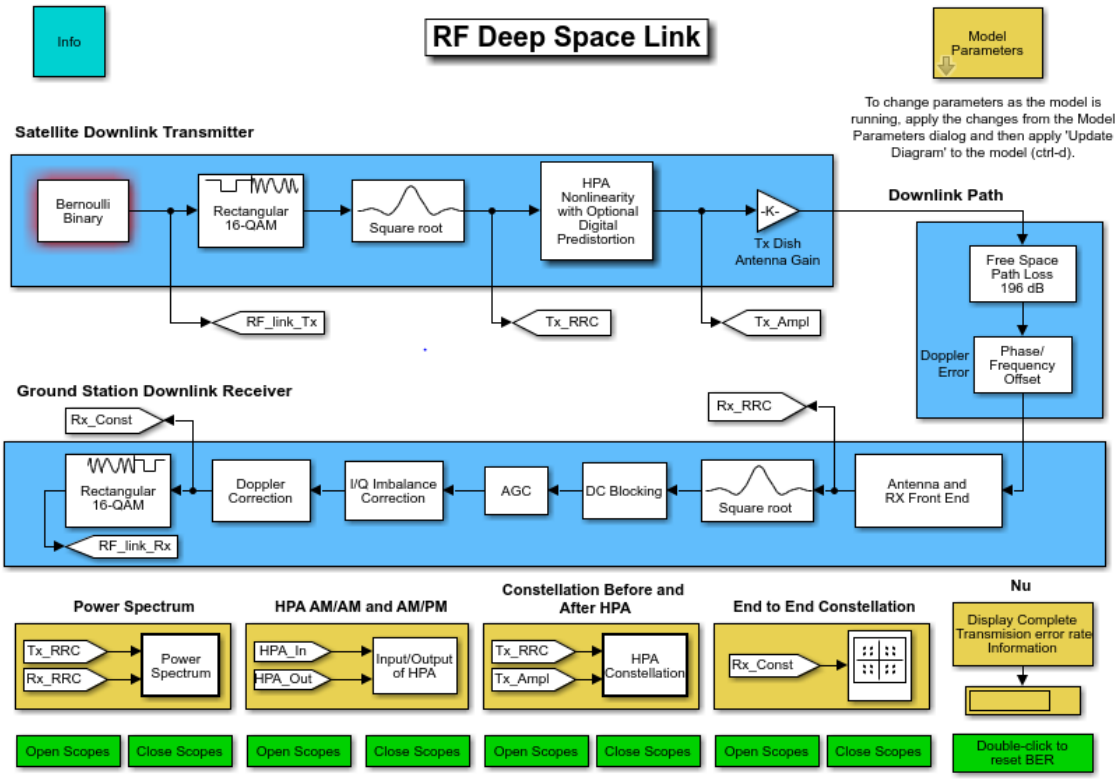
```
Saturn.Px = posSaturn(1)*1000; Saturn.Py = posSaturn(2)*1000; Saturn.Pz = posSaturn(3)*1000;
Saturn.Vx = velSaturn(1)*1000; Saturn.Vy = velSaturn(2)*1000; Saturn.Vz = velSaturn(3)*1000;

% Enceladus
Enceladus.M = 1.08E+20; Enceladus.R = 2.52E+5*GasGiantScaling; Enceladus.RGB = [0 0 0];
Enceladus.Px=posSaturn(1)*1000+238020*1000+Saturn.R/GasGiantScaling;
Enceladus.Py=posSaturn(2)*1000; Enceladus.Pz=posSaturn(3)*1000;
Enceladus.Vx=velSaturn(1)*1000;  Enceladus.Vy=velSaturn(2)*1000+12.64*1000;
Enceladus.Vz=velSaturn(3)*1000;

% Uranus Parameters
Uranus.M = 8.68e25;Uranus.R = 2.54e7*GasGiantScaling;Uranus.RGB = [0.8 0.8 0.8];
Uranus.Px = posUranus(1)*1000; Uranus.Py = posUranus(2)*1000;Uranus.Pz = posUranus(3)*1000;
Uranus.Vx = velUranus(1)*1000; Uranus.Vy = velUranus(2)*1000;Uranus.Vz = velUranus(3)*1000;

% Neptune Parameters
Neptune.M = 1.02e26;Neptune.R = 2.46e7*GasGiantScaling;Neptune.RGB = [0.8 0.8 0.8];
Neptune.Px = posNeptune(1)*1000; Neptune.Py = posNeptune(2)*1000;Neptune.Pz = posNeptune(3)*1000;
Neptune.Vx = velNeptune(1)*1000; Neptune.Vy = velNeptune(2)*1000;Neptune.Vz = velNeptune(3)*1000;



%-------------------------------------------------------
% All values are in SI units.
