AERO 3240 Term Assignment

Carleton University

By: Boaz Aharony

Student ID: 101191503

2022-12-07

Notes

All figures, code attachments, tables, code lines, and section references are clickable and will bring you to said reference. As well all 2-D plots are vectored images that allow unlimited zoom-in.

Contents

List of Figures

List of Tables

List of MATLAB Code Attachments

2 Two-Body Problem

2.5 Speed and Periods of Circular LEO

Created using code attachment 1 in Appendix A.

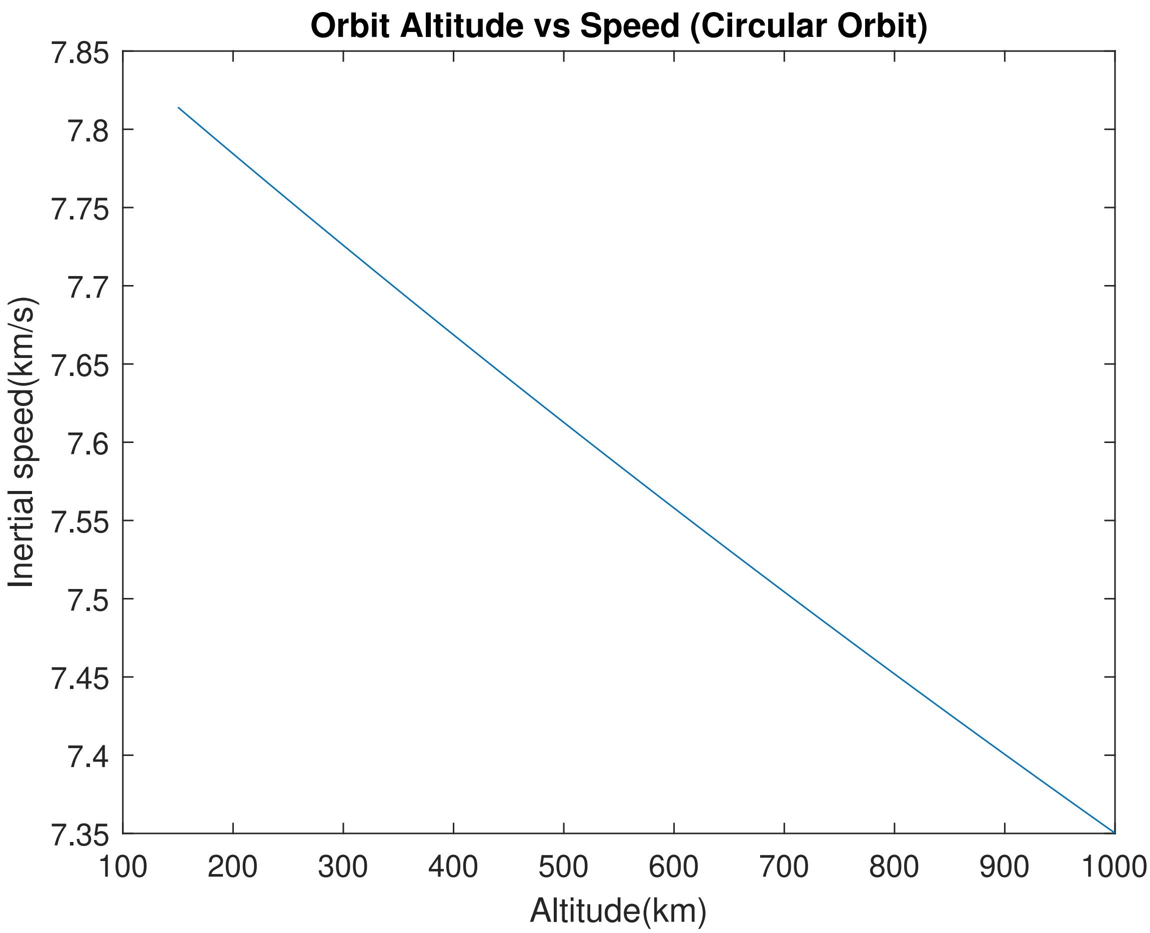

Figure 1: Orbital Speed vs Altitude

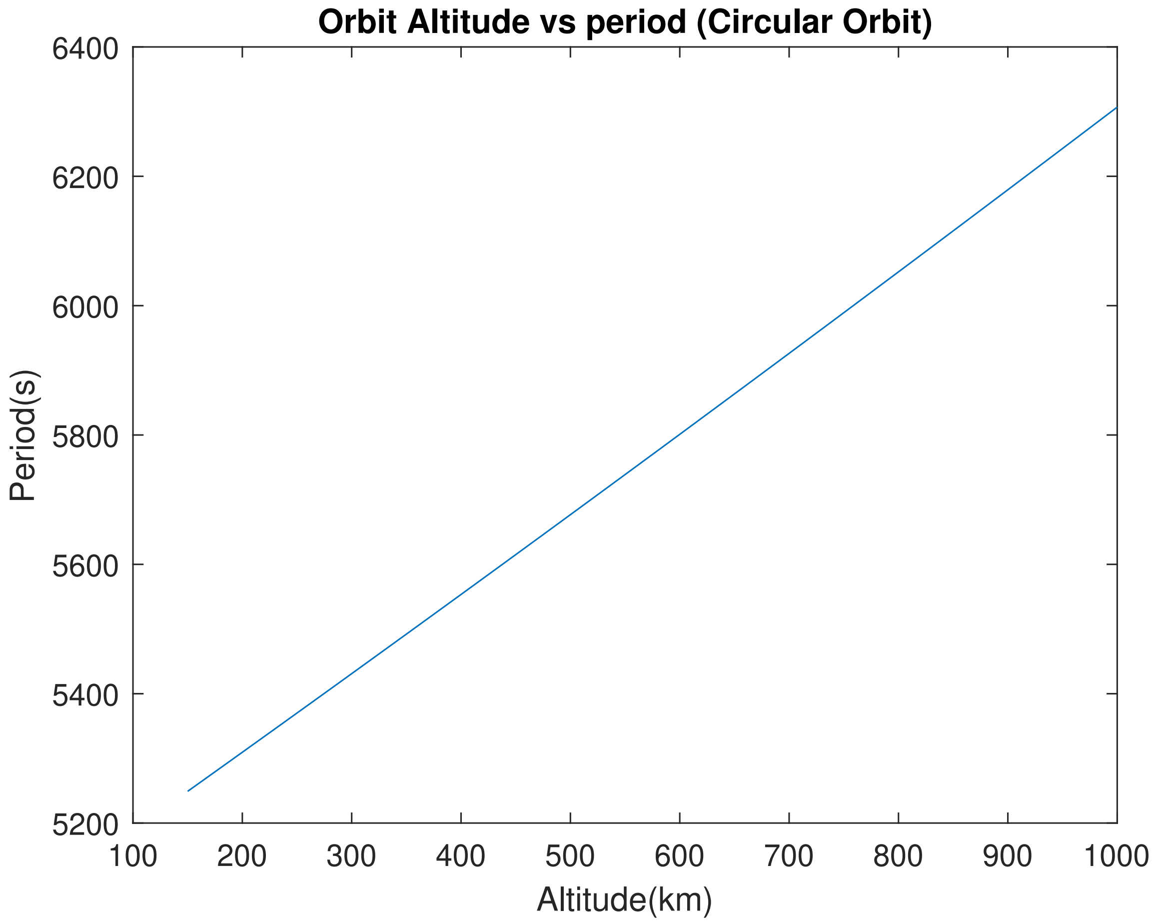

Figure 2: Orbital Period vs Altitude

2.13 PROBA-2 Without Perturbations

Created using `MATLAB` code and `SIMULINK` diagrams found in Appendix B.

2.13 a Plot of Orbit in the Orbital Plane

Created using code attachment 2 after line 65

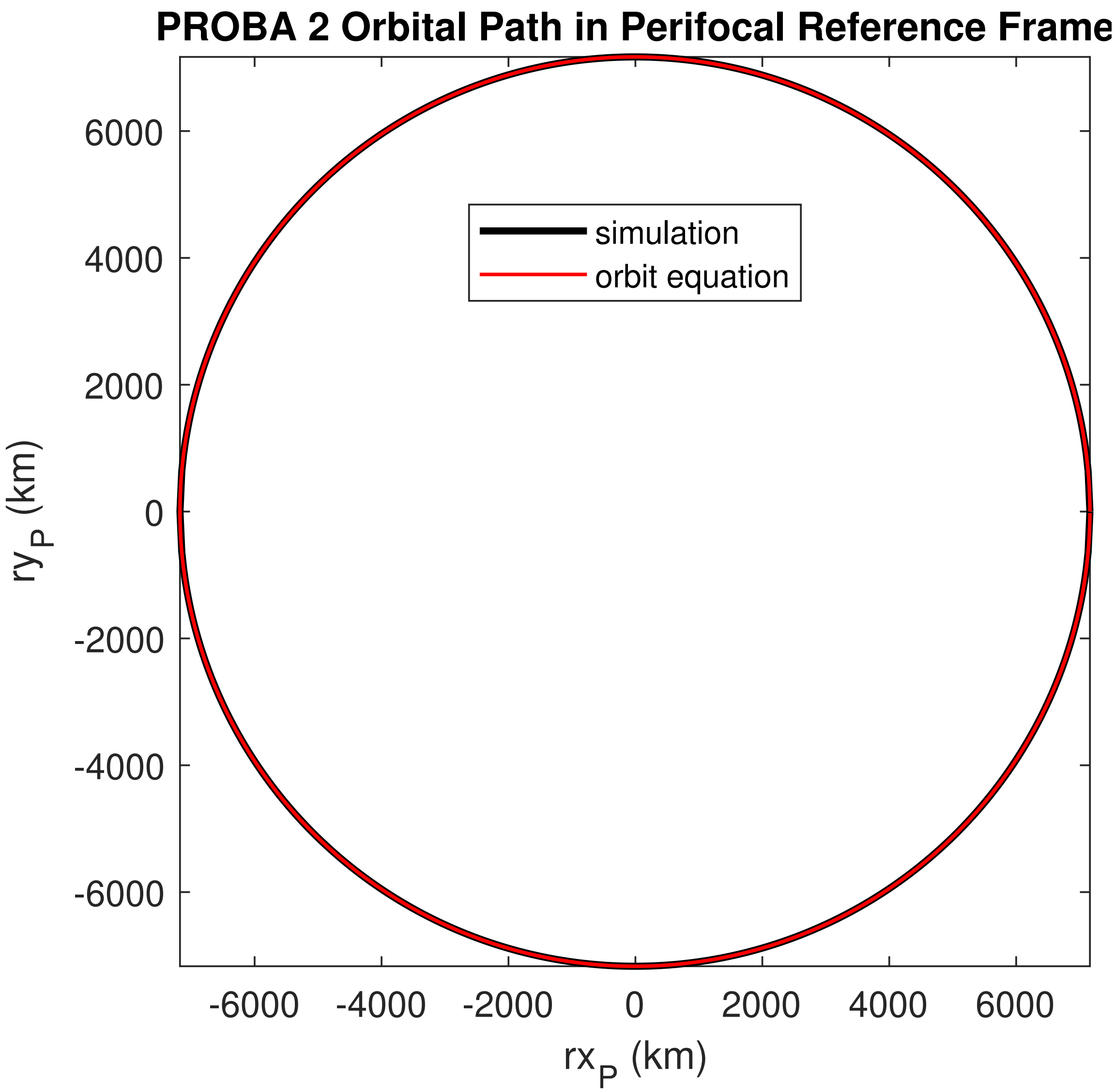

Figure 3: PROBA-2 Orbit in Perifocal Frame

2.13 b Initial Conditions For `SIMULINK` Integrators

Calculated in code attachment 2 after line 42 using rotation matrix functions found in appendix E.1

Table 1: Table of Initial PROBA-2 Position and Velocity

Component	$\mathbf{r}_{I,initial}$**(km)**	$\mathbf{v}_{I,initial}$**(km/s)**
X	0.0000	-1.0749
Y	-7.1618×10^3	0.0000
Z	0.0000	7.3862

2.13 c 3D plot of Orbit

An animation of one orbit cycle can be viewed by clicking this **youtube link** (done using attachment 12, does not show earth's spin)

