A graph of a function

Description automatically generated

Figure Determining minimum Delta V based on Alpha angle

A diagram of an orbit

Description automatically generated

Figure Plot of the Orbital Transfer using the Alpha for minimum Delta V

% This is the code for HW2 - Problem 2

% By Nicholas Luis (PSU ID 930841391)

clear; clc; close all;

%% Part A (used matlab as a calculator)

% Constants

MU = 398600; % km^3 / s^2

PI = 3.141592654;

% Departure Orbit

r1 = 7000; % km

e1 = 0;

a1 = r1;

v1 = getVelo(r1, a1);

I1 = 60;

% Arrival Orbit

r2 = 70000; % km

e2 = 0;

a2 = r2;

v2 = getVelo(r2, a2);

I2 = 20;

% Transfer Orbit

delta\_f = 180;

a\_T = get\_aMin(r1, r2, delta\_f);

p\_T = getPT(a\_T, r1, r2, delta\_f);

e\_T = (p\_T / r1) - 1;

v1T = getVelo(r1, a\_T);

v2T = getVelo(r2, a\_T);

IT = 40;

% Plane change angles

delta\_I1 = -20; % AKA alpha

delta\_I2 = -20; % AKA beta

gamma1 = 0;

gamma1T = 0;

gamma2T = 0;

gamma2 = 0;

% Delta V

deltaV1 = getDeltaV(v1, v1T, delta\_I1, gamma1, gamma1T);

deltaV2 = getDeltaV(v2, v2T, delta\_I2, gamma2, gamma2T);

%% Part B

% Question 1

alphaVec = -50:3:50; % Creates a list of alpha values to iterate through (in degrees)

deltaV1List = zeros(size(alphaVec)); % vector to store the delta V1's as we iterate

deltaV2List = zeros(size(alphaVec)); % vector to store the delta V2's as we iterate

for i=1 : length(alphaVec) % Iterates through each index

deltaV1List(i) = getDeltaV(v1, v1T, alphaVec(i), gamma1, gamma1T);

deltaV2List(i) = getDeltaV(v2, v2T, delta\_I2, gamma2, gamma2T);

end

deltaVTotalList = deltaV1List + deltaV2List; % vector that stores the total delta V's

% Question 2

figure(1)

plot(alphaVec, deltaVTotalList, LineWidth=2)

title('Plot of Alpha vs Delta V')

xlabel("Alpha (degrees)")

ylabel("Delta V (km/s)")

exportgraphics(gca,"HW2\_Problem2\_Figure1.jpg");

[minVal, minIndex] = min(deltaVTotalList);

fprintf("The alpha for minimum delta V is: \t\t%d degrees\n", alphaVec(minIndex))

fprintf("This results in a minimum delta V of: \t%.3f km/s\n", minVal)

% Question 3

fvec = 0:0.1:360; % Creates a list of true anomaly values to iterate through (in degrees)

r1vec = (a1\*(1-e1^2)) ./ (1+e1\*cosd(fvec)); % List of the radii of the departure orbit

xvec1 = r1vec.\*cosd(fvec); % List of the x coordinates of the departure orbit

yvec1 = r1vec.\*sind(fvec); % List of the y coordinates of the departure orbit

%zvec1 = r1vec.\*sind(I1);

rTvec = (a\_T\*(1-e\_T^2)) ./ (1+e\_T\*cosd(fvec)); % List of the radii of the transfer orbit

xvecT = rTvec.\*cosd(fvec); % List of the x coordinates of the transfer orbit

yvecT = rTvec.\*sind(fvec); % List of the y coordinates of the transfer orbit

%zvecT = r1vec.\*sind(IT);

r2vec = (a2\*(1-e2^2)) ./ (1+e2\*cosd(fvec)); % List of the radii of the arrival orbit

xvec2 = r2vec.\*cosd(fvec); % List of the x coordinates of the arrival orbit

yvec2 = r2vec.\*sind(fvec); % List of the y coordinates of the arrival orbit

%zvec2 = r1vec.\*sind(I2);

figure(2)

hold on

%plot(xvec1, yvec1, zvec1, LineWidth=2)

%plot(xvec2, yvec2, zvec2, LineWidth=2)

%plot(xvecT(1:1801), yvecT(1:1801), zvecT(1:1801), ':', LineWidth=2) % Only plotting half of the transfer orbit

plot(xvec1, yvec1, LineWidth=2)

plot(xvec2, yvec2, LineWidth=2)

plot(xvecT(1:1801), yvecT(1:1801), ':', LineWidth=2) % Only plotting half of the transfer orbit

title('Two-Impulse Minimum Energy Orbit Transfer')

xlabel("x (km)")

ylabel("y (km)")

%ylabel("z (km)")

legend('Departure Orbit', 'Arrival Orbit', 'Transfer Orbit')

hold off

exportgraphics(gca,"HW2\_Problem2\_Figure2.jpg");

%% Functions

function aMinT = get\_aMin(r1\_in, r2\_in, df)

% This function gets the semimajor axis of minimum energy transfer

% orbit given r1, r2, and the change in f

sqrtTerm = sqrt( r1\_in^2 + r2\_in^2 - 2\*r1\_in\*r2\_in\*cosd(df) ) ;

aMinT = 0.25 \* (r1\_in + r2\_in + sqrtTerm);

end

function P\_output = getPT(amin, r1\_in, r2\_in, df)

% This funcion gets the semilatus rectum of a transfer orbit given r1,

% r2, and the change in f

k = r1\_in\*r2\_in\*(1-cosd(df));

m = r1\_in\*r2\_in\*(1+cosd(df));

l = r1\_in + r2\_in;

P\_output = (k\*m - 2\*amin\*k\*l) / (4\*amin\*m - 2\*amin\*l\*l);

end

function velo = getVelo(r\_input, a\_input)

% This function gets the velocity using energy given MU, r, and a

MU = 398600; % km^3 / s^2

velo = sqrt(2 \* MU \* ( (1/r\_input) - (1 / (2\*a\_input)) ) );

end

function dvelo = getDeltaV(vi, vf, dI, gami, gamf)

dvelo = sqrt( vf^2 + vi^2 - 2\*vi\*vf\*( cosd(gamf-gami) - (1-cosd(dI))\*cos(gamf)\*cos(gami) ) );

end