Table of Contents

AERO 351 Group Project

```
clc;
clear;
close all;

% Adam Boegel, Alex Kessler, Ben Schmitt, Joe Thompson
% 11/28/2022

% Universal Variables
mu = 398600;
re = 6378;

% Initialize TLEs for Each Object (meaning propagating to sync up times)

APSTAR.Epoch= 22317.49472294;
APSTAR.Eccentricity = 0.0001196;
APSTAR.inclination = 3.6100;
APSTAR.perigee_height = 36137;
APSTAR.apogee_height = 36148;
APSTAR.RAAN = 84.6764;
APSTAR.argument_of_perigee = 297.1834;
APSTAR.revolutions_per_day = 0.99013431;
APSTAR.mean_anomaly_at_epoch =149.4792;

[R_APSTAR,V_APSTAR] = TLEtoRV(APSTAR);

Superbird.Epoch= 22317.39146487;
Superbird.Eccentricity = 0.0016364;
Superbird.inclination = 4.9936;
Superbird.perigee_height = 36119;
Superbird.apogee_height = 36258;
Superbird.RAAN = 78.5232;
Superbird.argument_of_perigee = 114.3141;
Superbird.revolutions_per_day = 0.98852896;
Superbird.mean_anomaly_at_epoch =137.2103;

[R_Superbird,V_Superbird] = TLEtoRV(Superbird);
```

```
Ariane.Epoch= 22317.41885961;
Ariane.Eccentricity = 0.7189723;
Ariane.inclination = 4.2580;
Ariane.perigee_height = 412;
Ariane.apogee_height = 35157;
Ariane.RAAN = 29.6657;
Ariane.argument_of_perigee = 32.6945;
Ariane.revolutions_per_day = 2.31135527;
Ariane.mean_anomaly_at_epoch =356.0749;

[R_Ariane,V_Ariane] = TLEtoRV(Ariane);

Vanguard.Epoch= 22317.43201831;
Vanguard.Eccentricity = 0.1846262;
Vanguard.inclination = 34.2528;
Vanguard.perigee_height = 649;
Vanguard.apogee_height = 3832;
Vanguard.RAAN = 40.2866;
Vanguard.argument_of_perigee = 348.1324;
Vanguard.revolutions_per_day = 10.85033307;
Vanguard.mean_anomaly_at_epoch =8.0865;

[R_Vanguard ,V_Vanguard] = TLEtoRV(Vanguard);

timetosyncto = 22318;
APSTARtimediff = (timetosyncto - APSTAR.Epoch)*24*60*60;
Superbirdtimediff = (timetosyncto - Superbird.Epoch)*24*60*60;
Arianetimediff = (timetosyncto - Ariane.Epoch)*24*60*60;
Vanguardtimediff = (timetosyncto - Vanguard.Epoch)*24*60*60;

[rAPSTAR0,vAPSTAR0,stateAPSTAR0,timeCoastAPSTAR0] =
 propagation(R_APSTAR,V_APSTAR,APSTARtimediff);
[rSuperbird0,vSuperbird0,stateSuperbird0,timeCoastASuperbird0] =
 propagation(R_Superbird,V_Superbird,Superbirdtimediff);
[rAriane0,vAriane0,stateAriane0,timeCoastAriane0] =
 propagation(R_Ariane,V_Ariane,Arianetimediff);
[rVanguard0,vVanguard0,stateVanguard0,timeCoastVanguard0] =
 propagation(R_Vanguard,V_Vanguard,Vanguardtimediff);
```

First Object: Starting There, 5 Orbits Later...

Need to propagate all objects forwards during the amount of time it takes for the first object + s/c to do 5 orbits together

```
propagateTime = (1/APSTAR.revolutions_per_day)*24*60*60*5;
rSC0 = rAPSTAR0;
vSC0 = vAPSTAR0;

[rAPSTAR1,vAPSTAR1,stateAPSTAR1,timeCoastAPSTAR1] =
 propagation(rAPSTAR0,vAPSTAR0,propagateTime);
[rSuperbird1,vSuperbird1,stateSuperbird1,timeCoastASuperbird1] =
 propagation(rSuperbird0,vSuperbird0,propagateTime);
[rAriane1,vAriane1,stateAriane1,timeCoastAriane1] =
 propagation(rAriane0,vAriane0,propagateTime);
```

```
[rVanguard1,vVanguard1,stateVanguard1,timeCoastVanguard1] =
 propagation(rVanguard0,vVanguard0,propagateTime);
[rSC1,vSC1,stateSC1,timeCoastSC1] = propagation(rSC0,vSC0,propagateTime);
```