The figure below was created using code attachment 2 after line 111

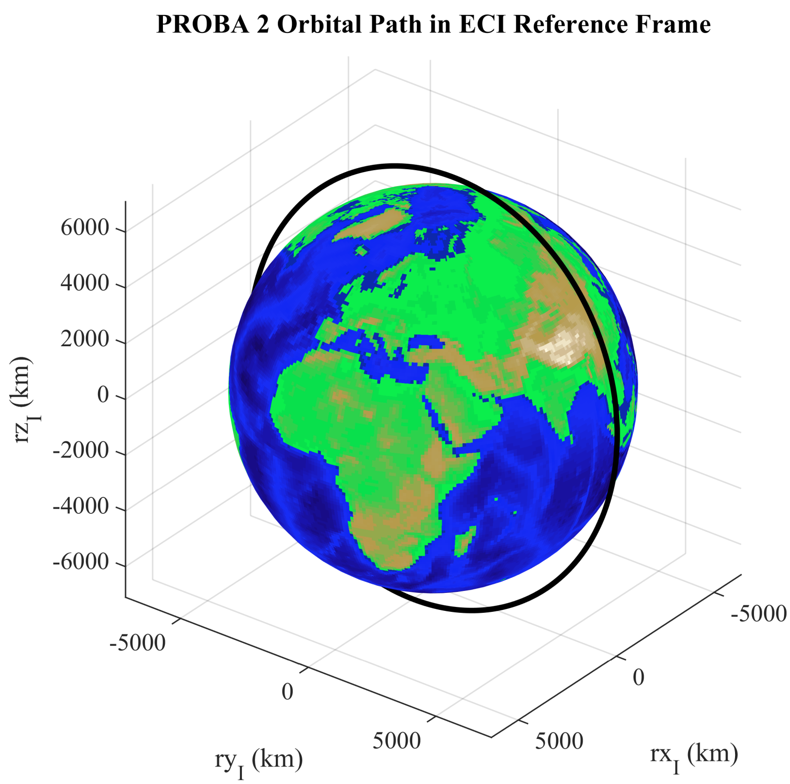

Figure 4: 3-D Plot of PROBA-2 Orbit Around Earth

2.13 d Components of Position and Velocity in ECIF

Created using code attachment 2 after line 139

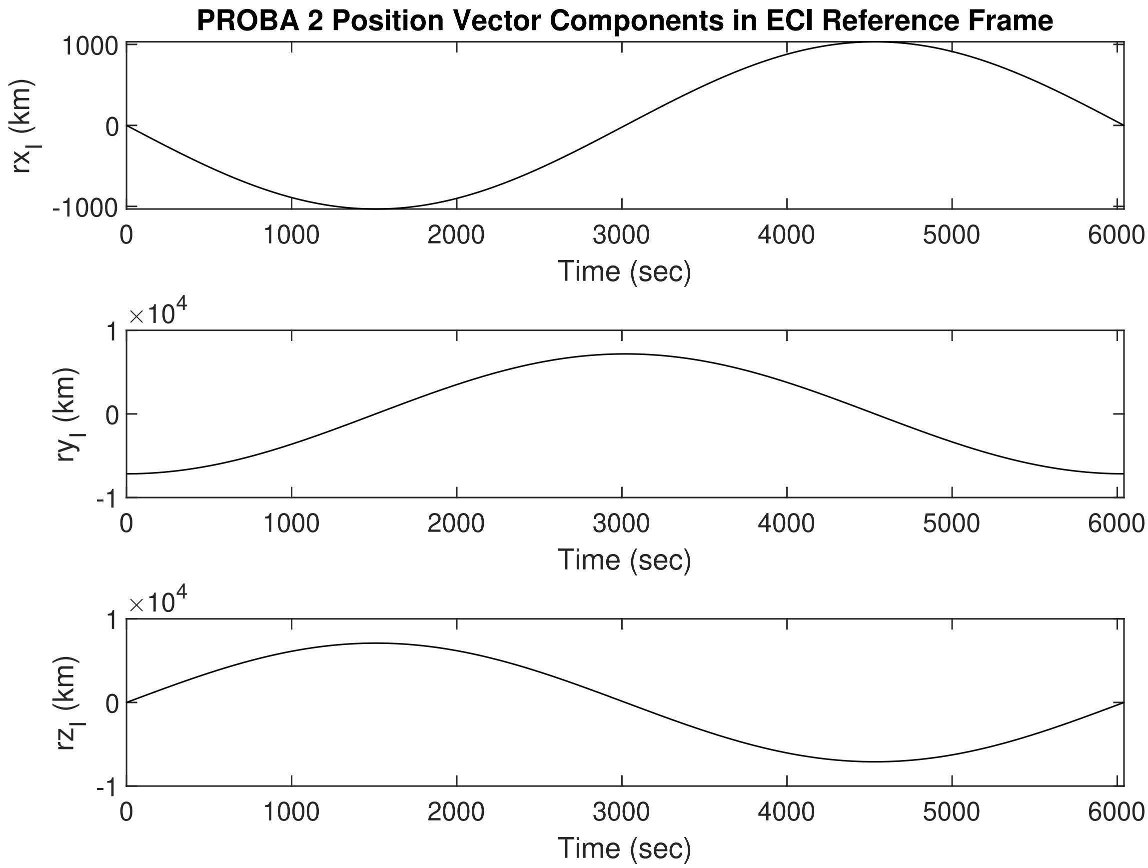

Figure 5: PROBA-2 Components of Position

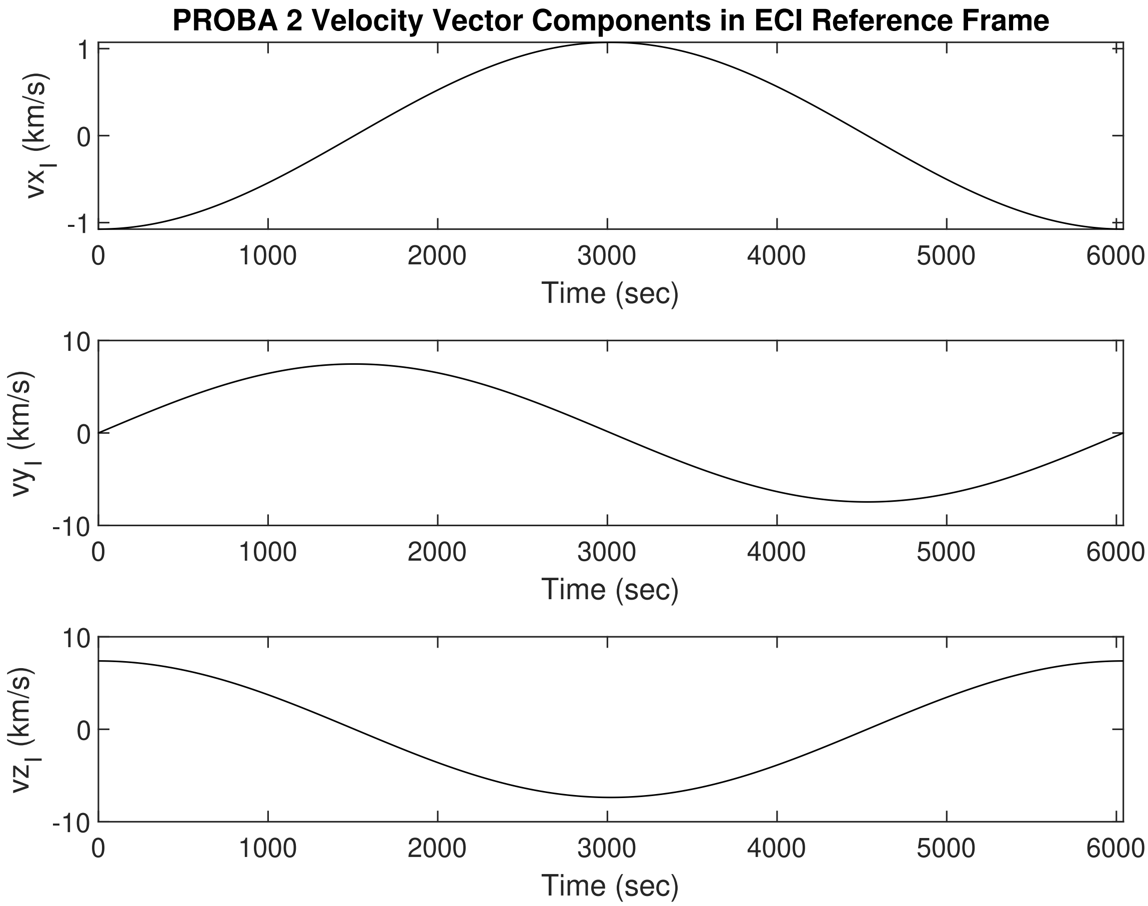

Figure 6: PROBA-2 Components of Velocity

2.13 e Radius and Speed in ECIF

Created using code attachment 2 after line 191

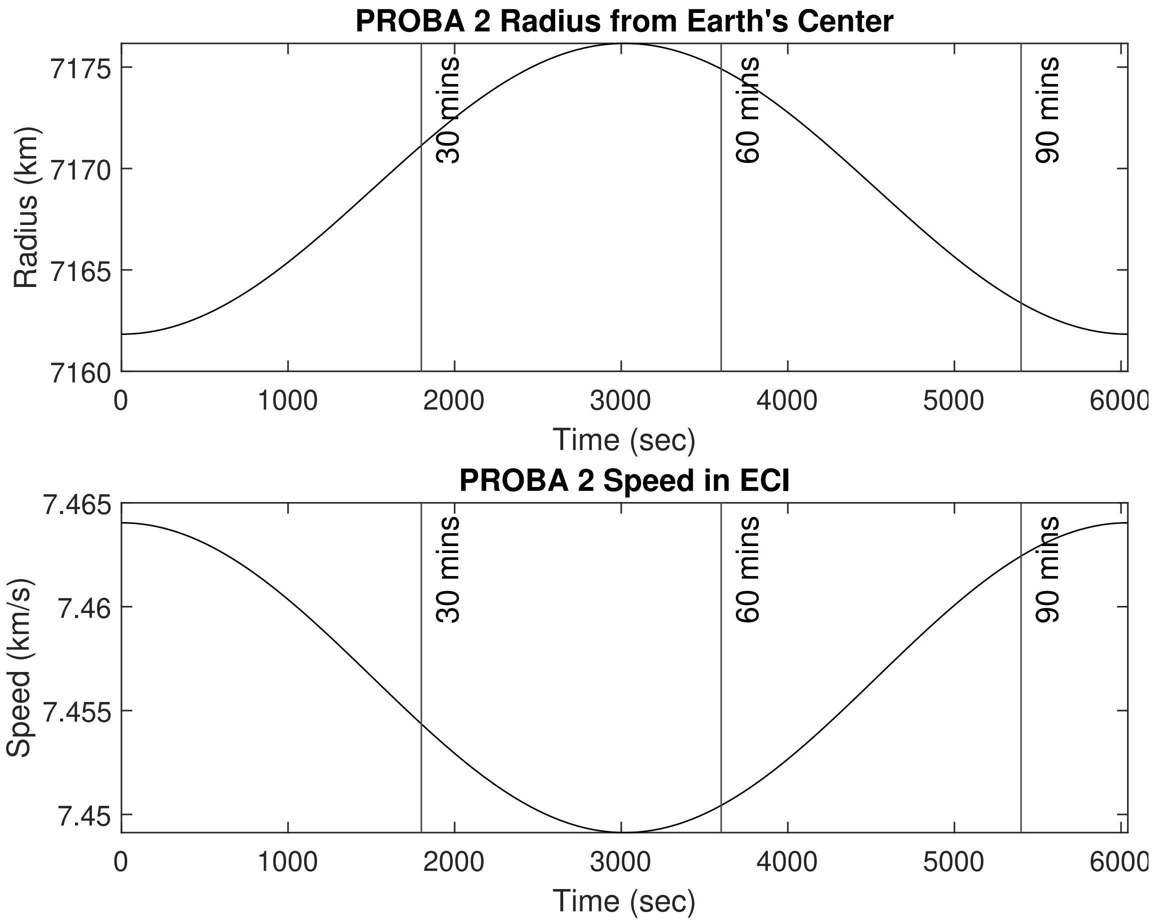

Figure 7: PROBA-2 Radius and Speed

Using the Vis-Viva equation as a function of radius r for a constant semi-major axis a the velocity at any point is determined by eq. 1

$$v = \sqrt{\mu \left(\frac{2}{r} - \frac{1}{a} \right)} \tag{1}$$

The radius was determined in code by finding the position in the array of the closest time point and taking its radius, which was visually compared with the above graphs. To verify the graphs, the results compared to eq. 1 are confirmed in table 2:

Table 2: Table of comparison between simulation and vis-viva results

Time (minutes)	$\mathbf{r}_I$**(km)**	**Calculated** $\mathbf{v}_I$**(km/s)**	**Simulated** $\mathbf{v}_I$**(km/s)**
30	7.1711×10^3	7.4544	7.4544
60	7.1749×10^3	7.4504	7.4504
90	7.1634×10^3	7.4624	7.4624

2.13 f Ground Plots

Video animated ground tracks are available by clicking this **youtube link** (done using attachment 13) As well as the plot below which was created using code attachment 2 after line 252:

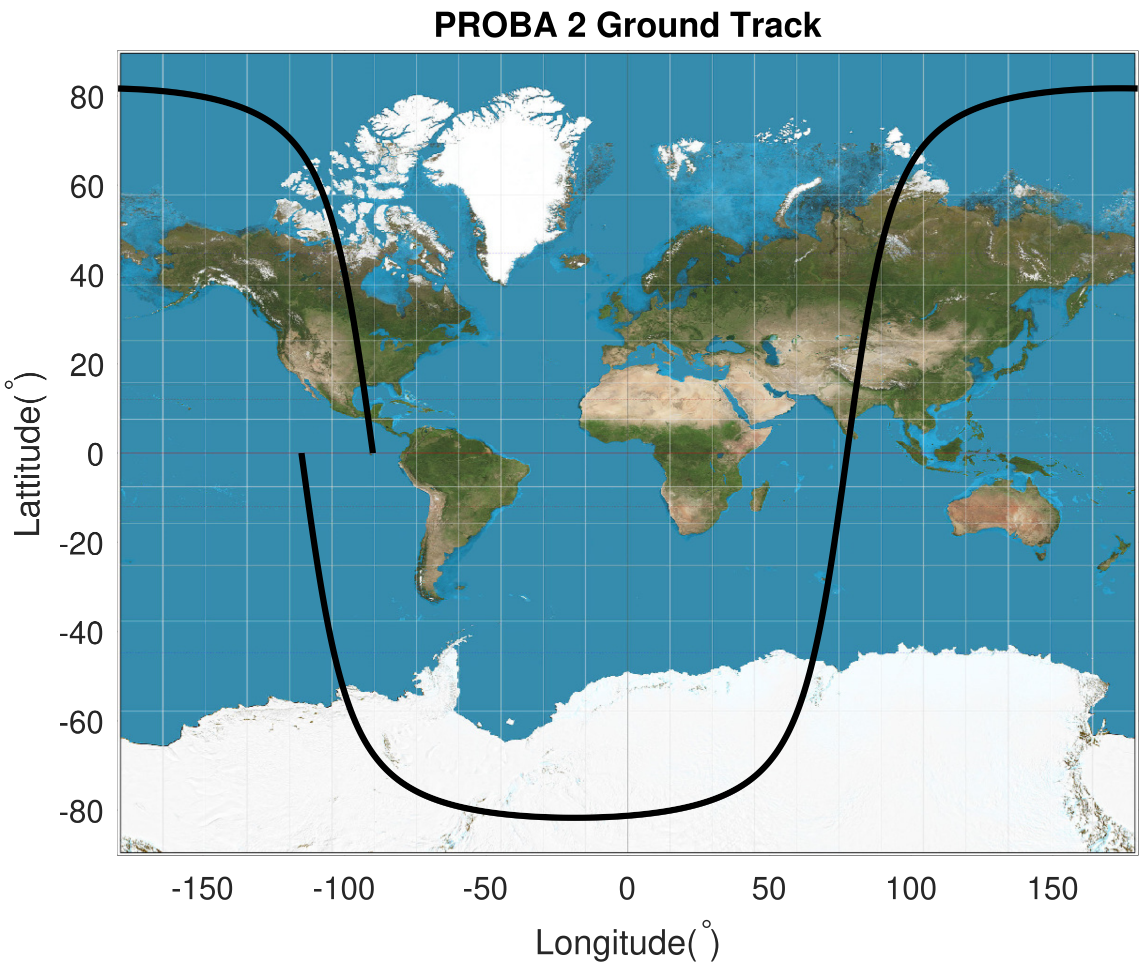

Figure 8: PROBA-2 ground tracks

3 Orbital Pertubations

3.1 PROBA-2 with Pertubations

Created using `MATLAB` code and `SIMULINK` diagrams found in Appendix C.

3.1 a Calculated Secular and Average Rate of Change

To determine the expected secular change in RAAN eq. 2 is used

$$\Delta\Omega = -\frac{3\pi J_2 R_{\oplus}^2}{p^2}\cos(i) \tag{2}$$

To determine the expected average rate of change eq. 3 is used:

$$\langle\dot{\Omega}\rangle = -\frac{3 J_2 R_{\oplus}^2}{2p^2} n \cos(i) \tag{3}$$

This is calculated in section A of `MATLAB` code attachment 5 after line 73. It is tabulated in Tables 3 and 4 for inclinations of 98.28° and 10° respectively. These tables are found in 3.1 e.

3.1 b Simulating with J_2

The simulation was done by adding the J_2 perturbation function in code attachment 6 to the `SIMULINK` diagram Figure 27 as a function block and summation with the acceleration caused by the two-body-problem. The main script attachment 5 uses the simulation for the rest of the question.

