MAE 3260 Final Group Work: Exploring a System of Interest

Report

Outline: Option #1

Title: State-spacemen

Topic of Interest: Autonomous orbital rendezvous using state space and PD control

Abstract: This project models the relative dynamics of a chaser satellite approaching a target satellite using the Clohessy-Wilstshire equations expressed as a state space model. We selected this system to demonstrate PD control of a MIMO system with coupled differential equations. In our work, we derive the A and B matrices of our state space model while implementing PD control tuned to achieve a gentle “docking” of the two satellites with optimal settling time. We will assume a circular orbit with zero inclination, relatively close proximity of the two satellites, zero disturbance, and zero sensor input noise in order to simplify our model.

Students/Roles:

Student Name	Task/Role	Portfolio Link
Jacob Miller	Researched defining equations and key system simplifications / constraints Linearized the CW equations and created a state space model Modeled the system in MATLAB, and created all MATLAB animations (linked below)	
Andrew Fujimoto	Created initial draft of block diagram Formatted ODE and state space matrices in final writeup Researched alternate uses for this model	https://cornell-mae-ug.github.io/fa25-portfolio-Andrew-Fujimoto/projects/Orbital-Rendezvous/
Qing Yi Chen	Produced block diagram image for model Helped format ODEs and state space model Helped define general layout of solving system Created references list in IEEE citation format Helped edit and proofread report write up	https://qingyichen-cornell-meche-portfolio.my.canva.site/autonomous-orbital-rendezvous-project
Juhi Singh	Helped define initial problem set up including assumptions and design parameters in context of practical applications Helped research and identify C-W model for use in ODEs and state space model	https://cornell-mae-ug.github.io/fa25-portfolio-js3669/projects/MAE-3260-Orbital-Rendezvous/

	Researched thrusters for spacecraft	

List of MAE 3260 concepts or skills used in this group work:

- Models
    - ODEs
    - TFs
    - State space model
    - Block diagrams
- Steady state behavior
- Step response
- Feedback control (PD control)

Description of System and Assumptions:

Description:

The system that we are modelling is an orbital rendezvous, which is a space maneuver where one spacecraft, the "chaser," arrives at the orbit of another spacecraft, the "target", with the two spacecraft coming into physical contact and docking together. The model we use assumes a circular target orbit and an elliptical or circular chaser orbit.

Applications:

In addition to the docking of two satellites, these equations can model many other situations in space. For example, some additional applications might include:

- An orbital vehicle latching onto a piece of space trash for disposal, such as a decommissioned satellite.
- A space mining vehicle approaching an asteroid to extract resources.
- A hostile satellite interfering with a target with the intent to maim, destroy, or otherwise incapacitate the target.

However, this model breaks down for other orbital applications, such as rendezvous with bodies that have a significant gravitational field.

Assumptions:

- The two orbits are in same plane → 2D system
- The Earth is a sphere
- Spacecraft are unaffected by other gravitational fields (i.e the moon)
- No system disturbance
- No sensor input noise

Design Criteria and Practical Considerations:

- There must be no overshoot, because otherwise the spacecraft will either crash or not dock immediately.
- There must be zero steady-state error, so that they can dock together.
- The final relative velocity should be approximately zero.

Thrusters:
Because the system is modelled as a step input response, we would require a thruster that is able to provide a continuous, approximately constant thrust force as opposed to an instantaneous one (impulse input) or one that decreases over time. In a practical scenario, the engineers would need to consider how long the thrusters need to be operating for, and so an orbital rendezvous would be timed appropriately such that the distance between the two spacecraft is minimized.

Preliminary research shows there are two primary types of spacecraft propulsion: chemical and electric. Chemical propulsion is used in situations that require very high and immediate thrust, whereas electric propulsion creates lower thrust but at a very high efficiency. Hybrid systems are therefore becoming more common, allowing satellites to perform maneuvers quickly but also make finer adjustments in situations like docking, where accuracy of alignment is very important. A hybrid propulsion system is therefore a potential option for our application.