First Transfer: Inc Change + + Hohmann to Superbird 4

```
% Actually doing Hohmann first:
[~,~,~,v,inc,o,w,~] = COES(rSC1, vSC1);
aSC = (APSTAR.perigee_height + APSTAR.apogee_height + 2*re)/2;
[aSuperbird,~,~,~,~,~,~,~] = COES(rSuperbird1, vSuperbird1);
% [~,Vbetter2] = COEStoRV(hSC, ((aSuperbird - aSC)/(aSuperbird +
 aSC)),v,inc,o,w);
e = ((aSuperbird - aSC)/(aSuperbird + aSC));
[RBetter2,Vbetter2] = COEStoRVperi(e,aSC,v,o,w,inc);
[deltaVHohmann,timeHohmannTransfer] = Hohmann(aSC,aSuperbird);

% propigate during homann

[rSCmiddle3,vSCmiddle3,stateSCmiddle3,timeCoastSCmiddle3] =
 propagation(RBetter2,Vbetter2,timeHohmannTransfer);
[rSuperbirdmiddle3,vSuperbirdmiddle3,stateSB2,~] =
 propagation(rSuperbird1,vSuperbird1,timeHohmannTransfer);

% Find intersection points of orbit planes
% Need to perform inc change here
interPoints = intersection(stateSCmiddle3,stateSB2);

% Find time to propagate to the intersection point
% Find eccentricity vectors to find true anomaly
% Use true anomaly to find time between
[aAPSTAR1,eccAPSTAR1,hAPSTAR1,~,~,~,~,~] = COES(rSCmiddle3, vSCmiddle3);
TA_inter = acos((norm(hAPSTAR1)^2/mu/norm(interPoints(1:3)) - 1)/
eccAPSTAR1); % [rad]
interIndex = find(stateSCmiddle3(:,1:3) == interPoints(1:3));
v_inter = stateSCmiddle3(interIndex(1),4:6);
if dot(interPoints(1:3), v_inter) < 0
    TA_inter = 2*pi - TA_inter;
end
TA_APSTAR1 = acos((norm(hAPSTAR1)^2/mu/norm(rAPSTAR1) - 1)/eccAPSTAR1); %
 [rad]
if dot(rAPSTAR1, vAPSTAR1) < 0
    TA_APSTAR1 = 2*pi - TA_APSTAR1;
end
% Find mean anomalies
E_inter = 2*atan(sqrt((1 - eccAPSTAR1)/(1 + eccAPSTAR1))*tan(TA_inter/2)); %
 [rad]
Me_inter = E_inter - eccAPSTAR1*sin(E_inter); % [rad]
E_APSTAR1 = 2*atan(sqrt((1 - eccAPSTAR1)/(1 +
 eccAPSTAR1))*tan(TA_APSTAR1/2)); % [rad]
Me_APSTAR1 = E_APSTAR1 - eccAPSTAR1*sin(E_APSTAR1); % [rad]
% Find time
```

```
n_APSTAR1 = sqrt(mu/aAPSTAR1^3);
propagateTime = (Me_inter - Me_APSTAR1)/n_APSTAR1;

% Propogate to intersection point
[rSuperbird2,vSuperbird2,stateSuperbird2,timeCoastASuperbird2] =
 propagation(rSuperbirdmiddle3,vSuperbirdmiddle3,propagateTime);
[rSC2,vSC2,stateSC2,timeCoastSC2] =
 propagation(rSCmiddle3,vSCmiddle3,propagateTime);

% Phantom orbit for the plots later
[rSCFake,vSCFake,stateSCFake,~] =
 propagation(rSCmiddle3,vSCmiddle3,propagateTime*20);

% % Great! Now they're in the same orbit, but where are they in relation to
% each other and how to we get there??

[~,~,hSC,~,~,~,w_SC,~] = COES(rSCmiddle3, vSCmiddle3);
[aSuperbird,eccSuperbird,hSuperbird,TA,~,~,w_Superbird,p] =
 COES(rSuperbirdmiddle3, vSuperbirdmiddle3);
T = p;