3.1 c Plot of Change in RAAN

Done in code attachment 5 after line 81

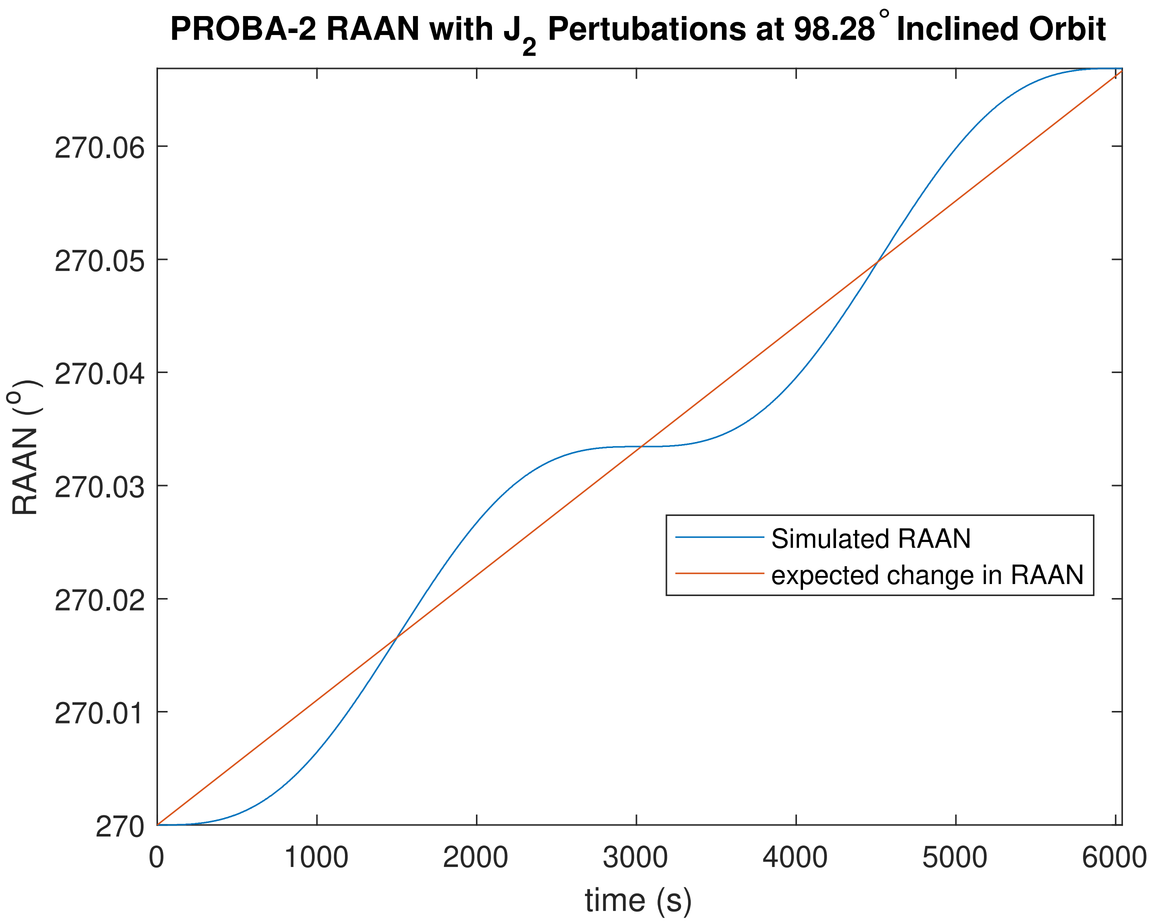

Figure 9: PROBA-2 Change in RAAN at 98.28° inclination

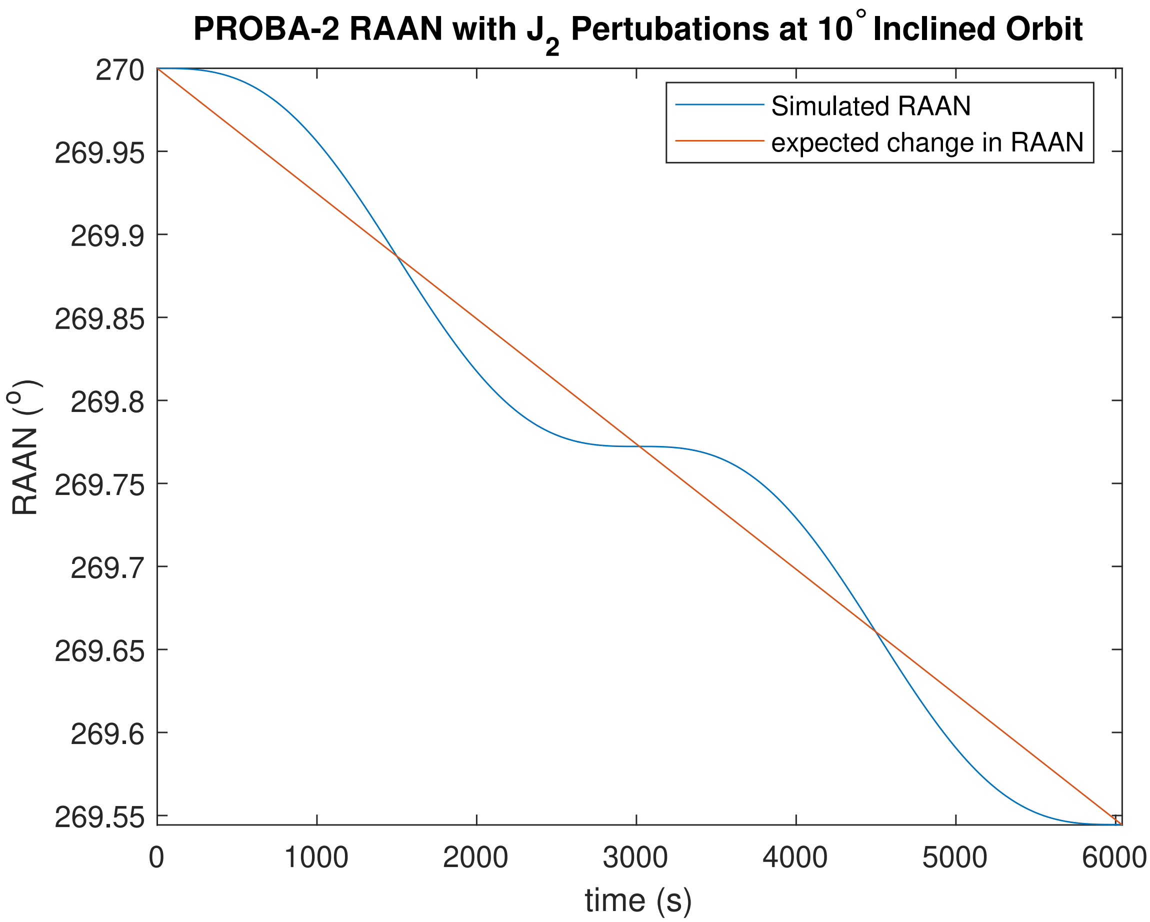

Figure 10: PROBA-2 Change in RAAN at 10° inclination

3.1 d Interpretation of Change in RAAN from Simulation

Calculated after line 113 of `MATLAB` code attachment 5 and tabulated in Tables 3 and 4 for inclinations of 98.28° and 10° respectively. These tables are found in 3.1 e.

3.1 e Comparison of Expected and Simulated Secular and Rate of Change in RAAN

Done in code attachment 5 after line 126

Table 3: Table of comparison between simulation and calculated RAAN results for inclination of 98.28°

	secular change (°/rev)	**average rate of change(°/s)**
expected	6.6639×10^{-2}	1.1031×10^{-5}
simulation	6.6876×10^{-2}	1.1071×10^{-5}
difference	2.3773×10^{-4}	3.9353×10^{-8}

Table 4: Table of comparison between simulation and calculated RAAN results for inclination of 10°

	secular change (°/rev)	**average rate of change(°/s)**
expected	-4.5570×10^{-1}	-7.5437×10^{-5}
simulation	-4.5547×10^{-1}	-7.5398×10^{-5}
difference	2.3206×10^{-4}	3.8416×10^{-8}

3.1 f Repeat for 10° Inclination

The repeat is done in Figure 10 and Table 4 by changing the initial i (inclination) value in code attachment 5 on line 29.

5 Spacecraft Formation Flying

5.1 Envisat Formation

5.1 f Confirmation of Answers with `MATLAB`

Created using code attached in Appendix D

5.1 f i Along-Track Formation

Initial conditions in table 5 are calculated in code attachment 7 after line 47.

Table 5: Table of initial formation conditions (along-track formation)

$\mathbf{v_{x_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{y_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{z_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{x_0}$(km)	$\mathbf{y_0}$(km)	$\mathbf{z_0}$(km)
0.000	0.000	0.000	0.000	-1.000×10^1	0.000

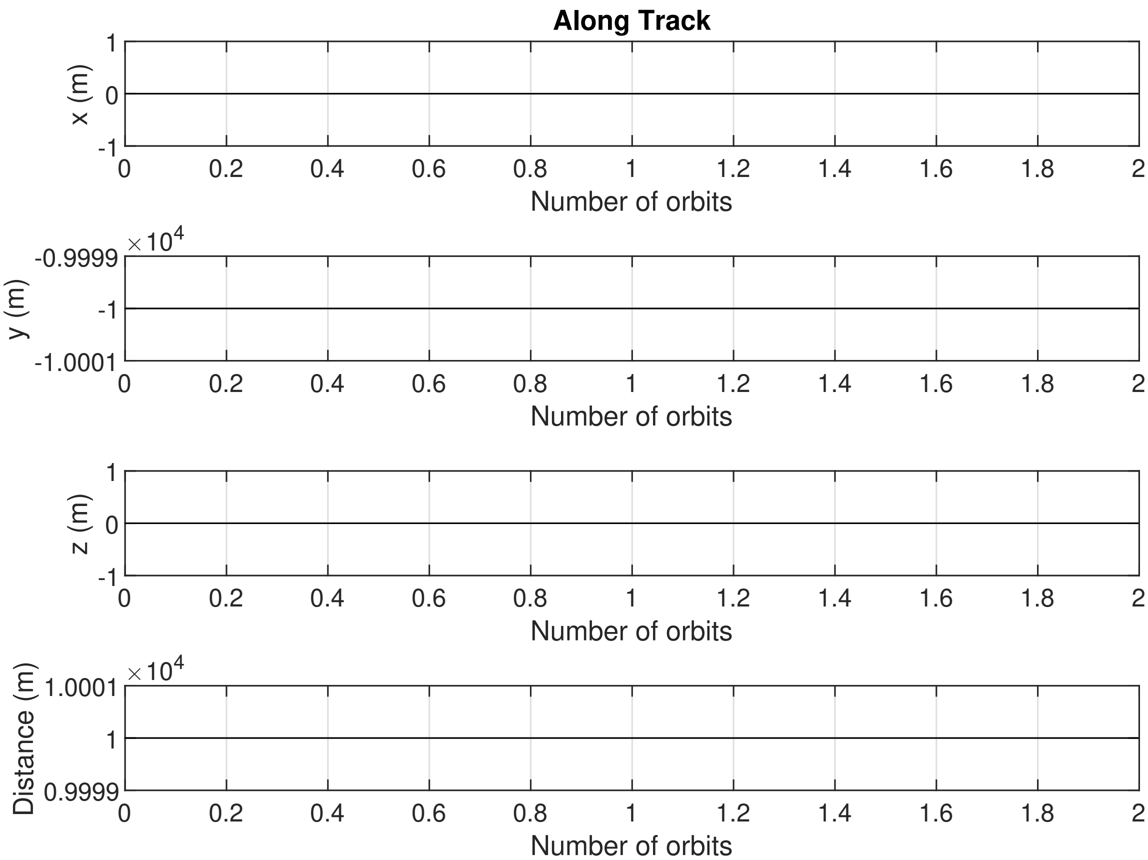

Figure 11: Position component plots (along-track formation)

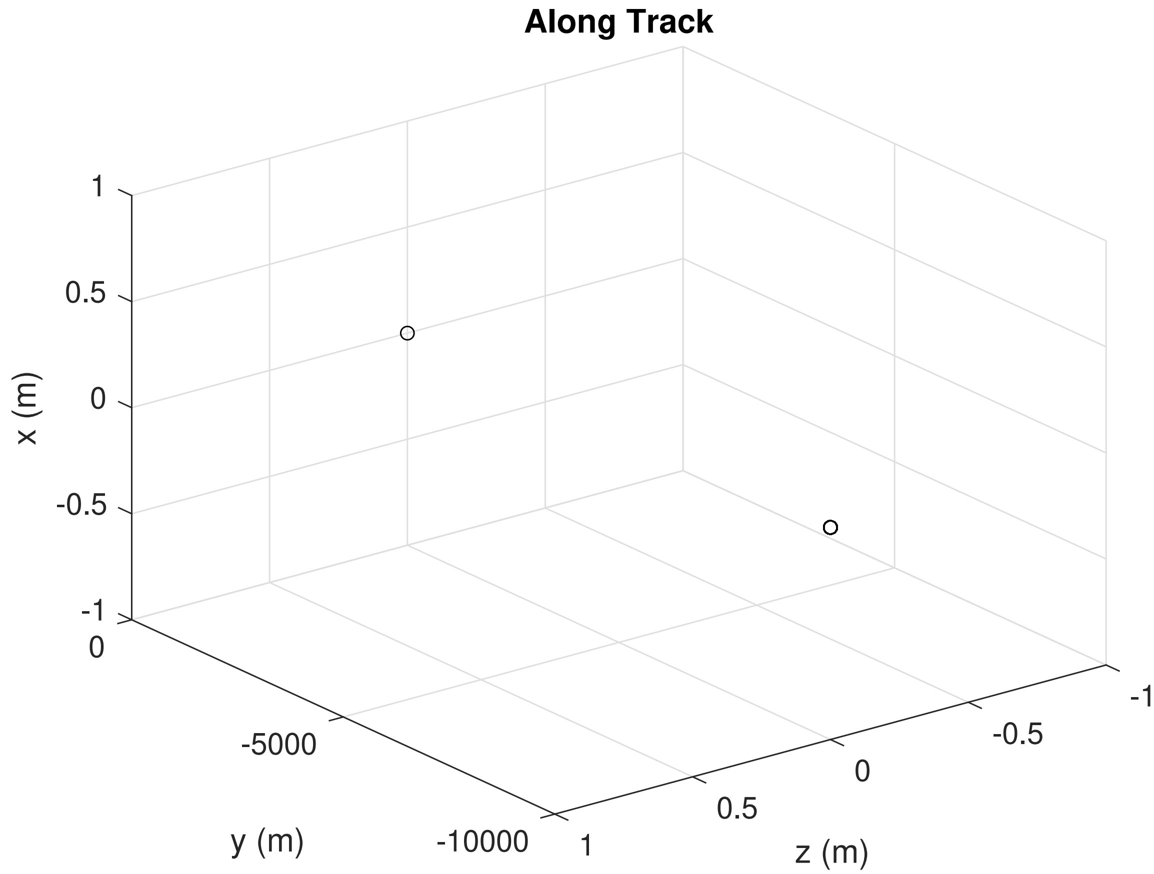

Figure 12: 3D-plot of path around target (along-track formation)

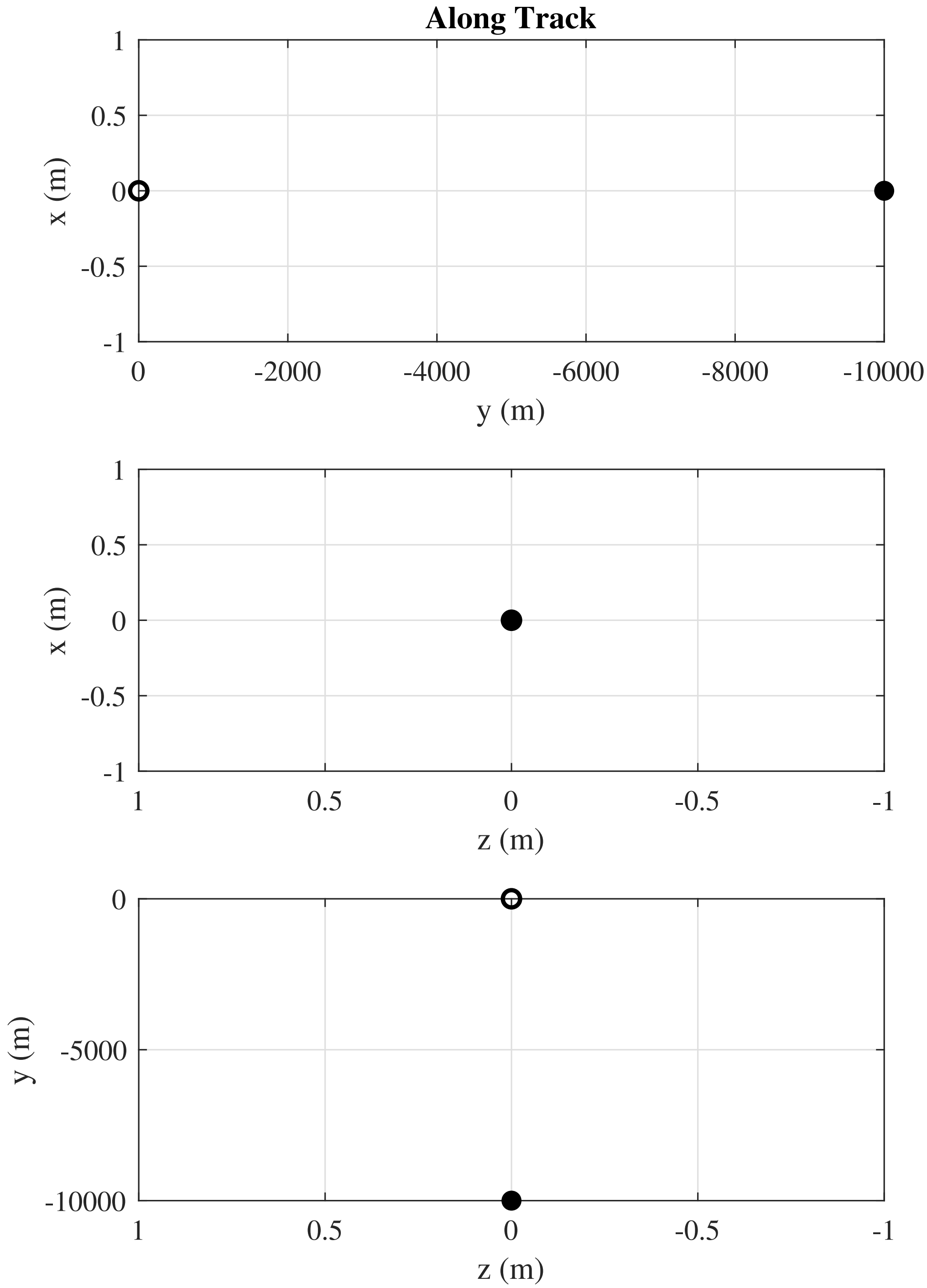

Figure 13: Path in projected planes (along-track formation)

5.1 f ii In-Track Formation

Initial conditions in table 6 are calculated in code attachment 7 after line 57.