Model:

Referencing previous literature on the topic, we used the Clohessy-Wiltshire equations, which describe a simplified model of orbital motion with the intended purpose of designing control systems for orbital rendezvous. The model simplifies the full, nonlinear equations of relative motion into a linear system, so it is only valid for small relative distances (relative to their altitude) and a circular reference orbit. As a further simplification to the model, we are ignoring the third equation that controls the z-coordinate when the two orbits and at a relative angle. The coordinate system is centered about the target, so both x and y are representations of the spacecraft's relative position.

Variables:
x = radial component (meters)
y = tangential component (meters)
n = $\sqrt{\frac{\mu}{a^3}}$ = orbital rate (radians/second), where μ is the standard gravitational parameter, and a is the radius of the target body's orbit.

ODEs:
$\ddot{x} - 2n\dot{y} - 3n^2x = u_x$
$\ddot{y} + 2n\dot{x} = u_y$

State Space Model:
$\dot{x} = A\vec{x} + B\vec{u}$
$\vec{Y} = C\vec{x} + D\vec{u}$

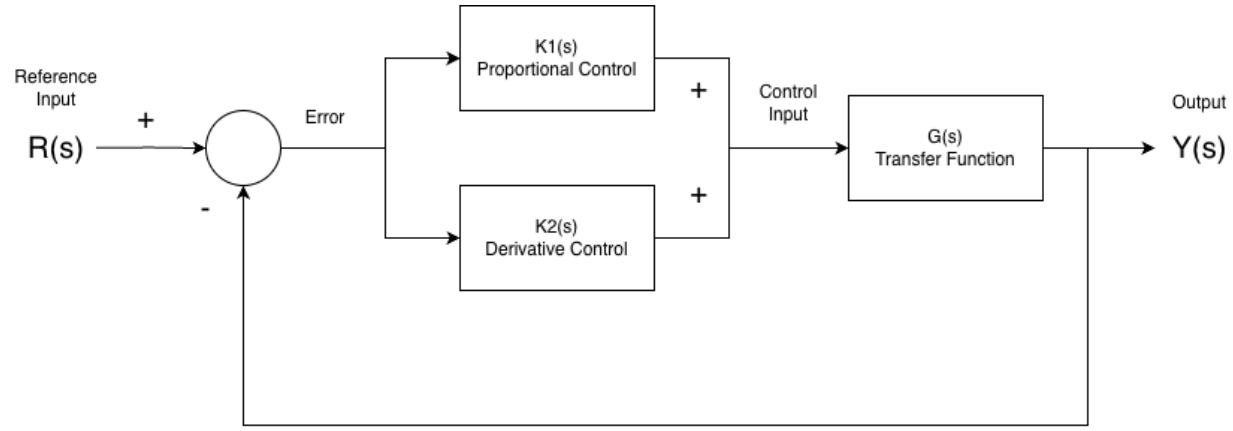
$$\vec{x} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \qquad \vec{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 2n \\ 0 & 0 & -2n & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Controller:

Because we are assuming zero disturbance in the system, integral control is not required to achieve a steady state error of zero. Therefore, we decided to use a PD controller.

