ASEN 5050 HW 6 Main Script

Table of Contents

Housekeeping 1
Problem 1 1
Problem 2 6
Sanity checks 10

By: Ian Faber

Housekeeping

```
clc; clear; close all;

addpath("..\utilities\")
```

Problem 1

Setup

```
muSun = 1.32712428e11; % km^3/s^2
AU2km = 149597870.7; % 1 AU = 149597870.7 km
R1 = [-2.686982e7; 1.326980e8; 5.752566e7]; % km, Earth position at t1
V1 = [-29.781722; -5.101774; -2.210394]; % km/s, Earth velocity at t1
t1 = 2457754.5; % TDB, Julian date
R2 = [-5.642901e7; -8.571048e7; -3.499466e7]; % km, Venus at position t2
V2 = [29.658341; -16.091100; -9.116674]; % km/s, Venus velocity at t2
t2 = 2457871.5; % TDB, Julian date
    % Lecture example problem parameters below
% muEarth = 398600.435507;
% muSun = muEarth;
% R1 = [-654; 13605; 1997];
% V1 = [-5.53; 0.849; 0.683];
% R2 = [7284; -19341; -3264];
% V2 = [3.07; 2.63; 0.444];
% t1 = 0;
% t2 = 5/24;

% Part b
fprintf("--- Problem 1 part b ---\n")
dTA_deg = acosd(dot(R1,R2)/(norm(R1)*norm(R2)));
dTA_1_deg = abs(dTA_deg);
dTA_2_deg = 360 - dTA_1_deg;

    % Only one transfer angle will be less than 180 degrees
if dTA_1_deg < 180
    dTA_deg = dTA_1_deg;
else
    dTA_deg = dTA_2_deg;
```

```
end

if dTA_deg < 180
    lt180 = 1;
else
    lt180 = 0;
end

r1 = norm(R1);
r2 = norm(R2);

c = sqrt(r1^2 + r2^2 - 2*r1*r2*cosd(dTA_deg));
s = 0.5*(r1 + r2 + c);

TOF = (t2 - t1)*60*60*24; % days -> sec

TOF_p = (1/3)*sqrt(2/muSun)*(s^(3/2) - (s-c)^(3/2));

if TOF_p < TOF
    fprintf("Elliptical transfer needed!\n")
    ellipse = true;
else
    fprintf("Hyperbolic transfer needed!\n")
    ellipse = false;
end

if ellipse
    amin = s/2; % Answer for part c
    nmin = sqrt(muSun/(amin)^3);
    alphamin = pi;
    betamin0 = 2*asin(sqrt((s-c)/s));
    if dTA_deg < 180; betamin = betamin0; else; betamin = -betamin0; end
    TOFmin = (1/nmin)*((alphamin - betamin) - (sin(alphamin) -
sin(betamin)));
else
    fprintf("You goofed...\n")
    return
end

if TOF < TOFmin
    shortTOF = 1;
else
    shortTOF = 0;
end

a = solveLambertsEq(muSun, s, c, TOF, shortTOF, lt180, ellipse)

alpha0 = 2*asin(sqrt(s/(2*a)));
beta0 = 2*asin(sqrt((s-c)/(2*a)));
if shortTOF
    alpha = alpha0;
else
    alpha = 2*pi - alpha0;
end
```

```
if lt180
    beta = beta0;
else
    beta = -beta0;
end

n = sqrt(muSun/(a^3));

TOF_check = (1/n)*((alpha - beta) - (sin(alpha) - sin(beta)));

p = ((4*a*(s-r1)*(s-r2))/(c^2))*(sin(0.5*(alpha+beta)))^2;

e = sqrt(1-(p/a))

% Part c
fprintf("--- Problem 1 part c ---\n")
amin

% Part d
fprintf("--- Problem 1 part d ---\n")

TA1_deg_init = acosd((1/e)*(p/r1 - 1));
TA2_deg_init = acosd((1/e)*(p/r2 - 1));

    % Check for correct TA combination
sit = 1;
minDiff = 1000; % Dummy variable for storing the last minimum difference
while true
    switch sit
        case 1
            TA1_deg = TA1_deg_init;
            TA2_deg = TA2_deg_init;
        case 2
            TA1_deg = -TA1_deg_init;
            TA2_deg = TA2_deg_init;
        case 3
            TA1_deg = TA1_deg_init;
            TA2_deg = -TA2_deg_init;
        case 4
            TA1_deg = -TA1_deg_init;
            TA2_deg = -TA2_deg_init;
        case 5
            TA1_deg = 360 - TA1_deg_init;
            TA2_deg = TA2_deg_init;
        case 6
            TA1_deg = -(360 - TA1_deg_init);
            TA2_deg = TA2_deg_init;
        case 7
            TA1_deg = 360 - TA1_deg_init;
            TA2_deg = -TA2_deg_init;
        case 8
            TA1_deg = -(360 - TA1_deg_init);
            TA2_deg = -TA2_deg_init;
```

```
        case 9
            TA1_deg = TA1_deg_init;
            TA2_deg = 360 - TA2_deg_init;
        case 10
            TA1_deg = -TA1_deg_init;
            TA2_deg = 360 - TA2_deg_init;
        case 11
            TA1_deg = TA1_deg_init;
            TA2_deg = -(360 - TA2_deg_init);
        case 12
            TA1_deg = -TA1_deg_init;
            TA2_deg = -(360 - TA2_deg_init);
        case 13
            TA1_deg = 360 - TA1_deg_init;
            TA2_deg = 360 - TA2_deg_init;
        case 14
            TA1_deg = -(360 - TA1_deg_init);
            TA2_deg = 360 - TA2_deg_init;
        case 15
            TA1_deg = 360 - TA1_deg_init;
            TA2_deg = -(360 - TA2_deg_init);
        case 16
            TA1_deg = -(360 - TA1_deg_init);
            TA2_deg = -(360 - TA2_deg_init);
    end
    dTA_check = TA2_deg - TA1_deg;

    if abs(dTA_check - dTA_deg) < minDiff
        minDiff = abs(dTA_check - dTA_deg);
        tempTA1 = TA1_deg;
        tempTA2 = TA2_deg;
    end

    if sit < 16 % Only test these 16 combinations
        sit = sit + 1; % Iterate to the next check
    else
        break % Tested all combinations!
    end
end
TA1_deg = tempTA1
TA2_deg = tempTA2

f = 1 - (r2/p)*(1-cosd(dTA_deg));
g = ((r2*r1)/sqrt(muSun*p))*sind(dTA_deg);
fDot = sqrt(muSun/p)*tand(dTA_deg/2)*((1-cosd(dTA_deg))/p - 1/r2 - 1/r1);
gDot = 1 - (r1/p)*(1-cosd(dTA_deg));

Vt1 = (1/g)*(R2 - f*R1)
Vt2 = fDot*R1 + gDot*Vt1

dV1 = Vt1 - V1;
dV1_mag = norm(dV1)
dV2 = V2 - Vt2;
dV2_mag = norm(dV2)
```