E = 2 * atan(sqrt((1-eccSuperbird)/(1+eccSuperbird))*tan((TA*pi/180)/2));
Me = E - eccSuperbird*sin(E);
timeFromPerigee = Me/(2*pi)*T; %outer phase required

TASC = 180;

Transfertime= T - timeFromPerigee; %literally on opposite sides of the orbit
 lmao

aTransfer=(6.67408e-20*5.97219e24*Transfertime^2/(4*pi^2))^(1/3);

Ra_Transfer= aTransfer*2 - (Superbird.perigee_height+re);

eccTransfer = (Ra_Transfer - (Superbird.perigee_height+re))/(Ra_Transfer +
 (Superbird.perigee_height+re));

h_Transfer= sqrt((Superbird.perigee_height+re)*(1+eccTransfer*cos(TASC))*mu);

Vp_SC_Transfer = h_Transfer/(Superbird.perigee_height+re);

% Matching Inclination AND RAAN to SB4
inc = Superbird.inclination - APSTAR.inclination;

vInc= 2*norm(vSC2)*sin(inc/2);

DeltaVTransfer = norm(vSCmiddle3) - Vp_SC_Transfer;

DeltaVTransfer = 2*DeltaVTransfer + vInc;

% Propagate forward
[RBetter3,Vbetter3] = COEStoRVperi(eccTransfer,(Superbird.perigee_height
+re),180,Superbird.RAAN,Superbird.argument_of_perigee,Superbird.inclination);
[rSCmiddle4,vSCmiddle4,stateSCmiddle4,timeCoastSCmiddle4] =
 propagation(RBetter3,Vbetter3,Transfertime);
```

```
[rSuperbirdmiddle4,vSuperbirdmiddle4,~,~] =
 propagation(rSuperbirdmiddle3,vSuperbirdmiddle3,Transfertime);

% Recircularize, and propagate every other object with amount of time taken
% REMEMBER to add BACK IN propagationtime FOR EVERY OTHER SC THAN SB AND SC
totalTimeCatchUp = propagateTime + Transfertime + timeHohmannTransfer;
[rArianeEnd2,vArianeEnd2,~,~] =
 propagation(rAriane1,vAriane1,totalTimeCatchUp);
[rVanguardEnd2,vVanguardEnd2,~,~] =
 propagation(rVanguard1,vVanguard1,totalTimeCatchUp);
deltaV1 = deltaVHohmann + DeltaVTransfer;

% Plots
figure(1)
plot3(stateSC1(:,1),stateSC1(:,2),stateSC1(:,3),'Color','#D95319');
hold on;
plot3(stateSCmiddle3(:,1),stateSCmiddle3(:,2),stateSCmiddle3(:,3),'--','Color','#7E2F8E','
hold on
plot3(RBetter2(:,1),RBetter2(:,2),RBetter2(:,3),'r*')
hold on
plot3(0,0,0,'gd')
title('Transfer 1, Part 1')
xlabel('x [km]')
ylabel('y [km]')
zlabel('z [km]')
grid on
hold off
legend('APSTAR Orbit','Hohmann Transfer', 'Burn Point', 'Earth');

figure(2)
plot3(stateSCFake(:,1),stateSCFake(:,2),stateSCFake(:,3), "b:");
hold on
% plot3(stateSCmiddle3(:,1),stateSCmiddle3(:,2),stateSCmiddle3(:,3));
% hold on;
plot3(stateSCmiddle4(:,1),stateSCmiddle4(:,2),stateSCmiddle4(:,3), '--','Color','#7E2F8E',
hold on;
plot3(stateSuperbird1(:,1),stateSuperbird1(:,2),stateSuperbird1(:,3),'m');
hold on
plot3(0,0,0,'gd')
hold off;
title('Transfer 1, Part 2')
xlabel('x [km]')
ylabel('y [km]')
zlabel('z [km]')
grid on
legend('Hohmann Orbit','Phasing Transfer', 'Superbird Orbit', 'Earth');
```