Table 6: Table of initial formation conditions (in-track formation)

$\mathbf{v_{x_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{y_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{z_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{x_0}$(km)	$\mathbf{y_0}$(km)	$\mathbf{z_0}$(km)
0.000	0.000	0.000	0.000	-1.000×10^{1}	6.900×10^{-1}

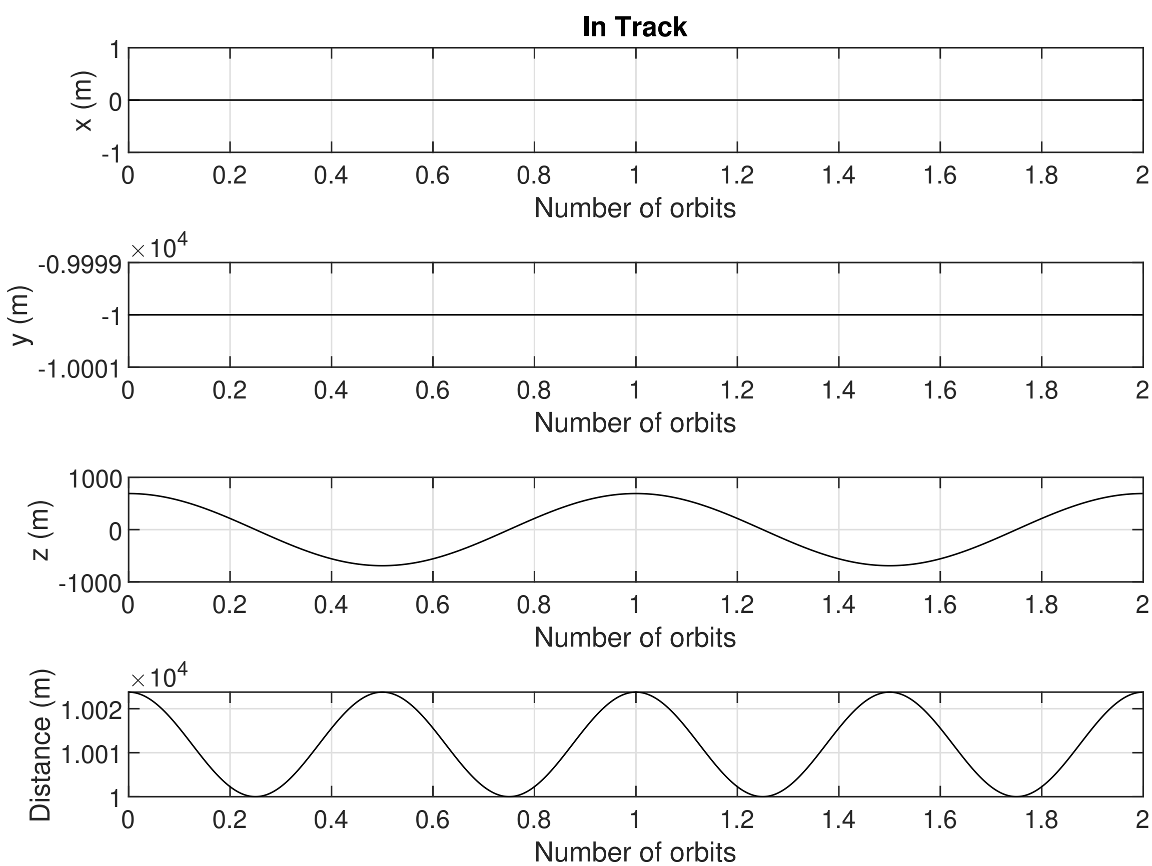

Figure 14: Position component plots (in-track formation)

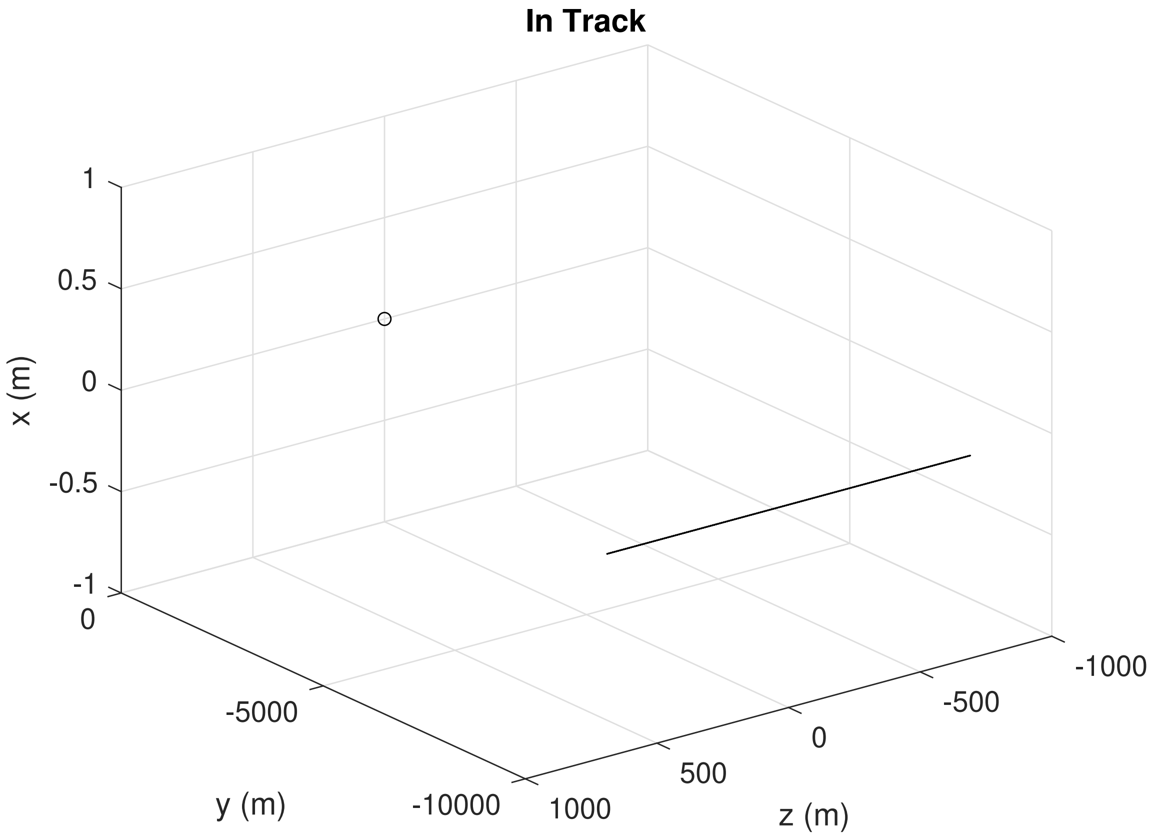

Figure 15: 3D-plot of path around target (in-track formation)

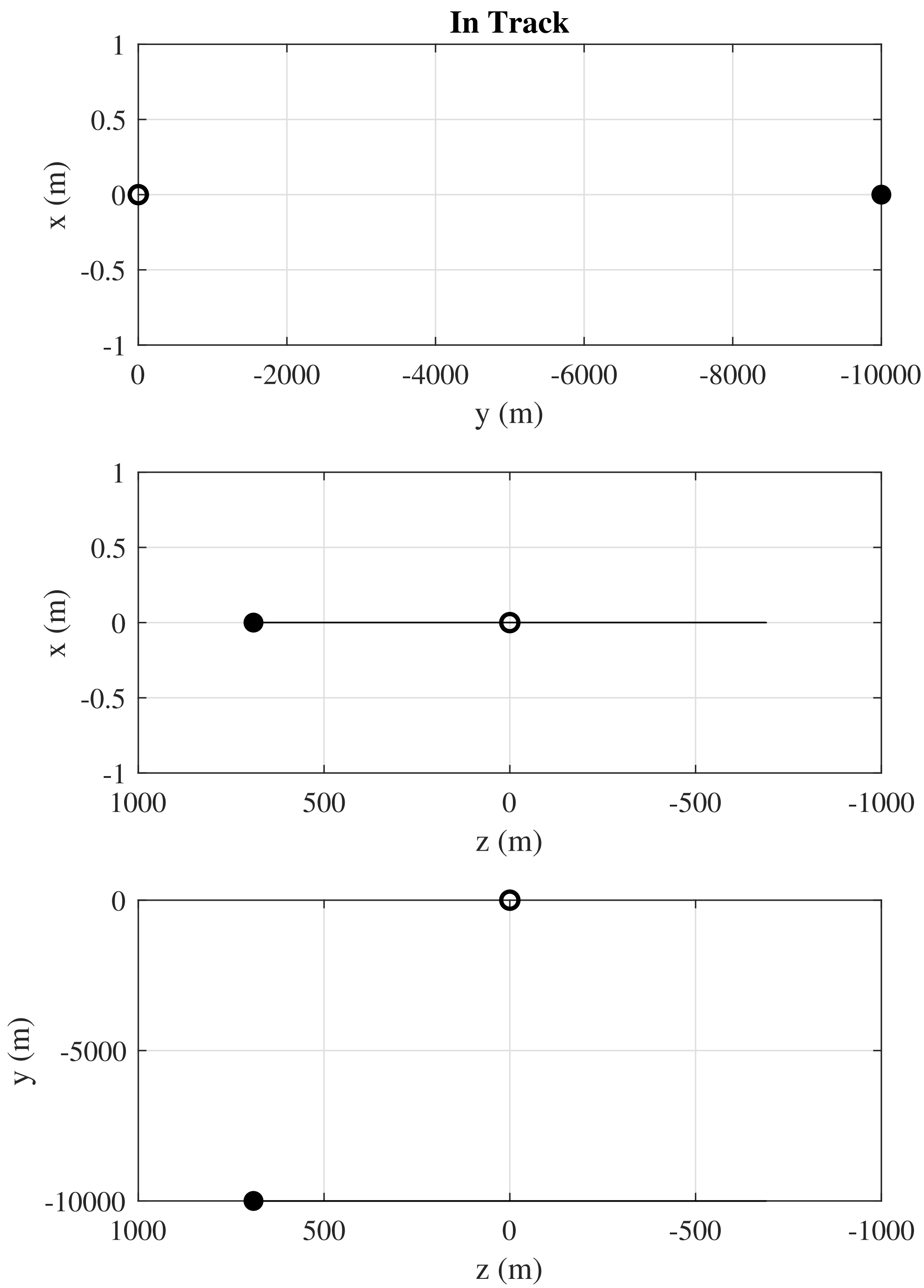

Figure 16: Path in projected planes (in-track formation)

5.1 f iii In-Plane Elliptical Formation

Initial conditions in table 7 are calculated in code attachment 7 after line 67. Assuming initial phase angle $\alpha = 0$ for initial conditions in table 7.

Table 7: Table of initial formation conditions (in-plane elliptical formation)

$\mathbf{v_{x_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{y_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{z_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{x_0}(\text{km})$	$\mathbf{y_0}(\text{km})$	$\mathbf{z_0}(\text{km})$
0.000	-5.228×10^{-4}	0.000	2.500×10^{-1}	0.000	0.000

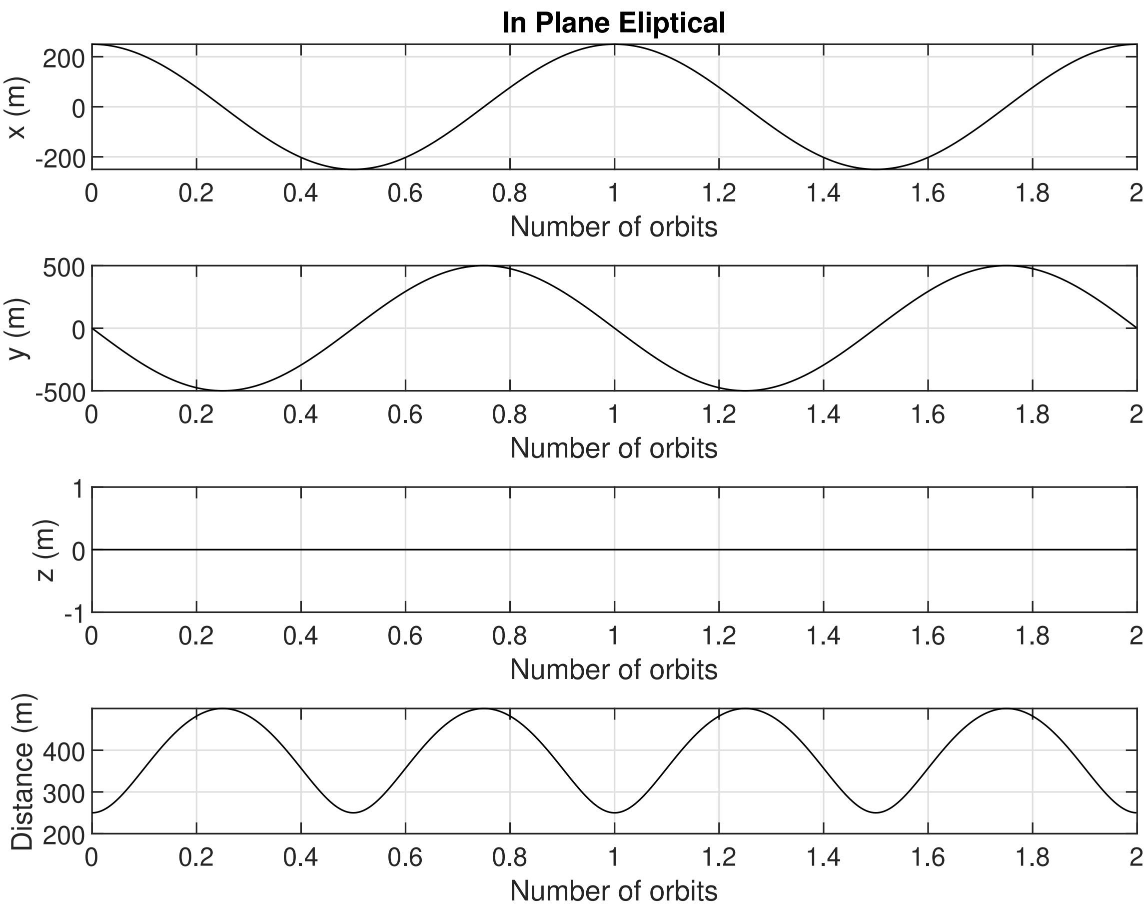

Figure 17: Position component plots (in-plane elliptical formation)