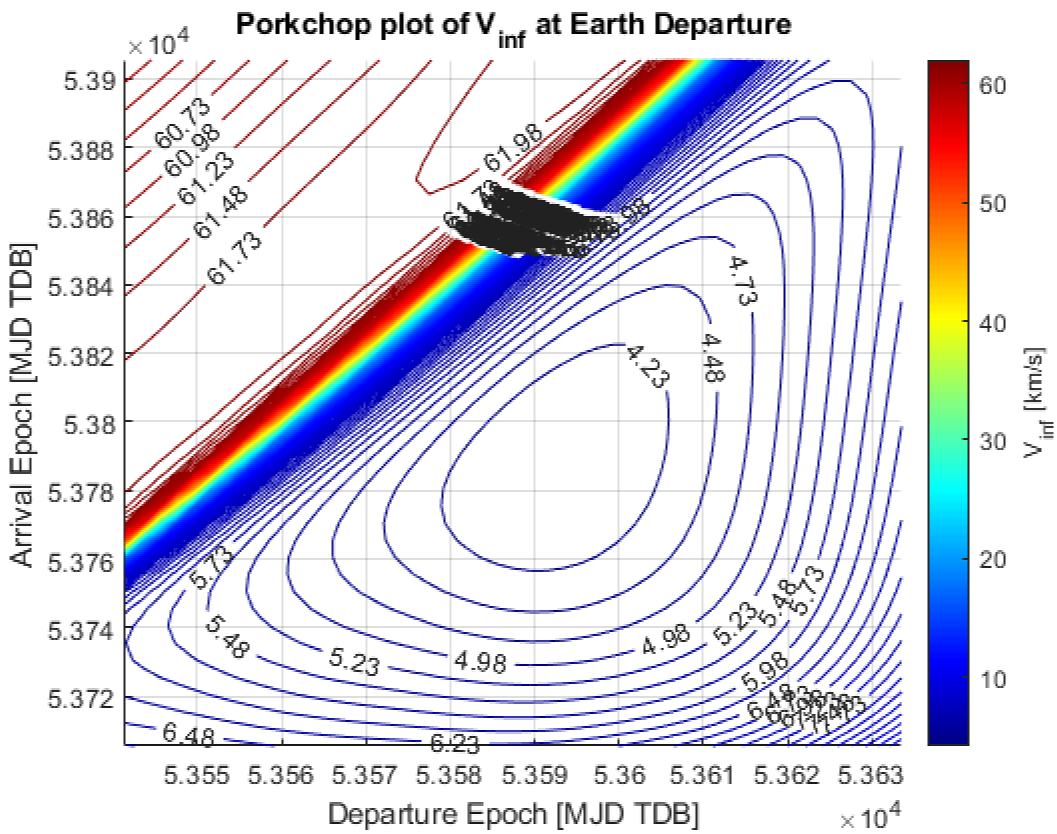
```
--- Problem 1 part b ---
Elliptical transfer needed!

a =

   1.2581e+08


e =

    0.1693

--- Problem 1 part c ---

amin =

   1.2364e+08

--- Problem 1 part d ---

TA1_deg =

 -179.4087


TA2_deg =

  -41.3130


Vt1 =

  -26.8951
   -4.9363
   -1.3228


Vt2 =

   33.6175
  -14.5349
   -7.1996


dV1_mag =

    3.0246


dV2_mag =

    4.6661
```

