Final Project Report

ASEN 5014: Linear Control Systems

Jack Huston & Tsuening Lee

University of Colorado Boulder
November 24th, 2025

1 TEAM MEMBER CONTRIBUTIONS

Jack Huston	**Tsuening Lee**
Primarily responsible for the development of the deliverables for Part B, and also contributed to the writeup and scripting for Part A.	Primarily responsible for the development of the deliverables for Part A, and also contributed to the writeup and review of Part B.

2 PART A: LINEAR SYSTEMS ANALYSIS

2.1 System Design and Equation Derivation

The linearized system can be described as follows:

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

with the following A, B, C, and D matrices defining the model:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{3\mu}{r_0^3} & 0 & 0 & 2\sqrt{\frac{\mu}{r_0}} \\ 0 & 0 & 0 & 1 \\ 0 & -2\sqrt{\frac{\mu}{r_0^5}} & 0 & 0 \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & \frac{1}{r_0} \end{bmatrix} \tag{4}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{6}$$

The state vector and control inputs represent small perturbations about a nominal circular orbit of radius r_0:

$$x = [\delta r \ \dot{\delta r} \ \delta\theta \ \dot{\delta\theta}]^T, \qquad u = [\delta u_1 \ \delta u_2]^T. \tag{7}$$

Here, δr and $\delta\theta$ denote the radial and in-track angular deviations from the reference orbit, while $\dot{\delta r}$ and $\dot{\delta\theta}$ represent the corresponding velocity perturbations. The inputs δu_1 and δu_2 are small thrust-generated accelerations in the radial and in-track directions. The measured outputs are

$$y = [\delta r \ \delta\theta]^T, \tag{8}$$

representing the sensed deviations in orbital radius and angular position.

This linear model arises from a first-order perturbation of the nonlinear orbital dynamics and is valid for small deviations around the circular reference orbit. It captures the dominant in-plane gravitational coupling and the effect of radial and tangential thrust on local relative motion. The purpose of the physical system is to maintain a desired orbit despite disturbances or initial offsets, and the model provides a useful approximation for designing such controllers.

Several simplifying assumptions limit the fidelity of the state-space representation. The model is strictly two-dimensional and therefore does not capture out-of-plane motion, plane changes, or cross-coupling between orbital elements. Earth is treated as a point-mass body, so perturbations such as J_2 oblateness, atmospheric drag, third-body effects, and solar radiation pressure are neglected. Additionally, actuator and sensor dynamics are idealized: thrust saturation, delays, and measurement noise are not represented. Finally, because the equations are linearized, the model loses accuracy for large deviations where nonlinear orbital mechanics dominate. Despite these limitations, the system accurately reflects the essential in-plane dynamics needed for analyzing small perturbation stabilization and orbit-holding performance.

2.2 Open Loop Response and Closed Loop Requirements

To build on the system description in Part A.1, the control objective for this project is to stabilize the spacecraft about its nominal circular orbit and to reject small perturbations introduced through initial condition offsets. This is demonstrated by identifying a system that drives all perturbation states in $x(t)$ to zero so that the radius and in-track angle remain at their desired reference values.

The desired closed-loop performance is characterized by the following time-domain specifications:

1. All closed-loop poles must lie strictly in the left-half complex plane (asymptotic stability);
2. The radial and angular response $y(t)$ must reach 95% of the reference value within one orbital period;
3. The radial and angular overshoot/undershoot of the desired value must remain below 20%;
4. The radial and angular steady-state tracking error must converge to zero.
5. The spacecraft acceleration should not exceed $0.01\,g$ for the initial deviation given (Assuming 1 lbf thruster and 100 lbm spacecraft bus)

These requirements imply that the dominant closed-loop poles must have sufficiently negative real parts to enforce the required settling time over one orbital period, while having adequate damping to satisfy the overshoot bound.

Before designing feedback, we examine the open-loop dynamics. Using the MATLAB command `eig(A)` (see Appendix code), the open-loop poles of the plant are computed to be

$$\lambda_{1,2} = 0, \qquad \lambda_{3,4} = \pm 0.0011569\, j, \tag{9}$$

As the real component for all eigenvalues is 0, the open-loop system is *marginally stable*. The purely imaginary eigenvalues generate undamped oscillations in the in-plane states, while the double eigenvalue at the origin introduces constant or slowly drifting modes. Because no eigenvalues have negative real parts, perturbations do not naturally decay, and because no eigenvalues have positive real parts, perturbations do not grow.

