

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

2	Two-Body Problem	5
2.5	Speed and Periods of Circular LEO	5
2.13	PROBA-2 Without Perturbations	6
2.13 a	Plot of Orbit in the Orbital Plane	6
2.13 b	Initial Conditions For SIMULINK Integrators	7
2.13 c	3D plot of Orbit	8
2.13 d	Components of Position and Velocity in ECIF	9
2.13 e	Radius and Speed in ECIF	10
2.13 f	Ground Plots	12
3	Orbital Pertubations	13
3.1	PROBA-2 with Pertubations	13
3.1 a	Calculated Secular and Average Rate of Change	14
3.1 b	Simulating with J_2	14
3.1 c	Plot of Change in RAAN	14
3.1 d	Interpretation of Change in RAAN from Simulation	16
3.1 e	Comparison of Expected and Simulated Secular and Rate of Change in RAAN	16
3.1 f	Repeat for 10° Inclination	17
5	Spacecraft Formation Flying	17
5.1	Envisat Formation	17
5.1 f	Confirmation of Answers with MATLAB	17
5.1 f i	Along-Track Formation	17
5.1 f ii	In-Track Formation	21
5.1 f iii	In-Plane Elliptical Formation	24
5.1 f iv	Circular Formation	27
5.1 f v	Projected Circular Formation	30

Appendix A	MATLAB code for 2.5	33
Appendix B	MATLAB Code for 2.13	34
B.1	General Script	34
B.2	SIMULINK diagram	41
B.3	additional functions	41
B.3 a	TBP function	41
B.3 b	ECI to ECEF function	42
Appendix C	MATLAB code for 3.1	42
C.1	general script	42
C.2	SIMULINK diagram	46
C.3	additional functions	47
C.3 a	J_2 Perturbation acceleration function	47
Appendix D	MATLAB code for 5.1	47
D.1	Main Script	47
D.2	Additional Functions	53
D.2 a	Hill's Equation function	53
Appendix E	General MATLAB Functions	53
E.1	Rotation matrix functions	53
E.1 a	rotation around x	53
E.1 b	rotation around y	54
E.1 c	rotation around z	54
E.2	Animation functions	54
E.2 a	Video Animate 3-D orbit	54
E.2 b	Video Animate ground tracks	55

List of Figures

1	Orbital Speed vs Altitude	5
2	Orbital Period vs Altitude	6
3	PROBA-2 Orbit in Perifocal Frame	7
4	3-D Plot of PROBA-2 Orbit Around Earth	8
5	PROBA-2 Components of Position	9
6	PROBA-2 Components of Velocity	10
7	PROBA-2 Radius and Speed	11
8	PROBA-2 ground tracks	13
9	PROBA-2 Change in RAAN at 98.28° inclination	15
10	PROBA-2 Change in RAAN at 10° inclination	16
11	Position component plots (along-track formation)	18
12	3D-plot of path around target (along-track formation)	19
13	Path in projected planes (along-track formation)	20
14	Position component plots (in-track formation)	21
15	3D-plot of path around target (in-track formation)	22
16	Path in projected planes (in-track formation)	23
17	Position component plots (in-plane elliptical formation)	24
18	3D-plot of path around target (in-plane elliptical formation)	25
19	Path in projected planes (in-plane elliptical formation)	26
20	Position component plots (circular formation)	27
21	3D-plot of path around target (circular formation)	28
22	Path in projected planes (circular formation)	29
23	Position component plots (projected circular formation)	30
24	3D-plot of path around target (projected circular formation)	31
25	Path in projected planes (projected circular formation)	32
26	SIMULINK model for question 2.13	41
27	SIMULINK model for question 3.1	46

List of Tables

1	Table of Initial PROBA-2 Position and Velocity	8
2	Table of comparison between simulation and vis-viva results	12
3	Table of comparison between simulation and calculated RAAN results for inclination of 98.28°	17

4	Table of comparison between simulation and calculated RAAN results for inclination of 10°	17
5	Table of initial formation conditions (along-track formation)	18
6	Table of initial formation conditions (in-track formation)	21
7	Table of initial formation conditions (in-plane elliptical formation) . .	24
8	Table of initial formation conditions (circular formation)	27
9	Table of initial formation conditions (projected circular formation) . .	30

List of MATLAB Code Attachments

1	Script used in question 2.5	33
2	Script used in question 2.13	40
3	Function to calculate a_I in SIMULINK model	42
4	Function used to convert vector from ECIF to ECEF	42
5	Script used in question 3.1	46
6	Function to calculate J_2 perturbation in SIMULINK	47
7	Script used in question 5.1	53
8	Function used to set up Hill's equation	53
9	Function to generate a rotation matrix around the x axis	54
10	Function to generate a rotation matrix around the y axis	54
11	Function to generate a rotation matrix around the z axis	54
12	Script to generate orbit animation video	55
13	Script to generate ground tracks animation video	56

2 Two-Body Problem

2.5 Speed and Periods of Circular LEO

Created using code attachment 1 in Appendix A.

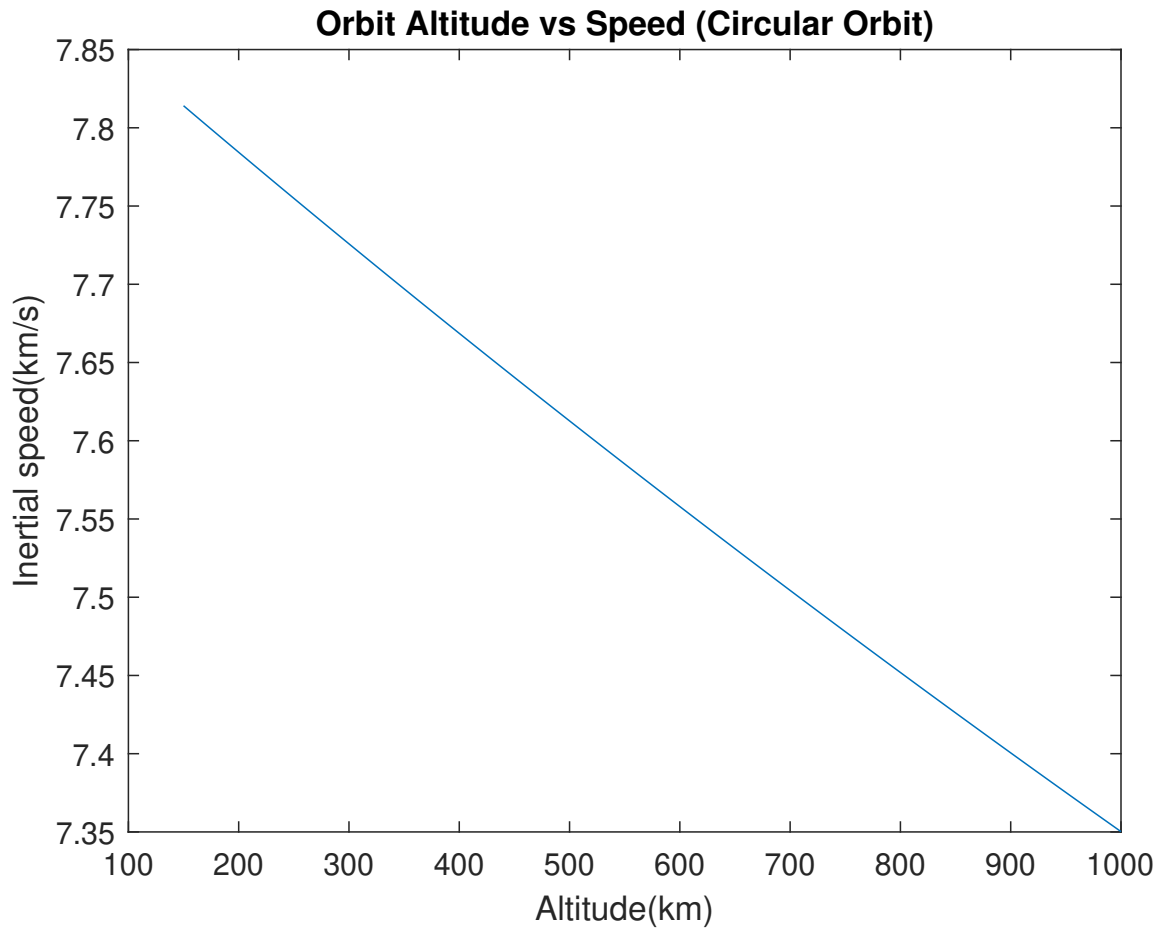


Figure 1: Orbital Speed vs Altitude

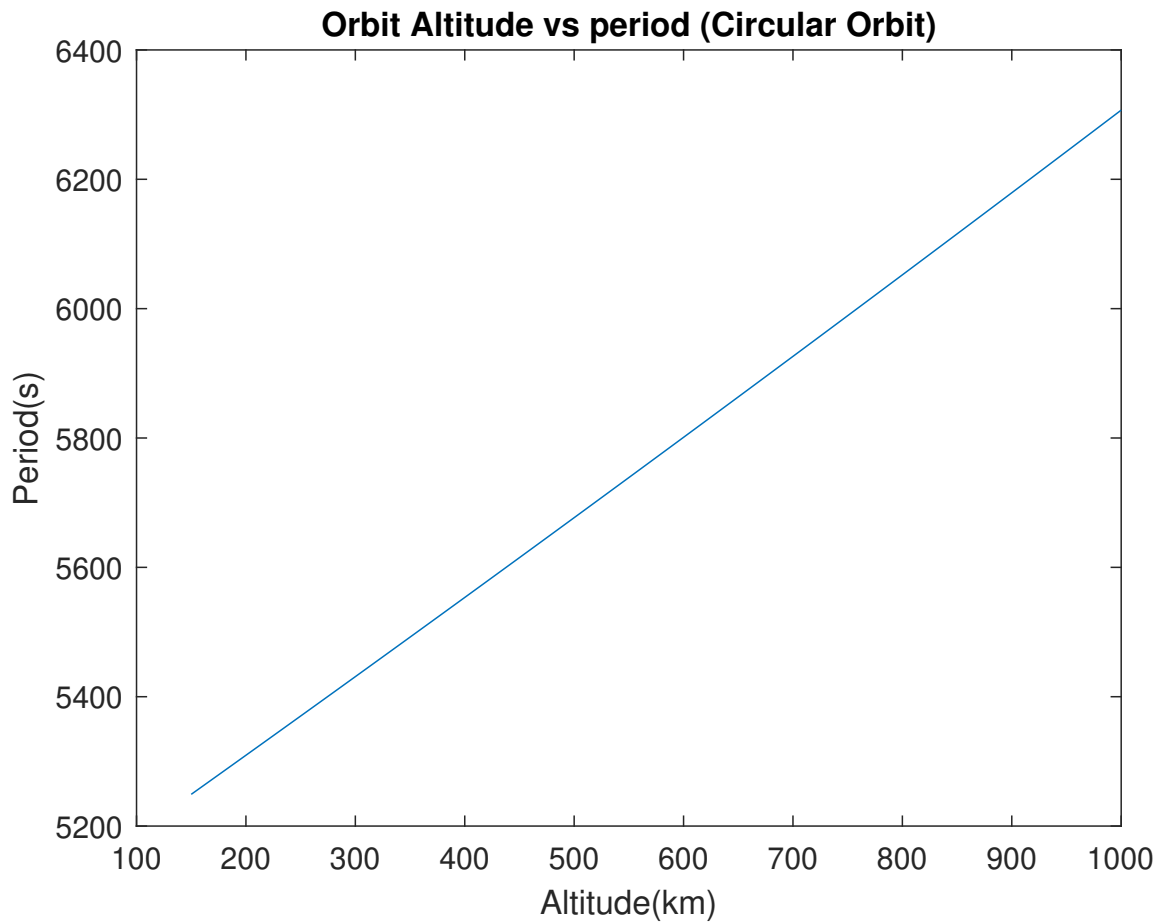


Figure 2: Orbital Period vs Altitude

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

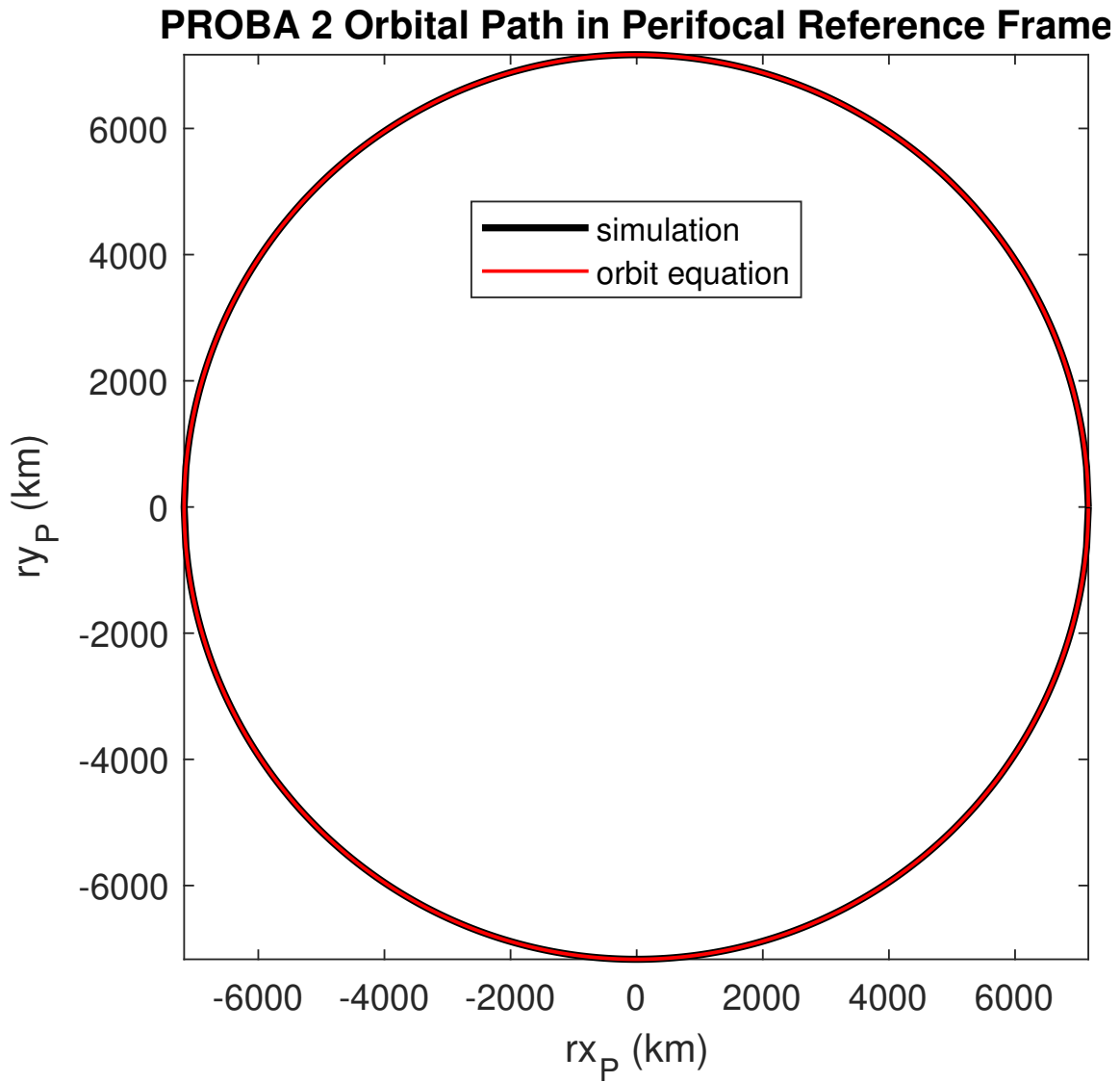


Figure 3: PROBA-2 Orbit in Perifocal Frame

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	$\mathbf{r}_{I,initial}(\text{km})$	$\mathbf{v}_{I,initial}(\text{km/s})$
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