Second Object: 5 Orbits Later...

```
% Propagating objects forward
propagateTime = (1/Superbird.revolutions_per_day)*24*60*60*5;
```

```
[rSuperbird3,vSuperbird3,stateSuperbird3,timeCoastASuperbird3] =
 propagation(rSuperbirdmiddle4,vSuperbirdmiddle4,propagateTime);
[rAriane3,vAriane3,stateAriane3,timeCoastAriane3] =
 propagation(rArianeEnd2,vArianeEnd2,propagateTime);
[rVanguard3,vVanguard3,stateVanguard3,timeCoastVanguard3] =
 propagation(rVanguardEnd2,vVanguardEnd2,propagateTime);
[rSC3,vSC3,stateSC3,timeCoastSC3] =
 propagation(rSuperbirdmiddle4,vSuperbirdmiddle4,propagateTime);
```

Second Transfer: Lamberts -> Ariane H10

Find 3d radius of Apogee for Lamberts

```
ArianeMags = sqrt(stateAriane3(:,1).^2 + stateAriane3(:,2).^2 +
 stateAriane3(:,3).^2);
ApogeeIndex = find(ArianeMags == max(ArianeMags));
ApogeeRadius_Ariane = [stateAriane3(ApogeeIndex,1);
 stateAriane3(ApogeeIndex,2); stateAriane3(ApogeeIndex,3)];

% Find time since perigee of apogee
% This equals one half the period
PerigeeIndex = find(ArianeMags == min(ArianeMags));
PerigeeRadius_Ariane = [stateAriane3(PerigeeIndex,1);
 stateAriane3(PerigeeIndex,2); stateAriane3(PerigeeIndex,3)];
ra_Ariane = norm(ApogeeRadius_Ariane);
rp_Ariane = norm(PerigeeRadius_Ariane);
a_Ariane = (ra_Ariane + rp_Ariane)/2;
T_Ariane = 2*pi*a_Ariane^(1.5)/sqrt(mu);
t_since_rp_ArianeApo = T_Ariane/2;

% Find time since perigee of Ariane in current state
[~,eccAriane3,hAriane3,~,~,~,~,~] = COES(rAriane3, vAriane3);
TA_Ariane3 = acos((norm(hAriane3)^2/mu/norm(rAriane3) - 1)/eccAriane3); %
 [rad]
if dot(rAriane3, vAriane3) < 0
    TA_Ariane3 = 2*pi - TA_Ariane3;
end
E_Ariane3 = 2*atan(sqrt((1 - eccAriane3)/(1 +
 eccAriane3))*tan(TA_Ariane3/2)); % [rad]
while E_Ariane3 < 0 || E_Ariane3 > 2*pi
    if  E_Ariane3 < 0
        E_Ariane3 = 2*pi + E_Ariane3;
    else
        E_Ariane3 = -2*pi + E_Ariane3;
    end
end
Me_Ariane3 = E_Ariane3 - eccAriane3*sin(E_Ariane3); % [rad]
t_since_rp_Ariane3 = Me_Ariane3*T_Ariane/2/pi;

% Time for Ariane to reach apogee
% Add an extra period to make Lambert's less expensive
dt_Transfer2 = t_since_rp_ArianeApo - t_since_rp_Ariane3 + T_Ariane;
tm = 1;
```

```
% Do Lamberts
[vSC_Start1, vSC_End1] = lambert(rSC3', ApogeeRadius_Ariane, dt_Transfer2, tm,
 mu);

% Propogate
vSCmiddle5 = vSC_Start1';
[rSC4,vSC4,stateSC4,timeCoastSC4] = propagation(rSC3,vSCmiddle5,dt_Transfer2);
[rArianeMid,vArianeMid,stateArianeMid,timeCoastArianeMid] =
 propagation(rAriane3,vAriane3,dt_Transfer2);

DeltaV_3_Total = abs(norm(vSCmiddle5)-norm(vSC_Start1) + norm(vArianeMid) -
 norm(vSC_End1));

figure
plot3(stateSC3(:,1),stateSC3(:,2),stateSC3(:,3),'m')
hold on
plot3(rSC3(:,1),rSC3(:,2),rSC3(:,3),'r*')
hold on
plot3(stateAriane3(:,1),stateAriane3(:,2),stateAriane3(:,3),'k')
hold on
%plot3(rAriane3(:,1),rAriane3(:,2),rAriane3(:,3),'*')
hold on
plot3(rArianeMid(:,1),rArianeMid(:,2),rArianeMid(:,3),'bs')
hold on
 plot3(0,0,0,'gd')
hold on
% plot3(rSC4(:,1),rSC4(:,2),rSC4(:,3),'^')
hold on
 plot3(stateSC4(:,1),stateSC4(:,2),stateSC4(:,3),'--','Color','#7E2F8E','LineWidth',2)
hold on
%
 plot3(ApogeeRadius_Ariane(1),ApogeeRadius_Ariane(2),ApogeeRadius_Ariane(3),'*')
title('Transfer 2')
xlabel('x [km]')
ylabel('y [km]')
zlabel('z [km]')
grid on
legend('Superbird Orbit', 'Burn point', 'Ariane Orbit ', 'Rendezvous
 Point', 'Earth',  'Transfer Orbit');
hold off


% Propogate Vanguard through Lamberts transfer time
[rVanguard4,vVanguard4,stateVanguard4,timeCoastVanguard4] =
 propagation(rVanguard3,vVanguard3,dt_Transfer2);
```