To verify the open-loop behavior, we simulate the plant with zero input using MATLAB. The initial perturbation state is

$$x(0) = \delta x = [0.1 \text{ km};\ 0 \text{ km/s};\ 0.5^\circ;\ 0 \text{ rad/s}],$$

which corresponds to a 100 m radial offset and a 0.5° in–track angle error relative to the circular reference orbit. No control input is applied ($u(t) = 0$). The resulting output trajectories for this disturbance are shown in Fig. 1.

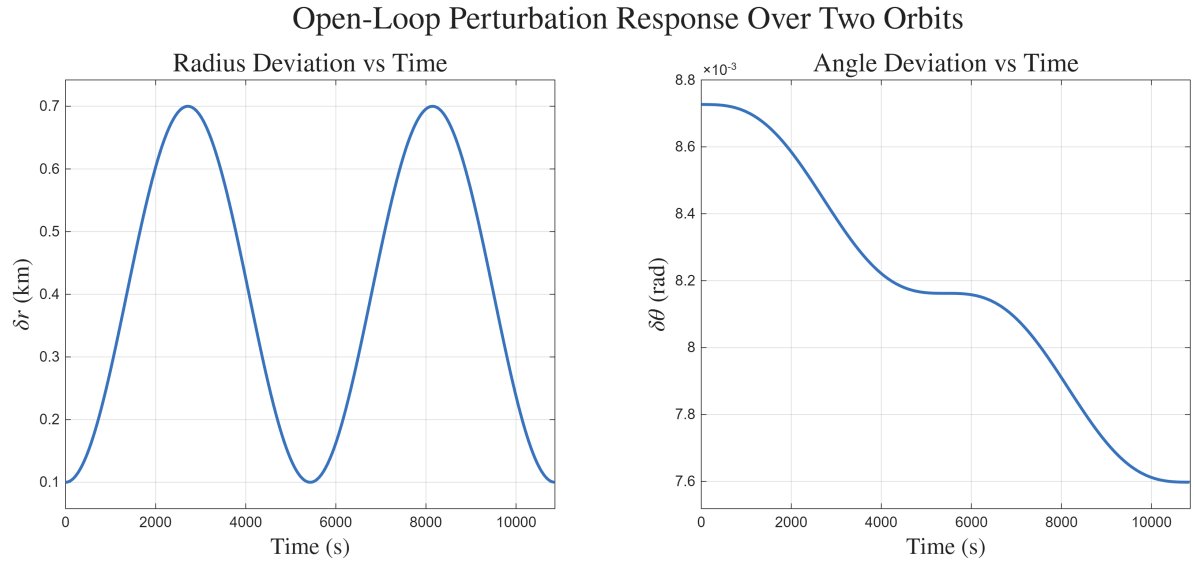

Figure 1: Open Loop Control Response

These responses follow directly from the open-loop pole locations. The radius $\delta r(t)$ exhibits a clear periodic oscillation with constant amplitude, matching the undamped sinusoidal motion generated by the purely imaginary pole pair $\pm 0.0011569j$. Because the real parts of these poles are zero, the system has no natural restoring or dissipative mechanism, so the oscillation neither grows nor decays over time. In contrast, the angular deviation $\delta\theta(t)$ shows a slow drift mixed with a very small oscillatory component. This behavior is characteristic of the double multiplicity eigenvalue at the origin, which can produce linear-in-time growth or decay. Importantly, neither state returns to the nominal value nor diverges catastrophically but instead both remain bounded but non-decaying, exactly as expected for a marginally stable system. The simulated trajectories therefore match the expected pole structure and illustrate why state feedback is required to achieve asymptotic stabilization.

2.3 System Reachability and Observability

To determine the reachability (controllability) and observability properties of the plant, we compute the standard controllability and observability matrices. The controllability matrix is defined as

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix}, \tag{10}$$

and the MATLAB command `ctrb(A,B)` confirms that

$$\operatorname{rank}(\mathcal{C}) = 4. \tag{11}$$

Since the rank equals the number of states $n = 4$, the system is fully reachable. Thus, state feedback can relocate all open-loop poles, and no uncontrollable modes are present.

The observability matrix is defined as

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix}, \tag{12}$$

and using the MATLAB command `obsv(A,C)`, we obtain

$$\operatorname{rank}(\mathcal{O}) = 4. \tag{13}$$

Because the rank equals the dimension of the state space, the system is fully observable. All state components influence the measured outputs in a way that allows them to be reconstructed by an observer.