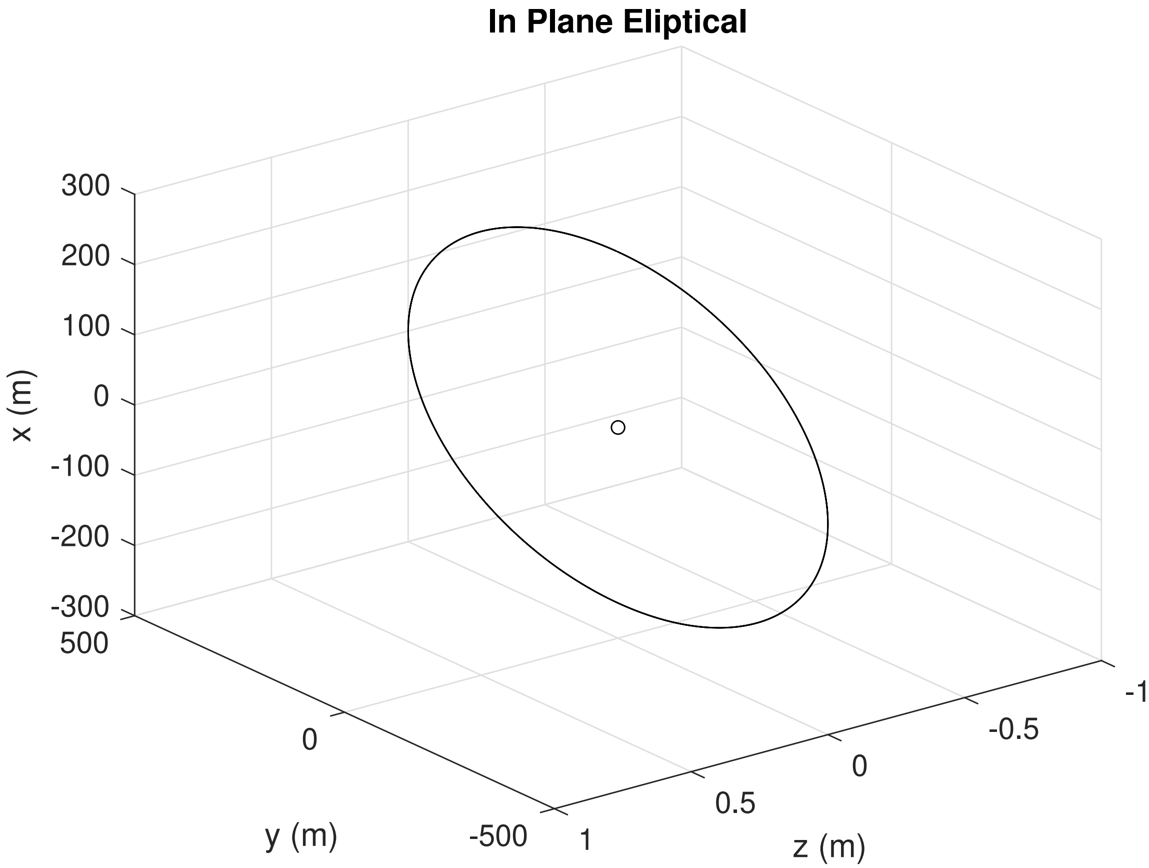

Figure 18: 3D-plot of path around target (in-plane elliptical formation)

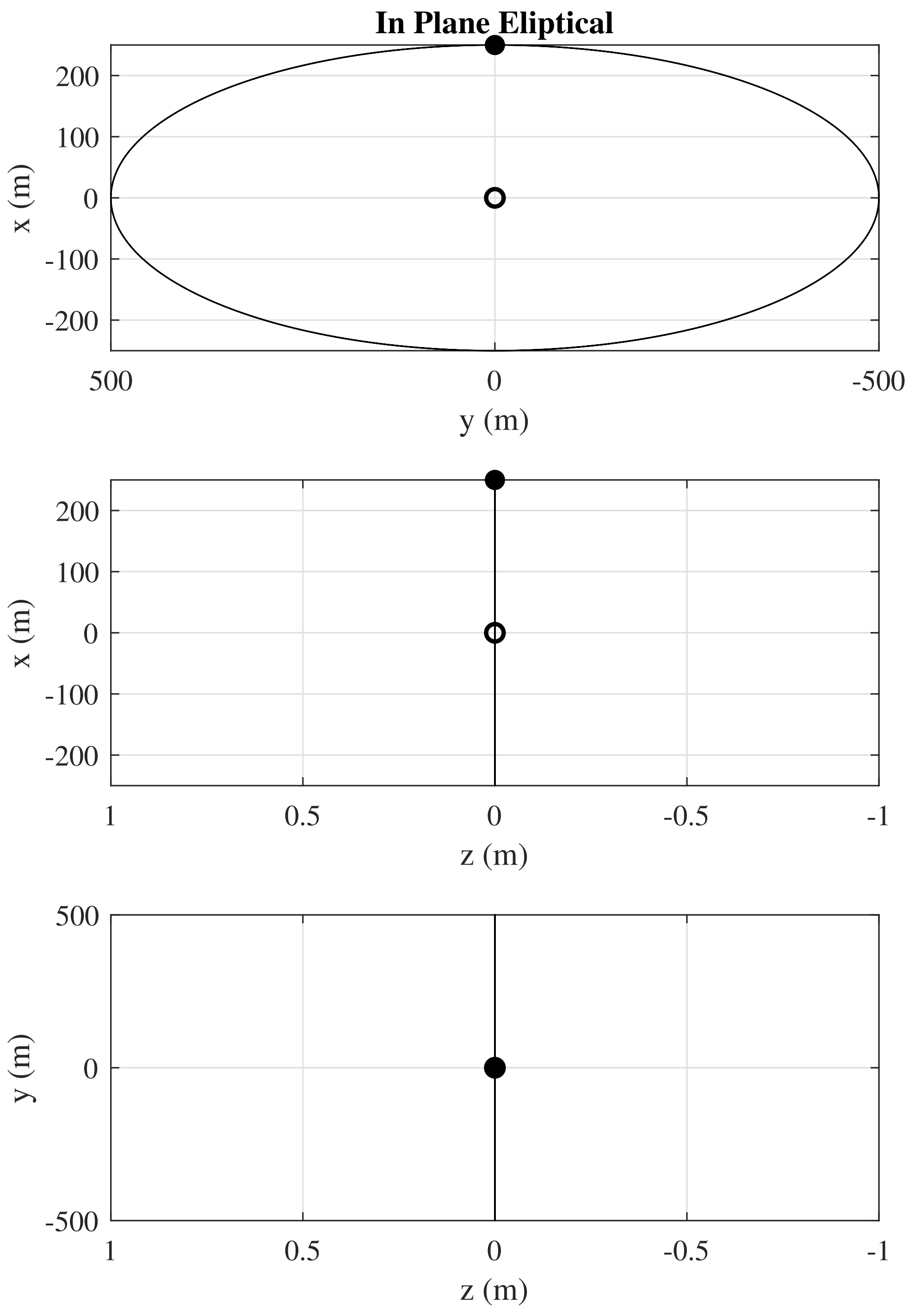

Figure 19: Path in projected planes (in-plane elliptical formation)

5.1 f iv Circular Formation

Initial conditions in table 8 are calculated in code attachment 7 after line 89.

Table 8: Table of initial formation conditions (circular formation)

$\mathbf{v_{x_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{y_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{z_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{x_0}$(km)	$\mathbf{y_0}$(km)	$\mathbf{z_0}$(km)
0.000	-2.091×10^{-5}	0.000	1.000×10^{-2}	0.000	1.732×10^{-2}

$\mathbf{z_0}$ and $\mathbf{v_{z_0}}$ in table 8 can also be negated and achieve the same formation type.

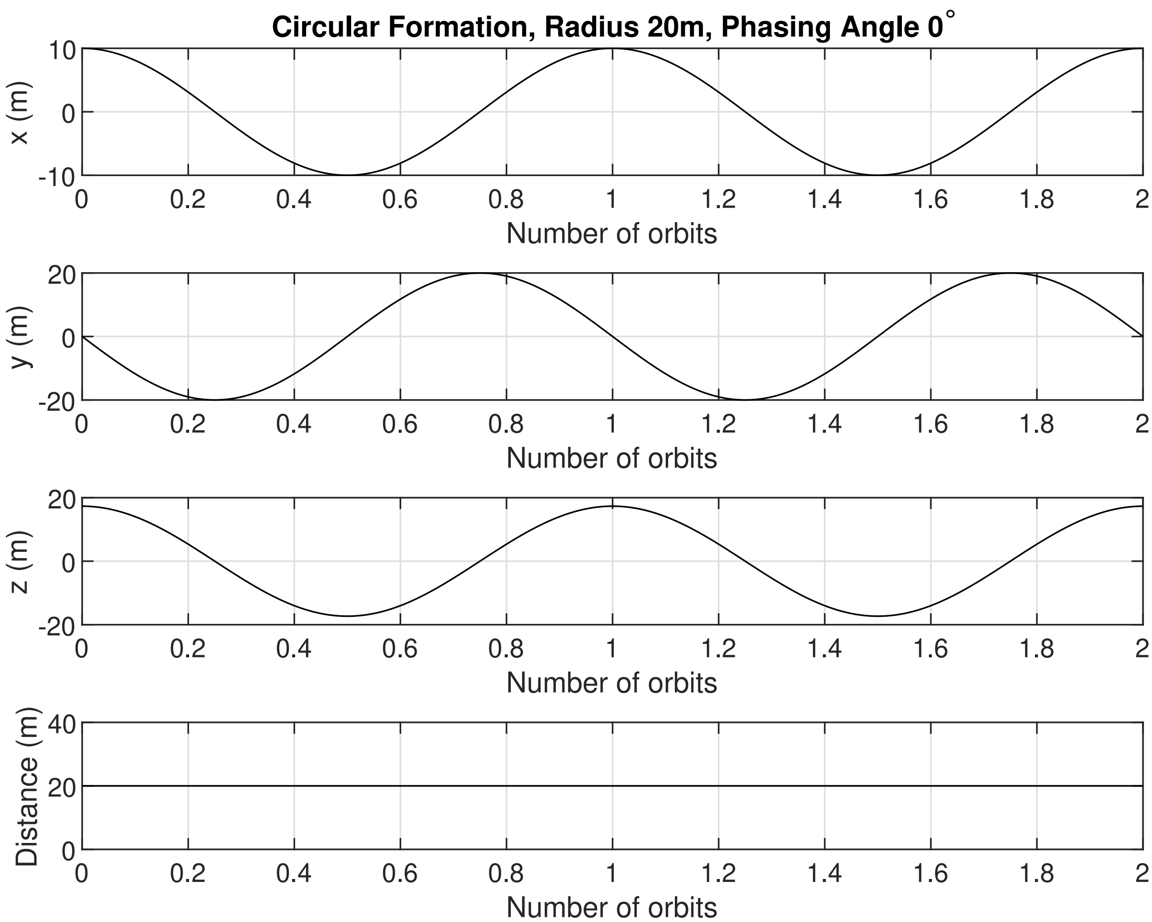

Figure 20: Position component plots (circular formation)

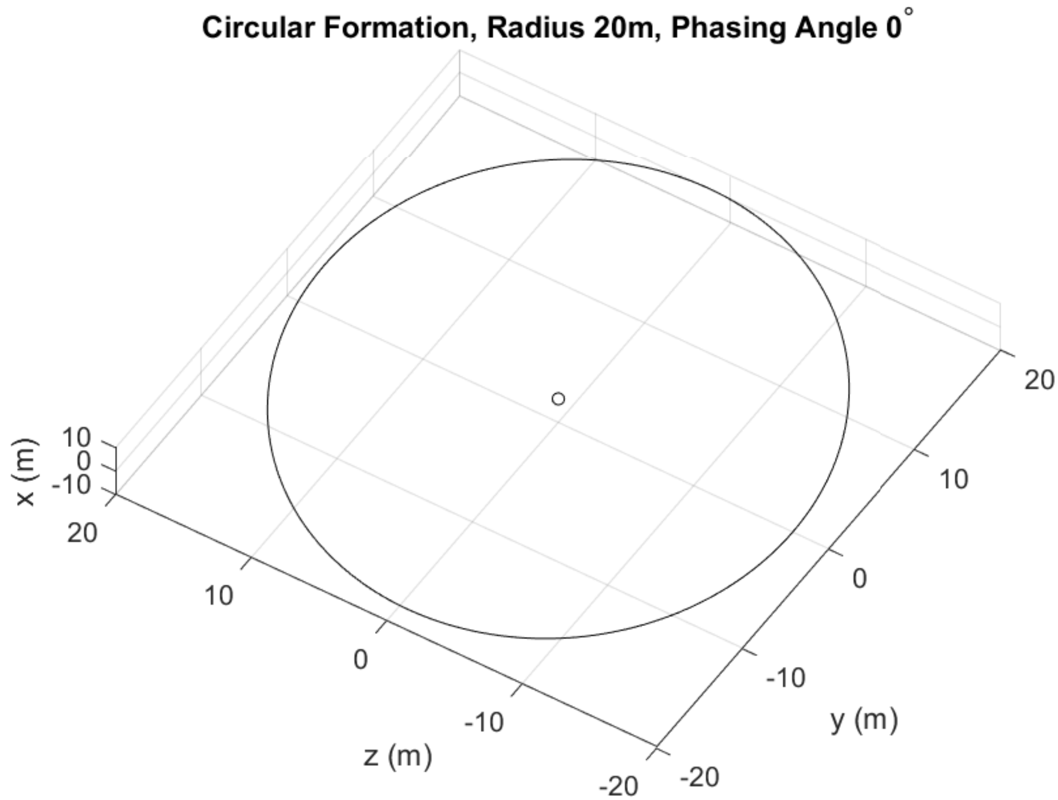

Figure 21: 3D-plot of path around target (circular formation)

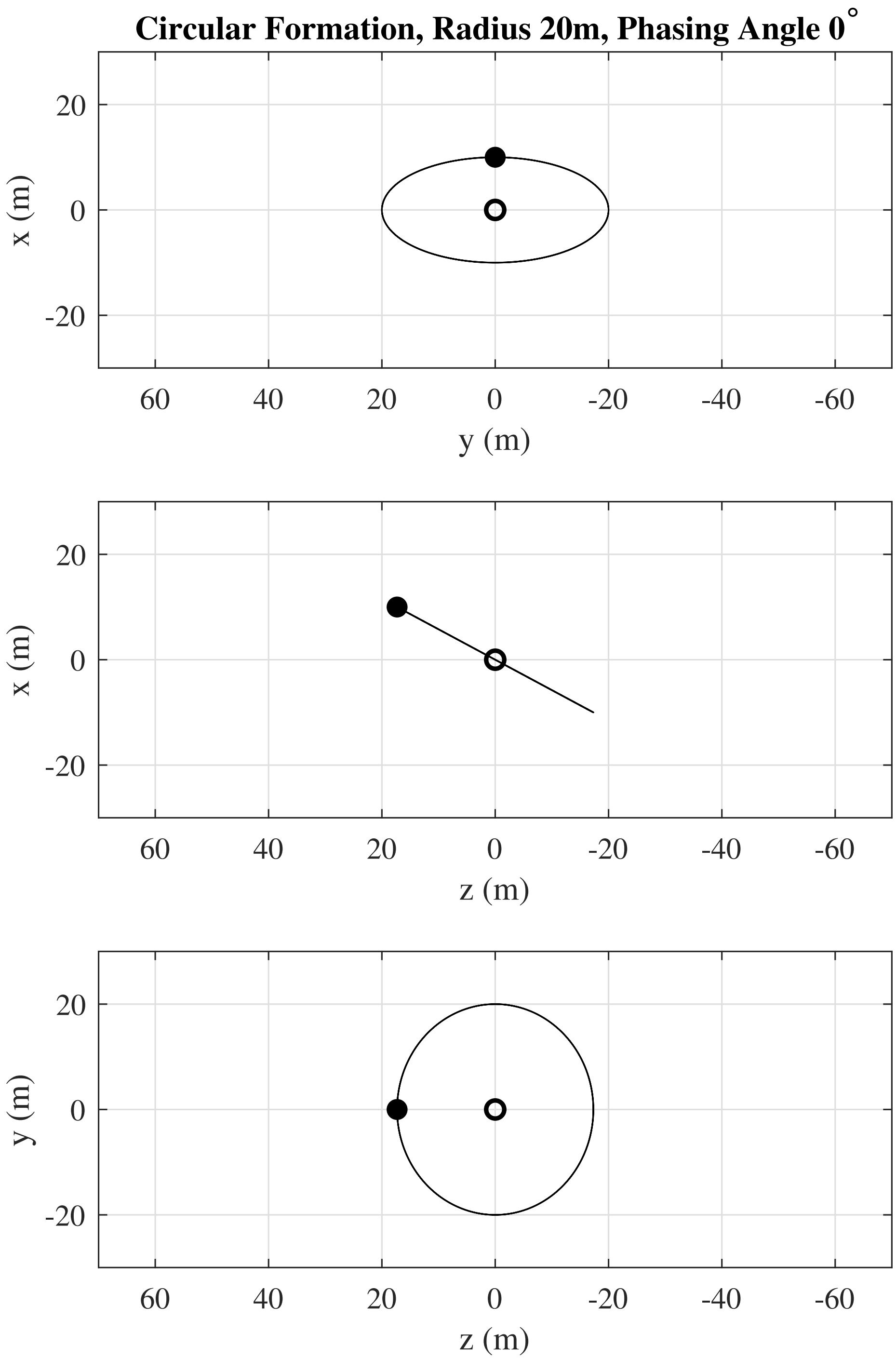

Figure 22: Path in projected planes (circular formation)

5.1 f v Projected Circular Formation

Initial conditions in table 9 are calculated in code attachment 7 after line 102.