Third Object: 5 Orbits Later...

```
% Propagating objects forward
[rSC5,vSC5,stateSC5,~] = propagation(rSC4,vArianeMid,T_Ariane*5);
[rVanguard5,vVanguard5,stateVanguard5,timeCoastVanguard5] =
 propagation(rVanguard4,vVanguard4,T_Ariane*5);
```

Third Transfer: Vanguard 1

```
% Propogate for further 40% of Ariane's orbit before performing Lambert's
 procedure
[rSCmiddle6,vSCmiddle6,stateSCmiddle6,timeCoastSCmiddle6] =
 propagation(rSC5,vSC5,0.45*T_Ariane);
[rVanguardMid1,vVanguardMid1,stateVanguardMid1,timeCoastVanguardMid1] =
 propagation(rVanguard5,vVanguard5,0.45*T_Ariane);

% Find 3D vector for perigee of Vanguard (and apogee and semi-major for later
 use)
% This will be our target r2 for Lamberts
VanguardMags = sqrt(stateVanguardMid1(:,1).^2 + stateVanguardMid1(:,2).^2 +
 stateVanguardMid1(:,3).^2);
ApogeeIndex = find(VanguardMags == max(VanguardMags));
ApogeeRadius_Vanguard = [stateVanguardMid1(ApogeeIndex,1);
 stateVanguardMid1(ApogeeIndex,2); stateVanguardMid1(ApogeeIndex,3)];
PerigeeIndex = find(VanguardMags == min(VanguardMags));
PerigeeRadius_Vanguard = [stateVanguardMid1(PerigeeIndex,1);
 stateVanguardMid1(PerigeeIndex,2); stateVanguardMid1(PerigeeIndex,3)];
ra_Vanguard = norm(ApogeeRadius_Vanguard);
rp_Vanguard = norm(PerigeeRadius_Vanguard);
a_Vanguard = (ra_Vanguard + rp_Vanguard)/2;
T_Vanguard = 2*pi*a_Vanguard^(1.5)/sqrt(mu);

% Find time since perigee for Vanguard
% We will use the difference of this with period for the transfer time for
 Lamberts
[~,eccVanguardMid1,hVanguardMid1,~,~,~,~,~] = COES(rVanguardMid1,
 vVanguardMid1);
TA_VanguardMid1 = acos((norm(hVanguardMid1)^2/mu/norm(rVanguardMid1) - 1)/
eccVanguardMid1); % [rad]

if dot(rVanguardMid1, vVanguardMid1) < 0
    TA_VanguardMid1 = 2*pi - TA_VanguardMid1;
end
E_VanguardMid1 = 2*atan(sqrt((1 - eccVanguardMid1)/(1 +
 eccVanguardMid1))*tan(TA_VanguardMid1/2)); % [rad]
while E_VanguardMid1 < 0 || E_VanguardMid1 > 2*pi
    if  E_VanguardMid1 < 0
        E_VanguardMid1 = 2*pi + E_VanguardMid1;
    else
        E_VanguardMid1 = -2*pi + E_VanguardMid1;
    end
end
Me_VanguardMid1 = E_VanguardMid1 - eccVanguardMid1*sin(E_VanguardMid1); %
 [rad]
t_since_rp_VanguardMid1 = Me_VanguardMid1*T_Vanguard/2/pi;

% Find transfer time
dt_Transfer3 = T_Vanguard - t_since_rp_VanguardMid1;

% Do Lamberts
```

```
[vSC_Start2, vSC_End2] = lambert(rSCmiddle6', PerigeeRadius_Vanguard,
 dt_Transfer3, tm, mu);

% Propogate
vSCmiddle5 = vSC_Start1';
[rSC6,vSC6,stateSC6,timeCoastSC6] =
 propagation(rSCmiddle6',vSC_Start2,dt_Transfer3);
[rVanguard6,vVanguard6,stateVanguard6,timeCoastVanguard6] =
 propagation(rVanguardMid1,vVanguardMid1,dt_Transfer3);

DeltaV_4_Total = norm(vSCmiddle6)-norm(vSC_Start2) + norm(vVanguard6) -
 norm(vSC_End2);

figure
plot3(stateSC5(:,1),stateSC5(:,2),stateSC5(:,3),'k')
hold on
%plot3(rSC5(:,1),rSC5(:,2),rSC5(:,3),'*')
hold on
plot3(stateVanguard5(:,1),stateVanguard5(:,2),stateVanguard5(:,3))
hold on
%plot3(rVanguard5(:,1),rVanguard5(:,2),rVanguard5(:,3),'*')
hold on
plot3(0,0,0,'gd')
hold on
plot3(rSCmiddle6(:,1),rSCmiddle6(:,2),rSCmiddle6(:,3),'r*')
hold on
plot3(rSC6(:,1),rSC6(:,2),rSC6(:,3),'bs')
hold on
plot3(stateSC6(:,1),stateSC6(:,2),stateSC6(:,3),'--','Color','#7E2F8E','LineWidth',2)
hold on
% plot3(rVanguardMid1(:,1),rVanguardMid1(:,2),rVanguardMid1(:,3),'x')
hold on
% plot3(rVanguard6(:,1),rVanguard6(:,2),rVanguard6(:,3),'d')
title('Transfer 3')
xlabel('x [km]')
ylabel('y [km]')
zlabel('z [km]')
grid on
legend('Ariane Orbit', 'Vanguard Orbit ', 'Earth', 'Burn point',  'Rendezvous
 Point',   'Transfer Orbit');
hold off



% Perform Lamberts
```