Since both $\mathcal{C}$ and $\mathcal{O}$ have full rank, the system possesses no unreachable or unobservable subspaces. Therefore, every pole of the open-loop system may be independently reassigned through state-feedback control, and the poles can be placed arbitrarily in the left-half plane. This guarantees that the full closed-loop pole set (controller poles and observer poles) can be shaped to meet the stabilization and transient performance specifications defined in Part A.2.

3 PART B: LINEAR CONTROL DESIGN

3.1 Manual State Feedback Design with Integral Action

In order to design a state feedback controller with a manual design approach that meets the specifications in part A.2 while enforcing zero steady-state error in the measured output, the integral control error technique from class is utilized. To begin, the augmented matrices are created, in the case of the system they are:

$$A_{\text{aug}} = \begin{bmatrix} A & 0_{4\times 2} \\ -C & 0_{2\times 2} \end{bmatrix}, \qquad A_{\text{aug}} \in R^{6\times 6}, \tag{14}$$

$$B_{\text{aug}} = \begin{bmatrix} B \\ 0_{2\times 2} \end{bmatrix}, \qquad B_{\text{aug}} \in R^{6\times 2}, \tag{15}$$

$$F_{\text{aug}} = \begin{bmatrix} 0_{4\times 2} \\ I_{2\times 2} \end{bmatrix}, \qquad F_{\text{aug}} \in R^{6\times 2}, \tag{16}$$

$$C_{\text{aug}} = \begin{bmatrix} C & 0_{2\times 2} \end{bmatrix}, \qquad C_{\text{aug}} \in R^{2\times 6}, \tag{17}$$

$$D_{\text{aug}} = 0_{2\times 2}. \tag{18}$$

$$u = -K_{\text{aug}}\, x_{\text{aug}} + r, \tag{19}$$

where r is a constant reference (here taken as zero perturbation) and

$$K_{\text{aug}} = [\, K_x \;\; K_i \,]$$

contains both the state feedback gains K_x and the integral gains K_i.

Closed-loop eigenvalues for the augmented system were selected to satisfy the qualitative requirements from Part A.2:

- Real parts negative and of order 10^{-3} s^{-1}, corresponding to settling within approximately one orbital period,
- And sufficient damping to avoid oscillatory behavior and keep overshoot/undershoot within the $\pm 20\%$ bands

With desired poles selected, the MATLAB command `place()` is used to find the augmented feedback K_{aug} such that $A_{\text{aug}} - B_{\text{aug}} K_{\text{aug}}$ has these eigenvalues. After trial and error, the following closed-loop poles are selected:

$$\lambda_{CL,manual} = \{-0.00123,\ -0.00115,\ -0.00110,\ -0.00100,\ -0.00018,\ -0.00010\} \tag{20}$$

The resulting simulated closed-loop perturbation response to the initial disturbance of $\delta x = [100$ m; 0 km/s; $0.5°$; 0 rad/s$]$ is shown in Fig. 2. Both radial and angular deviations enter the 5% settling band within the single orbital period (shown by the shaded yellow region) and remain inside of the $\pm 20\%$ over/undershoot limits throughout the response (shown by the purple shaded region). The response sees a small initial undershoot, which is consistent with the real, well-damped closed-loop poles in Eq. (20). The integral action guarantees that the steady-state perturbations converge to zero as required in our design guidelines.