% Dimensions are scaled for visualization purposes.
% Scaling has no impact on model dynamics.
clear,clc

% Model-Wide Parameters
G = 6.67e-11; % Universal gravitational constant

% Scaling factors
SunScaling = 0.5e2;
```

```
TerrestrialPlanetScaling = 1.2e3;
GasGiantScaling = 1; %2.5e2;

n=6.7; C1=-1; C2=1; C4=-4600; C3=85; %play with this line
jd=juliandate('30-apr-2020','dd-mmm-yyyy');
[posEarth,velEarth]=planetEphemeris(jd,'SolarSystem','Earth','405','km');
[posSaturn,velSaturn]=planetEphemeris(jd+1400,'SolarSystem','Saturn','432t','km'
);
[posJupiter,velJupiter]=planetEphemeris(jd+445.5,'SolarSystem','Jupiter','405','km'
);
[posSun,velSun]=planetEphemeris(jd,'SolarSystem','Sun','405','km');
[posMars,velMars]=planetEphemeris(jd,'SolarSystem','Mars','405','km');
[posMercury,velMercury]=planetEphemeris(jd,'SolarSystem','Mercury','405','km');
[posNeptune,velNeptune]=planetEphemeris(jd,'SolarSystem','Neptune','405','km');
[posUranus,velUranus]=planetEphemeris(jd,'SolarSystem','Uranus','405','km');
[posVenus,velVenus]=planetEphemeris(jd,'SolarSystem','Venus','405','km');


posEnceladus.Px=-posSaturn(1)*1000+3000000; posEnceladus.Py=-
posSaturn(2)*1000; posEnceladus.Pz=-posSaturn(3)*1000;
velEnceladus.Vx=-velSaturn(1)*1000; velEnceladus.Vy=-velSaturn(2)*1000;
velEnceladus.Vz=-velSaturn(3)*1000;

% Spacecraft
Spacecraft.M=2684;
Spacecraft.Px = -posEarth(1)*1000+C2*6.05e6*TerrestrialPlanetScaling;