Fourth Object: 5 Orbits Later...

```
[rSC5,vSC5,stateSC5,timeCoastSC5] = propagation(rSC6,vSC6,T_Vanguard*5);
```

Total DeltaV

```
deltaV = DeltaV_4_Total + DeltaV_3_Total + deltaV1;
```

```
totalTime = ((1/Superbird.revolutions_per_day)*24*60*60*5 + (1/
APSTAR.revolutions_per_day)*24*60*60*5 + T_Vanguard*5 + T_Ariane*5.45 +
 dt_Transfer2 + dt_Transfer3 + totalTimeCatchUp)/(60*60*24);
```

Published with MATLAB® R2021b

```
function [dstate] = coast(time, state, mu)

x = state(1);
y = state(2);
z = state(3);
dx = state(4);
dy = state(5);
dz = state(6);
r = norm([x y z]);
ddx = -mu*x/r^3;
ddy = -mu*y/r^3;
ddz = -mu*z/r^3;
dstate = [dx; dy; dz; ddx; ddy; ddz];

end
```

Published with MATLAB® R2021b

```
function [a,e,h,v,i,o,w,p] = COES(R, V)

% Defining some constants
mu = 398600;
khat = [0,0,1];
ihat = [1,0,0];

% Find our R and V magnitudes
Rmag = norm(R);
Vmag = norm(V);

% Solve for specific power and semi major axis a
SP = ((Vmag^2)/2 - mu/Rmag);
a = - (mu)/(2*SP);

% Calculate h and n
h = cross(R, V);
n = cross(khat, h);

% Solve for Eccentricity vector and magnitude
evec = (1/mu)*((((Vmag^2) - (mu/Rmag)).*R) - ((dot (R, V)).*V));
e = norm(evec);
% Plug and chug for angles i, o, w, and v
i = (acos((dot(khat, h))/(norm(h))))*(180/pi);

o = (acos((dot(ihat, n))/(norm(n))))*(180/pi);
if n(2) < 0
    o = 360 - o;
end

w = (acos((dot(n, evec))/(norm(n)*e)))*(180/pi);
if evec(3) < 0
    w = 360 - w;
end

v = (acos((dot(R, evec))/(Rmag*e)))*(180/pi);
if (dot(R, V)) < 0
    v = 360 - v;
end

%Calculating Period
p = 2*pi*sqrt((a^3)/mu);

% % Displaying all the things!
% disp([ 'Our semi-major axis is ', num2str(a), ' km.']);
% disp([ 'Our angular momentum is ', num2str(norm(h)), ' km^2/s.']);
% disp([ 'Our eccentricity is ', num2str(e), '.']);
% disp([ 'Our true anomaly is ', num2str(v), ' degrees.']);
% disp([ 'Our inclination is ', num2str(i), ' degrees.']);
% disp([ 'Our RAAN is ', num2str(o), ' degrees.']);
% disp([ 'Our argument of perigee is ', num2str(w), ' degrees.']);
% disp([ 'Our period is ', num2str(p), ' seconds.']);
```

```
end
```