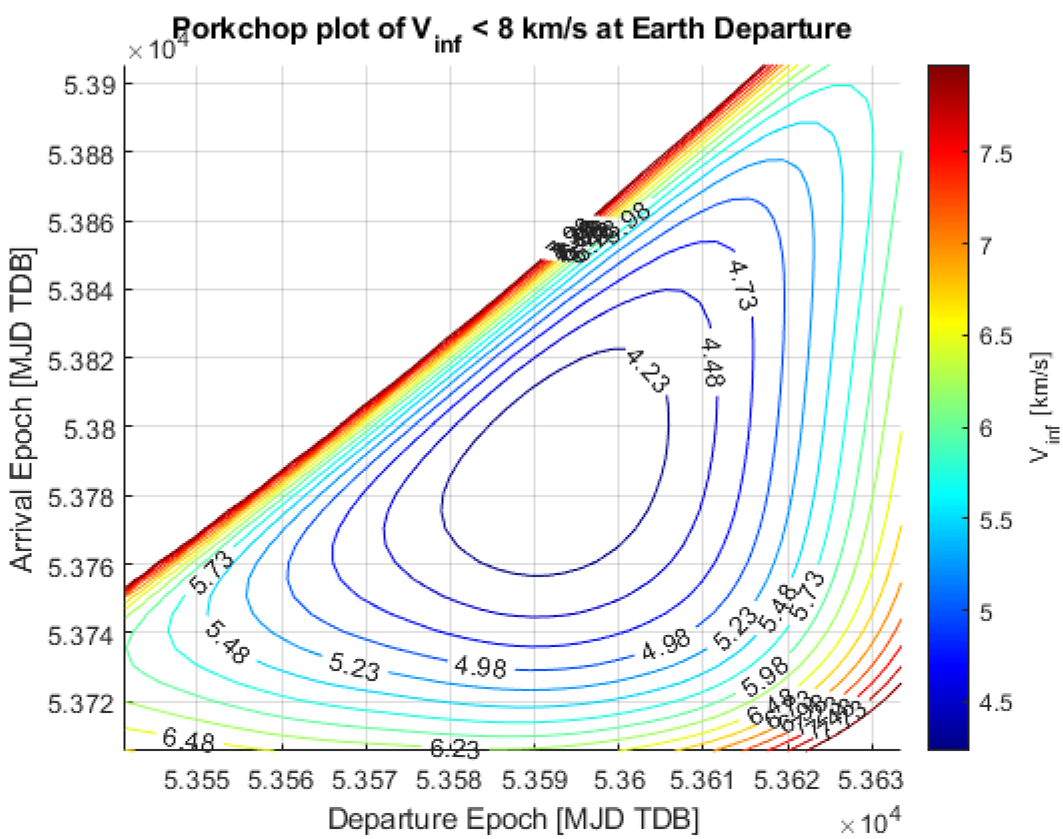
Problem 2

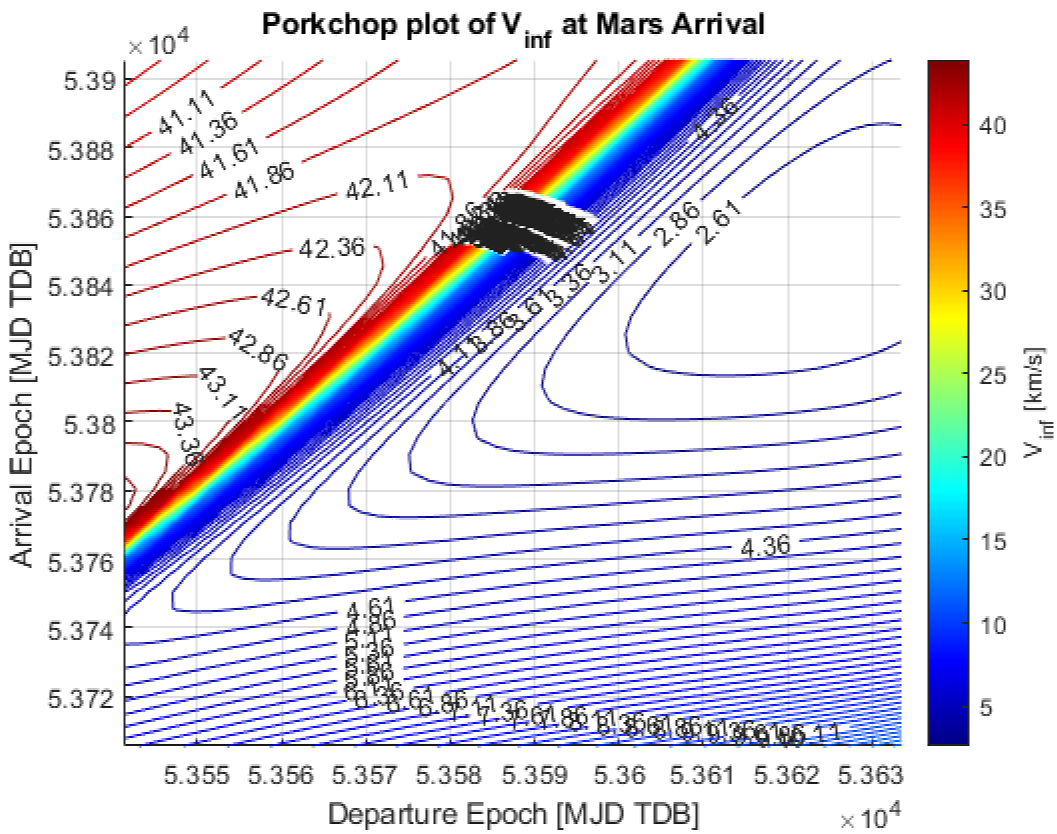
Check answer for problem 1

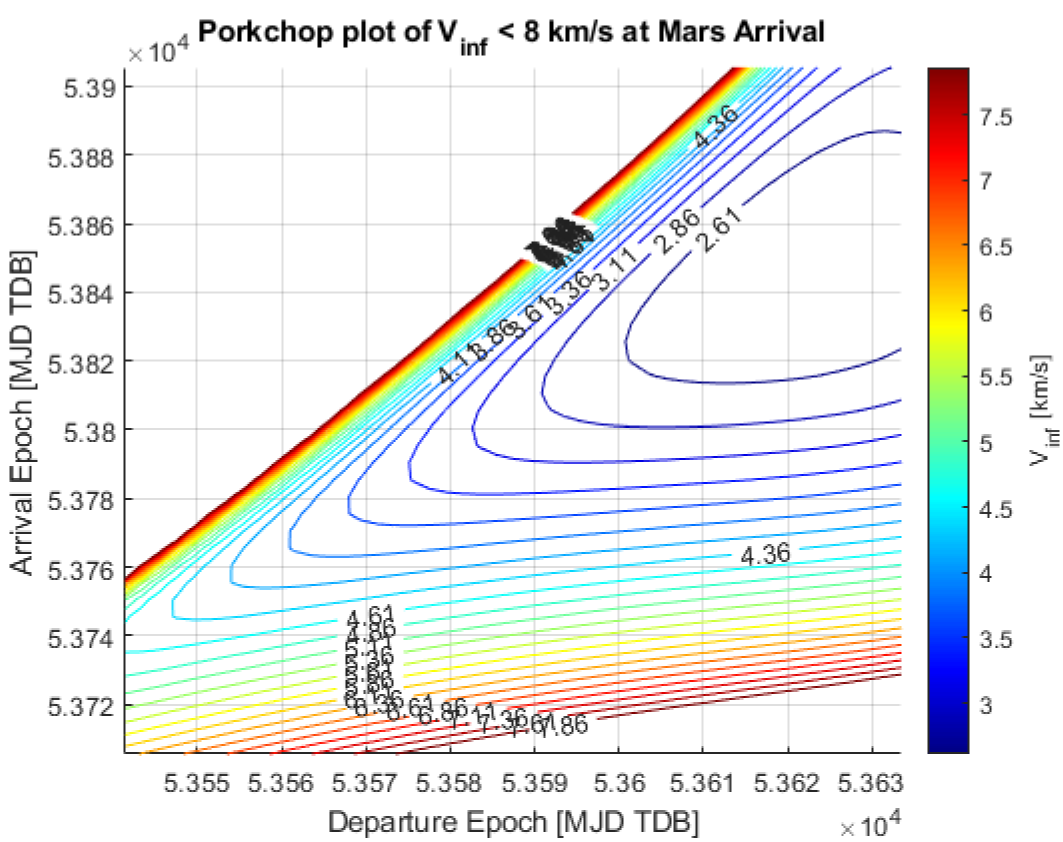
```
initialState_problem1 = [R1; V1];
finalState_problem1 = [R2; V2];
transfer_problem1 = solveLambertsProblem(initialState_problem1,
finalState_problem1, TOF, 1, muSun); % Check answer for problem 1

% Extract given ephemeris data
earthData = readEphemData("HW6_Ephem_Earth.txt");
marsData = readEphemData("HW6_Ephem_Mars.txt");

% Loop through valid epoch combinations and solve Lambert's Problem for
% each
lt180 = 1; % Interested in transfer angles < 180 deg
designSpace = [];
for k = 1:size(earthData, 1) % Loop through each departure time
    for kk = 1:size(marsData, 1) % Loop through each arrival time
        if marsData.JDTDB(kk) > earthData.JDTDB(k) % Transfer is only valid
if mars epoch is after earth epoch
            R1 = [earthData.X(k); earthData.Y(k); earthData.Z(k)];
            V1 = [earthData.VX(k); earthData.VY(k); earthData.VZ(k)];
            t1 = earthData.JDTDB(k)*24*60*60; % days -> sec
            R2 = [marsData.X(kk); marsData.Y(kk); marsData.Z(kk)];
            V2 = [marsData.VX(kk); marsData.VY(kk); marsData.VZ(kk)];
            t2 = marsData.JDTDB(kk)*24*60*60; % days -> sec

            initialState = [R1; V1];