Table 9: Table of initial formation conditions (projected circular formation)

$\mathbf{v_{x_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{y_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{v_{z_0}}\left(\frac{\text{km}}{\text{s}}\right)$	$\mathbf{x_0}(\text{km})$	$\mathbf{y_0}(\text{km})$	$\mathbf{z_0}(\text{km})$
1.046×10^{-5}	0.000	2.091×10^{-5}	0.000	2.000×10^{-2}	0.000

$\mathbf{z_0}$ and $\mathbf{v_{z_0}}$ in table 9 can also be negated and achieve the same formation type.

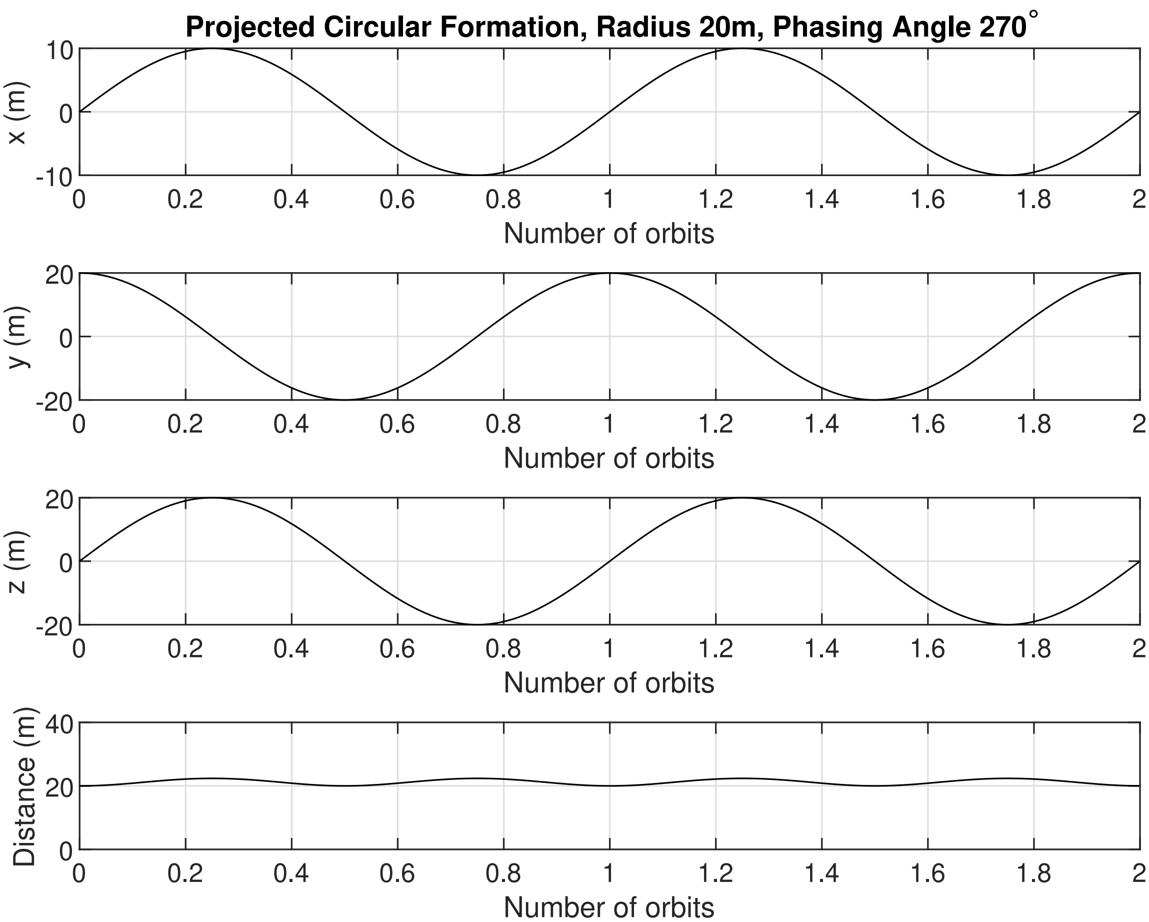

Figure 23: Position component plots (projected circular formation)

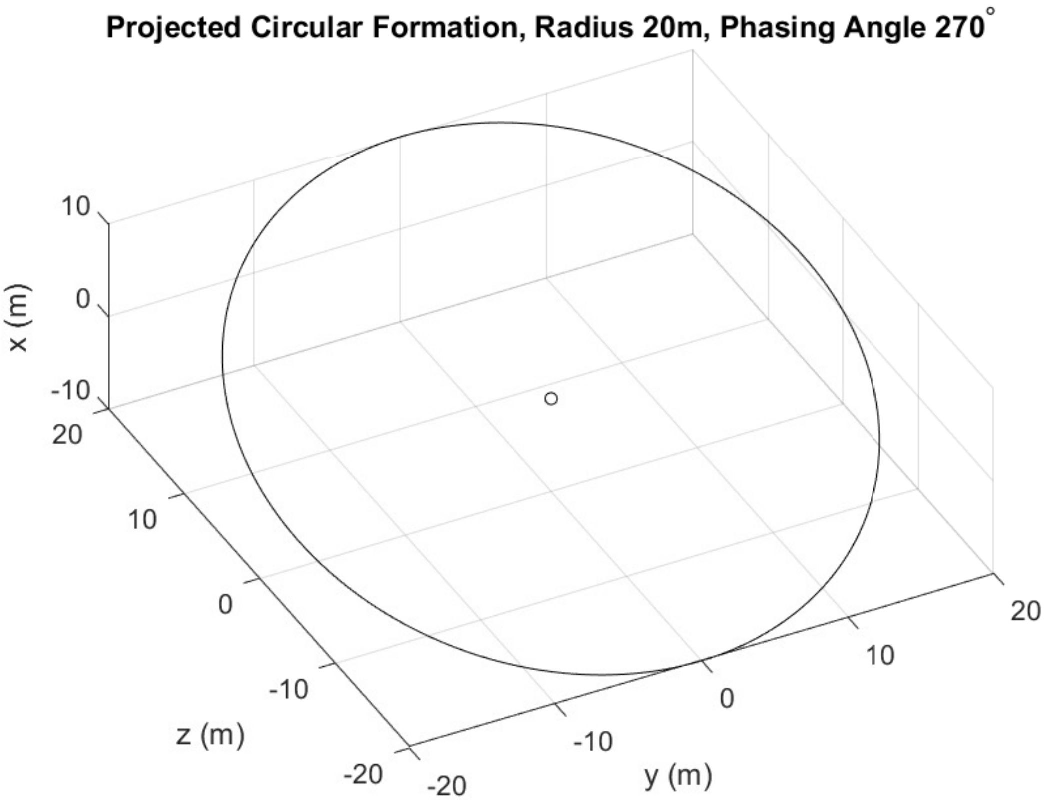

Figure 24: 3D-plot of path around target (projected circular formation)

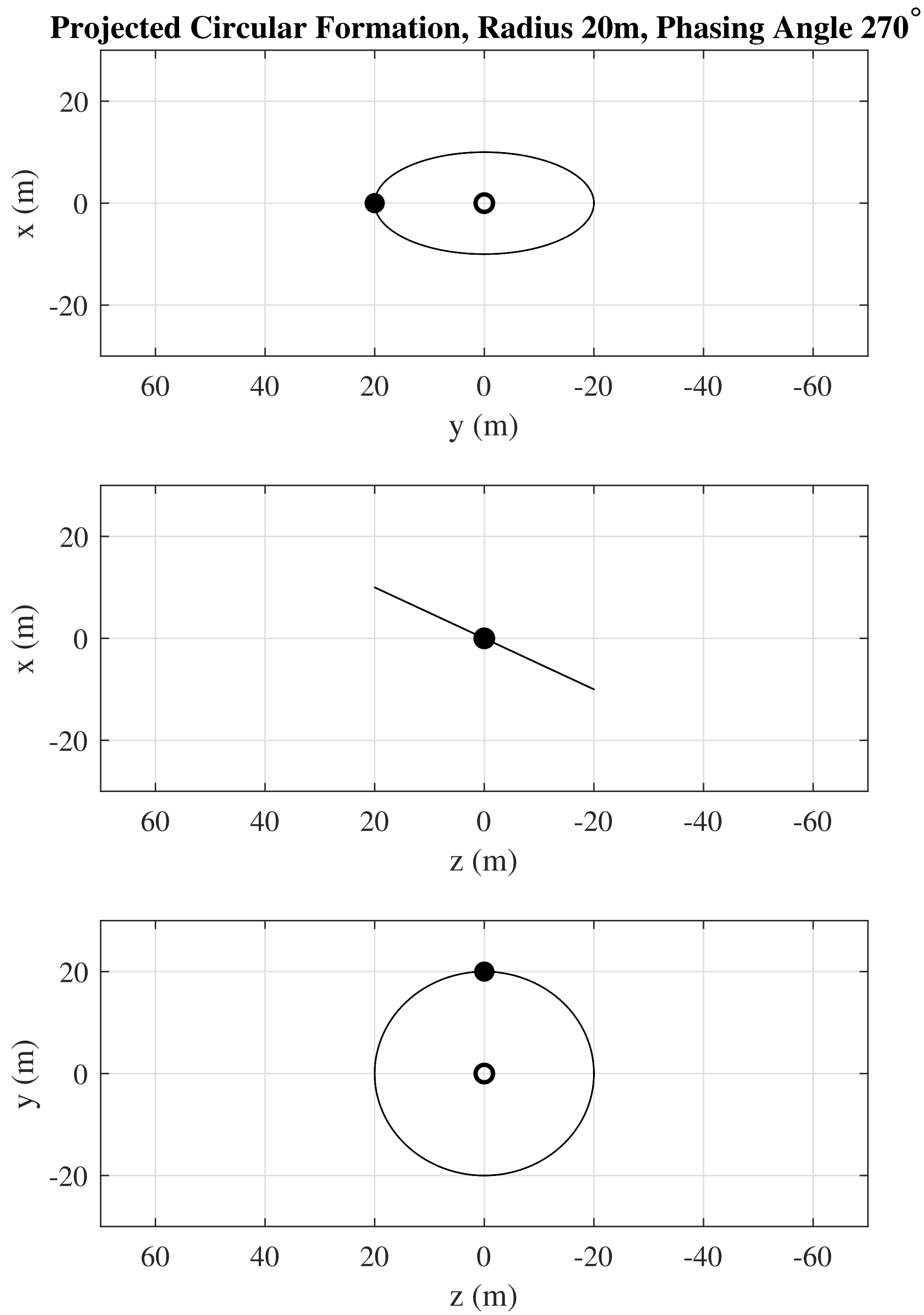

Figure 25: Path in projected planes (projected circular formation)

Appendix A MATLAB code for 2.5

```
clc
close all;
clear;

%constants
R_earth=6.378E3;
mu=3.986E5;
altitude = 150:1:1000;

r = altitude +R_earth; % define radius

theta_dot = sqrt(mu.*(r.^-3));  % calculate mean motion

T = (2*pi).*(theta_dot.^-1); %calculate periods
v = r.*theta_dot; %calculate velocity


%% Plot Figures
figure(1)
plot(altitude,v)
title("Orbit Altitude vs Speed (Circular Orbit)")
xlabel("Altitude(km)")
ylabel("Inertial speed(km/s)")
print('C:\Users\boaza\OneDrive - Carleton University\AERO 3240\output
↪  files\2.5\Orbit_Altitude_vs_Velocity','-depsc')

figure(2)
plot(altitude,T)
title("Orbit Altitude vs period (Circular Orbit)")
xlabel("Altitude(km)")
ylabel("Period(s)")
print('C:\Users\boaza\OneDrive - Carleton University\AERO 3240\output
↪  files\2.5\Orbit_Altitude_vs_period','-depsc')
```

Code Attachment 1: Script used in question 2.5

Appendix B `MATLAB` Code for 2.13

B.1 General Script

```
clc
clear all;
close all;

% save folder
mydir  = pwd;
idcs   = strfind(string(mydir),'\');
newdir = mydir(1:idcs(end)-1);
save_to = strcat(newdir,'\output files\2.13\');
% Add general Functions
addpath(strcat(newdir,'\general functions'));
clear mydir idcs newdir;
% Gravitational parameter
mu = 398600.4418; % km^3/s^2

% Earth radius
rE = 6378; % km

% Conversion constants
d2r = pi/180; % rad/deg
r2d = 1/d2r; % deg/rad

% Orbital elements
a        = rE + 791; % km
e        = 0.001;
i        = 98.28; % deg
w        = 0; % deg
RAAN     = 270; % deg
tp       = 0; % sec

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

% Eccentric anomaly at t = 0 sec
eano   = 0; % deg

% True anomaly at t = 0 sec
tano = 0; % deg

% Magnitude of position vector at t = 0 sec
r = p/(1+e*cosd(tano)); % km