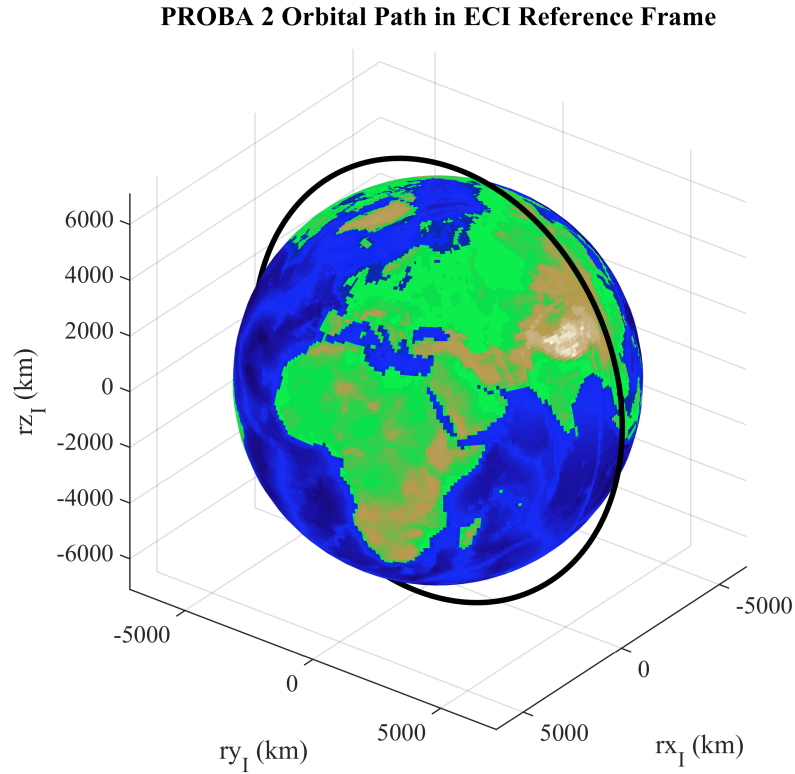


Figure 4: 3-D Plot of PROBA-2 Orbit Around Earth

2.13 d Components of Position and Velocity in ECIF

Created using code attachment 2 after line 139

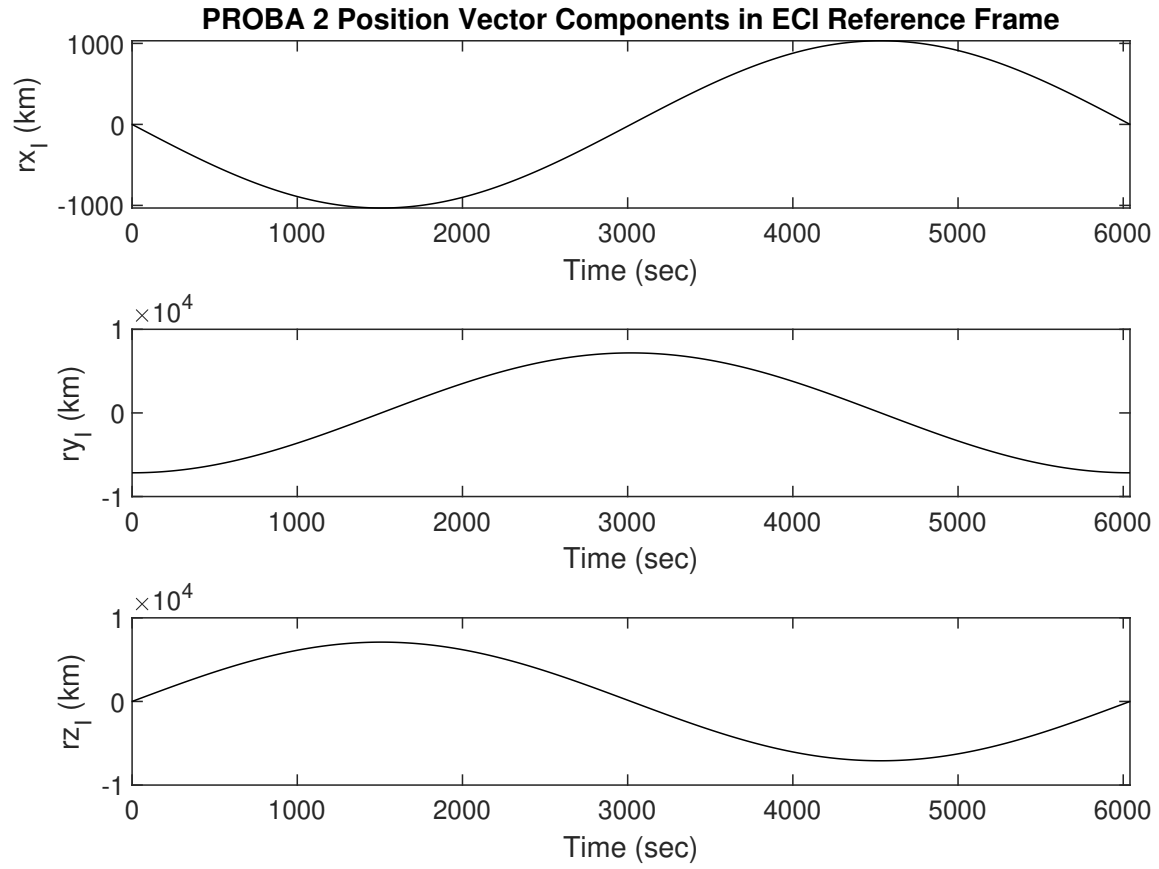


Figure 5: PROBA-2 Components of Position

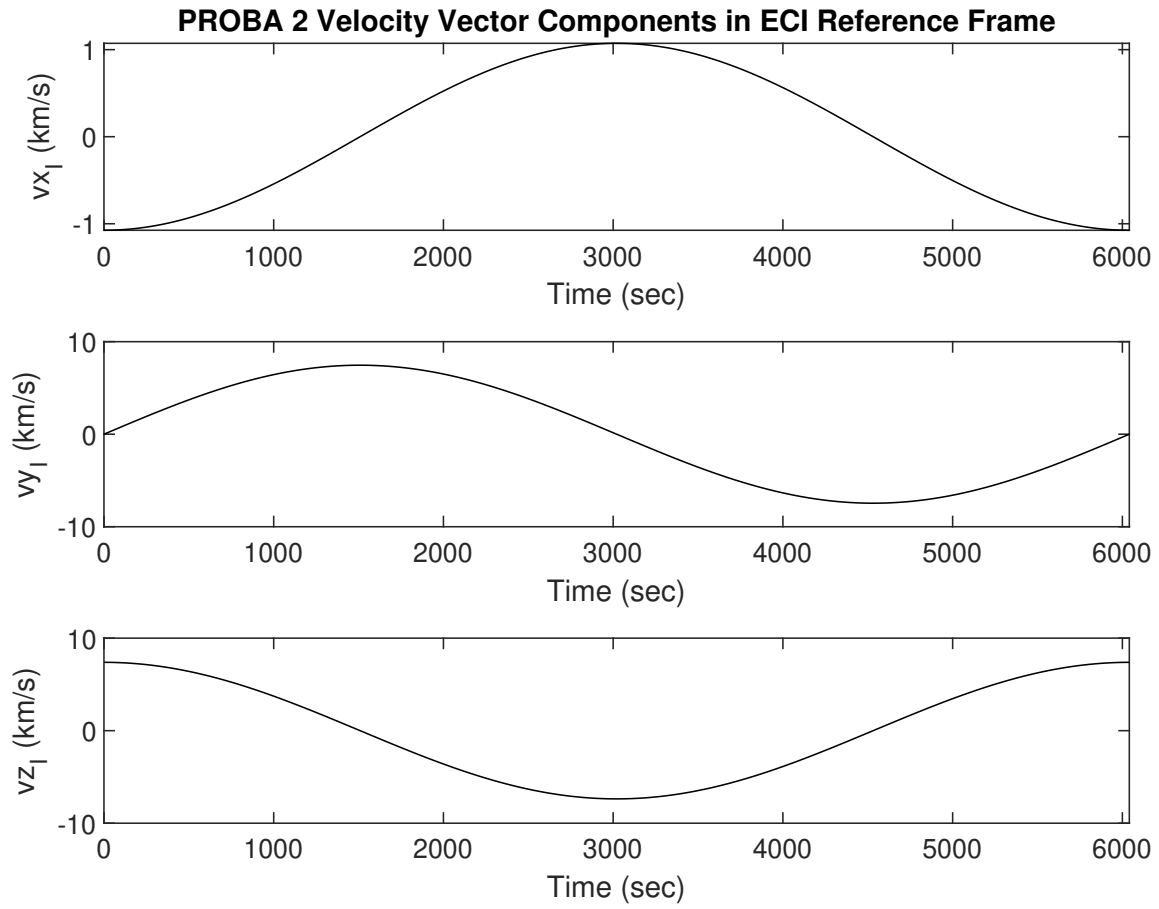


Figure 6: PROBA-2 Components of Velocity

2.13 e Radius and Speed in ECIF

Created using code attachment 2 after line 191

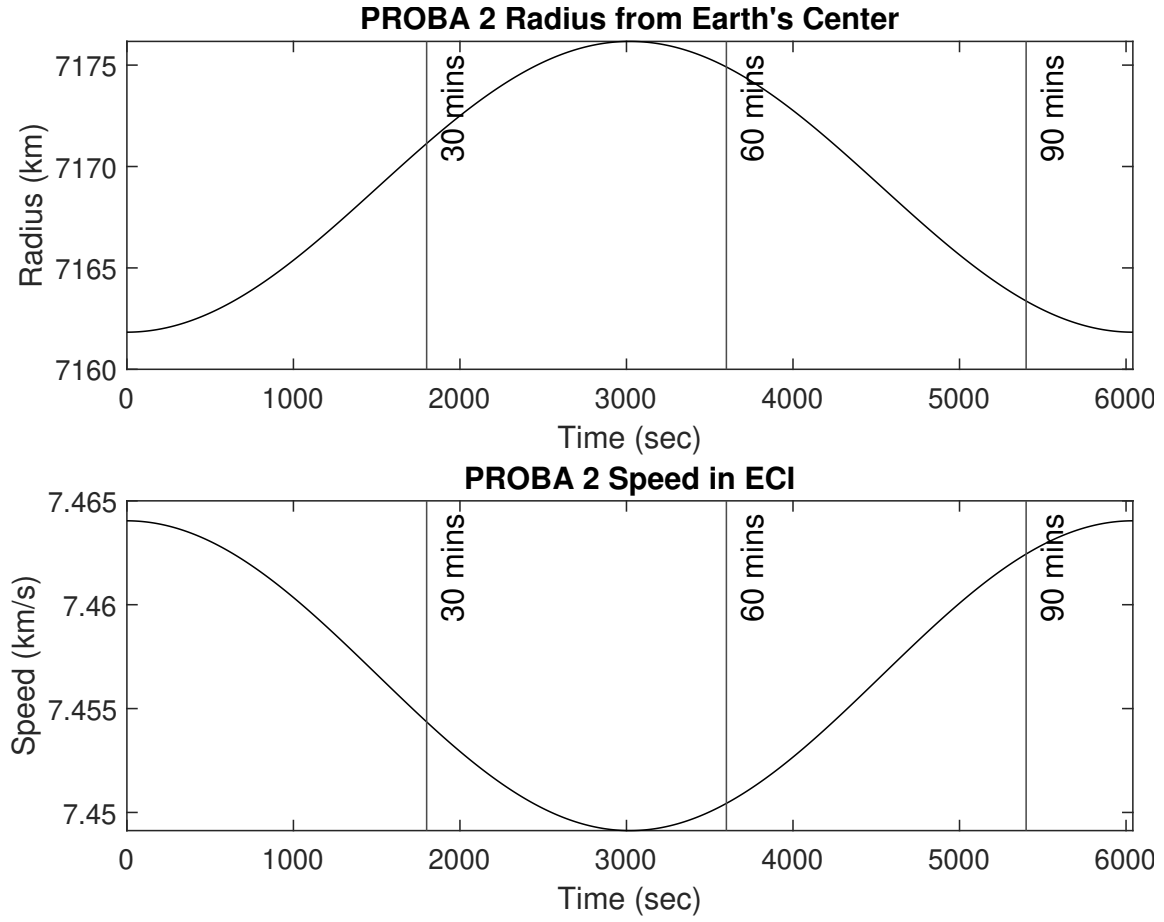


Figure 7: PROBA-2 Radius and Speed

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

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

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)	$r_I(\text{km})$	Calculated $v_I(\text{km/s})$	Simulated $v_I(\text{km/s})$
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:

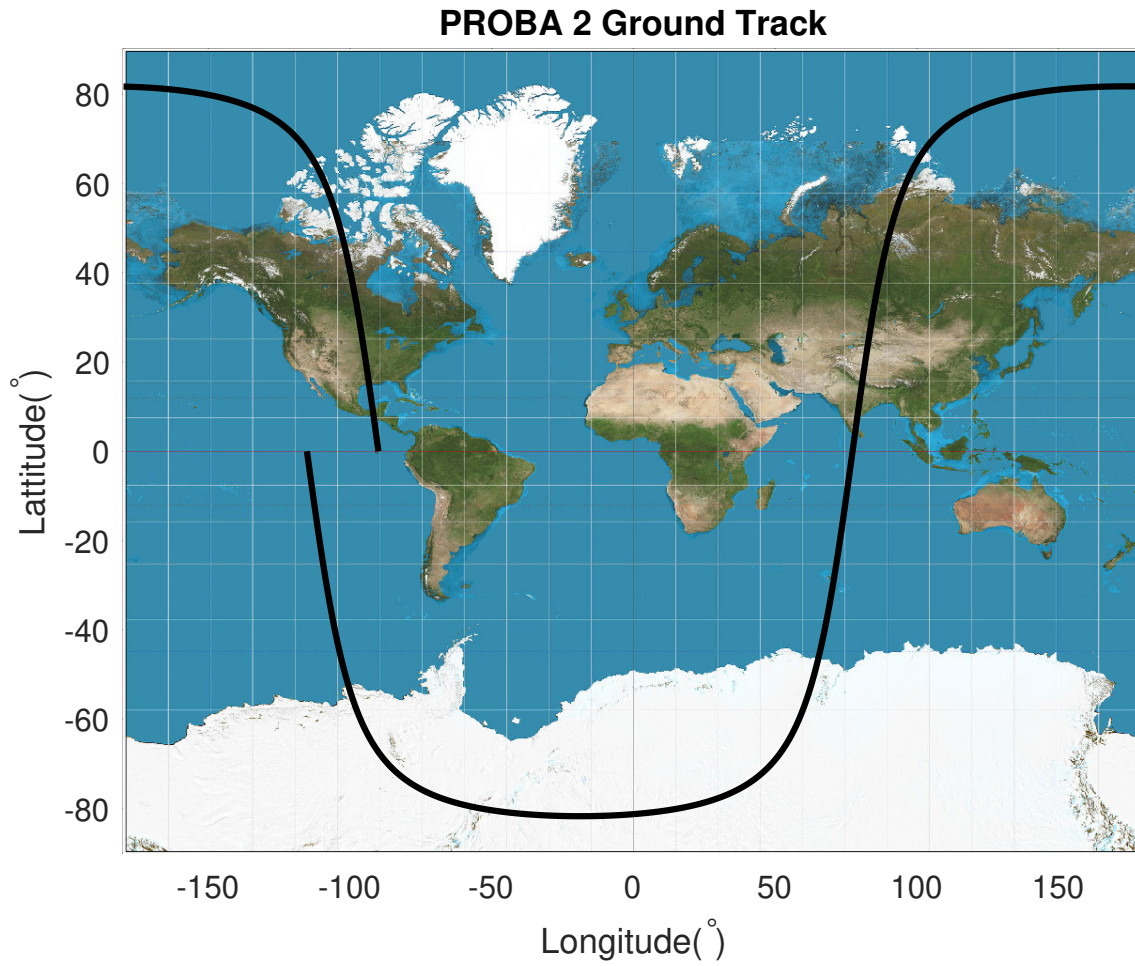


Figure 8: PROBA-2 ground tracks

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

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

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

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

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

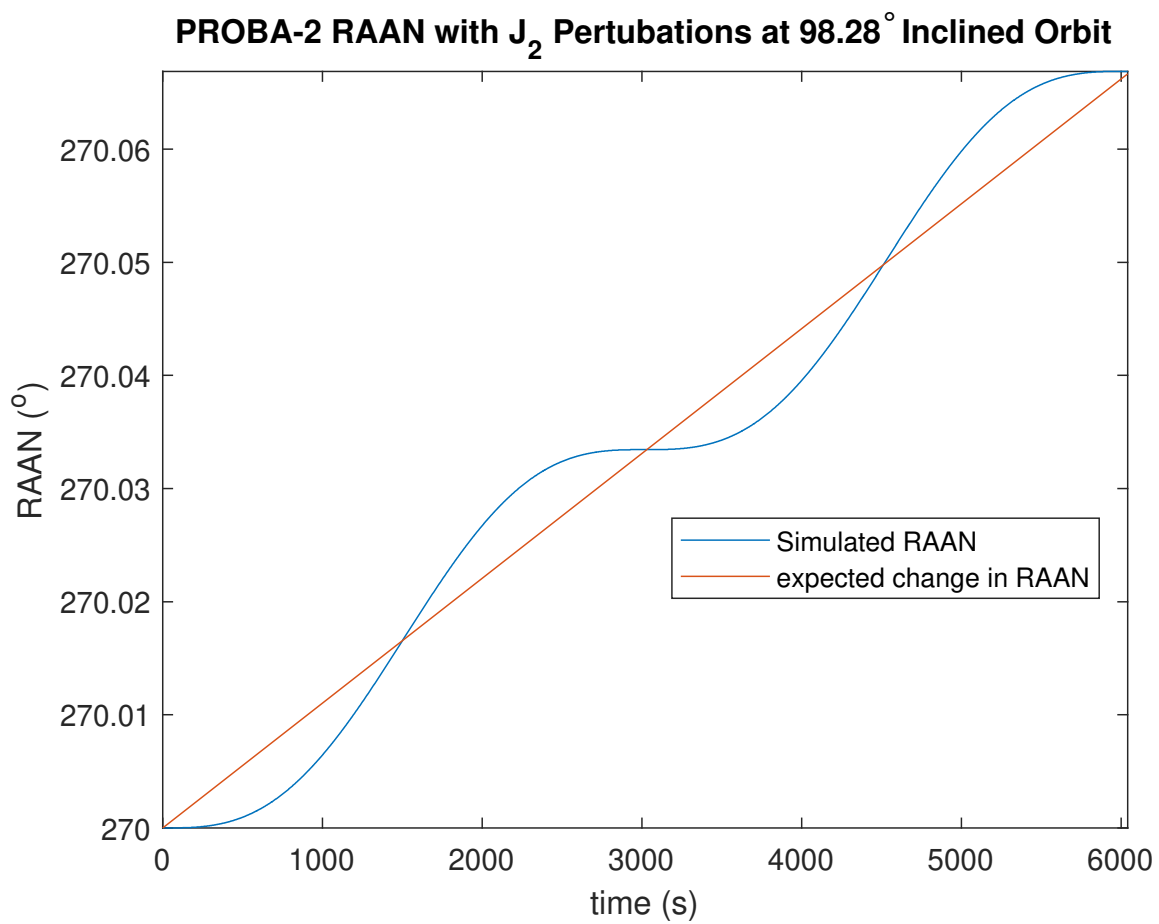


Figure 9: PROBA-2 Change in RAAN at 98.28° inclination

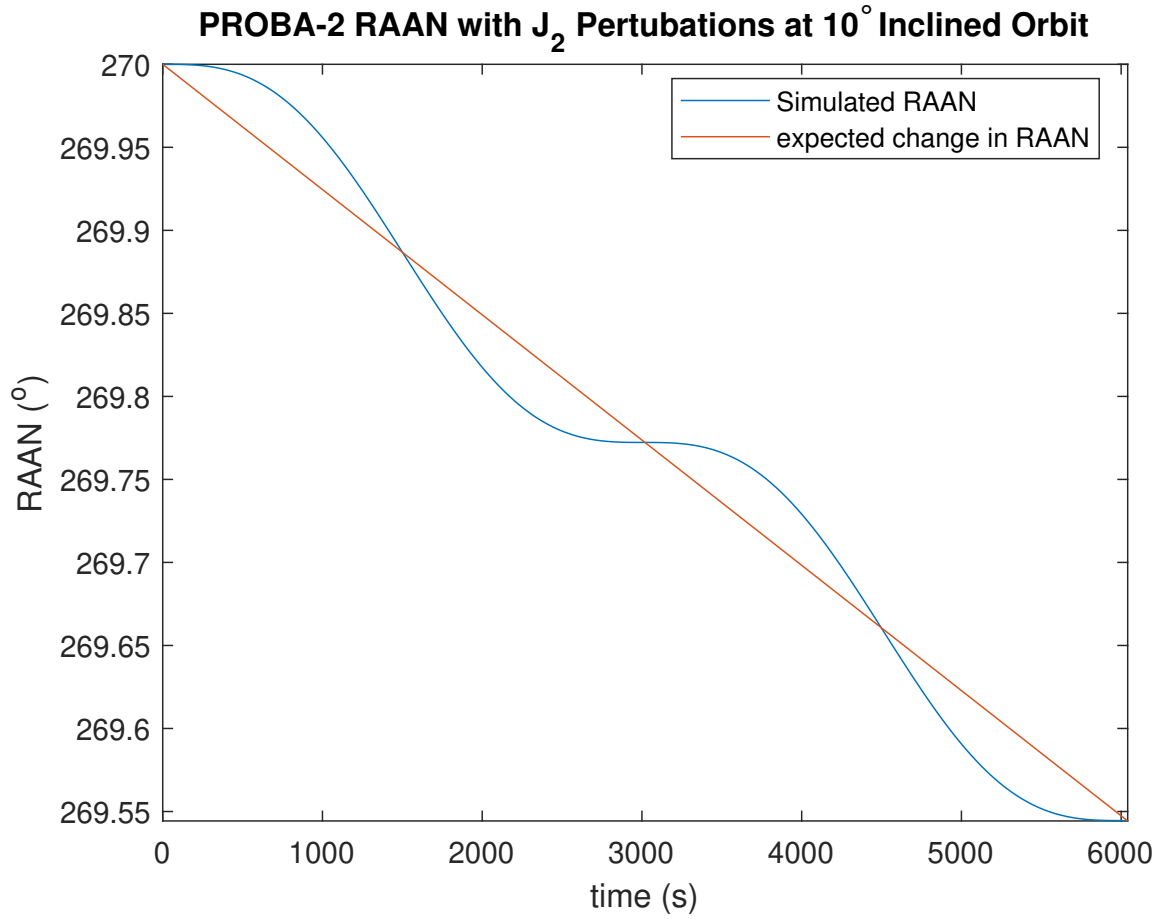


Figure 10: PROBA-2 Change in RAAN at 10° inclination

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 ($^\circ/\text{rev}$)	average rate of change($^\circ/\text{s}$)
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 ($^\circ/\text{rev}$)	average rate of change($^\circ/\text{s}$)
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)

$\mathbf{v}_{\mathbf{x}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{\mathbf{y}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{\mathbf{z}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{x}_0(\text{km})$	$\mathbf{y}_0(\text{km})$	$\mathbf{z}_0(\text{km})$
0.000	0.000	0.000	0.000	-1.000×10^1	0.000

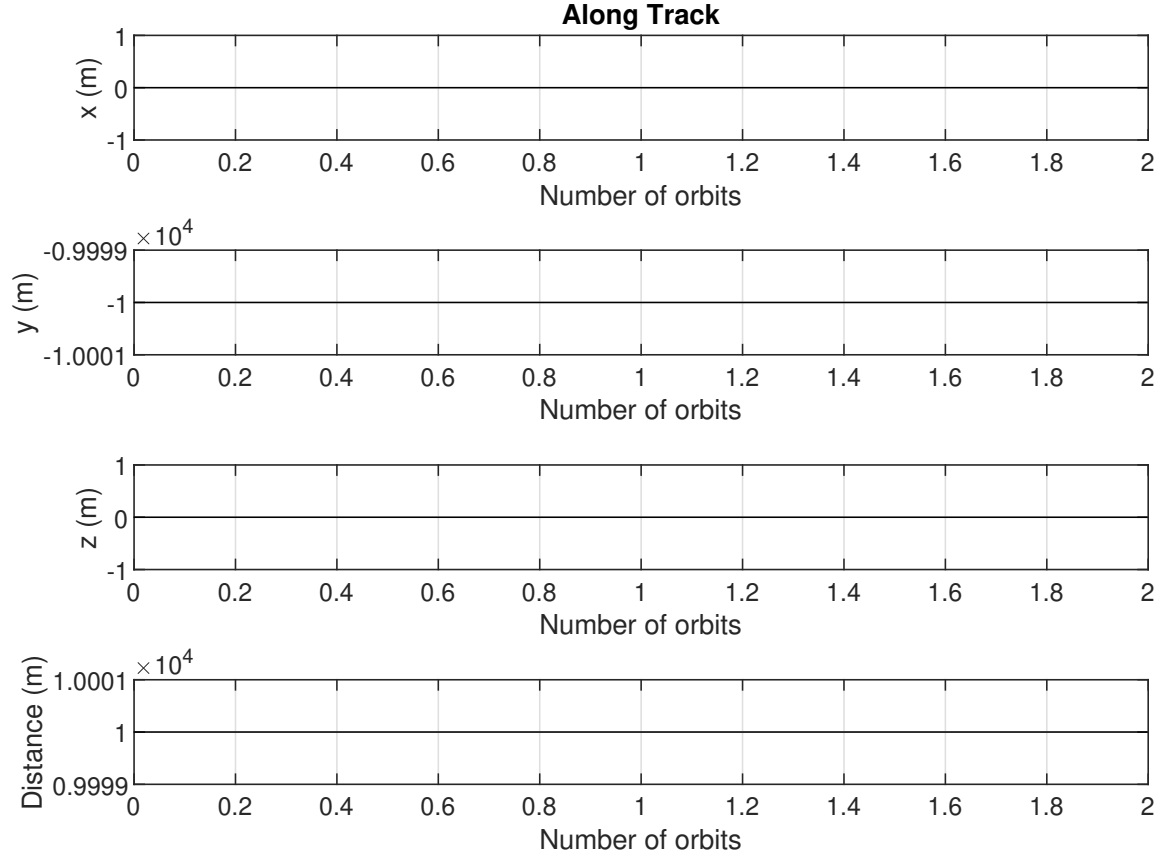


Figure 11: Position component plots (along-track formation)

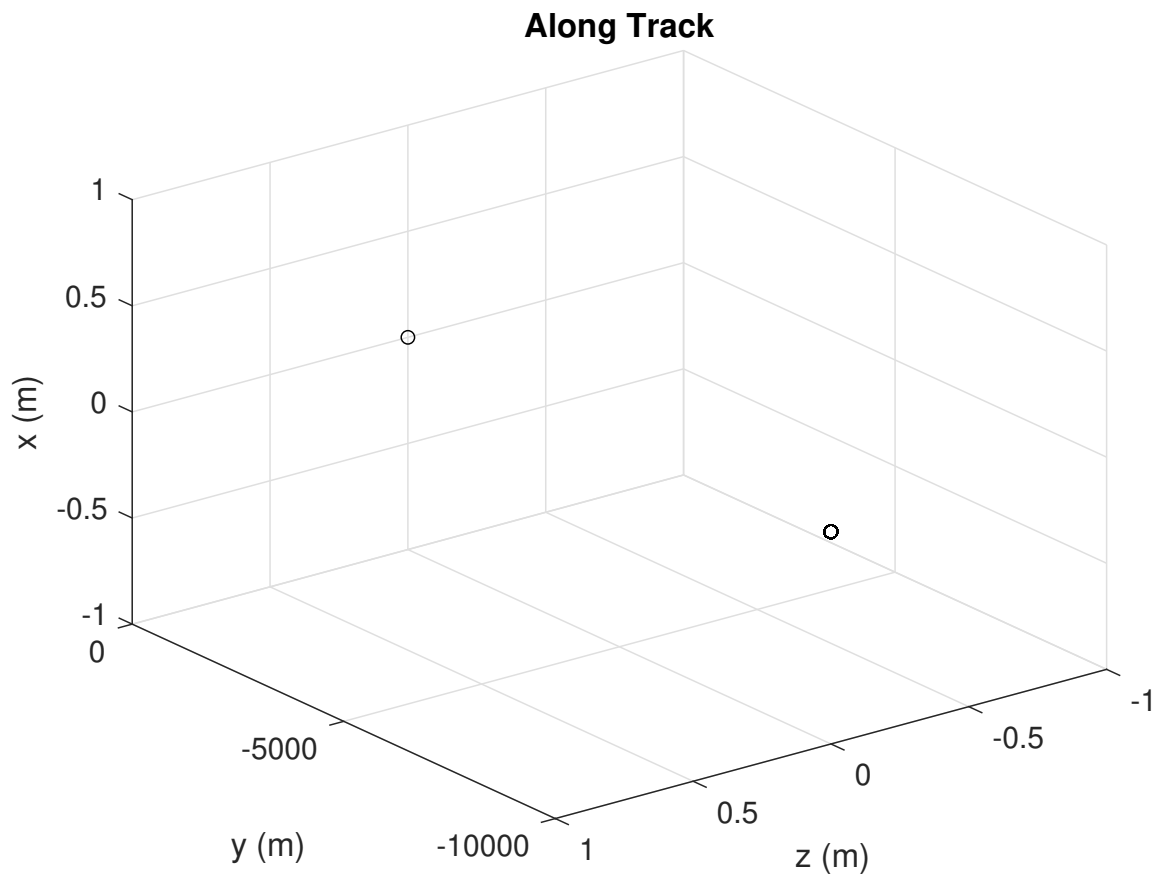


Figure 12: 3D-plot of path around target (along-track formation)

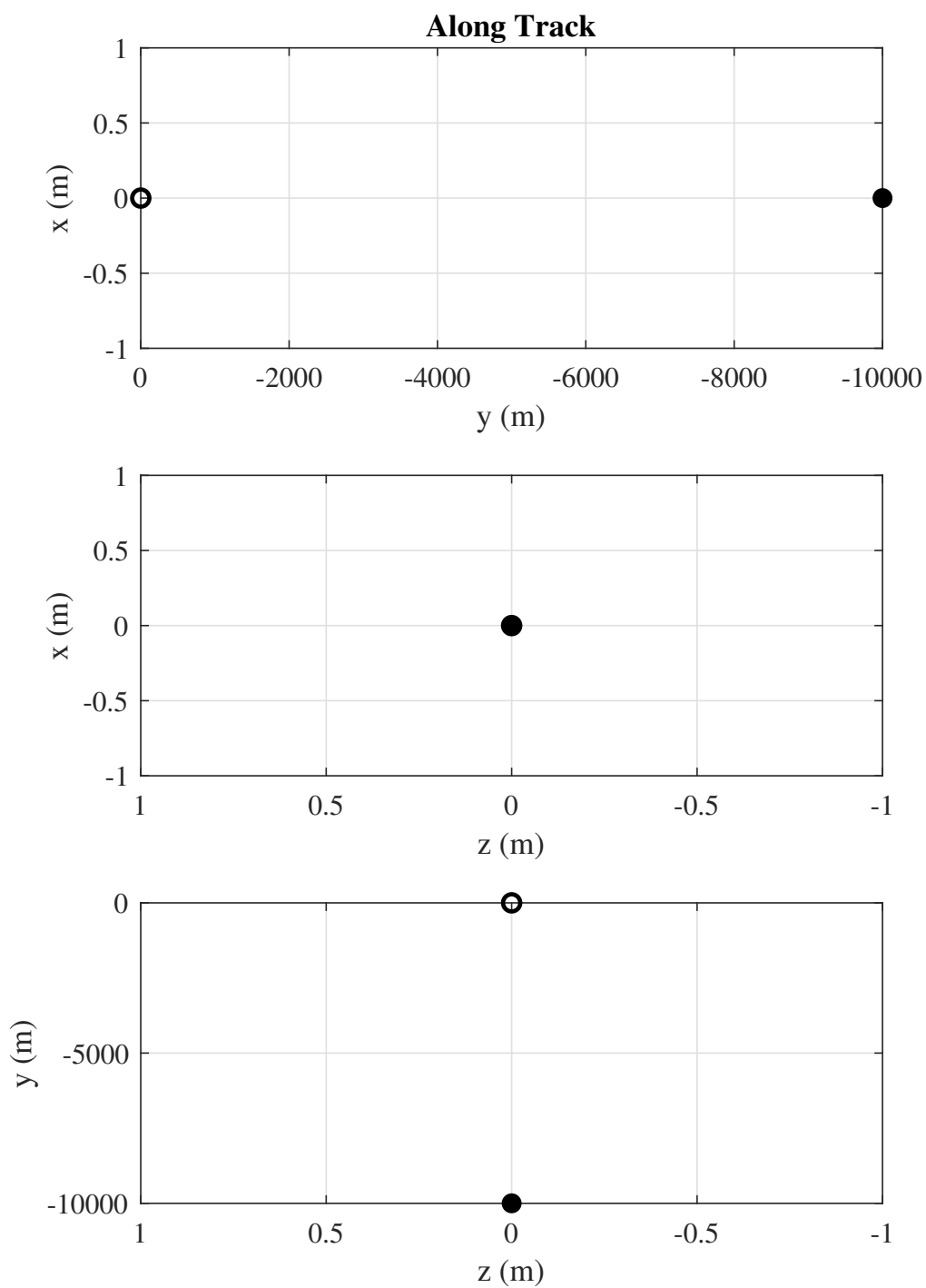


Figure 13: Path in projected planes (along-track formation)