Spacecraft.Py = -posEarth(2)*1000; Spacecraft.Pz = -posEarth(3)*1000;
Spacecraft.Vx = -(velEarth(1)*1000+n*1000); Spacecraft.Vy = -
(velEarth(2)*1000+1000*C1*(C3-n^2-(C4/1000)^2)^(1/2)); Spacecraft.Vz = -
(velEarth(3)*1000+C4);

% Sun Parameters
Sun.M = 1.99e30;Sun.R = 6.96e8*SunScaling;
Sun.Px = posSun(1)*1000; Sun.Py = posSun(2)*1000;Sun.Pz = posSun(3)*1000;
Sun.Vx = velSun(1)*1000; Sun.Vy = velSun(2)*1000;Sun.Vz = velSun(3)*1000;

% Mercury Parameters
Mercury.M =3.30e23; Mercury.R = 2.44e6*TerrestrialPlanetScaling;
Mercury.Px = posMercury(1)*1000; Mercury.Py = posMercury(2)*1000; Mercury.Pz
= posMercury(3)*1000;
Mercury.Vx = velMercury(1)*1000; Mercury.Vy = velMercury(2)*1000; Mercury.Vz
= velMercury(3)*1000;
```

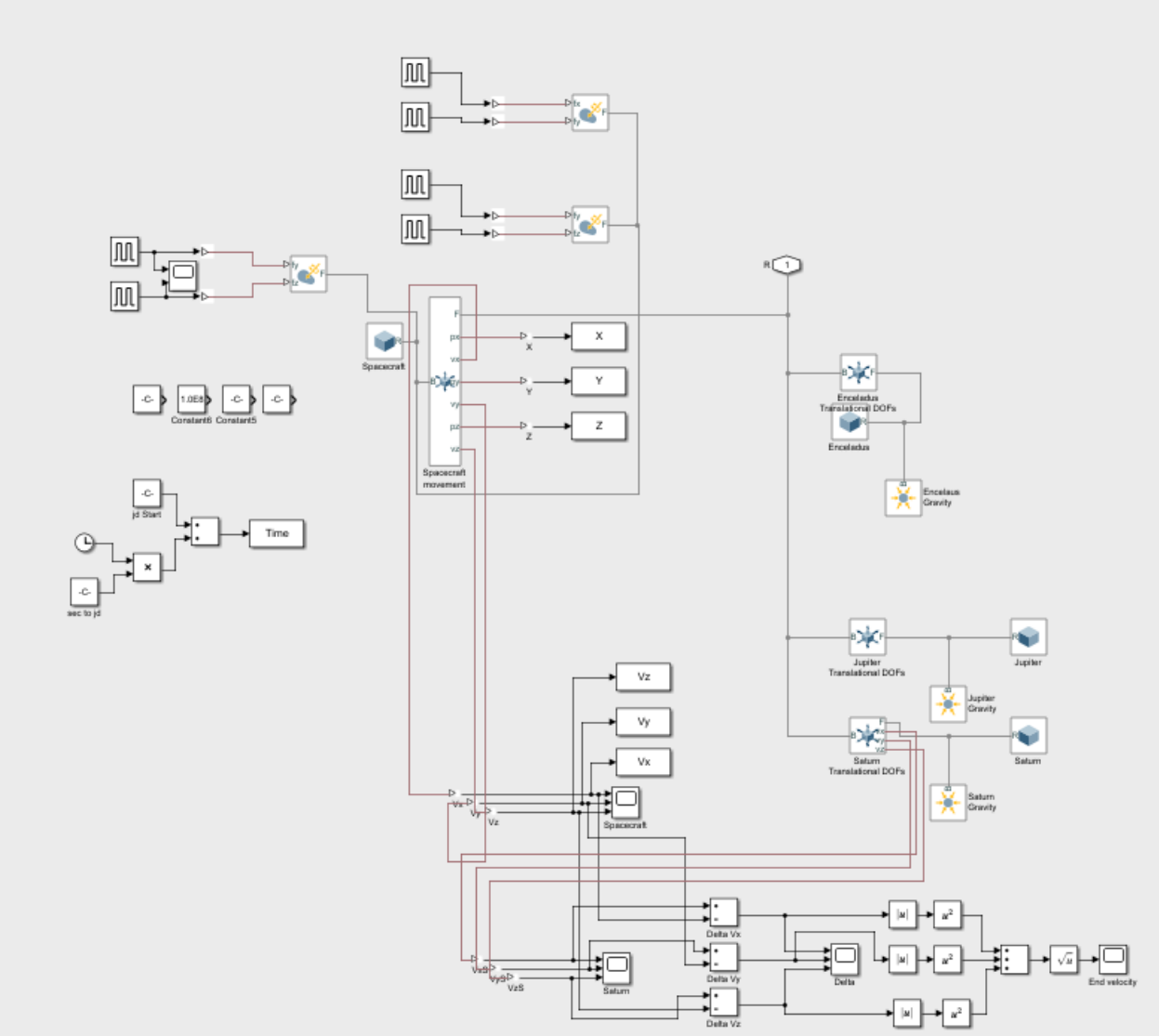
```
% Venus Parameters
Venus.M = 4.87e24;Venus.R = 6.05e6*TerrestrialPlanetScaling;
Venus.Px = posVenus(1)*1000; Venus.Py = posVenus(2)*1000;Venus.Pz = posVenus(3)*1000;
Venus.Vx = velVenus(1)*1000; Venus.Vy = velVenus(2)*1000;Venus.Vz = velVenus(3)*1000;

% Earth Parameters
Earth.M = 5.97e24;Earth.R = 6.05e6*TerrestrialPlanetScaling;
Earth.Px = posEarth(1)*1000; Earth.Py = posEarth(2)*1000;Earth.Pz = posEarth(3)*1000;
Earth.Vx = velEarth(1)*1000; Earth.Vy = velEarth(2)*1000;Earth.Vz = velEarth(3)*1000;

% Mars Parameters
Mars.M = 6.42e23;Mars.R = 3.39e6*TerrestrialPlanetScaling;