Published with MATLAB® R2021b

```
function [R,V] = COEStoRVperi(e,rP,TA,RAAN,w,inc)
mu = 398600;
h = sqrt(mu*rP*(1+e));
rperi = (h^2/mu) * (1+e*cos(TA))^-1*[cos(TA); sin(TA); 0];
vperi = (mu/h) * [-sin(TA); e + cos(TA); 0];
% Direction cosine e

Cz_RAAN =        [  cosd(RAAN)     sind(RAAN)     0;
                   -sind(RAAN)     cosd(RAAN)     0;
                    0             0             1]; % 3

Cx_inc =        [   1             0             0;
                    0             cosd(inc)      sind(inc);
                    0             -sind(inc)     cosd(inc)]; % 1

Cz_w =        [     cosd(w)        sind(w)        0;
                    -sind(w)       cosd(w)        0;
                    0             0             1]; % 3

Q = Cz_w * Cx_inc * Cz_RAAN;

% Transformation Matrix
% Geocentric equatorial position vector R
R = (Q' * rperi)';
% Geocentric equatorial velocity vector V
V = (Q' * vperi)';
end
```

Published with MATLAB® R2021b

```
function [deltaV,time] = Hohmann(rpInt,raFinal)

% This is all assuming circular and whatnot, also smaller to larger (keep your
 heads
% up, kings and queens)
% Setup
mue = 398600;
vInt = sqrt(mue/rpInt);

% Transfer Calcs
eccTrans = (raFinal - rpInt) / (raFinal + rpInt);
hTrans = sqrt(rpInt*mue*(1+eccTrans));
vTransInt = hTrans / rpInt;

% First burn
deltaVInt = vTransInt - vInt;

% Final burn
vFin = sqrt(mue/raFinal);
vTransFin = hTrans / raFinal;
deltaVFin = vFin - vTransFin;

% Total Delta V
deltaV = deltaVInt + deltaVFin;


% Find transfer orbit time
a = (raFinal+rpInt)/2;
T = 2*pi*sqrt((a^3)/mue);

time = T/2; % seconds

end
```

Published with MATLAB® R2021b

```matlab
function intersectionPoints = intersection(state1,state2)
% Find intersection points of two orbit planes to perform inc changes

% Define plane 1
A1 = [state1(1,1), state1(1,2), state1(1,3)];
B1 = [state1(2,1), state1(2,2), state1(2,3)];
C1 = [state1(3,1), state1(3,2), state1(3,3)];
vec1a = B1 - A1;
vec1b = C1 - A1;
N1 = cross(vec1a, vec1b);

% Define plane 2
A2 = [state2(1,1), state2(1,2), state2(1,3)];
B2 = [state2(2,1), state2(2,2), state2(2,3)];
C2 = [state2(3,1), state2(3,2), state2(3,3)];
vec2a = B2 - A2;
vec2b = C2 - A2;
N2 = cross(vec2a, vec2b);

% Find intersection points
% Compare unit vectors of each point on first orbit to unit vector of
% intersection line, find minimum differences for both sides
N3 = cross(N1, N2);
unitN3 = repmat(N3./norm(N3),length(state1(:,1)),1);
unitState1 = state1(:,1:3)./sqrt(state1(:,1).^2 + state1(:,2).^2 +
 state1(:,3).^2);
positive_side = unitState1 - unitN3;
negative_side = unitState1 + unitN3;
positive_side = sqrt(positive_side(:,1).^2 + positive_side(:,2).^2 +
 positive_side(:,3).^2);
negative_side = sqrt(negative_side(:,1).^2 + negative_side(:,2).^2 +
 negative_side(:,3).^2);
[~, index1] = min(positive_side);
[~, index2] = min(negative_side);
intersectionPoints = [state1(index1,1:3) state1(index2,1:3)];

end
```