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)

$\mathbf{v}_{x_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{y_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{z_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{x}_0(\text{km})$	$\mathbf{y}_0(\text{km})$	$\mathbf{z}_0(\text{km})$
0.000	0.000	0.000	0.000	-1.000×10^1	6.900×10^{-1}

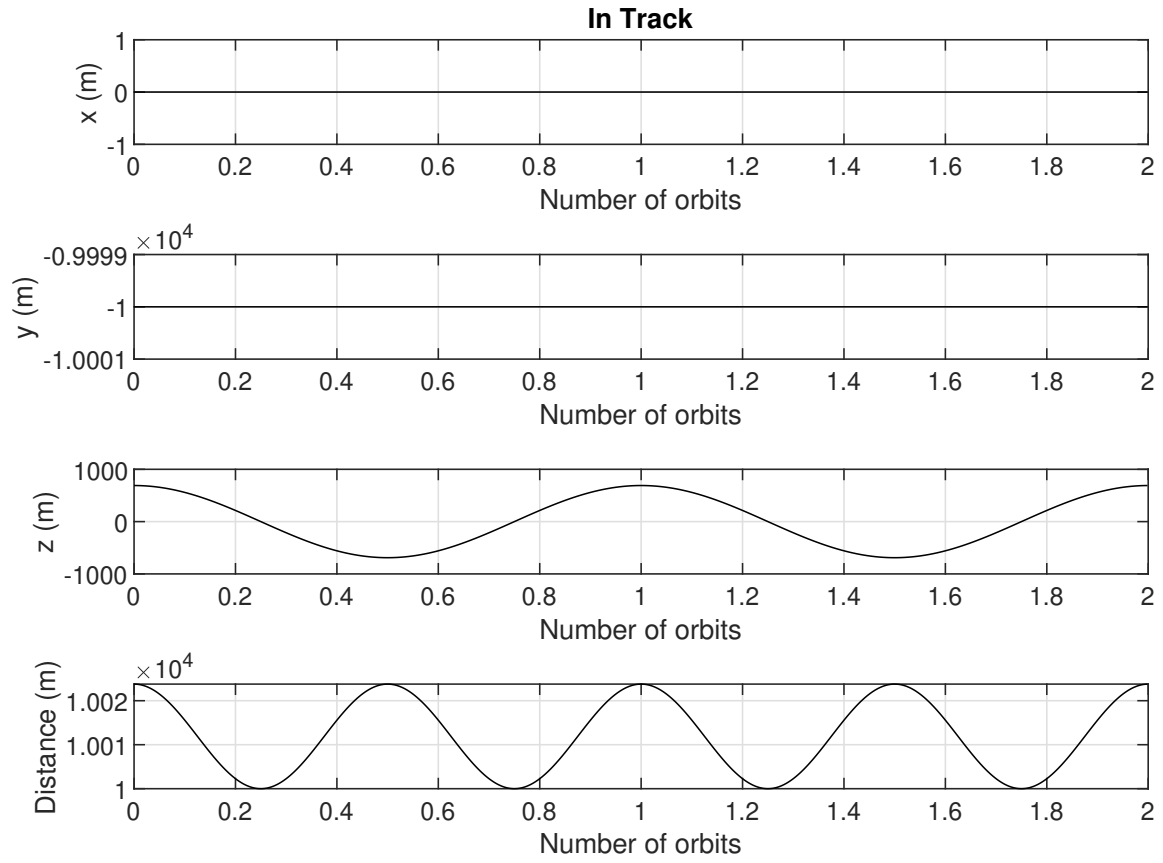


Figure 14: Position component plots (in-track formation)

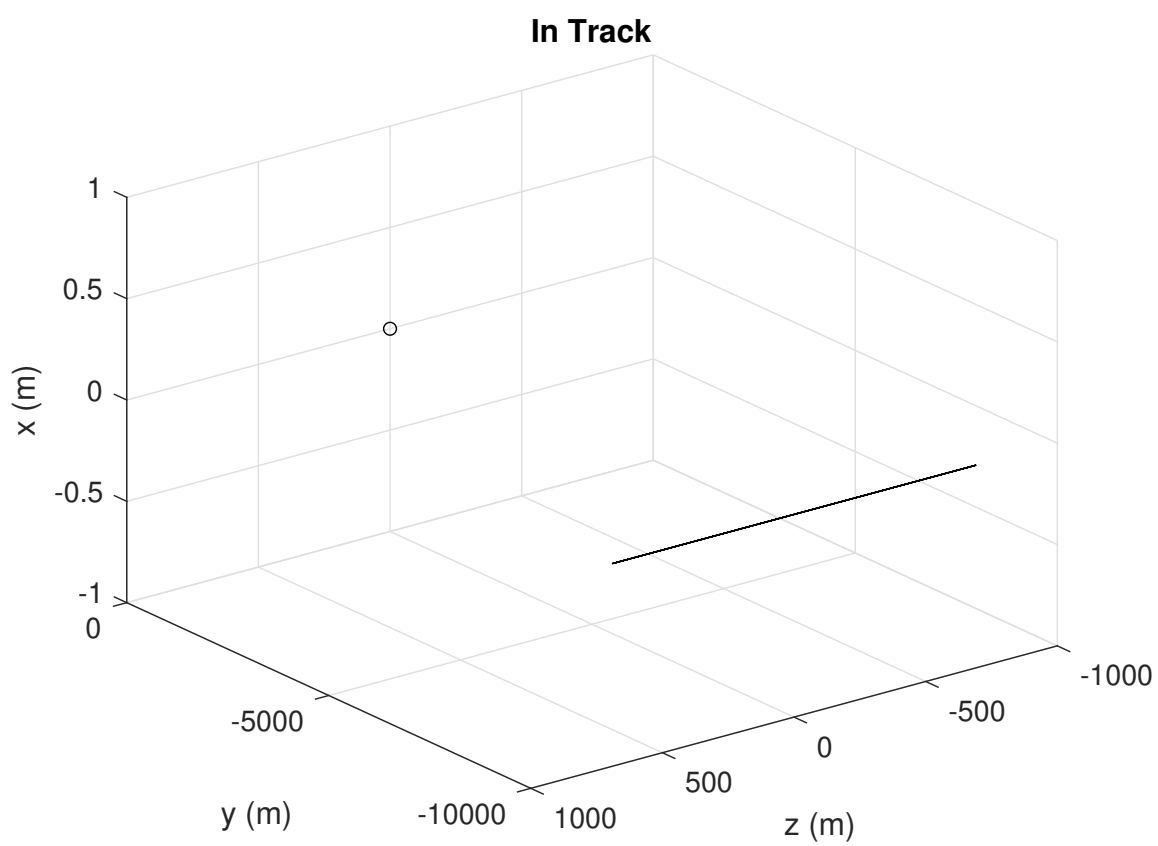


Figure 15: 3D-plot of path around target (in-track formation)

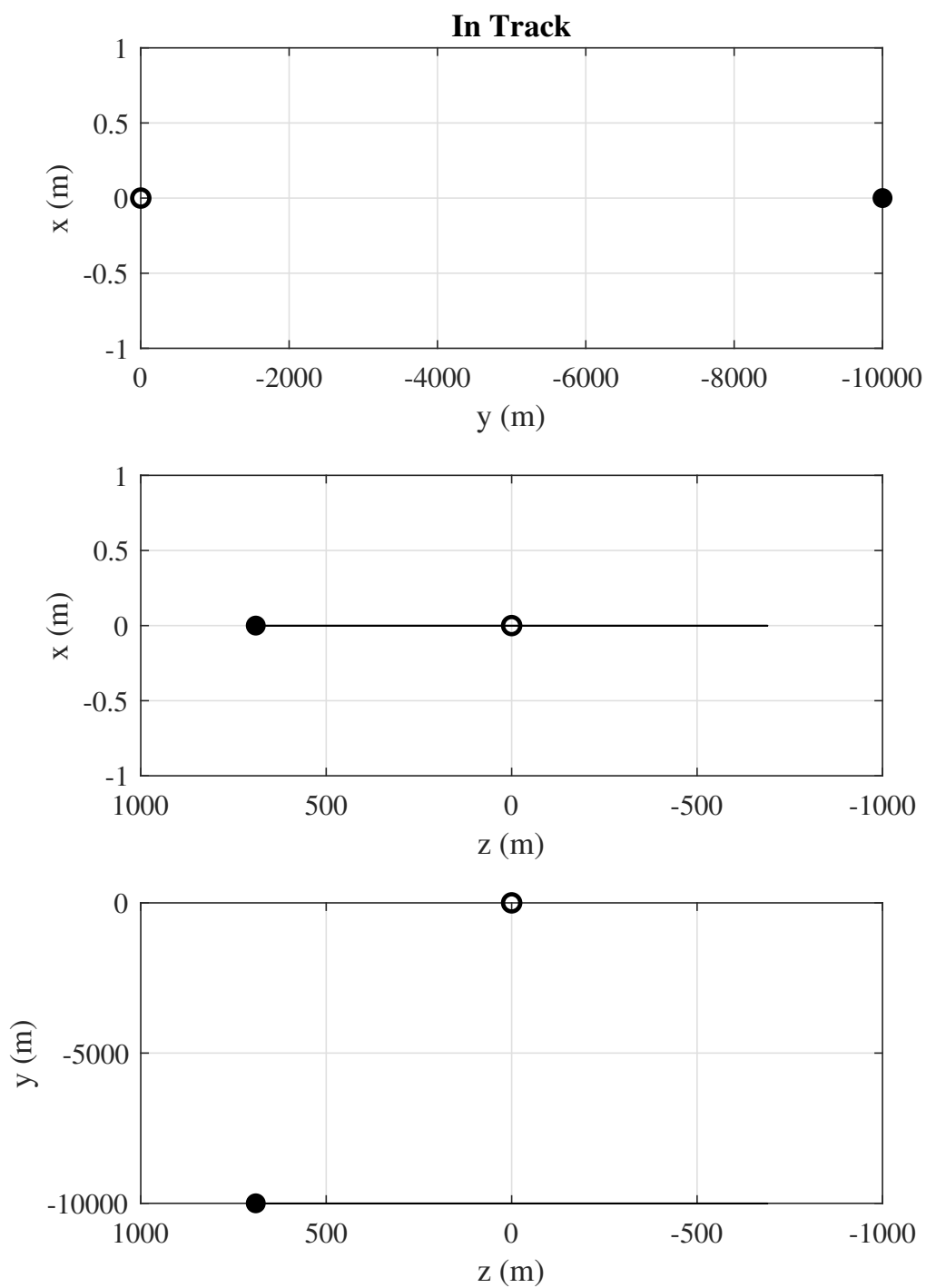


Figure 16: Path in projected planes (in-track formation)

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)

$\mathbf{v_{x0}} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v_{y0}} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v_{z0}} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{x_0}(\text{km})$	$\mathbf{y_0}(\text{km})$	$\mathbf{z_0}(\text{km})$
0.000	-5.228×10^{-4}	0.000	2.500×10^{-1}	0.000	0.000

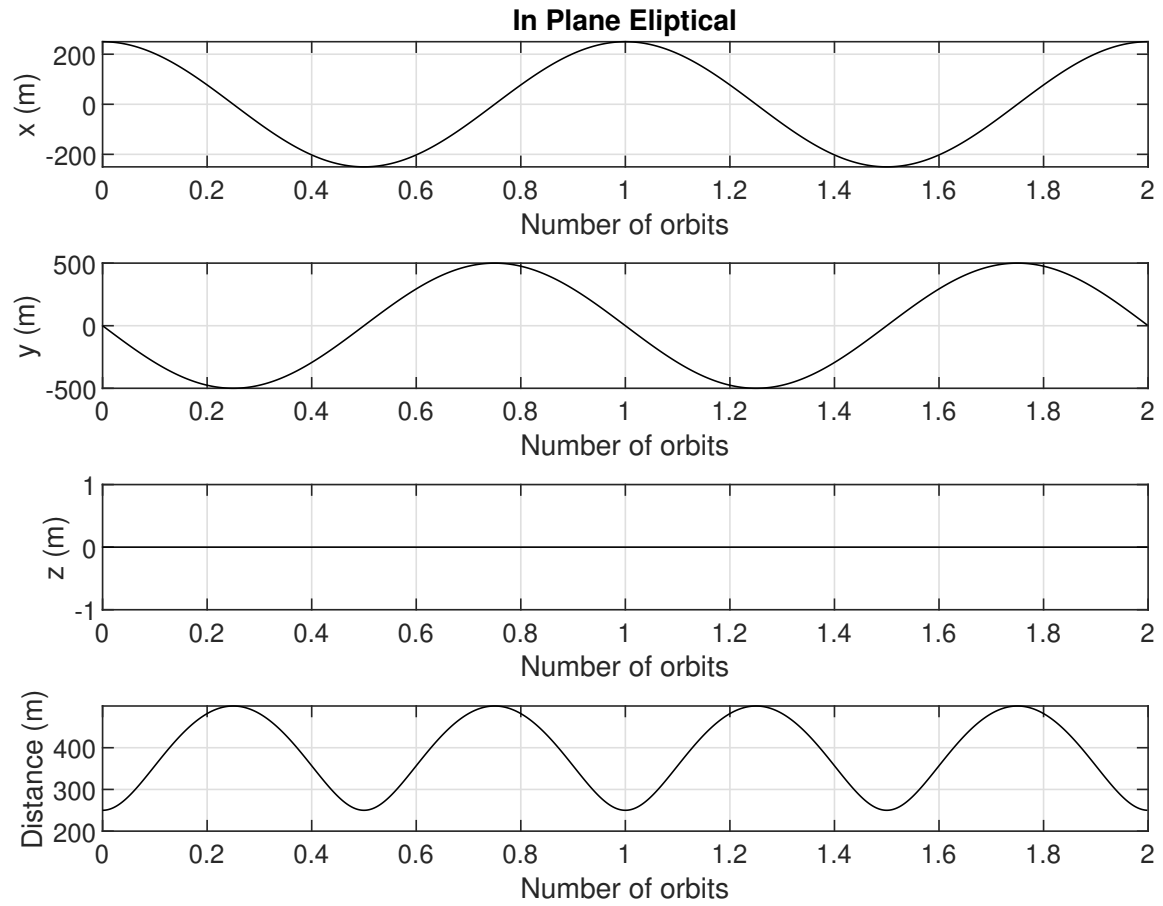


Figure 17: Position component plots (in-plane elliptical formation)

