A diagram of an object with a circle

Description automatically generated with medium confidence

Figure Labeled Plot of Problem 1’s Orbit Transfer

% This is the code for HW2 - Problem 1

% By Nicholas Luis (PSU ID 930841391)

clear; clc; close all;

% Constants

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

PI = 3.141592654;

% Departure Orbit

a1 = 8000; % km

e1 = 0.01;

f1 = 30; % degrees

r1 = getRadius(a1, e1, f1);

v1 = getVelo(r1, a1); % velocity at point 1 (on departure orbit)

% Arrival Orbit

a2 = 27000; % km

e2 = 0.6;

f2 = 210; % degrees

r2 = getRadius(a2, e2, f2);

v2 = getVelo(r2, a2); % velocity at point 2 (on arrival orbit)

% Transfer Orbit

delta\_f = f2-f1;

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); % velocity at point 1 (on transfer orbit)

v2T = getVelo(r2, a\_T); % velocity at point 2 (on transfer orbit)

t = PI \* sqrt((a\_T^3) / MU); % Transfer time

% Delta V

dV1 = abs(v1T - v1);

dV2 = abs(v2T - v2);

%% Plotting Values

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

% X Y values. Note that (0,0) will be located on the Earth

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

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

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

xvec2 = r2vec.\*cosd(fvec-30); % List of the x coordinates of the arrival orbit

yvec2 = r2vec.\*sind(fvec-30); % List of the y coordinates of the arrival orbit

figure(1)

hold on

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)")

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

hold off

exportgraphics(gca,"HW2\_Problem1\_Figure.jpg");

%% Functions

function radius = getRadius(a\_input, e\_input, f\_input)

p = a\_input \* (1 - (e\_input^2));

radius = p / (1+e\_input\*cosd(f\_input));

end

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