            finalState = [R2; V2];
            TOF = t2 - t1;
            transfer_lt180 = solveLambertsProblem(initialState, finalState,
TOF, lt180, muSun); % Calculate transfers with dTA < 180 deg
            transfer_lt180.startEpoch_MJD = t1/(24*3600) - 2.4e6;
            transfer_lt180.finalEpoch_MJD = t2/(24*3600) - 2.4e6;
            designSpace = [designSpace; transfer_lt180];
        end
    end
end

% Earth Vinfinity porkchop plot - full range of Vinfinity
maxVPlot = 8; % km/s
[X,Y] = meshgrid(unique([designSpace.startEpoch_MJD]),
unique([designSpace.finalEpoch_MJD]));
Z = reshape([designSpace.Vinfinity_1], size(X));

minV = round(min(min(Z)), 2);
maxV = round(max(max(Z)), 2);
levels = minV:0.25:maxV; % Specify contour levels for contour function

colors = 'jet';

figure; hold on; grid on;
```

```
title("Porkchop plot of V_{inf} at Earth Departure")
contour(X, Y, Z, levels, 'ShowText', 'on', 'LabelSpacing', 500);
colormap(colors)
c = colorbar; c.Label.String = "V_{inf} [km/s]";
xlabel("Departure Epoch [MJD TDB]"); ylabel("Arrival Epoch [MJD TDB]");

% Earth Vinfinity porkchop plot - truncated to Vinfinity < 10 km/s
minV = round(min(min(Z)), 2);
maxV = maxVPlot;
levels = minV:0.25:maxV; % Specify contour levels for contour function

colors = 'jet';

figure; hold on; grid on;