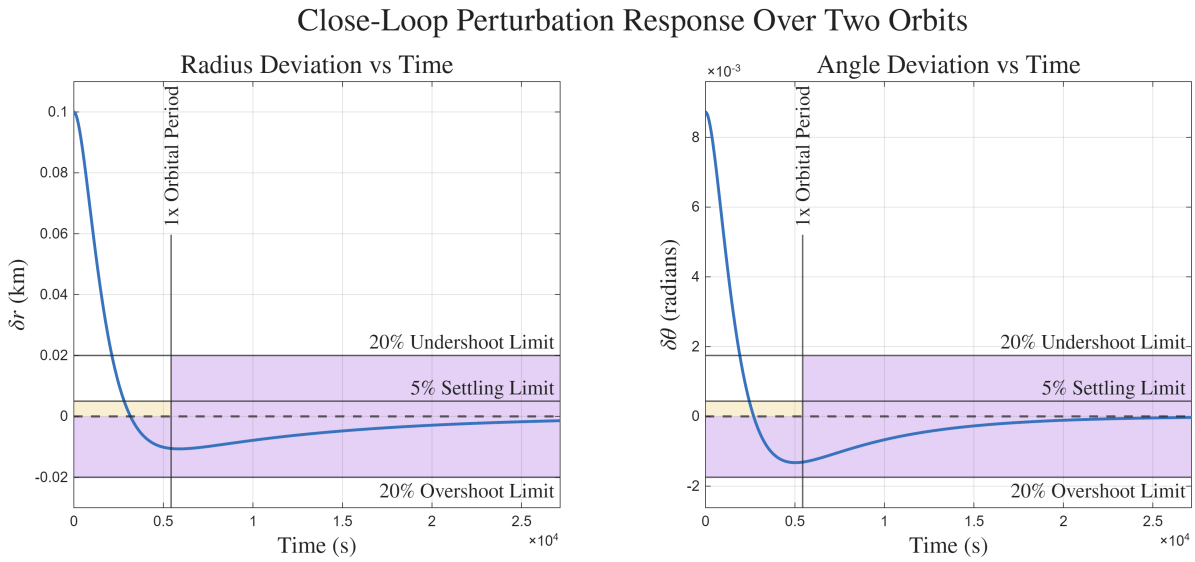

Figure 2: Manual Closed Loop Radial and Angular Response

The corresponding thrust accelerations are plotted in Fig. 3. The peak radial thrust is approximately $7.5 \times 10^{-3}\ g$, and the peak in–track thrust is about $1.0 \times 10^{-2}\ g$. The magnitude of the combined forcing vector fits inside the $\pm 0.01\ g$ actuator constraint band (shown by the shaded green region).

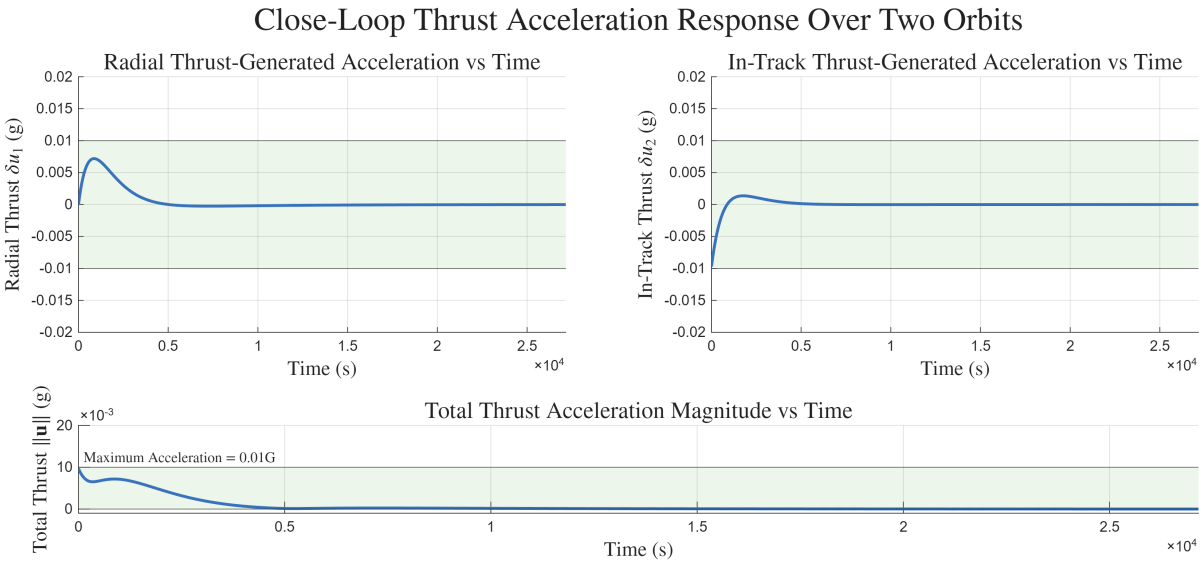

Figure 3: Manual Closed Loop Acceleration Forcing Response

This response could be achieved by a single 1 lbf thruster on a 100 lbm spacecraft with the appropriate engine throttling and spacecraft attitude control. The thrust decays rapidly after the first fraction of an orbit. As a basic measure of "actuator energy", we compute the integral of the required acceleration over time

$$E_{\text{man}} = \int_0^{T_{\text{sim}}} \|u(t)\|^2 \, dt, \tag{21}$$

where $u(t)$ is in units of km/s^2. The numerical value calculated here will be used later to compare against the optimal LQR design. Overall, the manually tuned pole placement achieves the time–domain specifications with physically reasonable control effort.