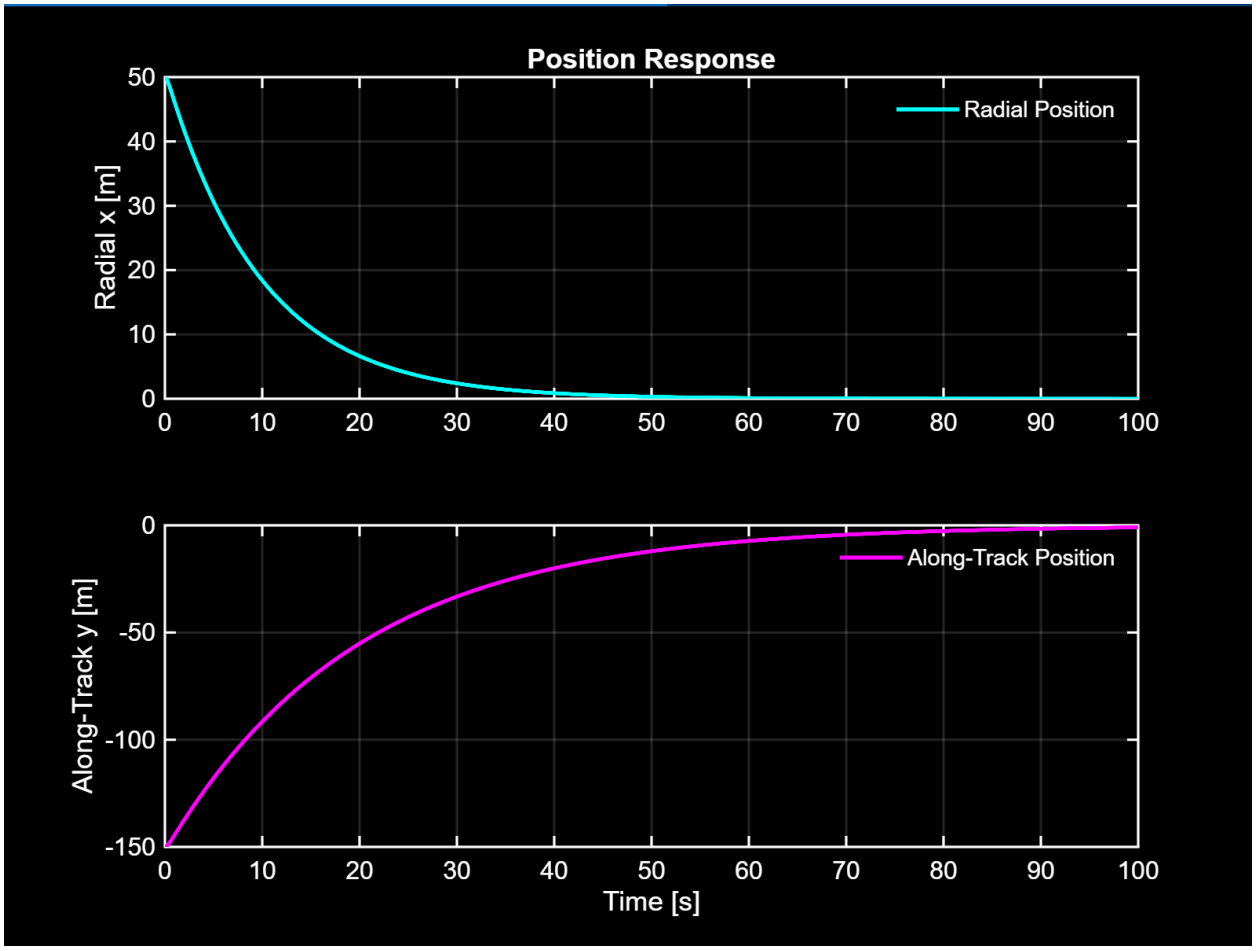
Block Diagram:

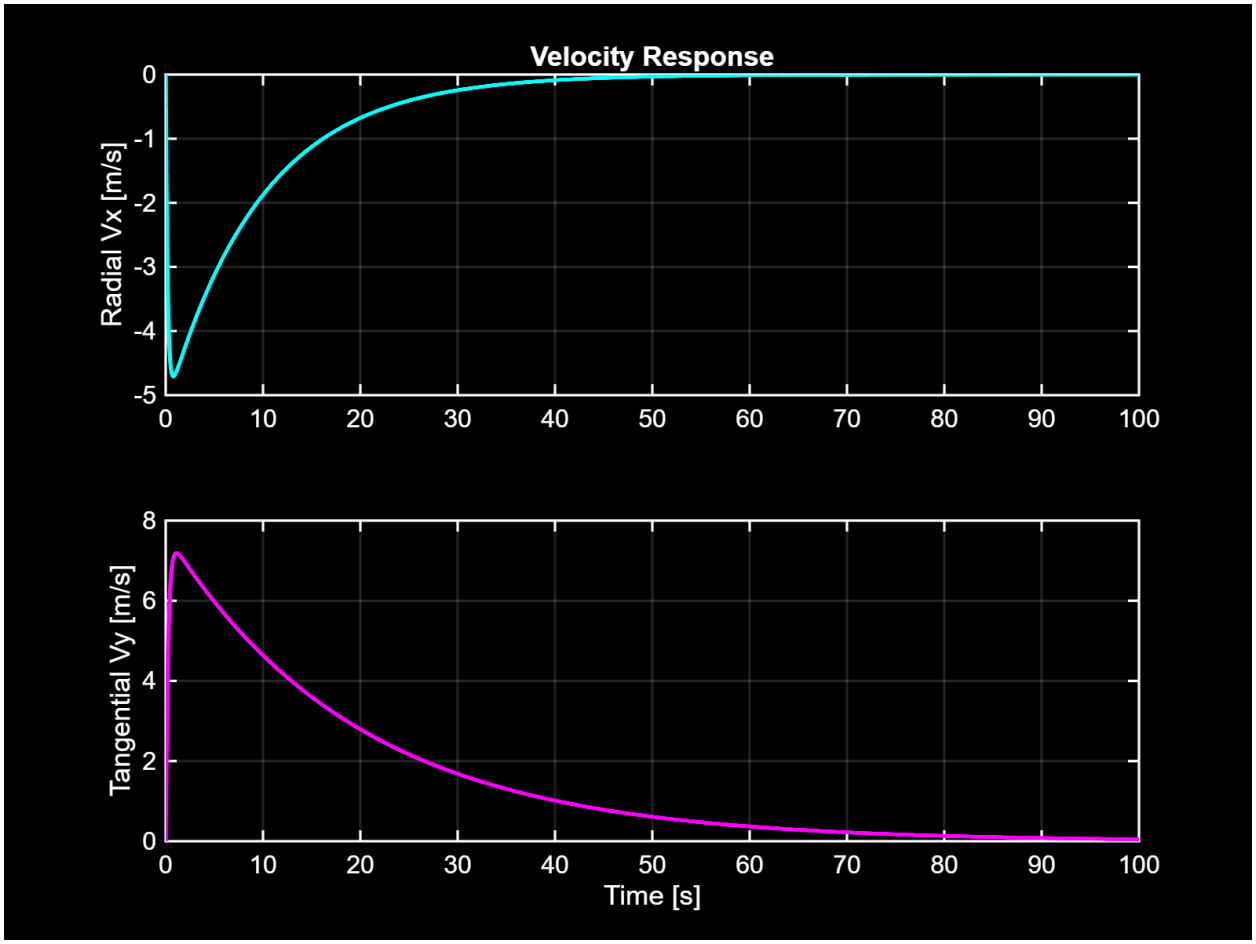


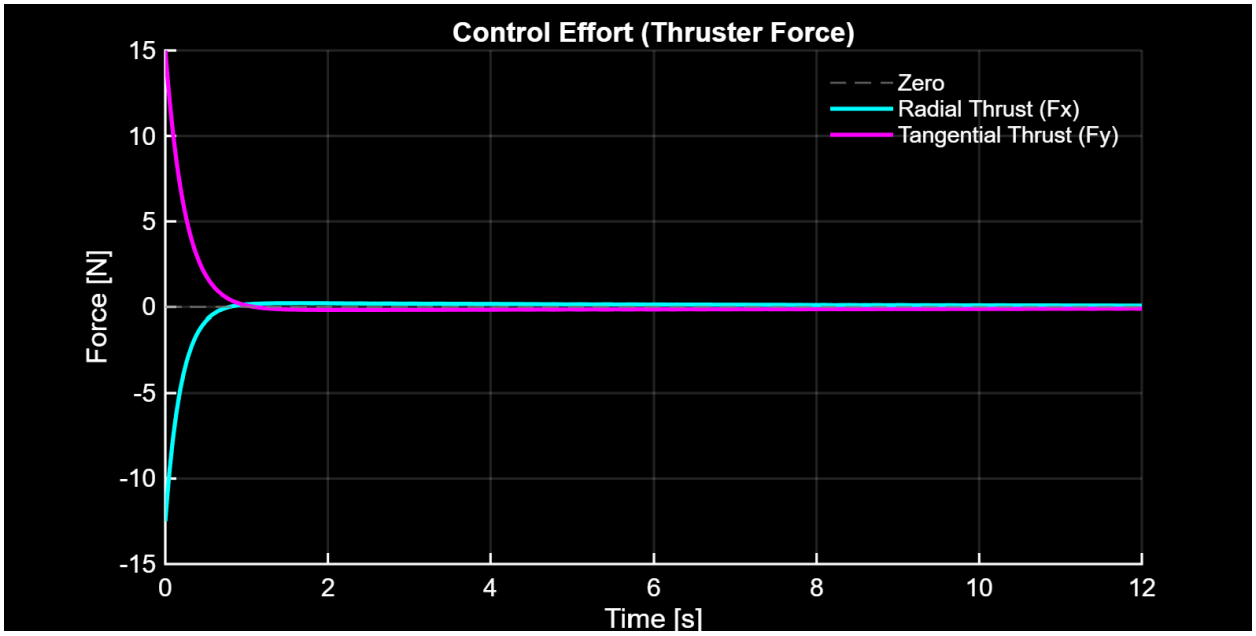
*Assume no disturbance
*Assume no sensor input noise

MATLAB Results and Animation (*please watch the animation*):

Watch the MATLAB animation video here:
https://drive.google.com/file/d/1KRw7WH9nfSKPOZuKE7A35M7JOyQRokmw/view?usp=sharing


Position Response
Radial Position
Radial x [m]
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
Along-Track Position
Along-Track y [m]
0
-50
-100
-150
Time [s]


Velocity Response
Radial Vx [m/s]
0
-1
-2
-3
-4
-5
0
10
20
30
40
50
60
70
80
90
100
Tangential Vy [m/s]
8
6
4
2
0
0
10
20
30
40
50
60
70
80
90
100
Time [s]


Control Effort (Thruster Force)
Zero
Radial Thrust (Fx)
Tangential Thrust (Fy)
Force [N]
15
10
5
0
-5
-10
-15
0
2
4
6
8
10
12
Time [s]

References:

[1] ensatellite, “Hill’s Equations: Derivation and Solution”, *ensatellite*. [Online]. Available: https://ensatellite.com/hills-equations/

[2] L. E. George, “Chapter 8 – Rendezvous,” in *Introduction to Orbital Mechanics*, Colorado Springs, CO: Pressbooks, 2023. Available: https://oer.pressbooks.pub/lynnanegeorge/chapter/chapter-8-ground-tracks-and-rendezvous/

[3] The Lee Company, “Comprehensive Overview of Satellite Propulsion Systems”, *The Lee Company.* [Online]. Available: https://www.theleeco.com/insights/comprehensive-overview-of-satellite-propulsion-systems/

Additionally, we used Google Gemini to help organize and refine our code.