titleText = sprintf("Porkchop plot of V_{inf} < %.0f km/s at Earth
Departure", maxVPlot);
title(titleText)
contour(X, Y, Z, levels, 'ShowText', 'on', 'LabelSpacing', 500);
colormap(colors)
c = colorbar; c.Label.String = "V_{inf} [km/s]";
xlabel("Departure Epoch [MJD TDB]"); ylabel("Arrival Epoch [MJD TDB]");

% Mars Vinfinity porkchop plot - full range of Vinfinity
Z = reshape([designSpace.Vinfinity_2], size(X));

minV = round(min(min(Z)), 2);
maxV = round(max(max(Z)), 2);
levels = minV:0.25:maxV; % Specify contour levels for contour function

figure; hold on; grid on;
title("Porkchop plot of V_{inf} at Mars Arrival")
contour(X, Y, Z, levels, 'ShowText', 'on', 'LabelSpacing', 500);
colormap(colors)
c = colorbar; c.Label.String = "V_{inf} [km/s]";
xlabel("Departure Epoch [MJD TDB]"); ylabel("Arrival Epoch [MJD TDB]");

% Mars Vinfinity porkchop plot - full range of Vinfinity
minV = round(min(min(Z)), 2);
maxV = maxVPlot;
levels = minV:0.25:maxV; % Specify contour levels for contour function

figure; hold on; grid on;
titleText = sprintf("Porkchop plot of V_{inf} < %.0f km/s at Mars Arrival",
maxVPlot);
title(titleText)
contour(X, Y, Z, levels, 'ShowText', 'on', 'LabelSpacing', 500);
colormap(colors)
c = colorbar; c.Label.String = "V_{inf} [km/s]";
xlabel("Departure Epoch [MJD TDB]"); ylabel("Arrival Epoch [MJD TDB]");
```