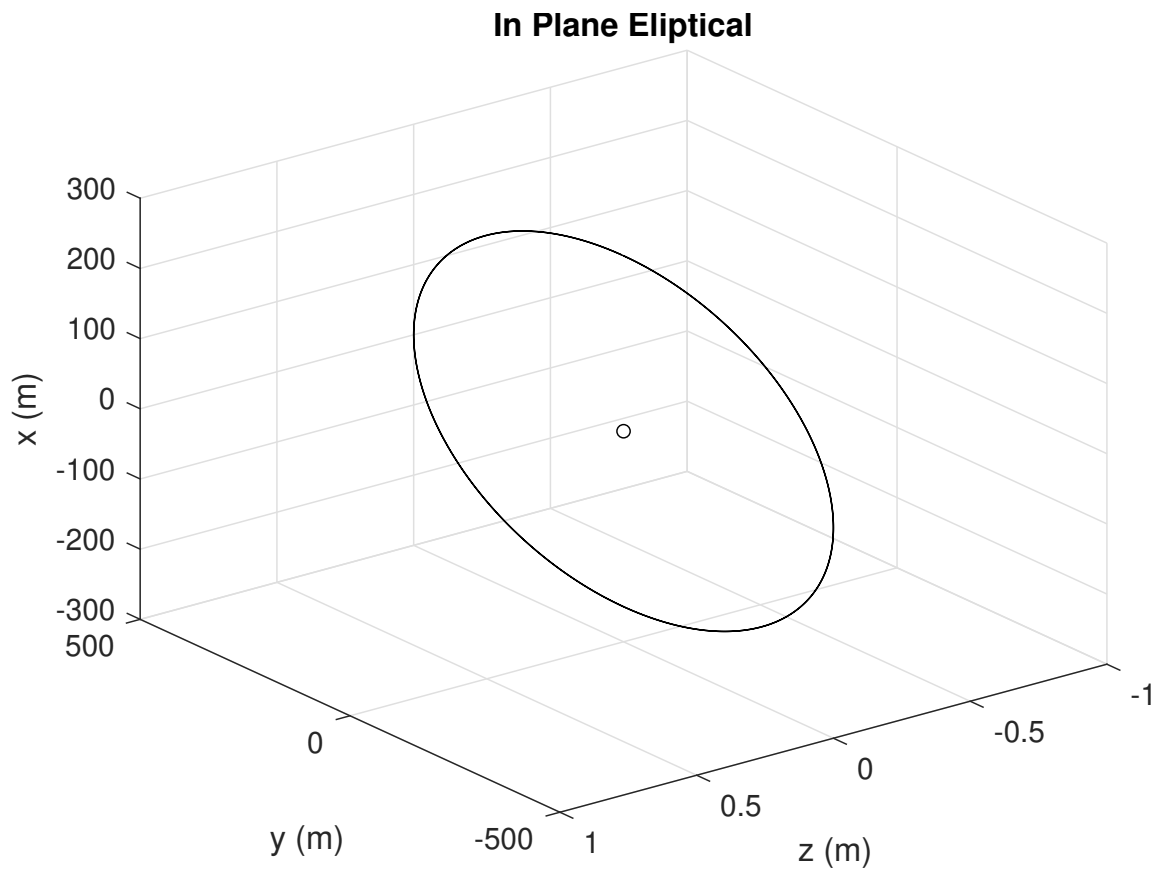


Figure 18: 3D-plot of path around target (in-plane elliptical formation)

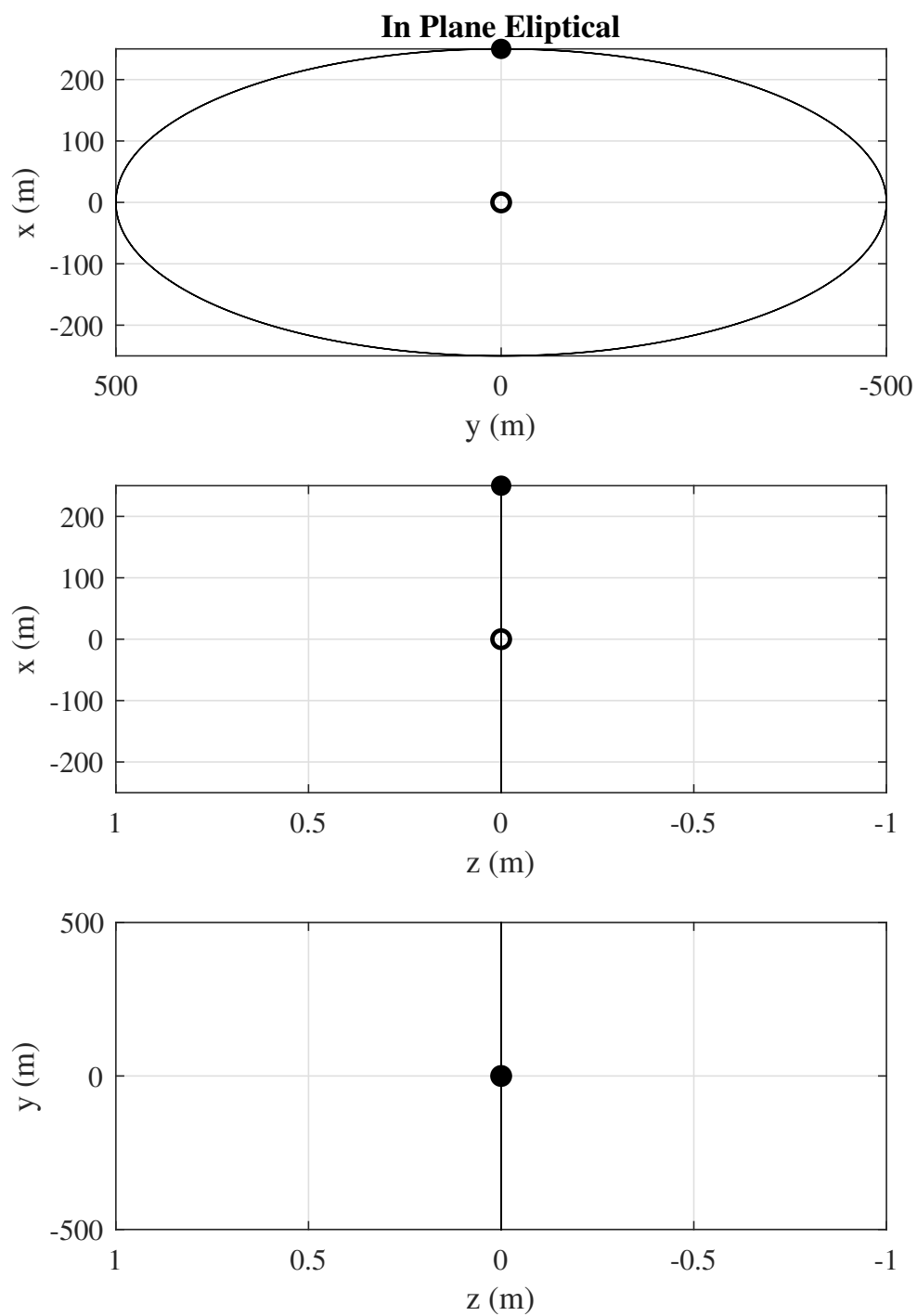


Figure 19: Path in projected planes (in-plane elliptical formation)

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)

$\mathbf{v}_{\mathbf{x}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{\mathbf{y}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{\mathbf{z}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{x}_0(\text{km})$	$\mathbf{y}_0(\text{km})$	$\mathbf{z}_0(\text{km})$
0.000	-2.091×10^{-5}	0.000	1.000×10^{-2}	0.000	1.732×10^{-2}

\mathbf{z}_0 and $\mathbf{v}_{\mathbf{z}_0}$ in table 8 can also be negated and achieve the same formation type.

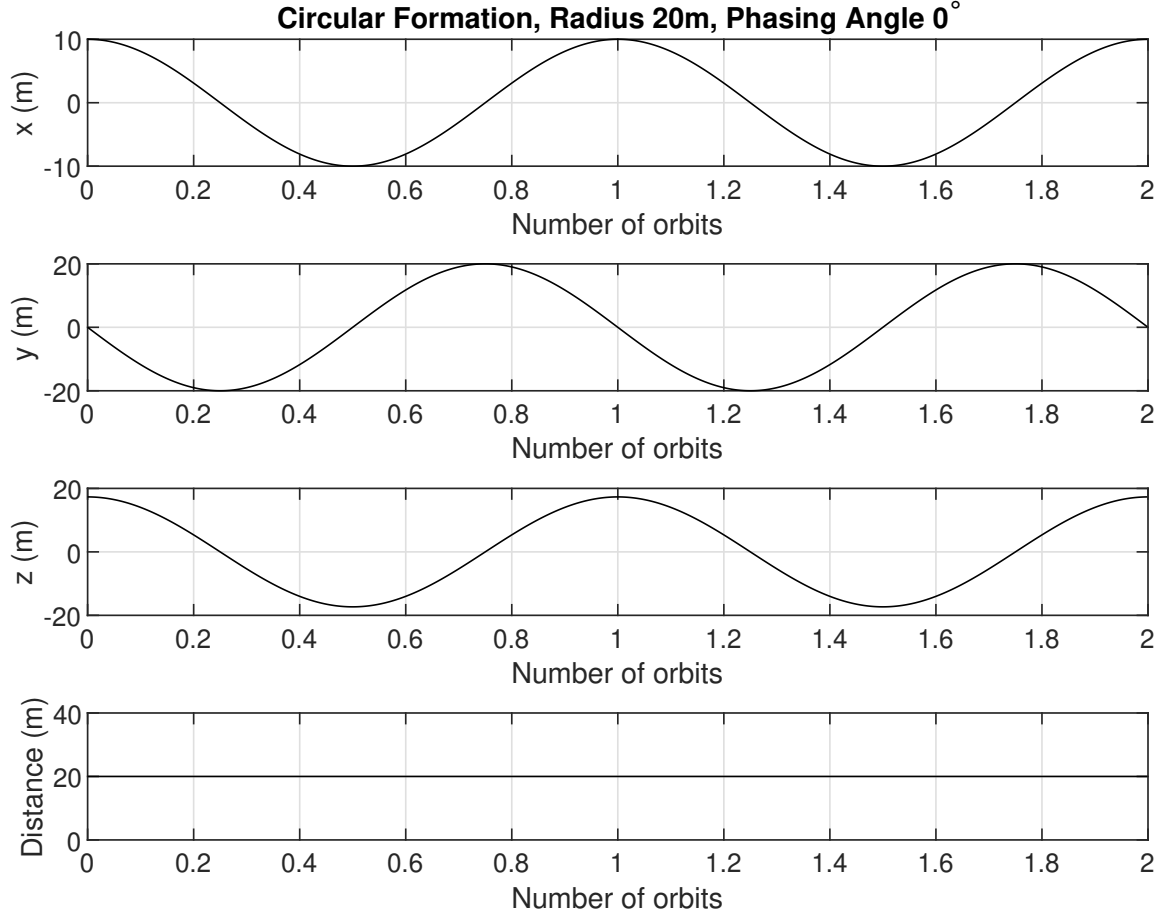


Figure 20: Position component plots (circular formation)

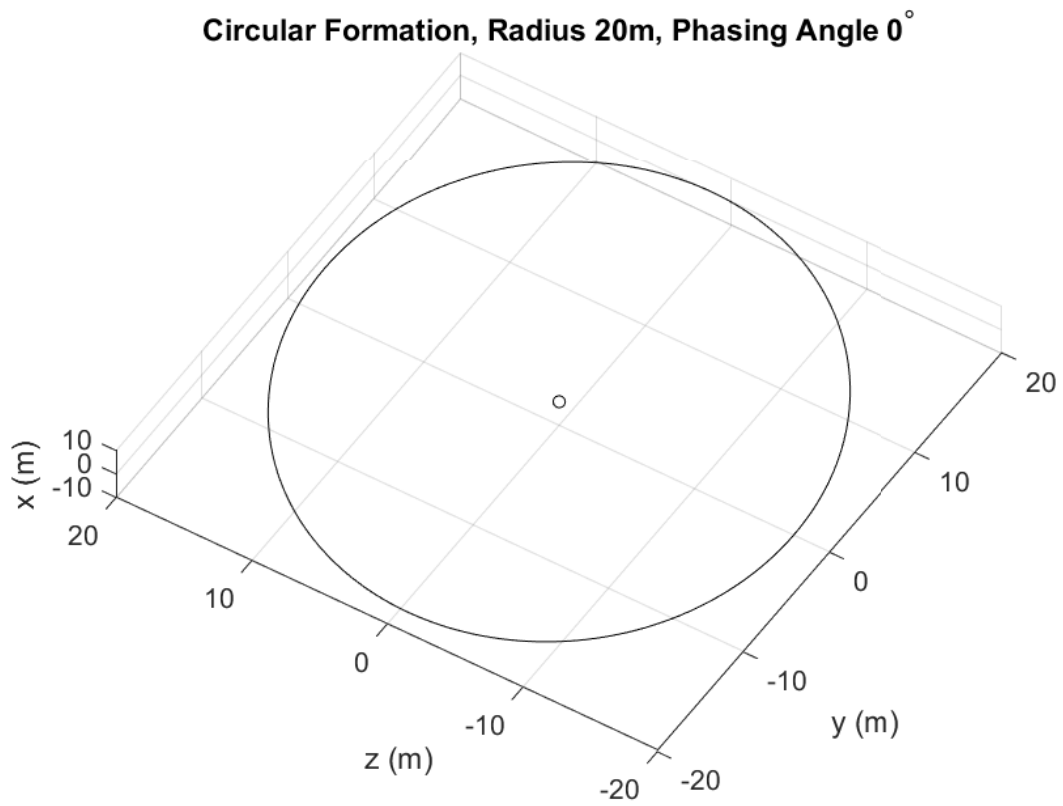


Figure 21: 3D-plot of path around target (circular formation)

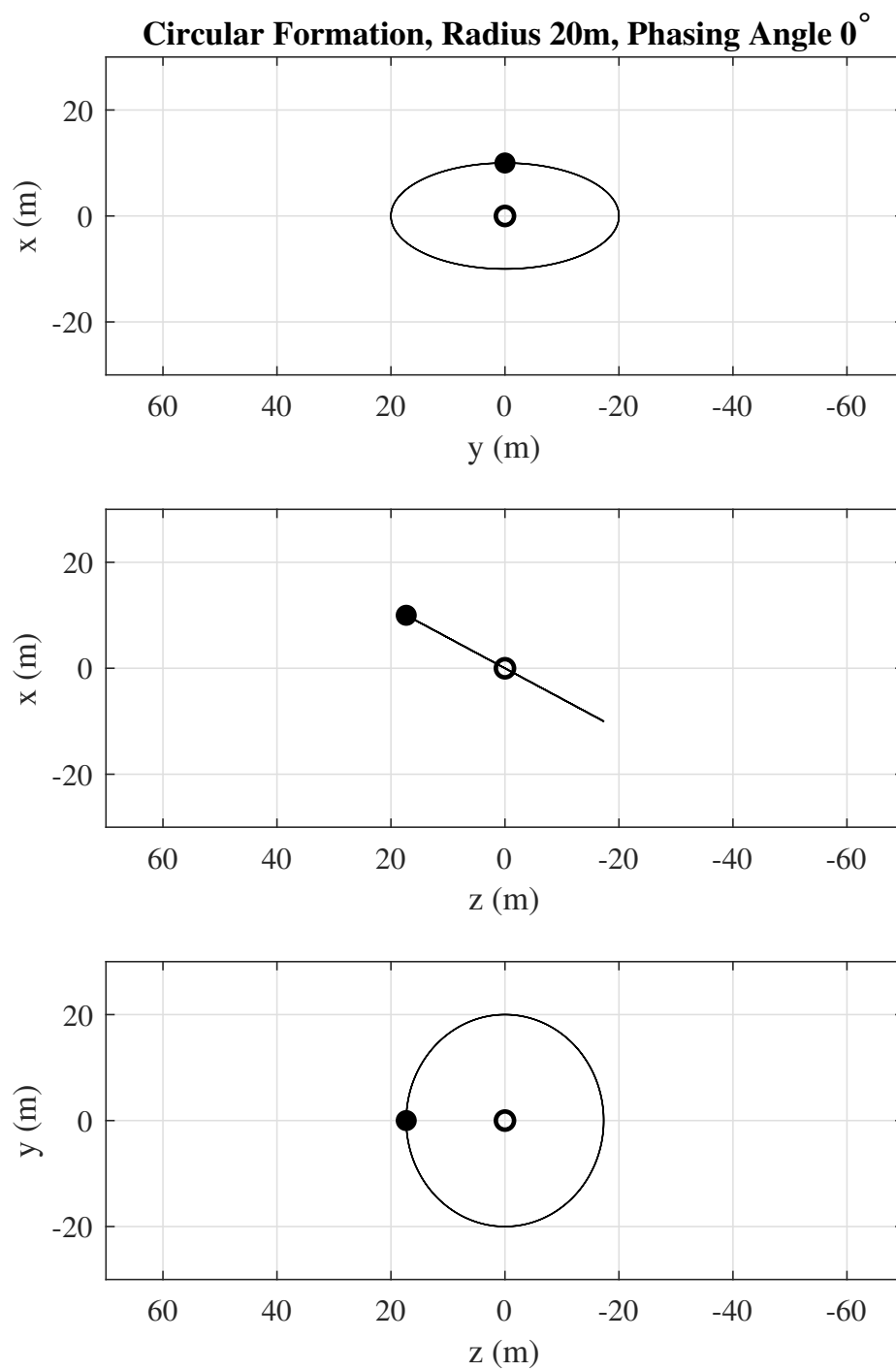


Figure 22: Path in projected planes (circular formation)

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)

$\mathbf{v}_{\mathbf{x}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{\mathbf{y}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{v}_{\mathbf{z}_0} \left(\frac{\text{km}}{\text{s}} \right)$	$\mathbf{x}_0(\text{km})$	$\mathbf{y}_0(\text{km})$	$\mathbf{z}_0(\text{km})$
1.046×10^{-5}	0.000	2.091×10^{-5}	0.000	2.000×10^{-2}	0.000

\mathbf{z}_0 and $\mathbf{v}_{\mathbf{z}_0}$ in table 9 can also be negated and achieve the same formation type.