```

```
% Components of position and velocity vectors in perifocal at t = 0 sec
r_P_ini = r*[1 0 0]';

v_P_ini = [0 sqrt(mu/p)*(e+1) 0]';

% Rotation matrix from perifocal to ECI, i.e., C_IP

C_IP = (C_3(w)*C_1(i)*C_3(RAAN))'; % using functions I made  for rot matrices

% Components of position and velocity vectors in ECI at t = 0 sec
r_I_ini = C_IP*r_P_ini;
v_I_ini = C_IP*v_P_ini;

T = 2*pi*sqrt(a^3/mu);

% Simulation from t = 0 to t = T
open_system('PROBA2mdl.slx')
set_param('PROBA2mdl', 'StopTime', 'T')
disp('Running Simulation...')
sim('PROBA2mdl')

%% Part A
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot components of the position vector in Perifocal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% r_P through conversion of simulation variables to perifocal
r_P =C_IP'*r_I';
% find r_p through equation

% Orbit equation definition with 2d vector componenets
syms r_P_func(theta)
r_P_func(theta) = matlabFunction(p/(1+e*cos(theta))*[cos(theta);sin(theta)]);

%calculation of orbit through orbit equation
r_P_eq = r_P_func(0:0.01:2*pi);
r_P_eq = double(cell2sym(r_P_eq)); % convert to symbolic to number

figure
plot(r_P(1,:),r_P(2,:),'k','LineWidth',2);
title('PROBA 2 Orbital Path in Perifocal Reference Frame')
xlabel('rx_P (km)')
ylabel('ry_P (km)')
hold on
plot(r_P_eq(1,:),r_P_eq(2,:),'r','LineWidth',1);
legend('simulation','orbit equation','Location','best')
pbaspect([1 1 1])
```

```
daspect([1 1 1])
hold off

print(strcat(save_to,'perifocal_plot.eps'),'-depsc')

%% Part B
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tabulize r_I_ini and v_I_ini
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
varNames = ["Component", "r_I_inital", "v_I_inital"];
varTypes = ["string","double","double"];
sz = [3 3];
initial_conditions = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);
initial_conditions.r_I_inital = r_I_ini;
initial_conditions.v_I_inital = v_I_ini;
initial_conditions.Component = ["X";"Y";"Z"];
initial_conditions.Properties.VariableNames = ["Component","r_{I,inital}(km)", "v_{I,inital}(km/s)"];
writetable(initial_conditions,strcat(save_to,'initial_conditions.csv'))



%% Part C
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot components of the position vector in ECI in 3D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
plot3(r_I(:,1),r_I(:,2),r_I(:,3),'k','linewidth', 2)
grid on

hold on
% Adding Earth
% Using Will Campbell (2022). Earth-sized
% Sphere with Topography
% (https://www.mathworks.com/matlabcentral
% /fileexchange/27123-earth-sized-sphere-with-topography),
% MATLAB Central File Exchange. Retrieved November 28, 2022.
earth_sphere(gca,'km')

set(gca,'FontSize',9,'FontName', 'Times')
title('PROBA 2 Orbital Path in ECI Reference Frame')
xlabel('rx_I (km)')
ylabel('ry_I (km)')
zlabel('rz_I (km)')
axis equal
```

```
print(strcat(save_to,'3d_plot.png'),'-dpng','-r600')
%% part D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot components of the position vector as function of time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
subplot(3,1,1)
plot(t,r_I(:,1),'k');
title('PROBA 2 Position Vector Components in ECI Reference Frame')
xlabel('Time (sec)')
ylabel('rx_I (km)')
xlim([0 max(t)])

subplot(3,1,2)
plot(t,r_I(:,2),'k');
xlabel('Time (sec)')
ylabel('ry_I (km)')
xlim([0 max(t)])

subplot(3,1,3)
plot(t,r_I(:,3),'k');
xlabel('Time (sec)')
ylabel('rz_I (km)')
xlim([0 max(t)])


print(strcat(save_to,'position_components.eps'),'-depsc')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot components of the velocity vector as function of time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
subplot(3,1,1)
plot(t,v_I(:,1),'k');
title('PROBA 2 Velocity Vector Components in ECI Reference Frame')
xlabel('Time (sec)')
ylabel('vx_I (km/s)')
xlim([0 max(t)])

subplot(3,1,2)
plot(t,v_I(:,2),'k');
xlabel('Time (sec)')
```

```
ylabel('vy_I (km/s)')
xlim([0 max(t)])

subplot(3,1,3)
plot(t,v_I(:,3),'k');
xlabel('Time (sec)')
ylabel('vz_I (km/s)')
xlim([0 max(t)])

print(strcat(save_to,'velocity_components.eps'),'-depsc')

%% part E
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot magnitude of position and velocity vectors as function of time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r_I_mag = vecnorm(r_I,2,2);
v_I_mag = vecnorm(v_I,2,2);

times = 60*[30 60 90];

figure
subplot(2,1,1)
plot(t,r_I_mag,'k');
title('PROBA 2 Radius from Earth''s Center')
xlabel('Time (sec)')
ylabel('Radius (km)')
xlim([0 max(t)])
xline(times(1),'k',"30 mins")
xline(times(2),'k',"60 mins")
xline(times(3),'k',"90 mins")

subplot(2,1,2)
plot(t,v_I_mag,'k');
title('PROBA 2 Speed in ECI')
xlabel('Time (sec)')
ylabel('Speed (km/s)')
xlim([0 max(t)])
xline(times(1),'k',"30 mins")
xline(times(2),'k',"60 mins")
xline(times(3),'k',"90 mins")

% Vis viva equation:
syms r
v(r) = sqrt(mu*(2/r-1/a));

% Create check table
```

```
varNames =
↪  ["Time","Radius","vis_viva_calculated_speed","simulation_calculated_speed"];
varTypes = ["double","double","double","double"];
sz = [length(times) length(varNames)];
vis_viva_check =
↪  table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);

% Calculate and tabulate the vis-viva check table
for loop_var = 1:length(times)
    [useless_var,index] = min(abs(t-times(loop_var))); %find index of time
    radius_at_point = r_I_mag(index); % Get radius at time
    vis_viva_check.Radius(loop_var) = radius_at_point;

    % Calculate velocity based on given radius
    vis_viva_check.vis_viva_calculated_speed(loop_var) =
↪  double(v(radius_at_point));
    vis_viva_check.simulation_calculated_speed(loop_var) = v_I_mag(index);

    vis_viva_check.Time(loop_var) = times(loop_var)/60;
    clear useless_var;
end
vis_viva_check.Properties.VariableNames = ["Time (minutes)", ...
    "Radius from Simulation(km)", ...
    "Calculated Speed from Vis-Viva equation (km/s)",...
    "Speed from Simulation (km/s)"];

writetable(vis_viva_check,strcat(save_to,'vis_viva_check.csv'))
print(strcat(save_to,'radius_and_speed.eps'),'-depsc')

%% part F
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot Ground path
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wE = 15.04/3600; %converting 15.04 deg/hr to deg/s;

thetaE = wE.*t; %Array of theta_GMT

% Calculate r_F
r_F = cell2mat(arrayfun(@(i) r_I2r_F(thetaE(i), r_I(i, :)), ...
    1:length(thetaE), 'UniformOutput', false))';


% Calculate lattitude
lattitude = asind(r_F(:,3)./r_I_mag);

% Calculate longitude and place in correct quadrant
```

```
negate_angle = (le(r_F(:,2), 0) -0.5).*-2;
longitude = negate_angle.*acosd(r_F(:,1)./((r_I_mag.*cosd(lattitude))));

% Remove horizontal lines
dont_connect = find(diff(longitude) > 250);
for loop_var = 1:length(dont_connect)
    index = dont_connect(loop_var) +loop_var -1;
    longitude =
    ↪  [longitude(1:index);NaN;longitude((index+1):(length(longitude)))];
    lattitude =
    ↪  [lattitude(1:index);NaN;lattitude((index+1):(length(lattitude)))];
end

figure
plot(longitude, lattitude,'k','LineWidth',2);
ylim([-90 90])
xlim([-180 180])
title('PROBA 2 Ground Track')
xlabel('Longitude(^\circ)')
ylabel('Lattitude(^\circ)')
hold on
I = imread('map.png');
h = image(xlim,-ylim,I);
uistack(h,'bottom')

print(strcat(save_to,'ground_tracks.eps'),'-depsc')
```

Code Attachment 2: Script used in question 2.13

B.2 SIMULINK diagram

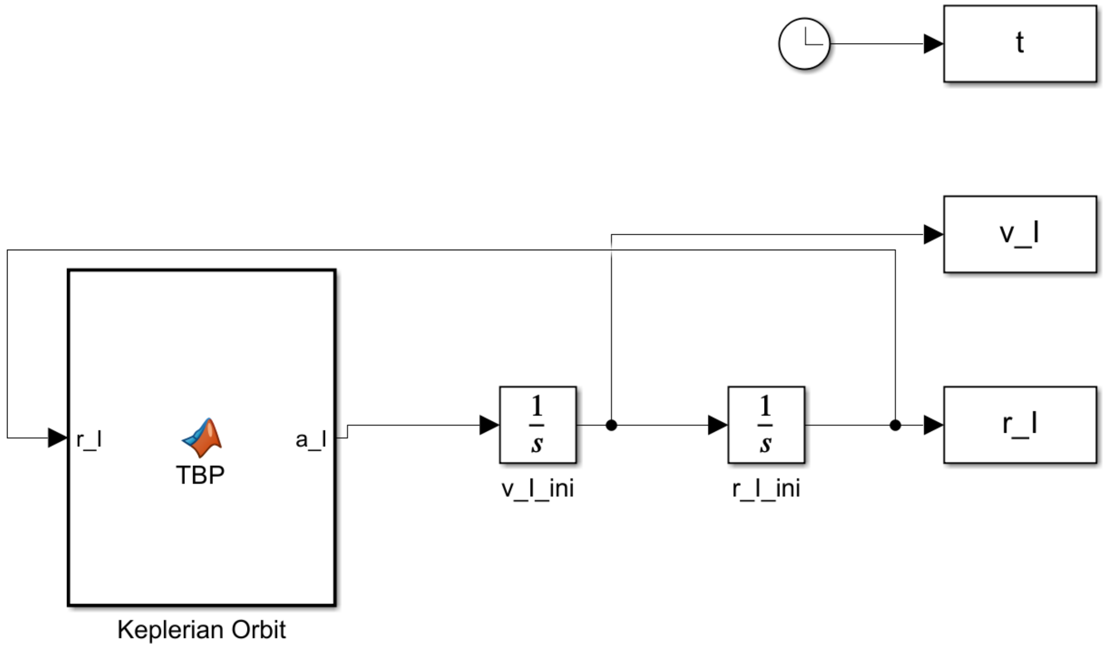

Figure 26: SIMULINK model for question 2.13

B.3 additional functions

B.3 a TBP function

```
function a_I  =  TBP(r_I, mu)
% ------------------------------------------------------------
% TBP
% ------------------------------------------------------------
% Description:
% Compute the spacecraft acceleration in ECI, using the two-body
% equation of motion.
% ------------------------------------------------------------
% Inputs:
% r_I => components of r in ECI (3 x 1 matrix)
% ------------------------------------------------------------
% Outputs:
% a_I => components of the acceleration in ECI (3 x 1 matrix)
% ------------------------------------------------------------
```

```
% Parameters:
% mu => gravitational constant of the Earth
% ---------------------------------------------------------------
% Copyright:
% Steve Ulrich, 2014
% ---------------------------------------------------------------

% Compute orbital radius
rmod = sqrt( r_I(1)*r_I(1) + r_I(2)*r_I(2) + r_I(3)*r_I(3) );

% Spacecraft acceleration
a_I = -mu/rmod^3*r_I;
end
```

Code Attachment 3: Function to calculate a_I in SIMULINK model

B.3 b ECI to ECEF function

```
function r_F_vector = r_I2r_F(rotation_angle,r_I_vector)
%r_I2r_F Rotates r_I_vector about z axis by specified rotation_angle in degrees
r_F_vector = C_3(rotation_angle)*(r_I_vector');
end
```

Code Attachment 4: Function used to convert vector from ECIF to ECEF

Appendix C MATLAB code for 3.1

C.1 general script

```
clc
clear;
close all;

% save folder
mydir  = pwd;
idcs   = strfind(string(mydir),'\');
newdir = mydir(1:idcs(end)-1);
save_to = strcat(newdir,'\output files\3.1\');
% Add general Functions
addpath(strcat(newdir,'\general functions'));
clear mydir idcs newdir;
% J2 constant
```

```
J_2 = 1.08264E-3;

% Gravitational parameter
mu = 398600.4418; % km^3/s^2

% Earth radius
rE = 6378; % km

% Conversion constants
d2r = pi/180; % rad/deg
r2d = 1/d2r; % deg/rad

% Orbital elements
a       = rE + 791; % km
e       = 0.001;
i       = 98.28; % deg
w       = 0; % deg
RAAN    = 270; % deg
tp      = 0; % sec

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

% Eccentric anomaly at t = 0 sec
eano   = 0; % deg

% True anomaly at t = 0 sec
tano = 0; % deg

% Magnitude of position vector at t = 0 sec
r = p/(1+e*cosd(tano)); % km

% Components of position and velocity vectors in perifocal at t = 0 sec
r_P_ini = r*[1 0 0]';

v_P_ini = [0 sqrt(mu/p)*(e+1) 0]';

% Rotation matrix from perifocal to ECI, i.e., C_IP

C_IP = (C_3(w)*C_1(i)*C_3(RAAN))'; % using functions I made  for rot matrices

% Components of position and velocity vectors in ECI at t = 0 sec
r_I_ini = C_IP*r_P_ini;
v_I_ini = C_IP*v_P_ini;

orbits = 1;
T = 2*pi*sqrt(a^3/mu)*orbits;
```

```
% Simulation from t = 0 to t = T
open_system('PROBA2mdl_Chapter3.slx')
set_param('PROBA2mdl_Chapter3', 'StopTime', 'T')
disp('Running Simulation...')
sim('PROBA2mdl_Chapter3')

% Fix for simulink adding a dimension
v_I = squeeze(v_I)';
r_I = squeeze(r_I)';

%% Part A
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate secular change in RAAN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RAAN_secular_change = -((3*pi*J_2*rE^2)/(p^2))*cosd(i)*orbits;% rad
RAAN_secular_change = RAAN_secular_change*r2d; % Convert to degrees
RAAN_average_change = RAAN_secular_change/T; % Average change of RAAN (deg/s)
%% Part C
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate and plot RAAN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h = cross(r_I,v_I); % Calculate angular momentum
N = cross(repmat([0 0 1],length(h),1),h); % Obtain vector pointing at RAAN

RAAN_Array = atan2d(N(:,2),N(:,1)); % find RAAN angle in degrees
RAAN_Array = le(RAAN_Array, 0)*360 + RAAN_Array; % add 360 if less than 0


figure
hold off

% Plot actual RAAN
plot(t, RAAN_Array)

%plot expected change based on secular change
hold on
plot([0,max(t)],[RAAN, RAAN+RAAN_secular_change])

% Graph formatting and saving
legend('Simulated RAAN', 'expected change in RAAN','Location','best')
xlabel('time (s)')
ylabel('RAAN (^o)')
```

```
title(strcat("PROBA-2 RAAN with J_2 Pertubations at ",num2str(i),'^{\circ} Inclined Orbit'))
xlim("tight")
ylim("tight")
print(strcat(save_to,'RAAN_',num2str(i),'_degrees.eps'), '-depsc')


%% Part D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate simulated secular and rate of change in RAAN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate change in RAAN from simulation
simulated_RAAN_secular_change = ...
    (RAAN_Array(length( RAAN_Array)) -  RAAN_Array(1))/orbits;

% Calculated  rate of change change in RAAN from simulation
simulated_RAAN_average_rate_of_change = ...
    simulated_RAAN_secular_change/(T/orbits);

%% Part E
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tabulize expected and simulated secular and rate of change in RAAN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate differences
RAAN_secular_difference = abs(RAAN_secular_change-simulated_RAAN_secular_change);
RAAN_roc_difference = abs(RAAN_average_change-simulated_RAAN_average_rate_of_change);
varNames = ["secular_change", "rate_of_change"];
varTypes = repmat("double", 2,1);
sz = [3 2];
comparison_table = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);

comparison_table.secular_change = ...
[RAAN_secular_change;simulated_RAAN_secular_change;RAAN_secular_difference];

comparison_table.rate_of_change = ...
[RAAN_average_change; simulated_RAAN_average_rate_of_change; RAAN_roc_difference];

comparison_table.Properties.VariableNames = ["secular change (degrees/rev)",...
    "average rate of change(degrees/s)"];
comparison_table.Properties.RowNames = ["expected","simulation", "difference"];

```

```
writetable(comparison_table,strcat(save_to,'RAAN_check_for_inc_',num2str(i),'.csv'),'WriteRowNames',
```