Porkchop plot of V_inf at Earth Departure
Arrival Epoch [MJD TDB]
Departure Epoch [MJD TDB]
V_inf [km/s]
×10^4
5.39
5.388
5.386
5.384
5.382
5.38
5.378
5.376
5.374
5.372
5.355
5.356
5.357
5.358
5.359
5.36
5.361
5.362
5.363
60
50
40
30
20
10
60.73
60.98
61.23
61.48
61.73
61.98
4.73
4.48
4.23
4.98
5.23
5.48
5.73
5.98
6.23
6.48


Porkchop plot of V_inf < 8 km/s at Earth Departure
Arrival Epoch [MJD TDB]
Departure Epoch [MJD TDB]
V_inf [km/s]
×10^4
5.39
5.388
5.386
5.384
5.382
5.38
5.378
5.376
5.374
5.372
5.355
5.356
5.357
5.358
5.359
5.36
5.361
5.362
5.363
7.5
7
6.5
6
5.5
5
4.5
4.73
4.48
4.23
4.98
5.23
5.48
5.73
5.98
6.23
6.48


Porkchop plot of V_{inf} at Mars Arrival
Arrival Epoch [MJD TDB]
Departure Epoch [MJD TDB]
V_{inf} [km/s]
$\times 10^4$
5.39
5.388
5.386
5.384
5.382
5.38
5.378
5.376
5.374
5.372
5.355
5.356
5.357
5.358
5.359
5.36
5.361
5.362
5.363
40
35
30
25
20
15
10
5
41.11
41.36
41.61
41.86
42.11
42.36
42.61
42.86
43.11
43.36
4.36
2.86
2.61
3.11
4.61

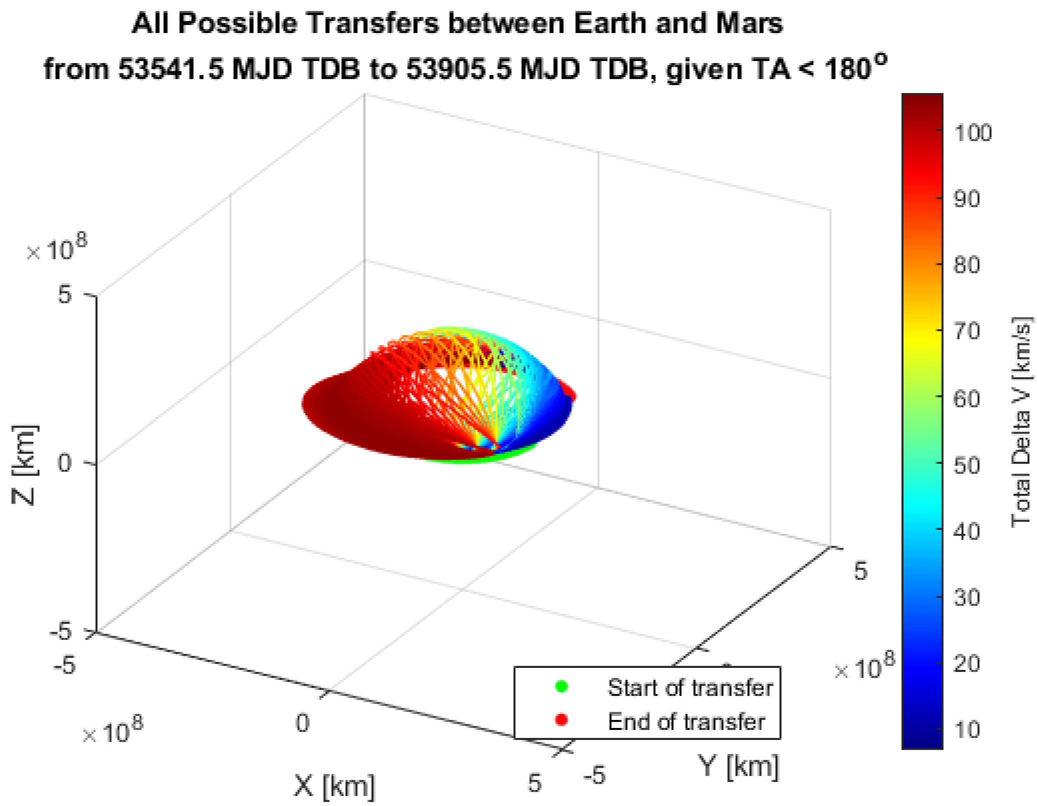

Porkchop plot of V_{inf} < 8 km/s at Mars Arrival
Arrival Epoch [MJD TDB]
Departure Epoch [MJD TDB]
V_{inf} [km/s]
$\times 10^4$
5.39
5.388
5.386
5.384
5.382
5.38
5.378
5.376
5.374
5.372
5.355
5.356
5.357
5.358
5.359
5.36
5.361
5.362
5.363
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
4.36
2.86
2.61
3.11
4.61