Published with MATLAB® R2021b

```
function [v1, v2] = lambert(r1, r2, dt, tm, mu)

% Setup
r1mag = sqrt(dot(r1,r1));
r2mag = sqrt(dot(r2,r2));
rx = cross(r1,r2);
z = rx(3);

if tm > 0

    if z < 0

        deltaTheta = 360 - acosd(dot(r1,r2)/r1mag/r2mag);

    else

        deltaTheta = acosd(dot(r1,r2)/r1mag/r2mag);

    end

else

    if z < 0

        deltaTheta = acosd(dot(r1,r2)/r1mag/r2mag);

    else

        deltaTheta = 360 - acosd(dot(r1,r2)/r1mag/r2mag);

    end

end

% Calculating A
A = sind(deltaTheta)*sqrt(r1mag*r2mag/(1 - cosd(deltaTheta)));

% Calculating z (Bisection Method!)

if A == 0

    error('deltaTheta = 180 degrees, use Hohmann instead');

end

z = 0;
S = 1/6;
C = 1/2;

y = r1mag + r2mag + A*((z*S - 1)/sqrt(C));
chi = sqrt(y/C);
lb = -4*pi^2; % Guess
```

```
ub = 4*pi^2; % Guess
tol = 1e-8;
dtloop = chi^3*S/sqrt(mu) + A*sqrt(y)/sqrt(mu);

while abs(dtloop - dt) > tol

    z = (ub + lb)/2;

    if z < 0

        S = (sinh(sqrt(-z)) - sqrt(-z))/(sqrt(-z))^3;
        C = (cosh(sqrt(-z)) - 1)/(-z);

    elseif z > 0

        S = (sqrt(z) - sin(sqrt(z)))/(sqrt(z))^3;
        C = (1 - cos(sqrt(z)))/z;

    else

        S = 1/6;
        C = 1/2;

    end

    y = r1mag + r2mag + A*((z*S - 1)/sqrt(C));
    chi = sqrt(y/C);
    dtloop = chi^3*S/sqrt(mu) + A*sqrt(y)/sqrt(mu);

    if dtloop > dt

        ub = z;

    else

        lb = z;

    end

end

% Calculating f/g and their dots
f = 1 - y/r1mag;
g = A*sqrt(y/mu);
fdot = sqrt(mu*y/C)*(z*S - 1)/r1mag/r2mag;
gdot = 1 - y/r2mag;

% Calculating v1 and v2
v1 = (r2 - f*r1)/g;
v2 = fdot*r1 + gdot*v1;

end
```

Published with MATLAB® R2021b

```matlab
function [rF, vF,state1, timeCoast] = propagation(r0,v0,timediff)
mue = 398600;
timespan = [0 timediff];
options = odeset('RelTol', 1e-8,'AbsTol',1e-8);
state0 = [r0 v0];
[timeCoast, state1] = ode45(@coast, timespan, state0, options, mue);
rF = state1(length(state1),1:3);
vF = state1(length(state1),4:6);
end
```

Published with MATLAB® R2021b

```
function [R,V] = TLEtoRV(Craft)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
Epoch = Craft.Epoch;
Eccentricity = Craft.Eccentricity;
Inclination = Craft.inclination ;
perigee_height = Craft.perigee_height;
apogee_height = Craft.apogee_height;
RAAN = Craft.RAAN;
argument_of_perigee = Craft.argument_of_perigee;
revolutions_per_day = Craft.revolutions_per_day;
mean_anomaly = Craft.mean_anomaly_at_epoch;

% semimajor_Axis = 1/2*(perigee_height+apogee_height);

G = 6.67430*10^-11;
mu = 398600;

rp = perigee_height + 6378;
Angular_Momentum = sqrt(rp*mu*(1+Eccentricity));

Eguess = mean_anomaly - Eccentricity/2 ;
i = 0;
Edif = 1;

while (Edif > 10^-8)

    Enew = Eguess - (mean_anomaly - Eguess + Eccentricity*sin(Eguess))/
(Eccentricity*cos(Eguess) - 1);
    Edif = abs(Enew - Eguess);
    Eguess = Enew;

    i = i+1;
end

E = Enew;
TA = 2 * atan(sqrt((1+Eccentricity)/(1-Eccentricity))*tan(E/2));

if TA < 0

    TA = (2*pi + TA)*180/pi;

elseif TA > (2*pi)

    TA = (2*pi - TA)*180/pi;

else

    TA = TA*180/pi;

end
```

```
rx = Angular_Momentum^2/mu*(1/(1 +
 Eccentricity*cosd(TA)))*[cosd(TA);sind(TA);0];
vx = mu/Angular_Momentum*[-sind(TA); (Eccentricity +cosd(TA));0];
% Direction cosine matrix

QXx = [cosd(argument_of_perigee), sind(argument_of_perigee),0;-
sind(argument_of_perigee),cosd(argument_of_perigee),0;0,0,1]*...
    [1,0,0;0,cosd(Inclination),sind(Inclination);0,-
sind(Inclination),cosd(Inclination)]*...
    [cosd(RAAN), sind(RAAN),0;-sind(RAAN),cosd(RAAN),0;0,0,1];

% Transformation Matrix
QxX = inv(QXx);
% Geocentric equatorial position vector R
R = QxX*rx;
% Geocentric equatorial velocity vector V
V = QxX*vx;
end
```

Published with MATLAB® R2021b