Mars.Px = posMars(1)*1000; Mars.Py = posMars(2)*1000;Mars.Pz = posMars(3)*1000;
Mars.Vx = velMars(1)*1000; Mars.Vy = velMars(2)*1000;Mars.Vz = velMars(3)*1000;

% Jupiter Parameters
Jupiter.M = 1.90e27;Jupiter.R = 6.99e7*GasGiantScaling;Jupiter.RGB = [0.8 0.8 0.8];
Jupiter.Px = posJupiter(1)*1000; Jupiter.Py = posJupiter(2)*1000;Jupiter.Pz = posJupiter(3)*1000;
Jupiter.Vx = velJupiter(1)*1000; Jupiter.Vy = velJupiter(2)*1000;Jupiter.Vz = velJupiter(3)*1000;

% Saturn Parameters
Saturn.M = 5.68e26;Saturn. R = 5.82e7*GasGiantScaling; Saturn.RGB = [0.8 0.8 0.8];
Saturn.Px = posSaturn(1)*1000; Saturn.Py = posSaturn(2)*1000; Saturn.Pz = posSaturn(3)*1000;
Saturn.Vx = velSaturn(1)*1000; Saturn.Vy = velSaturn(2)*1000; Saturn.Vz = velSaturn(3)*1000;

% Enceladus
Enceladus.M = 1.08E+20; Enceladus.R = 2.52E+5*TerrestrialPlanetScaling; Enceladus.RGB = [0 0 0];
```

```
Enceladus.Px=posSaturn(1)*1000+238020*1000+Saturn.R;
Enceladus.Py=posSaturn(2)*1000; Enceladus.Pz=posSaturn(3)*1000;
Enceladus.Vx=velSaturn(1)*1000;  Enceladus.Vy=velSaturn(2)*1000+12.64*1000;
Enceladus.Vz=velSaturn(3)*1000;

% Uranus Parameters
Uranus.M = 8.68e25;Uranus.R = 2.54e7*GasGiantScaling;Uranus.RGB = [0.8 0.8
0.8];
Uranus.Px = posUranus(1)*1000; Uranus.Py = posUranus(2)*1000;Uranus.Pz =
posUranus(3)*1000;
Uranus.Vx = velUranus(1)*1000; Uranus.Vy = velUranus(2)*1000;Uranus.Vz =
velUranus(3)*1000;

% Neptune Parameters