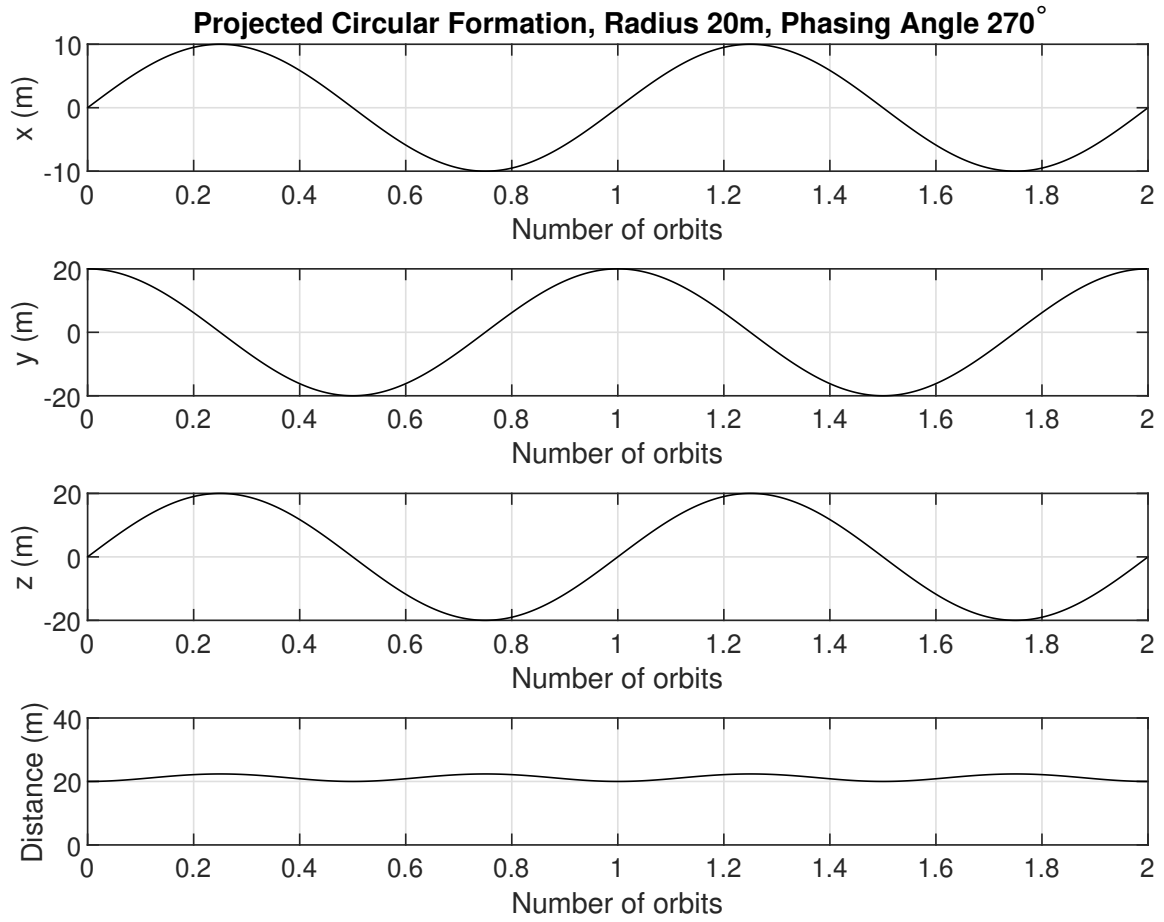


Figure 23: Position component plots (projected circular formation)

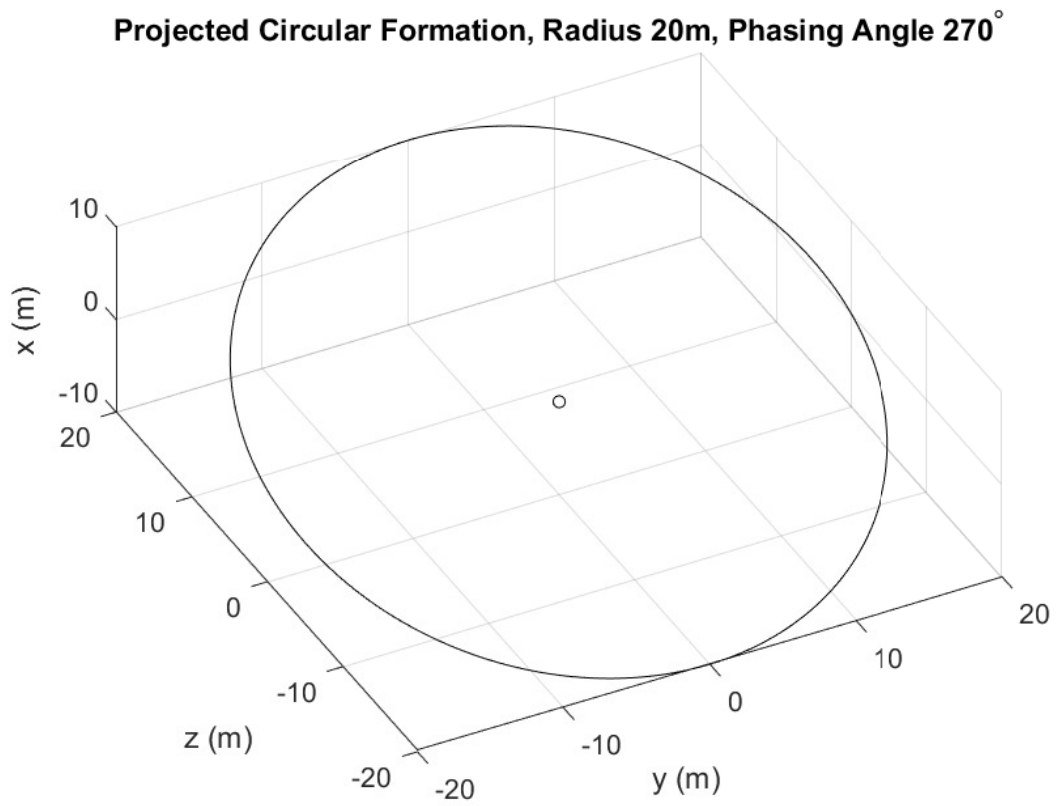


Figure 24: 3D-plot of path around target (projected circular formation)

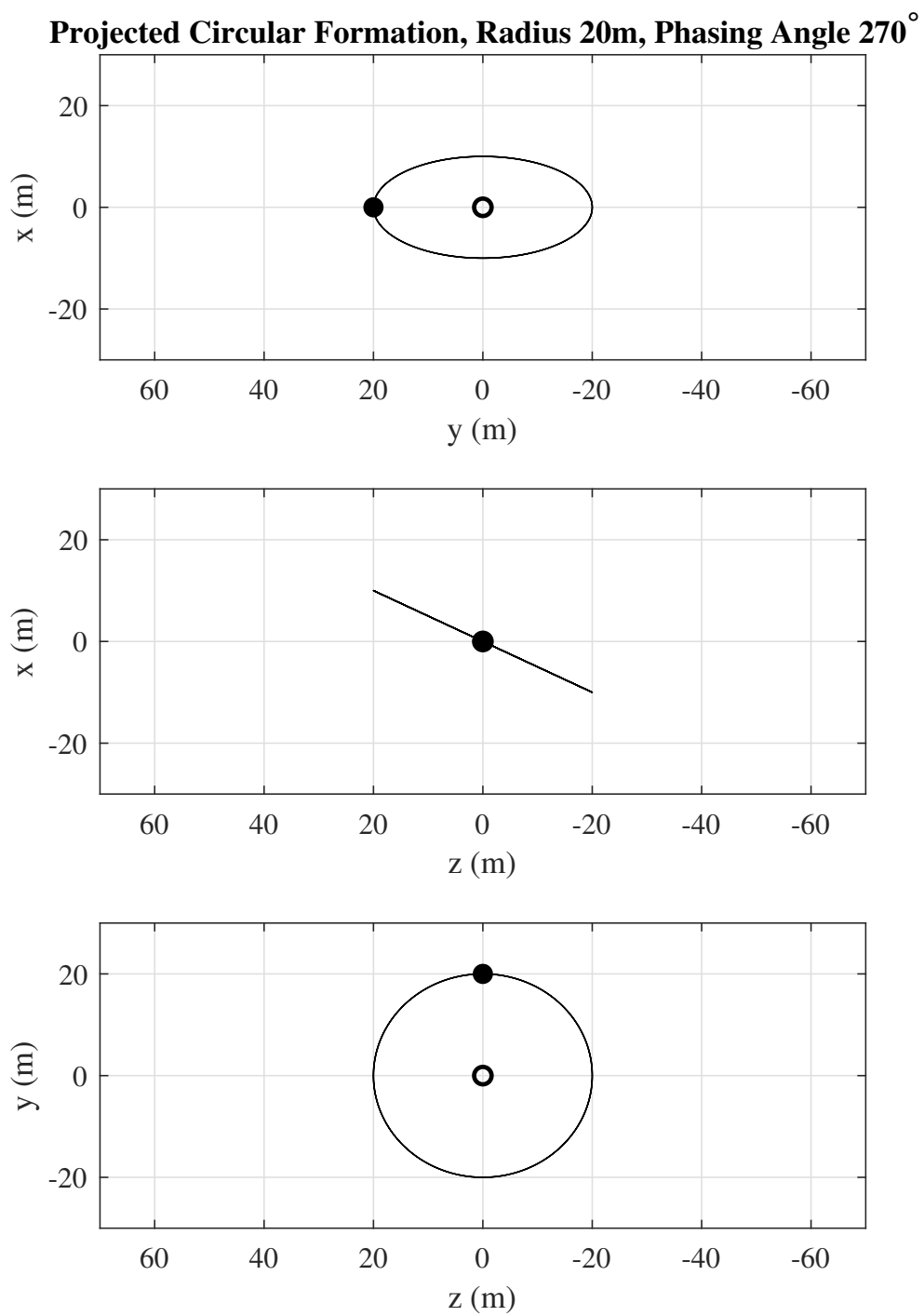


Figure 25: Path in projected planes (projected circular formation)

Appendix A MATLAB code for 2.5

```
1  clc
2  close all;
3  clear;
4
5  %constants
6  R_earth=6.378E3;
7  mu=3.986E5;
8  altitude = 150:1:1000;
9
10 r = altitude +R_earth; % define radius
11
12 theta_dot = sqrt(mu.*(r.^-3)); % calculate mean motion
13
14 T = (2*pi).*(theta_dot.^-1); %calculate periods
15 v = r.*theta_dot; %calculate velocity
16
17
18 %% Plot Figures
19 figure(1)
20 plot(altitude,v)
21 title("Orbit Altitude vs Speed (Circular Orbit)")
22 xlabel("Altitude(km)")
23 ylabel("Inertial speed(km/s)")
24 print('C:\Users\boaza\OneDrive - Carleton University\AERO 3240\output
    ↪ files\2.5\Orbit_Altitude_vs_Velocity','-depsc')
25
26 figure(2)
27 plot(altitude,T)
28 title("Orbit Altitude vs period (Circular Orbit)")
29 xlabel("Altitude(km)")
30 ylabel("Period(s)")
31 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

```
1  clc
2  clear all;
3  close all;
4
5  % save folder
6  mydir = pwd;
7  idcs = strfind(string(mydir), '\');
8  newdir = mydir(1:idcs(end)-1);
9  save_to = strcat(newdir, '\output files\2.13\');
10 % Add general Functions
11 addpath(strcat(newdir, '\general functions'));
12 clear mydir idcs newdir;
13 % Gravitational parameter
14 mu = 398600.4418; % km^3/s^2
15
16 % Earth radius
17 rE = 6378; % km
18
19 % Conversion constants
20 d2r = pi/180; % rad/deg
21 r2d = 1/d2r; % deg/rad
22
23 % Orbital elements
24 a = rE + 791; % km
25 e = 0.001;
26 i = 98.28; % deg
27 w = 0; % deg
28 RAAN = 270; % deg
29 tp = 0; % sec
30
31 p = a*(1-e^2);
32
33 % Eccentric anomaly at t = 0 sec
34 eano = 0; % deg
35
36 % True anomaly at t = 0 sec
37 tano = 0; % deg
38
39 % Magnitude of position vector at t = 0 sec
40 r = p/(1+e*cosd(tano)); % km
41
```

```

42 % Components of position and velocity vectors in perifocal at t = 0 sec
43 r_P_ini = r*[1 0 0]';
44
45 v_P_ini = [0 sqrt(mu/p)*(e+1) 0]';
46
47 % Rotation matrix from perifocal to ECI, i.e., C_IP
48
49 C_IP = (C_3(w)*C_1(i)*C_3(RAAN))'; % using functions I made for rot matrices
50
51 % Components of position and velocity vectors in ECI at t = 0 sec
52 r_I_ini = C_IP*r_P_ini;
53 v_I_ini = C_IP*v_P_ini;
54
55 T = 2*pi*sqrt(a^3/mu);
56
57 % Simulation from t = 0 to t = T
58 open_system('PROBA2mdl.slx')
59 set_param('PROBA2mdl', 'StopTime', 'T')
60 disp('Running Simulation...')
61 sim('PROBA2mdl')
62
63 %% Part A
64 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65 % Plot components of the position vector in Perifocal
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 % r_P through conversion of simulation variables to perifocal
68 r_P = C_IP'*r_I';
69 % find r_p through equation
70
71 % Orbit equation definition with 2d vector componenets
72 syms r_P_func(theta)
73 r_P_func(theta) = matlabFunction(p/(1+e*cos(theta))*[cos(theta);sin(theta)]);
74
75 %calculation of orbit through orbit equation
76 r_P_eq = r_P_func(0:0.01:2*pi);
77 r_P_eq = double(cell2sym(r_P_eq)); % convert to symbolic to number
78
79 figure
80 plot(r_P(1,:),r_P(2:,:), 'k', 'LineWidth', 2);
81 title('PROBA 2 Orbital Path in Perifocal Reference Frame')
82 xlabel('rx_P (km)')
83 ylabel('ry_P (km)')
84 hold on
85 plot(r_P_eq(1,:),r_P_eq(2,:), 'r', 'LineWidth', 1);
86 legend('simulation', 'orbit equation', 'Location', 'best')
87 pbaspect([1 1 1])

```

```

88 daspect([1 1 1])
89 hold off
90
91 print(strcat(save_to,'perifocal_plot.eps'),'-depsc')
92
93 %% Part B
94 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
95 % Tabulize r_I_ini and v_I_ini
96 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
97 varNames = ["Component", "r_I_initial", "v_I_initial"];
98 varTypes = ["string", "double", "double"];
99 sz = [3 3];
100 initial_conditions =
    ⇨ table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);
101 initial_conditions.r_I_initial = r_I_ini;
102 initial_conditions.v_I_initial = v_I_ini;
103 initial_conditions.Component = ["X";"Y";"Z"];
104 initial_conditions.Properties.VariableNames = ["Component", "r_{I,initial}(km)",
    ⇨ "v_{I,initial}(km/s)"];
105 writetable(initial_conditions,strcat(save_to,'initial_conditions.csv'))
106
107
108
109 %% Part C
110 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111 % Plot components of the position vector in ECI in 3D
112 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113 figure
114 plot3(r_I(:,1),r_I(:,2),r_I(:,3),'k','linewidth', 2)
115 grid on
116
117 hold on
118 % Adding Earth
119 % Using Will Campbell (2022). Earth-sized
120 % Sphere with Topography
121 % (https://www.mathworks.com/matlabcentral
122 % /fileexchange/27123-earth-sized-sphere-with-topography),
123 % MATLAB Central File Exchange. Retrieved November 28, 2022.
124 earth_sphere(gca,'km')
125
126 set(gca,'FontSize',9,'FontName','Times')
127 title('PROBA 2 Orbital Path in ECI Reference Frame')
128 xlabel('rx_I (km)')
129 ylabel('ry_I (km)')
130 zlabel('rz_I (km)')
131 axis equal

```