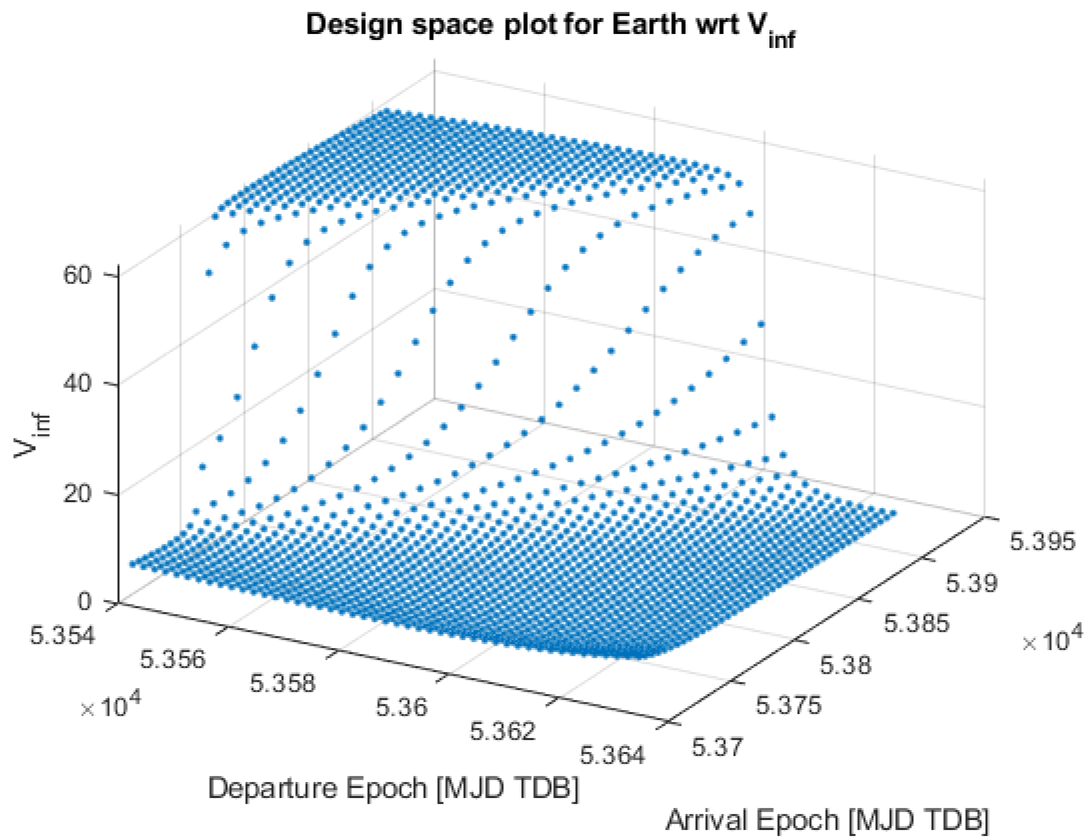
Sanity checks

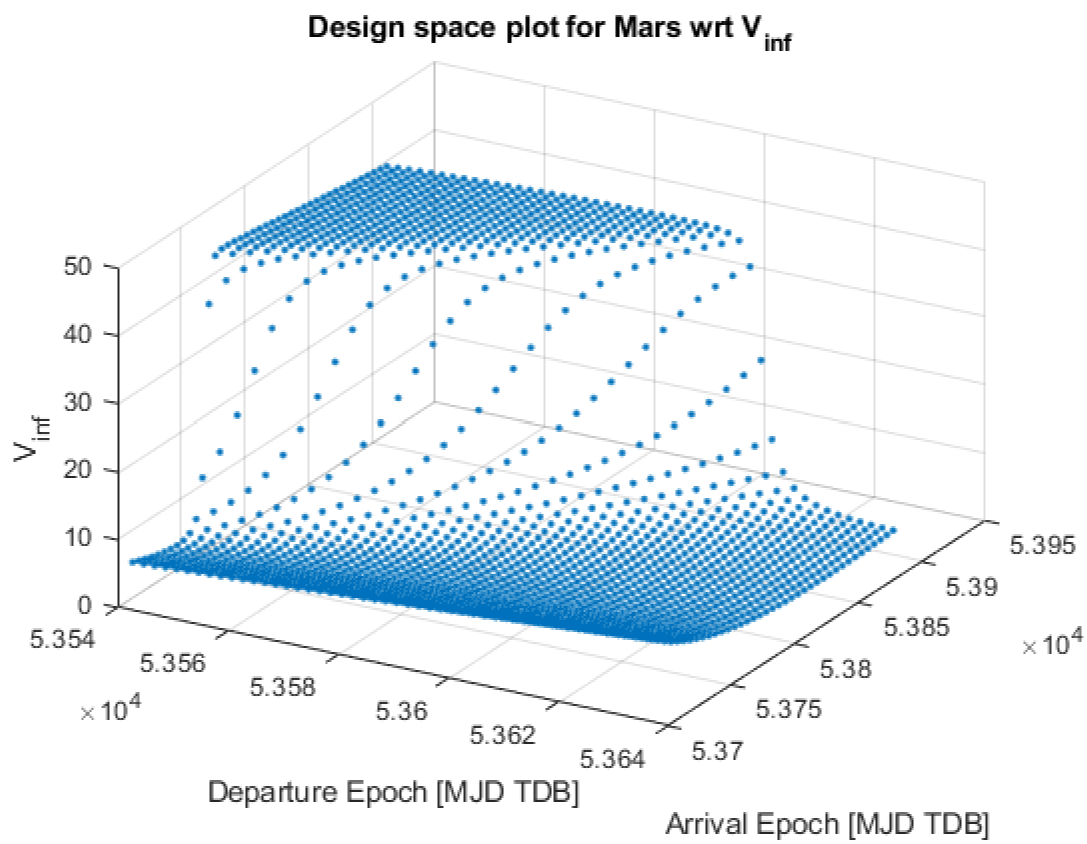
DCM function handle

```
NO = @(theta, inc, RAAN)EA2DCM([-theta, -inc, -RAAN], [3, 1, 3]); % orbital
-> inertial

    % Plot all transfers
figure; hold on; grid on;
if lt180; sym = "<"; else; sym = ">"; end
titleText = sprintf("All Possible Transfers between Earth and Mars \n from
%.1f MJD TDB to %.1f MJD TDB, given TA %s 180^o", earthData.JDTDB(1) -
2.4e6, marsData.JDTDB(end) - 2.4e6, sym);
title(titleText)
for k = 1:length(designSpace)
    a = designSpace(k).a;
    e = designSpace(k).e;
    inc = designSpace(k).orbElems.inc;
    RAAN = designSpace(k).orbElems.RAAN;
    argPeri = designSpace(k).orbElems.argPeri;
    TA =
deg2rad(real(designSpace(k).TA1_deg):0.5:real(designSpace(k).TA2_deg));
    theta = TA + argPeri;
    r = (a*(1-e^2))./(1+e*cos(TA));
    % rhat-thetahat-hhat frame
    R = [r;zeros(size(r));zeros(size(r))];
    % rhat-thetahat-hhat -> Cartesian frame
    for kk = 1:length(theta)
        R(:,kk) = real(NO(theta(kk), inc, RAAN)*R(:,kk));
    end
    % plot3(R(1,:), R(2,:), R(3,:))
    if designSpace(k).dV_mag_total% < 80 % Limit plotting based on delta V's
        color_line3d(designSpace(k).dV_mag_total*ones(size(R(1,:))), R(1,:),
R(2,:), R(3,:)); % Create colored line based on total delta V required
        startTrans = plot3(R(1,1), R(2,1), R(3,1), 'g.', 'markerSize', 15);
% Start of transfer