Neptune.M = 1.02e26;Neptune.R = 2.46e7*GasGiantScaling;Neptune.RGB = [0.8
0.8 0.8];
Neptune.Px = posNeptune(1)*1000; Neptune.Py = posNeptune(2)*1000;Neptune.Pz
= posNeptune(3)*1000;
Neptune.Vx = velNeptune(1)*1000; Neptune.Vy = velNeptune(2)*1000;Neptune.Vz
= velNeptune(3)*1000;
```

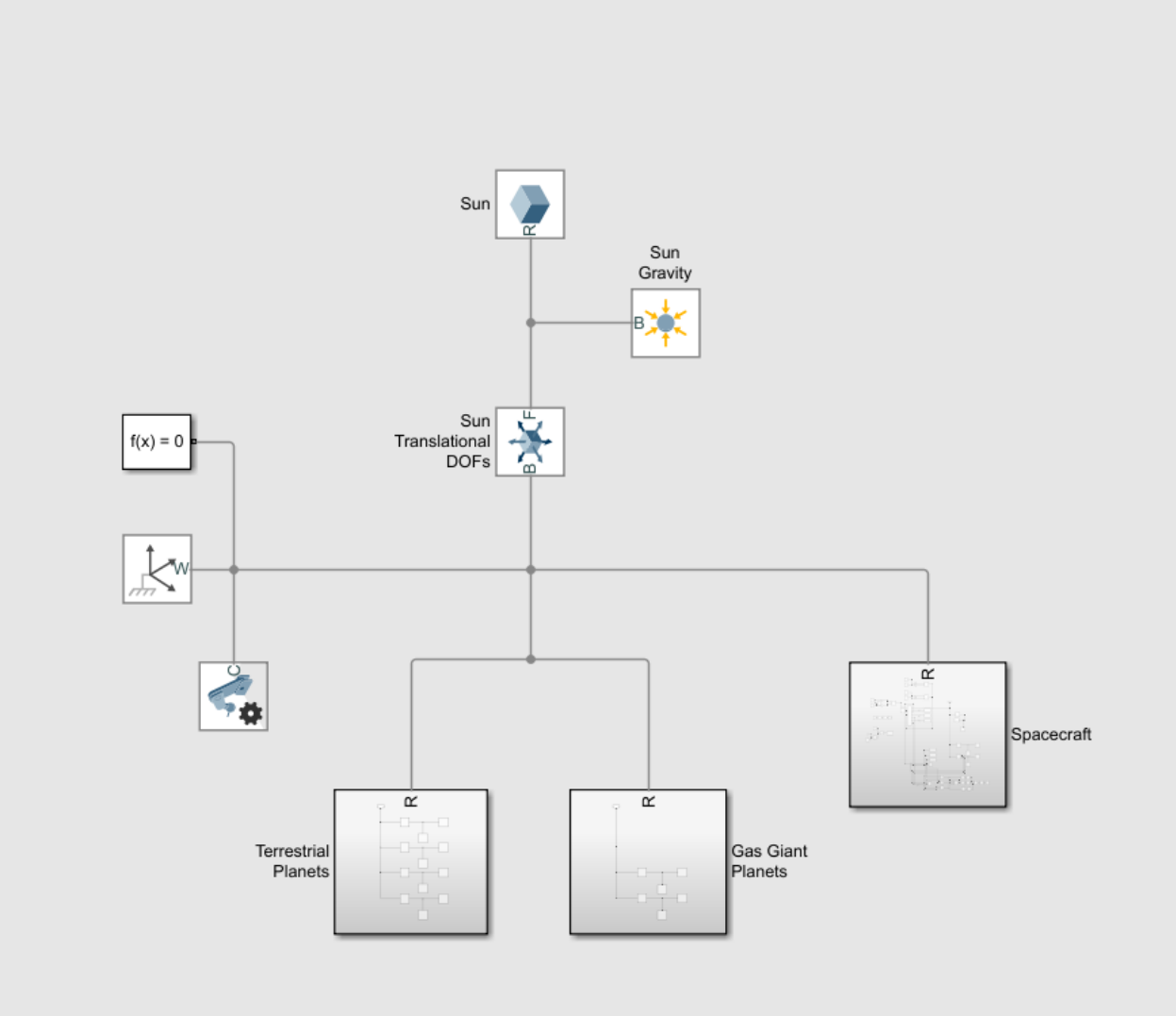
Appendix 2: Simulink code



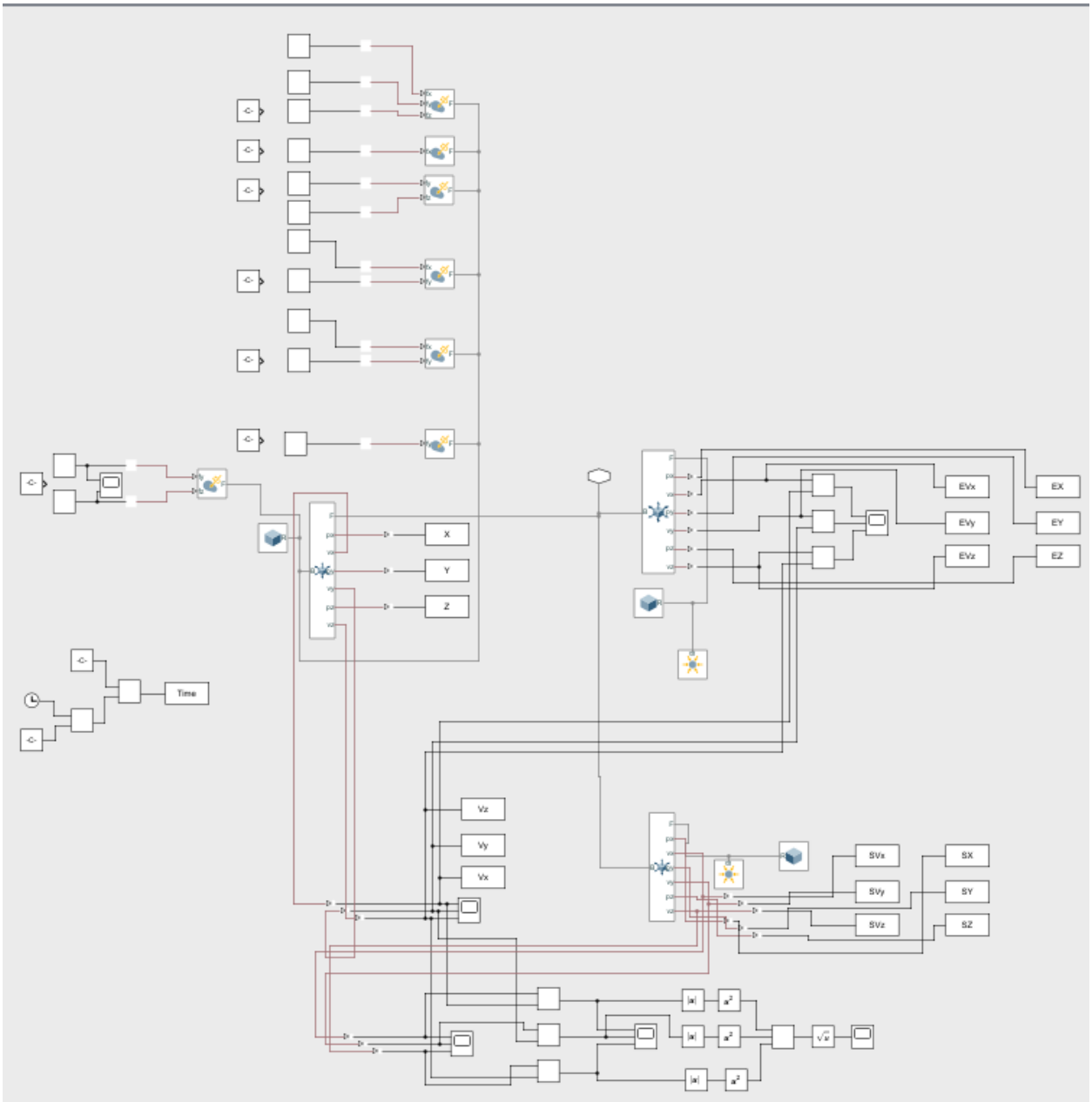
Comms Model



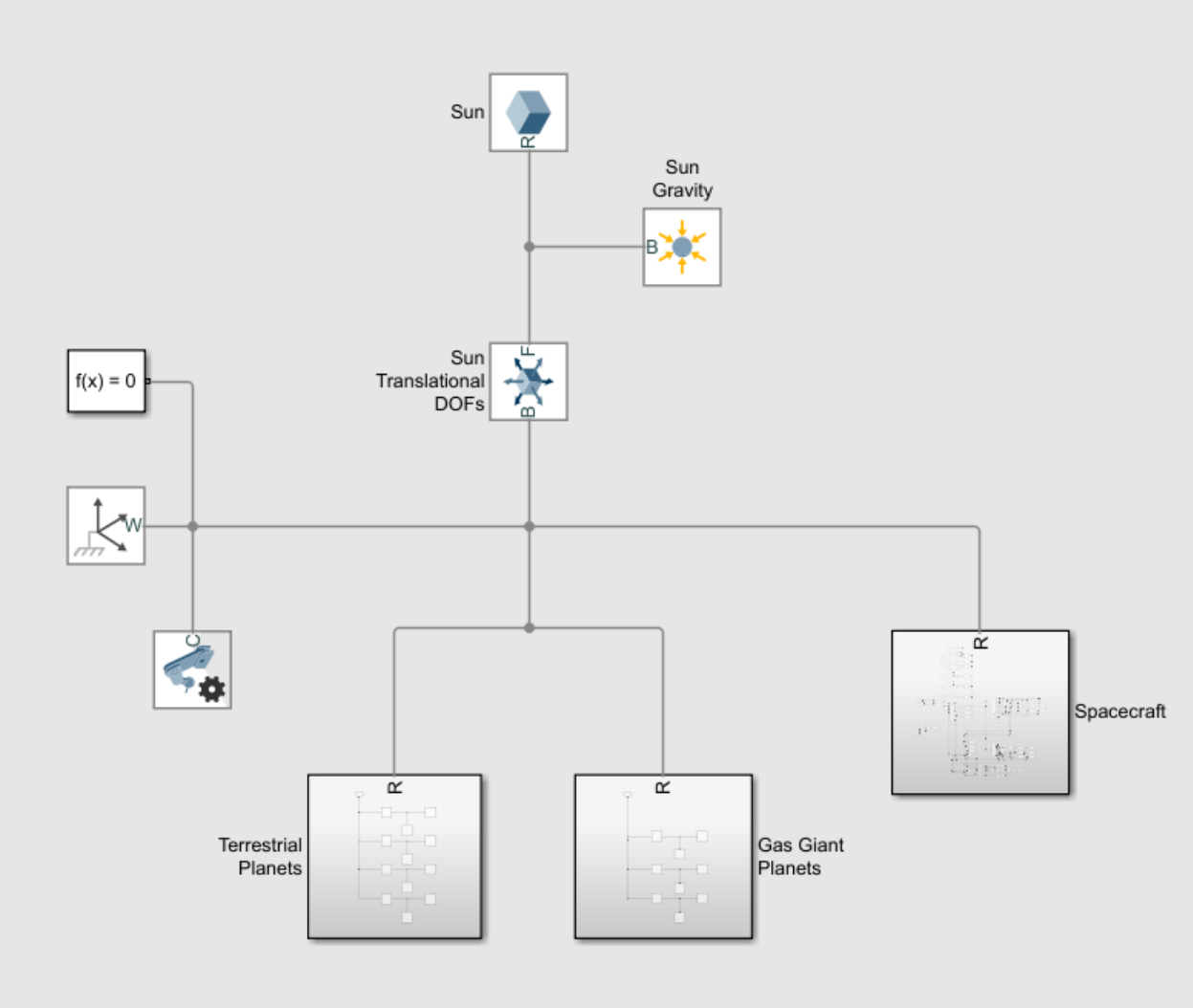
Honors Simulation, Spacecraft block



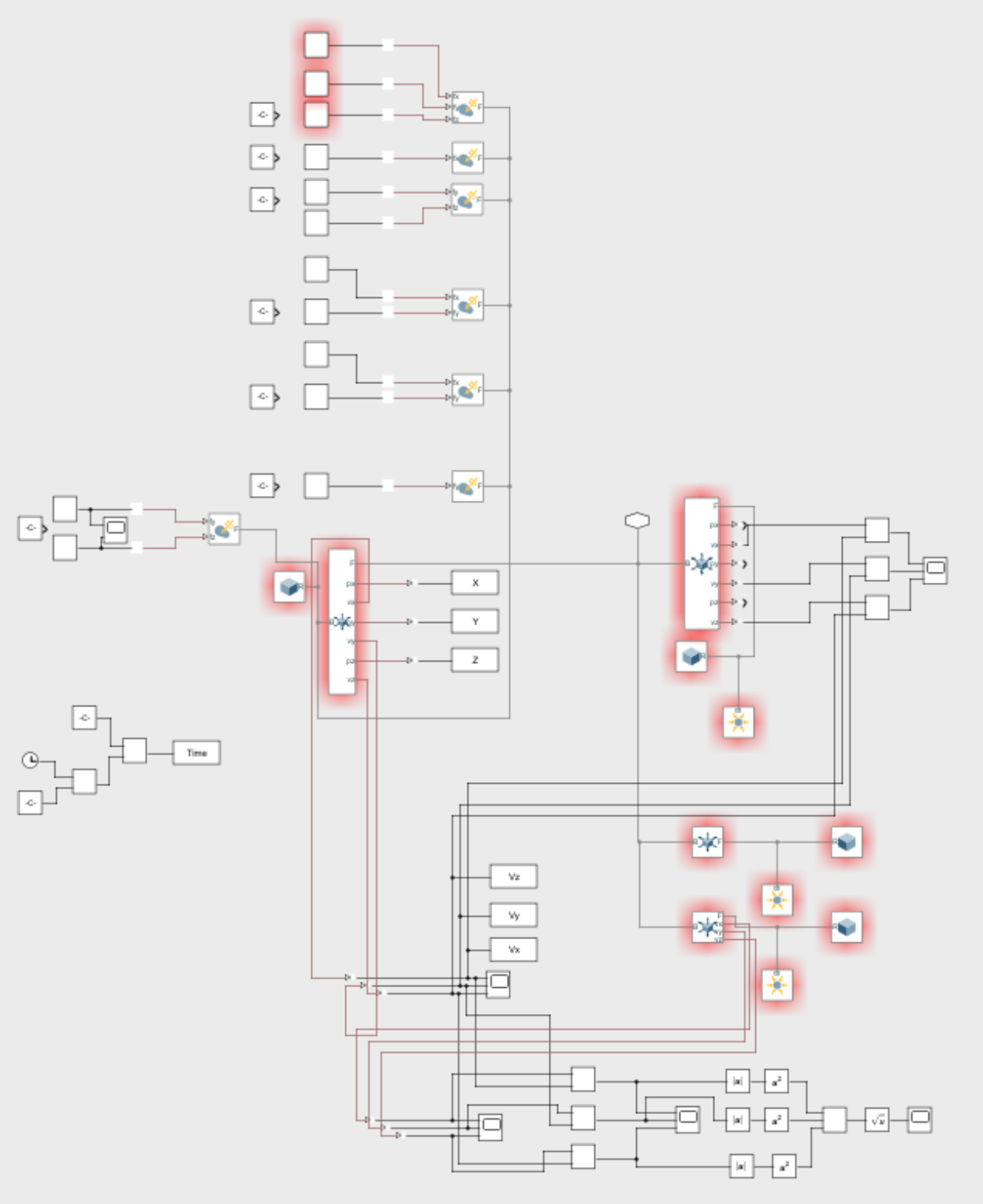
Honors Simulation: overall



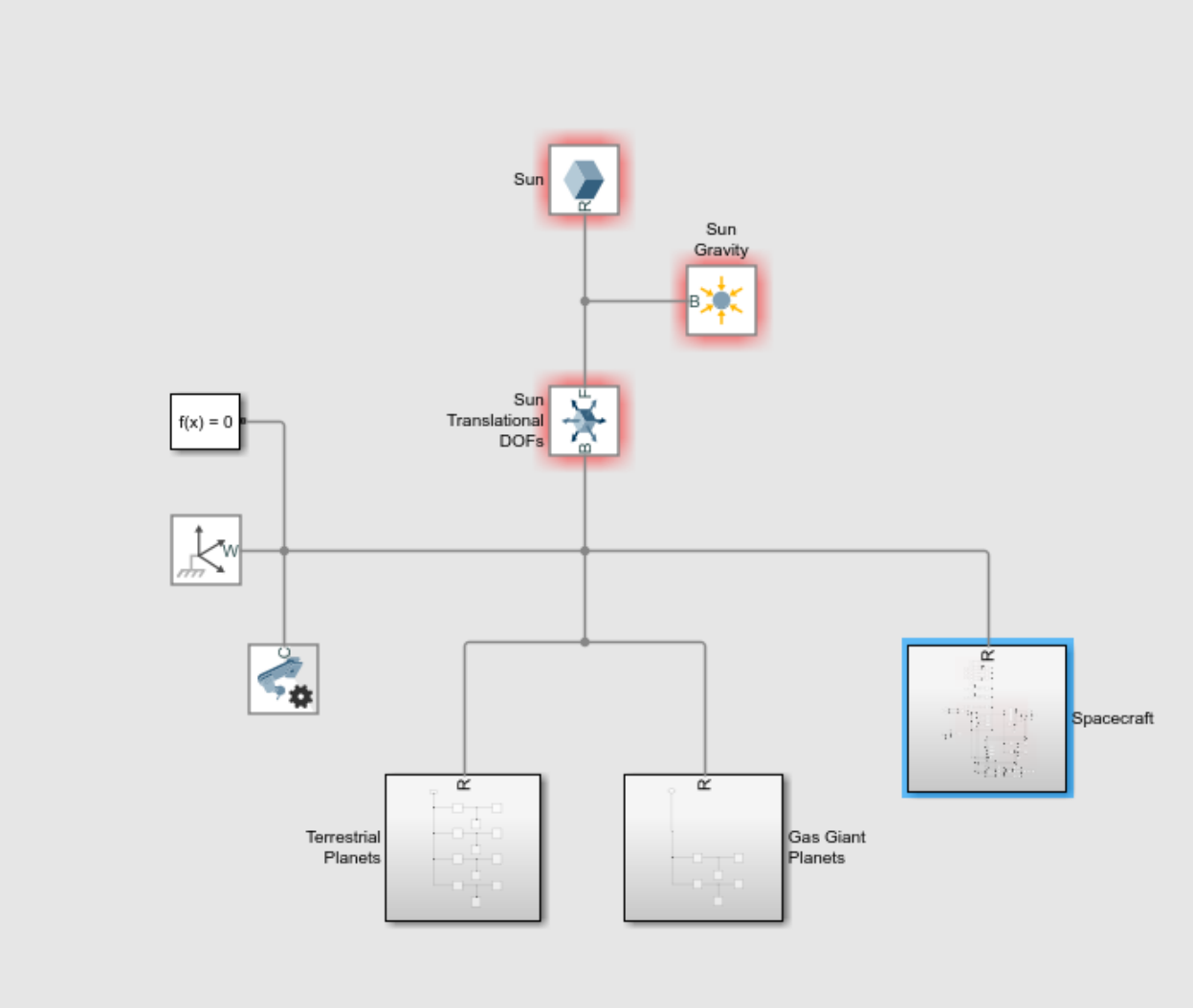
Class simulation, Spacecraft



Class Simulation: overall



Simulation at Saturn: Spacecraft



Simulation at Saturn: overall