```

132
133
134
135
136 print(strcat(save_to, '3d_plot.png'), '-dpng', '-r600')
137 %% part D
138 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
139 % Plot components of the position vector as function of time
140 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
141 figure
142 subplot(3,1,1)
143 plot(t,r_I(:,1), 'k');
144 title('PROBA 2 Position Vector Components in ECI Reference Frame')
145 xlabel('Time (sec)')
146 ylabel('rx_I (km)')
147 xlim([0 max(t)])
148
149 subplot(3,1,2)
150 plot(t,r_I(:,2), 'k');
151 xlabel('Time (sec)')
152 ylabel('ry_I (km)')
153 xlim([0 max(t)])
154
155 subplot(3,1,3)
156 plot(t,r_I(:,3), 'k');
157 xlabel('Time (sec)')
158 ylabel('rz_I (km)')
159 xlim([0 max(t)])
160
161
162 print(strcat(save_to, 'position_components.eps'), '-depsc')
163
164 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
165 % Plot components of the velocity vector as function of time
166 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
167 figure
168 subplot(3,1,1)
169 plot(t,v_I(:,1), 'k');
170 title('PROBA 2 Velocity Vector Components in ECI Reference Frame')
171 xlabel('Time (sec)')
172 ylabel('vx_I (km/s)')
173 xlim([0 max(t)])
174
175 subplot(3,1,2)
176 plot(t,v_I(:,2), 'k');
177 xlabel('Time (sec)')

```

```

178 ylabel('vy_I (km/s)')
179 xlim([0 max(t)])
180
181 subplot(3,1,3)
182 plot(t,v_I(:,3),'k');
183 xlabel('Time (sec)')
184 ylabel('vz_I (km/s)')
185 xlim([0 max(t)])
186
187 print(strcat(save_to,'velocity_components.eps'),'-depsc')
188
189 %% part E
190 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
191 % Plot magnitude of position and velocity vectors as function of time
192 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
193 r_I_mag = vecnorm(r_I,2,2);
194 v_I_mag = vecnorm(v_I,2,2);
195
196 times = 60*[30 60 90];
197
198 figure
199 subplot(2,1,1)
200 plot(t,r_I_mag,'k');
201 title('PROBA 2 Radius from Earth's Center')
202 xlabel('Time (sec)')
203 ylabel('Radius (km)')
204 xlim([0 max(t)])
205 xline(times(1),'k',"30 mins")
206 xline(times(2),'k',"60 mins")
207 xline(times(3),'k',"90 mins")
208
209 subplot(2,1,2)
210 plot(t,v_I_mag,'k');
211 title('PROBA 2 Speed in ECI')
212 xlabel('Time (sec)')
213 ylabel('Speed (km/s)')
214 xlim([0 max(t)])
215 xline(times(1),'k',"30 mins")
216 xline(times(2),'k',"60 mins")
217 xline(times(3),'k',"90 mins")
218
219 % Vis viva equation:
220 syms r
221 v(r) = sqrt(mu*(2/r-1/a));
222
223 % Create check table

```

```

224 varNames =
    ↪ ["Time","Radius","vis_viva_calculated_speed","simulation_calculated_speed"];
225 varTypes = ["double","double","double","double"];
226 sz = [length(times) length(varNames)];
227 vis_viva_check =
    ↪ table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);
228
229 % Calculate and tabulate the vis-viva check table
230 for loop_var = 1:length(times)
231     [useless_var,index] = min(abs(t-times(loop_var))); %find index of time
232     radius_at_point = r_I_mag(index); % Get radius at time
233     vis_viva_check.Radius(loop_var) = radius_at_point;
234
235     % Calculate velocity based on given radius
236     vis_viva_check.vis_viva_calculated_speed(loop_var) =
    ↪ double(v(radius_at_point));
237     vis_viva_check.simulation_calculated_speed(loop_var) = v_I_mag(index);
238
239     vis_viva_check.Time(loop_var) = times(loop_var)/60;
240     clear useless_var;
241 end
242 vis_viva_check.Properties.VariableNames = ["Time (minutes)", ...
243     "Radius from Simulation(km)", ...
244     "Calculated Speed from Vis-Viva equation (km/s)",...
245     "Speed from Simulation (km/s)"];
246
247 writetable(vis_viva_check, strcat(save_to, 'vis_viva_check.csv'))
248 print(strcat(save_to, 'radius_and_speed.eps'), '-depsc')
249
250 %% part F
251 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
252 % Plot Ground path
253 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
254 wE = 15.04/3600; %converting 15.04 deg/hr to deg/s;
255
256 thetaE = wE.*t; %Array of theta_GMT
257
258 % Calculate r_F
259 r_F = cell2mat(arrayfun(@(i) r_I2r_F(thetaE(i), r_I(i, :)), ...
260     1:length(thetaE), 'UniformOutput', false))';
261
262
263 % Calculate latitude
264 latitude = asind(r_F(:,3)./r_I_mag);
265
266 % Calculate longitude and place in correct quadrant

```



```

267 negate_angle = (le(r_F(:,2), 0) -0.5).*-2;
268 longitude = negate_angle.*acosd(r_F(:,1)./((r_I_mag.*cosd(latitude))));
269
270 % Remove horizontal lines
271 dont_connect = find(diff(longitude) > 250);
272 for loop_var = 1:length(dont_connect)
273     index = dont_connect(loop_var) +loop_var -1;
274     longitude =
275     ↪ [longitude(1:index);NaN;longitude((index+1):(length(longitude)))];
276     latitude =
277     ↪ [latitude(1:index);NaN;latitude((index+1):(length(latitude)))];
278 end
279
280 figure
281 plot(longitude, latitude, 'k', 'LineWidth', 2);
282 ylim([-90 90])
283 xlim([-180 180])
284 title('PROBA 2 Ground Track')
285 xlabel('Longitude(^{\circ})')
286 ylabel('Latitude(^{\circ})')
287 hold on
288 I = imread('map.png');
289 h = image(xlim,ylim,I);
290 uistack(h, 'bottom')
291
292 print(strcat(save_to, 'ground_tracks.eps'), '-depsc')
293
294

```

Code Attachment 2: Script used in question 2.13

B.2 SIMULINK diagram

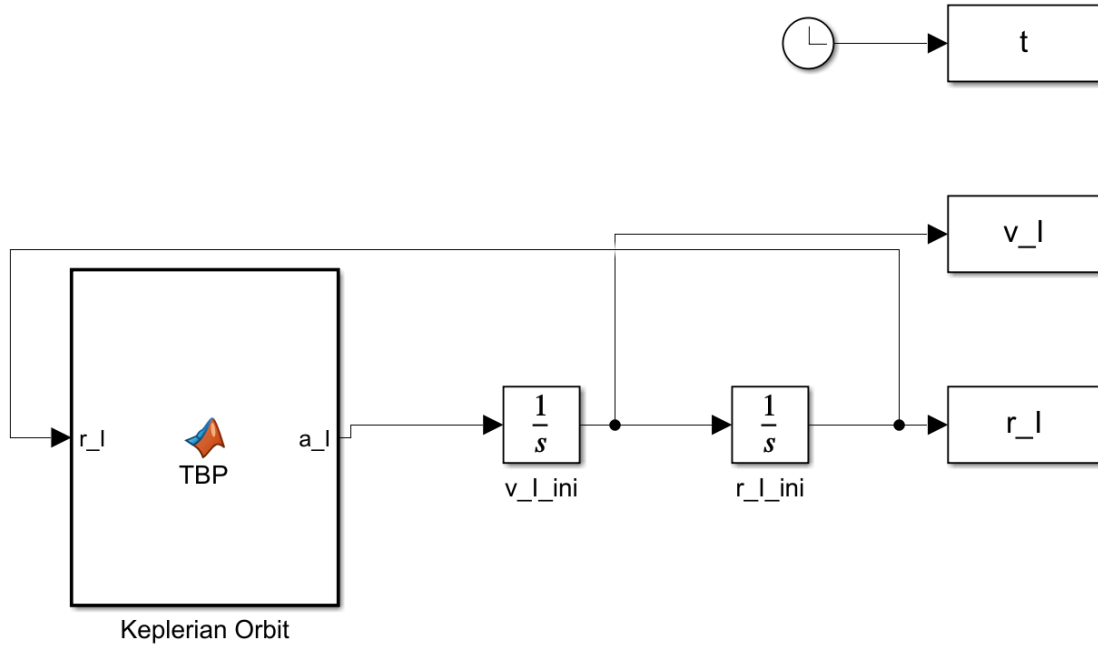


Figure 26: SIMULINK model for question 2.13

B.3 additional functions

B.3 a TBP function

```

1 function a_I = TBP(r_I, mu)
2 % -----
3 % TBP
4 % -----
5 % Description:
6 % Compute the spacecraft acceleration in ECI, using the two-body
7 % equation of motion.
8 % -----
9 % Inputs:
10 % r_I => components of r in ECI (3 x 1 matrix)
11 % -----
12 % Outputs:
13 % a_I => components of the acceleration in ECI (3 x 1 matrix)
14 % -----

```

```

15 % Parameters:
16 % mu => gravitational constant of the Earth
17 % -----
18 % Copyright:
19 % Steve Ulrich, 2014
20 % -----
21
22 % Compute orbital radius
23 rmod = sqrt( r_I(1)*r_I(1) + r_I(2)*r_I(2) + r_I(3)*r_I(3) );
24
25 % Spacecraft acceleration
26 a_I = -mu/rmod^3*r_I;
27 end

```

Code Attachment 3: Function to calculate a_I in SIMULINK model

B.3 b ECI to ECEF function

```

1 function r_F_vector = r_I2r_F(rotation_angle,r_I_vector)
2 %r_I2r_F Rotates r_I_vector about z axis by specified rotation_angle in degrees
3 r_F_vector = C_3(rotation_angle)*(r_I_vector');
4 end

```

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

Appendix C MATLAB code for 3.1

C.1 general script

```

1 clc
2 clear;
3 close all;
4
5 % save folder
6 mydir = pwd;
7 idcs = strfind(string(mydir),'\');
8 newdir = mydir(1:idcs(end)-1);
9 save_to = strcat(newdir,'\output files\3.1\');
10 % Add general Functions
11 addpath(strcat(newdir,'\general functions'));
12 clear mydir idcs newdir;
13 % J2 constant

```

```

14 J_2 = 1.08264E-3;
15
16 % Gravitational parameter
17 mu = 398600.4418; % km^3/s^2
18
19 % Earth radius
20 rE = 6378; % km
21
22 % Conversion constants
23 d2r = pi/180; % rad/deg
24 r2d = 1/d2r; % deg/rad
25
26 % Orbital elements
27 a      = rE + 791; % km
28 e      = 0.001;
29 i      = 98.28; % deg
30 w      = 0; % deg
31 RAAN   = 270; % deg
32 tp     = 0; % sec
33
34 p = a*(1-e^2);
35
36 % Eccentric anomaly at t = 0 sec
37 eano   = 0; % deg
38
39 % True anomaly at t = 0 sec
40 tano   = 0; % deg
41
42 % Magnitude of position vector at t = 0 sec
43 r = p/(1+e*cosd(tano)); % km
44
45 % Components of position and velocity vectors in perifocal at t = 0 sec
46 r_P_ini = r*[1 0 0]';
47
48 v_P_ini = [0 sqrt(mu/p)*(e+1) 0]';
49
50 % Rotation matrix from perifocal to ECI, i.e., C_IP
51
52 C_IP = (C_3(w)*C_1(i)*C_3(RAAN))'; % using functions I made for rot matrices
53
54 % Components of position and velocity vectors in ECI at t = 0 sec
55 r_I_ini = C_IP*r_P_ini;
56 v_I_ini = C_IP*v_P_ini;
57
58 orbits = 1;
59 T = 2*pi*sqrt(a^3/mu)*orbits;

```

```

60
61 % Simulation from t = 0 to t = T
62 open_system('PROBA2mdl_Chapter3.slx')
63 set_param('PROBA2mdl_Chapter3', 'StopTime', 'T')
64 disp('Running Simulation...')
65 sim('PROBA2mdl_Chapter3')
66
67 % Fix for simulink adding a dimension
68 v_I = squeeze(v_I)';
69 r_I = squeeze(r_I)';
70
71 %% Part A
72 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73 % Calculate secular change in RAAN
74 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
75
76 RAAN_secular_change = -((3*pi*J_2*rE^2)/(p^2))*cosd(i)*orbits;% rad
77 RAAN_secular_change = RAAN_secular_change*r2d; % Convert to degrees
78 RAAN_average_change = RAAN_secular_change/T; % Average change of RAAN (deg/s)
79 %% Part C
80 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
81 % Calculate and plot RAAN
82 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
83
84 h = cross(r_I,v_I); % Calculate angular momentum
85 N = cross( repmat([0 0 1],length(h),1),h); % Obtain vector pointing at RAAN
86
87 RAAN_Array = atan2d(N(:,2),N(:,1)); % find RAAN angle in degrees
88 RAAN_Array = le(RAAN_Array, 0)*360 + RAAN_Array; % add 360 if less than 0
89
90
91 figure
92 hold off
93
94 % Plot actual RAAN
95 plot(t, RAAN_Array)
96
97 %plot expected change based on secular change
98 hold on
99 plot([0,max(t)], [RAAN, RAAN+RAAN_secular_change])
100
101 % Graph formatting and saving
102 legend('Simulated RAAN', 'expected change in RAAN','Location','best')
103 xlabel('time (s)')
104 ylabel('RAAN (^o)')

```

```

105 title(strcat("PROBA-2 RAAN with J_2 Pertubations at ",num2str(i),'^\circ{
    ↳ Inclined Orbit'))
106 xlim("tight")
107 ylim("tight")
108 print(strcat(save_to,'RAAN_',num2str(i),'_degrees.eps'), '-depsc')
109
110
111 %% Part D
112 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113 % Calculate simulated secular and rate of change in RAAN
114 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
115
116 % Calculate change in RAAN from simulation
117 simulated_RAAN_secular_change = ...
118     (RAAN_Array(length( RAAN_Array)) - RAAN_Array(1))/orbits;
119
120 % Calculated rate of change change in RAAN from simulation
121 simulated_RAAN_average_rate_of_change = ...
122     simulated_RAAN_secular_change/(T/orbits);
123
124 %% Part E
125 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
126 % Tabulize expected and simulated secular and rate of change in RAAN
127 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
128
129 % Calculate differences
130 RAAN_secular_difference = abs(RAAN_secular_change-simulated_RAAN_secular_change);
131 RAAN_roc_difference =
132     ↳ abs(RAAN_average_change-simulated_RAAN_average_rate_of_change);
132 varNames = ["secular_change", "rate_of_change"];
133 varTypes = repmat("double", 2,1);
134 sz = [3 2];
135 comparison_table =
136     ↳ table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);
136
137 comparison_table.secular_change = ...
138 [RAAN_secular_change;simulated_RAAN_secular_change;RAAN_secular_difference];
139
140 comparison_table.rate_of_change = ...
141 [RAAN_average_change; simulated_RAAN_average_rate_of_change;
142     ↳ RAAN_roc_difference];
142
143 comparison_table.Properties.VariableNames = ["secular change (degrees/rev)",...
144     "average rate of change(degrees/s)"];
145 comparison_table.Properties.RowNames = ["expected","simulation", "difference"];
146

```