Appendix:

MATLAB Code (created with Gemini assistance):

```
%% Satellite Docking Animation (Dual View)
% Description: Simulates docking with Hill's (CW) equations.
% Includes a Relative Frame (Docking Cam) and an Inertial Frame (Orbit Cam)
% Frame: Local Vertical Local Horizontal (LVLH)
clear; clc; close all;
%% --- 1. CONFIGURATION ---
% Physics
m_chaser = 500;         % Mass (kg)
R_earth = 6378;         % Earth Radius (km)
h_orbit = 500;          % Orbit Altitude (km)
R_sat = R_earth + h_orbit;
n = sqrt(3.986e5/R_sat^3);           % Mean motion
% Initial State (Relative positions in Kilometers)
% X = Radial (Up), Y = Along-Track (Forward)
r0_x = 0.05;            % 50 meters above
r0_y = -0.15;           % 150 meters behind
v0_x = 0.0;
v0_y = 0.0;
% PD Controller Gains (Overdamped approach)
Kp_x = 0.5; Kd_x = 5.0;
Kp_y = 0.2; Kd_y = 4.0;
% Timing
t_total = 120;          % Duration (s)
dt = 0.05;              % Time step (s)
```

```
t_start_control = 5.0;  % Wait 5s before engaging
% Visualization
sat_scale = 0.008;      % Size of satellite drawing in Relative View
vis_mag = 5000;         % Exaggeration factor for Orbital View separation
%% --- 2. DYNAMICS SETUP (CW Equations) ---
A = [0     0     1     0;
    0     0     0     1;
    3*n^2 0     0     2*n;
    0     0    -2*n   0]; %Neglect z direction
B = [0 0; 0 0; 1 0; 0 1];
K = [Kp_x 0 Kd_x 0; 0 Kp_y 0 Kd_y];
z = [r0_x; r0_y; v0_x; v0_y];
%% --- 3. GRAPHICS INITIALIZATION ---
fig = figure('Name', 'Docking Mission', 'Color', 'k', ...
             'Position', [50, 50, 1400, 700]);
tlo = tiledlayout(1, 2, 'Padding', 'compact', 'TileSpacing', 'compact');
% --- VIEW 1: RELATIVE FRAME (DOCKING) ---
ax1 = nexttile;
set(ax1, 'Color', 'k', 'XColor', 'w', 'YColor', 'w', ...
   'NextPlot', 'add', 'DataAspectRatio', [1 1 1]);
grid(ax1, 'on');
set(ax1, 'GridColor', 'w', 'GridAlpha', 0.15);
xlabel(ax1, 'Radial (km)'); ylabel(ax1, 'Along-Track (km)');
title(ax1, 'Relative Frame (Docking Cam)', 'Color', 'w');
% Geometry: Target (Center)
tgt_body_x = [-1 1 1 -1] * sat_scale;
tgt_body_y = [-1 -1 1 1] * sat_scale;
patch(ax1, tgt_body_x, tgt_body_y, [0 0.5 1], 'EdgeColor', 'w');
% Geometry: Chaser (Rotated: Flat side +Y, Point -Y)
% Broad end facing +Y (Target is at 0, Chaser approaches from -Y)
chaser_w = 0.8 * sat_scale;
chaser_h = 0.8 * sat_scale;
chaser_shape_x = [-chaser_w,  chaser_w,       0];
chaser_shape_y = [ chaser_h,  chaser_h, -chaser_h*1.5]; % Flat top, pointy bottom
h_chaser = patch(ax1, chaser_shape_x + z(1), chaser_shape_y + z(2), ...
                [1 0.2 0], 'EdgeColor', 'w', 'LineWidth', 2);
h_trail = plot(ax1, z(1), z(2), 'm:', 'LineWidth', 1.5);
% HUD Text
h_info = text(ax1, 0.05, 0.95, 'Init...', 'Units', 'normalized', ...
             'Color', 'w', 'FontName', 'Courier', 'FontSize', 11);
% Limits
margin = 0.05;
xlim(ax1, [min(0, r0_x)-margin, max(0, r0_x)+margin]);
ylim(ax1, [min(0, r0_y)-margin, max(0, r0_y)+margin]);
% --- VIEW 2: INERTIAL FRAME (ORBIT) ---