Code Attachment 5: Script used in question 3.1

C.2 SIMULINK diagram

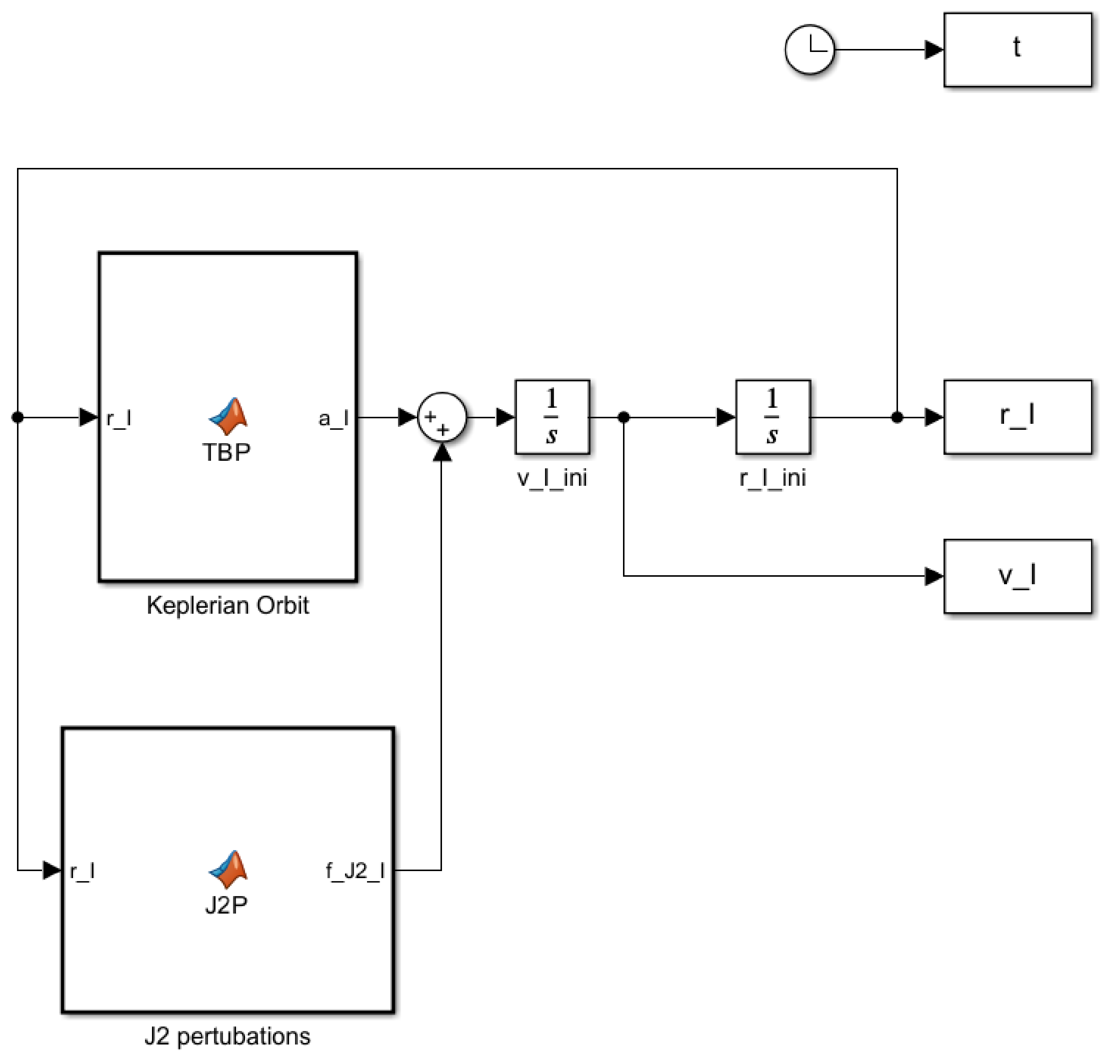

Figure 27: SIMULINK model for question 3.1

C.3 additional functions

Refer to code attachment 3 for TBP function used in Figure 27.

C.3 a J_2 Perturbation acceleration function

```
function f_J2_I =  J2P(rE, J_2, mu, r_I)

% ---------------------------------------------------------------
% J2P
% ---------------------------------------------------------------
% Description:
% Compute the spacecraft acceleration due to J2 perturbation ECI
% ---------------------------------------------------------------
% Inputs:
% r_I => components of r in ECI (3 x 1 matrix)
% ---------------------------------------------------------------
% Outputs:
% f_J2_I => components of the acceleration in ECI (3 x 1 matrix)
% ---------------------------------------------------------------
% Parameters:
% mu => gravitational constant of the Earth
% J_2 => J2 coefficient for oblate body of earth
% rE => Radius of earth
% ---------------------------------------------------------------

% J_2 equation
f_J2_I = 3*mu*J_2*rE^2/(2*(norm(r_I)^5))...
    *(...
    (5*(r_I(3)^2)/(norm(r_I)^2)-1)*r_I - 2*[0; 0; r_I(3)]...
    );

end
```

Code Attachment 6: Function to calculate J_2 perturbation in SIMULINK

Appendix D MATLAB code for 5.1

D.1 Main Script

```
clc
clear;
close all;
```

```
% save folder
mydir  = pwd;
idcs   = strfind(string(mydir),'\');
newdir = mydir(1:idcs(end)-1);
save_to = strcat(newdir,'\output files\5.1\');
clear mydir idcs newdir;

global n
ALONG_TRACK = 1;
IN_TRACK = 2;
IN_PLANE_ELIPTICAL = 3;
ORBIT_I = 4;
ORBIT_II = 5;

%%%%%%%%%%%%%%%%%%%%%
formation_type = ORBIT_I;
%%%%%%%%%%%%%%%%%%%%%
% Gravitational parameter
mu = 398600.4418; % km^3/s^2

% Earth's radius
rE = 6378; % km

% Orbital elements of Envisat
a = rE+766; % km
e = 1E-4;
i = 98.4; % deg
RAAN = 27.2; % deg
w = 71; % deg

% Earth angular velocity
wE = 72.922e-6; % rad/s

% Mean motion
n = sqrt(mu/a^3); % rad/s

% Orbital period
T = 2*pi/n ; % sec

% Initial conditions of relative motion
if formation_type == ALONG_TRACK
    % Calculate initial conditions of along track formation
    name = 'Along Track';
    x0 = 0; % km
```

```
    z0 = 0; % km
    vx0 = 0; % km
    vy0 = 0; % km
    vz0 = 0; % km

    y0 = -10; % km
elseif formation_type == IN_TRACK
    % Calculate initial conditions of in track formation
    name = 'In Track';
    x0 = 0; % km
    vx0 = 0; % km
    vy0 = 0; % km
    vz0 = 0; % km

    y0 = -10; % km
    z0 = -(wE/n*sind(i))*y0; % km
elseif formation_type == IN_PLANE_ELIPTICAL
    % Calculate initial conditions of in plane eliptical formation
    name = 'In Plane Eliptical';
    alpha = 0; % deg
    A_0 = 0.5/2; %km

    z0 = 0; % km
    vz0 = 0; % km

    % solve in plane eliptical equations
    syms vx0 x0
    eqns = [
        alpha == atand(-vx0/(n*x0));
        A_0 == sqrt((vx0/n)^2+x0^2)
        ];

    [vx0, x0] = vpasolve(eqns,[vx0 x0]);

    vx0 = round(double(vx0), 10);
    x0 =  round(double(x0), 10);
    vy0 = -2*n*x0; % km
    y0 = 2*vx0/n; % km
elseif formation_type == ORBIT_I
    % Calculate initial conditions of 5.1) E)I formation
    name = 'Circular Formation, Radius 20m, Phasing Angle 0^\circ';
    r = 20E-3; % km
    phase = 0;  % deg

    x0 = (r/2)*cosd(phase); %km
    vx0 = -(r*n/2)*sind(phase); %km
```

```
    z0 = sqrt(3)*x0; % km
    vz0 = sqrt(3)*vx0; % km

    vy0 = -2*n*x0; % km
    y0 = 2*vx0/n; % km
elseif formation_type == ORBIT_II
    % Calculate initial conditions of 5.1) E)II formation
    name = 'Projected Circular Formation, Radius 20m, Phasing Angle 270^\circ';
    r = 20E-3; % km
    phase = 270;  % deg

    x0 = (r/2)*cosd(phase); %km
    vx0 = -(r*n/2)*sind(phase); %km
    z0 = 2*x0; % km
    vz0 = 2*vx0; % km

    vy0 = -2*n*x0; % km
    y0 = 2*vx0/n; % km
end
% Naming of output graphs
formation_type = num2str(formation_type);

%creating table of inital conditions
initial_conditions = array2table([vx0 vy0 vz0 x0 y0 z0]);
varNames = ["vx0", "vy0", "vz0", "x0", "y0", "z0"];
initial_conditions.Properties.VariableNames = varNames;
writetable(initial_conditions,strcat(save_to,'initial_conditions_',formation_type,'.csv'))

% Integration
options=odeset('RelTol',1e-9,'AbsTol',1e-9);
[t,X]=ode45(@CWHdyn,[0:10:2*T],[vx0 vy0 vz0 x0 y0 z0]',options);

vx = X(:,1);
vy = X(:,2);
vz = X(:,3);
x = X(:,4);
y = X(:,5);
z = X(:,6);

% Plot components of the relative position vector as function of time
figure
subplot(4,1,1)
plot(t/T,x*1000,'k');
title(name)
xlabel('Number of orbits')
ylabel('x (m)')
```

```
grid on
subplot(4,1,2)
plot(t/T,y*1000,'k');
xlabel('Number of orbits')
ylabel('y (m)')
grid on
subplot(4,1,3)
plot(t/T,z*1000,'k');
xlabel('Number of orbits')
ylabel('z (m)')
grid on
subplot(4,1,4)
plot(t/T,sqrt((x*1000).^2+(y*1000).^2+(z*1000).^2),'k');
xlabel('Number of orbits')
ylabel('Distance (m)')
axis([0 2 0 40]);
grid on


print(strcat(save_to,'Position_component_plots_',formation_type,'.eps'),
↪  '-depsc') % save figure

% Plot the components of the relative position vector in 3D
figure
plot3(z*1000, y*1000, x*1000,'k');
if str2double(formation_type) == ALONG_TRACK
    hold on
    scatter3(z*1000, y*1000, x*1000,20,'k');
end
hold on
scatter3(0,0,0,20,'k');
xlabel('z (m)')
ylabel('y (m)')
zlabel('x (m)')
title(name)
grid on
set(gca,'XDir','reverse')

print(strcat(save_to,'3D-plot_of_path_around_target_',formation_type,'.eps'),
↪  '-depsc') % save figure

% Plot and animate the in-plane and out-of-plane motion
planes = figure
set(planes,'Position',[556 33 454 665])
subplot(3,1,1)
plot(y*1000,x*1000,'k');
```

```
ip(1) = line(y(1)*1000, x(1)*1000, 'Marker', '.', 'MarkerSize', 22, 'Color', 'k');
ip(2) = line(0, 0,'Marker', 'o', 'MarkerSize', 6, 'LineWidth', 1.5, 'Color', 'k' );
axis([-70 70 -30 30]);
set(gca,'XDir','reverse')
set(gca,'FontSize',10,'FontName', 'Times')
xlabel('y (m)')
ylabel('x (m)')
title(name)
grid on

subplot(3,1,2)
plot(z*1000,x*1000,'k');
axis([-70 70 -30 30]);
oop(1) = line(z(1)*1000, x(1)*1000, 'Marker', '.', 'MarkerSize', 22, 'Color', 'k');
oop(2) = line(0, 0, 'Marker', 'o', 'MarkerSize', 6, 'LineWidth', 1.5, 'Color', 'k');
set(gca,'XDir','reverse')
set(gca,'FontSize',10,'FontName', 'Times')
xlabel('z (m)')
ylabel('x (m)')
grid on

subplot(3,1,3)
plot(z*1000,y*1000,'k');
axis([-70 70 -30 30]);
ct(1) = line(z(1)*1000, y(1)*1000, 'Marker', '.', 'MarkerSize', 22, 'Color', 'k');
ct(2) = line(0, 0, 'Marker', 'o', 'MarkerSize', 6, 'LineWidth', 1.5, 'Color', 'k');
set(gca,'XDir','reverse')
set(gca,'FontSize',10,'FontName', 'Times')
xlabel('z (m)')
ylabel('y (m)')
grid on
print(strcat(save_to,'Path_in_projected_planes_',formation_type,'.eps'), '-depsc')

for i = 1:length(x)
    set(ip(1),  'XData', y(i)*1000, 'YData', x(i)*1000);
    set(oop(1), 'XData', z(i)*1000, 'YData', x(i)*1000);
    set(ct(1), 'XData', z(i)*1000, 'YData', y(i)*1000);
    drawnow
end
```

Code Attachment 7: Script used in question 5.1

D.2 Additional Functions

D.2 a Hill's Equation function

```
function X = CWHdyn(t,states)

global n

vx = states(1);
vy = states(2);
vz = states(3);
x = states(4);
y = states(5);
z = states(6);

% Hill's equations of motion
x_ddot = 3*n^2*x + 2*n*vy;
y_ddot = -2*n*vx;
z_ddot = -n^2*z;
x_dot  = vx;
y_dot  = vy;
z_dot  = vz;

X = [x_ddot y_ddot z_ddot x_dot y_dot z_dot]';

end
```

Code Attachment 8: Function used to set up Hill's equation

Appendix E General MATLAB Functions

E.1 Rotation matrix functions

E.1 a rotation around x

```
function rotation_matrix_x = C_1(rotation_angle)
%C_1 roatates by given angle in degrees around x axis

rotation_matrix_x = [1          0                       0;
```

```
                     0 cosd(rotation_angle) sind(rotation_angle);
                     0 -sind(rotation_angle) cosd(rotation_angle)
    ];
end
```

Code Attachment 9: Function to generate a rotation matrix around the x axis

E.1 b rotation around y

```
function rotation_matrix_y = C_2(rotation_angle)
%C_2 roatates by given angle in degrees around y axis

rotation_matrix_y = [cosd(rotation_angle) 0 -sind(rotation_angle);
                     0                    1                  0   ;
                     sind(rotation_angle) 0 cosd(rotation_angle)
    ];
end
```

Code Attachment 10: Function to generate a rotation matrix around the y axis

E.1 c rotation around z

```
function rotation_matrix_z = C_3(rotation_angle)
%C_3 roatates by given angle in degrees around z axis

rotation_matrix_z = [cosd(rotation_angle) sind(rotation_angle) 0;
                     -sind(rotation_angle) cosd(rotation_angle) 0;
                         0                      0              1
    ];
end
```

Code Attachment 11: Function to generate a rotation matrix around the z axis

E.2 Animation functions

E.2 a Video Animate 3-D orbit

```
%The following code must be run in matlab live after calculating r_I in
%PROBA2.m

```

```
figure

grid on

earth_sphere(gca,'km')

set(gca,'FontSize',9,'FontName', 'Times')
title('PROBA 2 Orbital Path in ECI Reference Frame')
xlabel('rx_I (km)')
ylabel('ry_I (km)')
zlabel('rz_I (km)')
axis equal
view([62 32])
for i = 1:500:length(r_I(:,1))
    hold on
    plot3(r_I(1:i,1),r_I(1:i,2),r_I(1:i,3),'k','linewidth', 2)
    pause(0.00001);
    drawnow;
end
plot3(r_I(:,1),r_I(:,2),r_I(:,3),'k','linewidth', 2)
drawnow;
```

Code Attachment 12: Script to generate orbit animation video

E.2 b Video Animate ground tracks

```
%The following code must be run in matlab live after calculating longitude
% and lattitude in PROBA2.m

figure
plot(longitude(1), lattitude(1),'k','LineWidth',2);
hold on


ylim([-90 90])
xlim([-180 180])
title('PROBA 2 Ground Track')
xlabel('longitude(^\circ)')
ylabel('lattitude(^\circ)')
    I = imread('map.png');
    image(xlim,-ylim,I);
dt = [diff(t, 500);0];
for i = 1:1000:length(longitude)
    hold on
    plot(longitude(1:i), lattitude(1:i),'k','LineWidth',2);
```

```
    drawnow;
    pause(1/1000);


end
    plot(longitude, lattitude,'k','LineWidth',2);
drawnow;
```

Code Attachment 13: Script to generate ground tracks animation video