```

147 writetable(comparison_table, strcat(save_to, 'RAAN_check_for_inc_', num2str(i), '.csv'), 'WriteRowNames',
148

```

Code Attachment 5: Script used in question 3.1

C.2 SIMULINK diagram

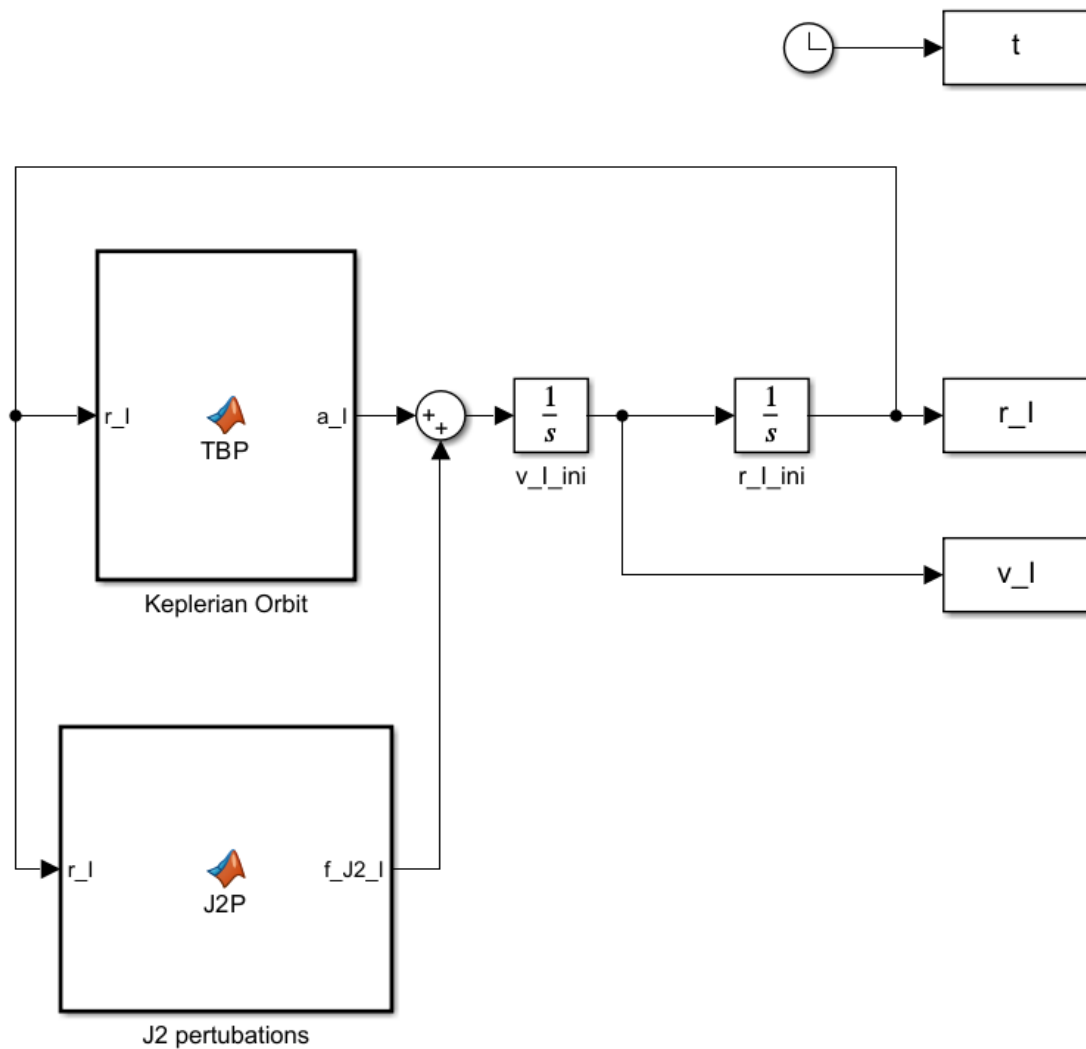


Figure 27: SIMULINK model for question 3.1

C.3 additional functions

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

C.3 a J₂ Perturbation acceleration function

```
1 function f_J2_I = J2P(rE, J_2, mu, r_I)
2
3 % -----
4 % J2P
5 % -----
6 % Description:
7 % Compute the spacecraft acceleration due to J2 perturbation ECI
8 % -----
9 % Inputs:
10 % r_I => components of r in ECI (3 x 1 matrix)
11 % -----
12 % Outputs:
13 % f_J2_I => components of the acceleration in ECI (3 x 1 matrix)
14 % -----
15 % Parameters:
16 % mu => gravitational constant of the Earth
17 % J_2 => J2 coefficient for oblate body of earth
18 % rE => Radius of earth
19 % -----
20
21 % J_2 equation
22 f_J2_I = 3*mu*J_2*rE^2/(2*(norm(r_I)^5))...
23     *(...
24     (5*(r_I(3)^2)/(norm(r_I)^2)-1)*r_I - 2*[0; 0; r_I(3)]...
25     );
26
27 end
```

Code Attachment 6: Function to calculate J₂ perturbation in SIMULINK

Appendix D MATLAB code for 5.1

D.1 Main Script

```
1 clc
2 clear;
3 close all;
```



```

4
5
6  % save folder
7  mydir = pwd;
8  idcs = strfind(string(mydir),'\');
9  newdir = mydir(1:idcs(end)-1);
10 save_to = strcat(newdir,'\output files\5.1\');
11 clear mydir idcs newdir;
12
13 global n
14 ALONG_TRACK = 1;
15 IN_TRACK = 2;
16 IN_PLANE_ELIPTICAL = 3;
17 ORBIT_I = 4;
18 ORBIT_II = 5;
19
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 formation_type = ORBIT_I;
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 % Gravitational parameter
24 mu = 398600.4418; % km^3/s^2
25
26 % Earth's radius
27 rE = 6378; % km
28
29 % Orbital elements of Envisat
30 a = rE+766; % km
31 e = 1E-4;
32 i = 98.4; % deg
33 RAAN = 27.2; % deg
34 w = 71; % deg
35
36 % Earth angular velocity
37 wE = 72.922e-6; % rad/s
38
39 % Mean motion
40 n = sqrt(mu/a^3); % rad/s
41
42 % Orbital period
43 T = 2*pi/n ; % sec
44
45 % Initial conditions of relative motion
46 if formation_type == ALONG_TRACK
47     % Calculate initial conditions of along track formation
48     name = 'Along Track';
49     x0 = 0; % km

```

```

50     z0 = 0; % km
51     vx0 = 0; % km
52     vy0 = 0; % km
53     vz0 = 0; % km
54
55     y0 = -10; % km
56 elseif formation_type == IN_TRACK
57     % Calculate initial conditions of in track formation
58     name = 'In Track';
59     x0 = 0; % km
60     vx0 = 0; % km
61     vy0 = 0; % km
62     vz0 = 0; % km
63
64     y0 = -10; % km
65     z0 = -(wE/n*sind(i))*y0; % km
66 elseif formation_type == IN_PLANE_ELIPTICAL
67     % Calculate initial conditions of in plane elliptical formation
68     name = 'In Plane Elliptical';
69     alpha = 0; % deg
70     A_0 = 0.5/2; %km
71
72     z0 = 0; % km
73     vz0 = 0; % km
74
75     % solve in plane elliptical equations
76     syms vx0 x0
77     eqns = [
78         alpha == atand(-vx0/(n*x0));
79         A_0 == sqrt((vx0/n)^2+x0^2)
80     ];
81
82     [vx0, x0] = vpasolve(eqns,[vx0 x0]);
83
84     vx0 = round(double(vx0), 10);
85     x0 = round(double(x0), 10);
86     vy0 = -2*n*x0; % km
87     y0 = 2*vx0/n; % km
88 elseif formation_type == ORBIT_I
89     % Calculate initial conditions of 5.1) E)I formation
90     name = 'Circular Formation, Radius 20m, Phasing Angle 0^\circ';
91     r = 20E-3; % km
92     phase = 0; % deg
93
94     x0 = (r/2)*cosd(phase); %km
95     vx0 = -(r*n/2)*sind(phase); %km

```

```

96     z0 = sqrt(3)*x0; % km
97     vz0 = sqrt(3)*vx0; % km
98
99     vy0 = -2*n*x0; % km
100    y0 = 2*vx0/n; % km
101    elseif formation_type == ORBIT_II
102        % Calculate initial conditions of 5.1) E)II formation
103        name = 'Projected Circular Formation, Radius 20m, Phasing Angle 270^\circ';
104        r = 20E-3; % km
105        phase = 270; % deg
106
107        x0 = (r/2)*cosd(phase); %km
108        vx0 = -(r*n/2)*sind(phase); %km
109        z0 = 2*x0; % km
110        vz0 = 2*vx0; % km
111
112        vy0 = -2*n*x0; % km
113        y0 = 2*vx0/n; % km
114    end
115    % Naming of output graphs
116    formation_type = num2str(formation_type);
117
118    %creating table of initial conditions
119    initial_conditions = array2table([vx0 vy0 vz0 x0 y0 z0]);
120    varNames = ["vx0", "vy0", "vz0", "x0", "y0", "z0"];
121    initial_conditions.Properties.VariableNames = varNames;
122    writetable(initial_conditions, strcat(save_to, 'initial_conditions_', formation_type, '.csv'))
123
124    % Integration
125    options=odeset('RelTol',1e-9,'AbsTol',1e-9);
126    [t,X]=ode45(@CWHdyn,[0:10:2*T],[vx0 vy0 vz0 x0 y0 z0]',options);
127
128    vx = X(:,1);
129    vy = X(:,2);
130    vz = X(:,3);
131    x = X(:,4);
132    y = X(:,5);
133    z = X(:,6);
134
135    % Plot components of the relative position vector as function of time
136    figure
137    subplot(4,1,1)
138    plot(t/T,x*1000,'k');
139    title(name)
140    xlabel('Number of orbits')
141    ylabel('x (m)')

```

```

142 grid on
143 subplot(4,1,2)
144 plot(t/T,y*1000,'k');
145 xlabel('Number of orbits')
146 ylabel('y (m)')
147 grid on
148 subplot(4,1,3)
149 plot(t/T,z*1000,'k');
150 xlabel('Number of orbits')
151 ylabel('z (m)')
152 grid on
153 subplot(4,1,4)
154 plot(t/T,sqrt((x*1000).^2+(y*1000).^2+(z*1000).^2),'k');
155 xlabel('Number of orbits')
156 ylabel('Distance (m)')
157 axis([0 2 0 40]);
158 grid on
159
160
161 print(strcat(save_to,'Position_component_plots_',formation_type,'.eps'),
    ↪ '-depsc') % save figure
162
163 % Plot the components of the relative position vector in 3D
164 figure
165 plot3(z*1000, y*1000, x*1000,'k');
166 if str2double(formation_type) == ALONG_TRACK
167     hold on
168     scatter3(z*1000, y*1000, x*1000,20,'k');
169 end
170 hold on
171 scatter3(0,0,0,20,'k');
172 xlabel('z (m)')
173 ylabel('y (m)')
174 zlabel('x (m)')
175 title(name)
176 grid on
177 set(gca,'XDir','reverse')
178
179 print(strcat(save_to,'3D-plot_of_path_around_target_',formation_type,'.eps'),
    ↪ '-depsc') % save figure
180
181 % Plot and animate the in-plane and out-of-plane motion
182 planes = figure
183 set(planes,'Position',[556 33 454 665])
184 subplot(3,1,1)
185 plot(y*1000,x*1000,'k');

```

```

186 ip(1) = line(y(1)*1000, x(1)*1000, 'Marker', '.', 'MarkerSize', 22, 'Color',
    ↪ 'k');
187 ip(2) = line(0, 0, 'Marker', 'o', 'MarkerSize', 6, 'LineWidth', 1.5, 'Color', 'k'
    ↪ );
188 axis([-70 70 -30 30]);
189 set(gca, 'XDir', 'reverse')
190 set(gca, 'FontSize', 10, 'FontName', 'Times')
191 xlabel('y (m)')
192 ylabel('x (m)')
193 title(name)
194 grid on
195
196 subplot(3,1,2)
197 plot(z*1000, x*1000, 'k');
198 axis([-70 70 -30 30]);
199 oop(1) = line(z(1)*1000, x(1)*1000, 'Marker', '.', 'MarkerSize', 22, 'Color',
    ↪ 'k');
200 oop(2) = line(0, 0, 'Marker', 'o', 'MarkerSize', 6, 'LineWidth', 1.5, 'Color',
    ↪ 'k');
201 set(gca, 'XDir', 'reverse')
202 set(gca, 'FontSize', 10, 'FontName', 'Times')
203 xlabel('z (m)')
204 ylabel('x (m)')
205 grid on
206
207 subplot(3,1,3)
208 plot(z*1000, y*1000, 'k');
209 axis([-70 70 -30 30]);
210 ct(1) = line(z(1)*1000, y(1)*1000, 'Marker', '.', 'MarkerSize', 22, 'Color',
    ↪ 'k');
211 ct(2) = line(0, 0, 'Marker', 'o', 'MarkerSize', 6, 'LineWidth', 1.5, 'Color',
    ↪ 'k');
212 set(gca, 'XDir', 'reverse')
213 set(gca, 'FontSize', 10, 'FontName', 'Times')
214 xlabel('z (m)')
215 ylabel('y (m)')
216 grid on
217 print(strcat(save_to, 'Path_in_projected_planes_', formation_type, '.eps'),
    ↪ '-depsc')
218
219 for i = 1:length(x)
220     set(ip(1), 'XData', y(i)*1000, 'YData', x(i)*1000);
221     set(oop(1), 'XData', z(i)*1000, 'YData', x(i)*1000);
222     set(ct(1), 'XData', z(i)*1000, 'YData', y(i)*1000);
223     drawnow
224 end

```

Code Attachment 7: Script used in question 5.1

D.2 Additional Functions

D.2 a Hill's Equation function

```
1 function X = CWHdyn(t,states)
2
3 global n
4
5 vx = states(1);
6 vy = states(2);
7 vz = states(3);
8 x = states(4);
9 y = states(5);
10 z = states(6);
11
12 % Hill's equations of motion
13 x_ddot = 3*n^2*x + 2*n*vy;
14 y_ddot = -2*n*vx;
15 z_ddot = -n^2*z;
16 x_dot = vx;
17 y_dot = vy;
18 z_dot = vz;
19
20 X = [x_ddot y_ddot z_ddot x_dot y_dot z_dot]';
21
22 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

```
1 function rotation_matrix_x = C_1(rotation_angle)
2 %C_1 roatates by given angle in degrees around x axis
3
4 rotation_matrix_x = [1          0          0;
```

```

5         0 cosd(rotation_angle) sind(rotation_angle);
6         0 -sind(rotation_angle) cosd(rotation_angle)
7     ];
8 end

```

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

E.1 b rotation around y

```

1 function rotation_matrix_y = C_2(rotation_angle)
2 %C_2 roatates by given angle in degrees around y axis
3
4 rotation_matrix_y = [cosd(rotation_angle) 0 -sind(rotation_angle);
5                     0 1 0 ;
6                     sind(rotation_angle) 0 cosd(rotation_angle)
7 ];
8 end

```

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

E.1 c rotation around z

```

1 function rotation_matrix_z = C_3(rotation_angle)
2 %C_3 roatates by given angle in degrees around z axis
3
4 rotation_matrix_z = [cosd(rotation_angle) sind(rotation_angle) 0;
5                     -sind(rotation_angle) cosd(rotation_angle) 0;
6                     0 0 1
7 ];
8 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

```

1 %The following code must be run in matlab live after calculating r_I in
2 %PROBA2.m
3

```

```

4  figure
5
6  grid on
7
8  earth_sphere(gca, 'km')
9
10 set(gca, 'FontSize', 9, 'FontName', 'Times')
11 title('PROBA 2 Orbital Path in ECI Reference Frame')
12 xlabel('rx_I (km)')
13 ylabel('ry_I (km)')
14 zlabel('rz_I (km)')
15 axis equal
16 view([62 32])
17 for i = 1:500:length(r_I(:,1))
18     hold on
19     plot3(r_I(1:i,1), r_I(1:i,2), r_I(1:i,3), 'k', 'linewidth', 2)
20     pause(0.00001);
21     drawnow;
22 end
23 plot3(r_I(:,1), r_I(:,2), r_I(:,3), 'k', 'linewidth', 2)
24 drawnow;

```

Code Attachment 12: Script to generate orbit animation video

E.2 b Video Animate ground tracks

```

1  %The following code must be run in matlab live after calculating longitude
2  % and latitude in PROBA2.m
3
4  figure
5  plot(longitude(1), latitude(1), 'k', 'LineWidth', 2);
6  hold on
7
8
9  ylim([-90 90])
10 xlim([-180 180])
11 title('PROBA 2 Ground Track')
12 xlabel('longitude(^{\circ})')
13 ylabel('latitude(^{\circ})')
14 I = imread('map.png');
15 image(xlim, ylim, I);
16 dt = [diff(t, 500); 0];
17 for i = 1:1000:length(longitude)
18     hold on
19     plot(longitude(1:i), latitude(1:i), 'k', 'LineWidth', 2);

```



```
20     drawnow;
21     pause(1/1000);
22
23
24 end
25     plot(longitude, lattitude, 'k', 'LineWidth', 2);
26 drawnow;
```

Code Attachment 13: Script to generate ground tracks animation video