        stopTrans = plot3(R(1,end), R(2,end), R(3,end), 'r.', 'markerSize',
15); % End of transfer
    end
end
c = colorbar; c.Label.String = "Total Delta V [km/s]"; colormap(colors)
xlabel("X [km]"); ylabel("Y [km]"); zlabel("Z [km]");
xlim([-5e8, 5e8]), ylim([-5e8, 5e8]); zlim([-5e8, 5e8]); view([30 35])
legend([startTrans, stopTrans], ["Start of transfer", "End of transfer"],
"location", 'best')

    % Plot design space wrt Vinfinity
figure; hold on; grid on;
title("Design space plot for Earth wrt V_{inf}")
plot3([designSpace.startEpoch_MJD], [designSpace.finalEpoch_MJD],
[designSpace.Vinfinity_1], '.')
view([30 35])
xlabel("Departure Epoch [MJD TDB]"); ylabel("Arrival Epoch [MJD TDB]");
```

```
zlabel("V_{inf}")

figure; hold on; grid on;
title("Design space plot for Mars wrt V_{inf}")
plot3([designSpace.startEpoch_MJD], [designSpace.finalEpoch_MJD],
[designSpace.Vinfinity_2], '.')
view([30 35])
xlabel("Departure Epoch [MJD TDB]"); ylabel("Arrival Epoch [MJD TDB]");
zlabel("V_{inf}")
```




Design space plot for Earth wrt V_{inf}
V_{inf}
60
40
20
0
5.354
5.356
5.358
5.36
5.362
5.364
$\times 10^4$
Departure Epoch [MJD TDB]
5.37
5.375
5.38
5.385
5.39
5.395
$\times 10^4$
Arrival Epoch [MJD TDB]


Design space plot for Mars wrt V_{inf}
V_{inf}
50
40
30
20
10
0
5.354
5.356
5.358
5.36
5.362
5.364
$\times 10^4$
Departure Epoch [MJD TDB]
5.37
5.375
5.38
5.385
5.39
5.395
$\times 10^4$
Arrival Epoch [MJD TDB]

Published with MATLAB® R2023b