ax2 = nexttile;
set(ax2, 'Color', 'k', 'XColor', 'w', 'YColor', 'w', ...
```

```
   'NextPlot', 'add', 'DataAspectRatio', [1 1 1]);
axis(ax2, 'off');
title(ax2, 'Inertial Frame (Orbit Cam)', 'Color', 'w');
% Draw Earth
theta = linspace(0, 2*pi, 100);
fill(ax2, R_earth*cos(theta), R_earth*sin(theta), [0 0.1 0.3], 'EdgeColor',
'none');
plot(ax2, R_earth*cos(theta), R_earth*sin(theta), 'c', 'LineWidth', 1);
% Draw Orbit Path
plot(ax2, R_sat*cos(theta), R_sat*sin(theta), 'w--', 'Color', [0.3 0.3 0.3]);
% Satellites (Dots)
h_sat_tgt = plot(ax2, 0, R_sat, 'co', 'MarkerFaceColor', 'c', 'MarkerSize', 6);
h_sat_chaser = plot(ax2, 0, R_sat, 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 6);
% Zoom text
text(ax2, 0, 0, 'EARTH', 'Color', [0 0.2 0.5], 'HorizontalAlignment', 'center');
text(ax2, 0.05, 0.05, sprintf('Visual Separation x%d', vis_mag), ...
    'Units', 'normalized', 'Color', 'y', 'FontSize', 8);
axis(ax2, [-1 1 -1 1] * (R_sat + 1000));
drawnow;
%% --- 4. SIMULATION LOOP ---
history_x = [];
history_y = [];
sim_start = tic;
for t = 0:dt:t_total

   % --- Control ---
   if t < t_start_control
       u = [0; 0];
       mode = 'DRIFT';
   else
       u = -K * z;
       mode = 'AUTO';
   end

   % --- Dynamics ---
   z_dot = A * z + B * u;
   z = z + z_dot * dt;

   % --- Update View 1 (Relative) ---
   set(h_chaser, 'XData', chaser_shape_x + z(1), ...
                 'YData', chaser_shape_y + z(2));
   history_x = [history_x, z(1)];
   history_y = [history_y, z(2)];
   set(h_trail, 'XData', history_x, 'YData', history_y);

   dist = norm(z(1:2));
```

```
    % --- Update View 2 (Inertial) ---
    % Calculate true anomaly for Target (assuming circular orbit)
    nu = n * t + pi/2; % Start at top of orbit (90 deg)

    % Target Position (Inertial)
    pos_tgt_I = [R_sat * cos(nu); R_sat * sin(nu)];

    % Chaser Position (Inertial approximation)
    % Map LVLH (x=radial, y=along-track) to Inertial Frame
    % We exaggerate the LVLH offsets so they are visible on the full orbit
    % Radial (x) adds to Radius. Along-track (y) adds to Angle.

    r_vis = R_sat + z(1) * vis_mag; % Exaggerated radius change
    nu_vis = nu - (z(2) * vis_mag / R_sat); % Exaggerated angle change (-y is
trailing)

    pos_chaser_I = [r_vis * cos(nu_vis); r_vis * sin(nu_vis)];

    set(h_sat_tgt, 'XData', pos_tgt_I(1), 'YData', pos_tgt_I(2));
    set(h_sat_chaser, 'XData', pos_chaser_I(1), 'YData', pos_chaser_I(2));

    % --- HUD ---
    info_str = {
        sprintf('T: %5.1f', t);
        sprintf('Range: %5.1f m', dist*1000);
        sprintf('V_rel: %5.2f m/s', norm(z(3:4))*1000);
    };
    set(h_info, 'String', info_str);

    % Color Logic for Contact
    if dist < (sat_scale/2)
        set(h_chaser, 'FaceColor', 'g');
    end

    % --- Timing ---
    elapsed = toc(sim_start);
    if t > elapsed
        pause(t - elapsed);
    else
        drawnow limitrate;
    end

    if ~ishandle(fig), break; end
end
```