3.2 Luenberger Observer Design and Performance

As can be seen in our measurement vector, y, not all states are directly measured. Only the outputs

$$y = [\delta r \;\; \delta\theta]^T$$

are available. To reconstruct the full state vector, we design a Luenberger observer of the form

$$\dot{\hat{x}} = A\hat{x} + Bu + L\big(y - C\hat{x}\big), \tag{22}$$

where $\hat{x}$ is the state estimate and L is the observer gain. The estimation error $e = x - \hat{x}$ can therefore be written as

$$\dot{e} = (A - LC)\, e. \tag{23}$$

Because the plant is fully observable (Part A.3), the eigenvalues of $A - LC$ can be placed anywhere.

From lecture, a guideline is to choose the observer poles to be faster than, but not orders of magnitude faster than, the closed–loop controller poles. This ensures that the state estimate converges rapidly relative to the plant dynamics without amplifying sensor noise excessively. Here we select the observer poles to be five times faster than the four dominant controller poles:

$$\lambda_{\text{obs}} = 5 \cdot \{ \; -0.00123, \; -0.00115, \; -0.00110, \; -0.00100 \; \} \tag{24}$$

$$= \{-0.00615, \; -0.00575, \; -0.00550, \; -0.00500\} \tag{25}$$

and compute the gain using $L = \texttt{place}(A^T, C^T, \lambda_{\text{obs}})^T$.

The controller and observer are combined by forming an 8–state augmented system in which the true plant and the observer evolve together. With zero initial observer error ($\hat{x}(0) = x(0)$), the estimated and true states are numerically indistinguishable as expected. Without error, the two states should behave identically. In the case with a large initial mismatch, we set a 50% error in the initial estimate, $\hat{x}(0) = 0.5\,x(0)$. The resulting trajectories are shown in Fig. 4. For all four states, the estimated trajectories converge rapidly to the true trajectories within roughly half an orbit, and the residual estimation error is negligible over the remainder of the simulation.

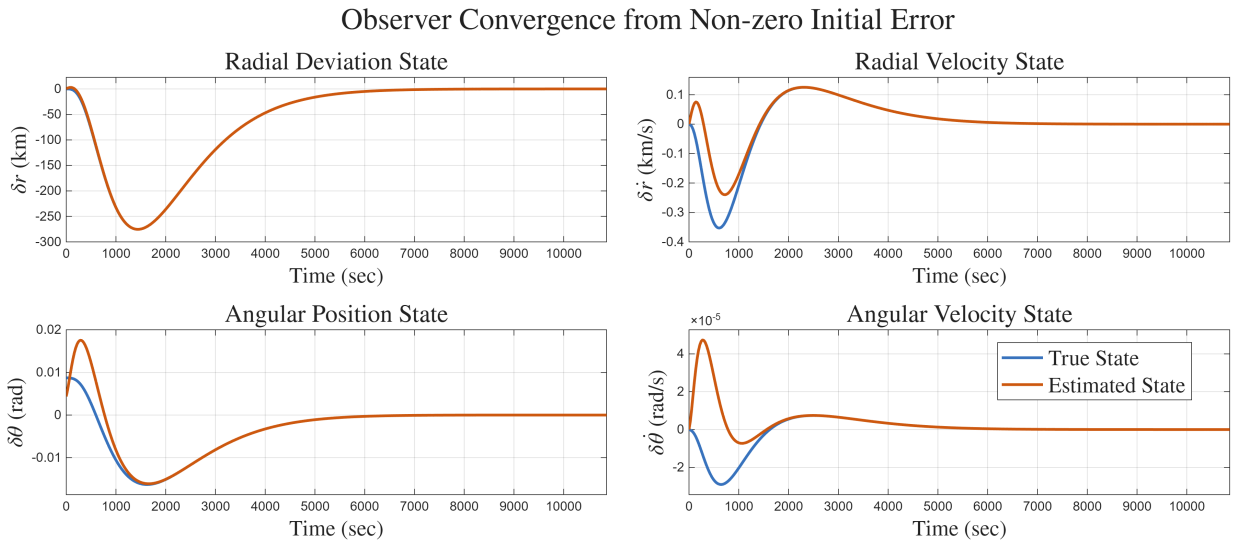

Figure 4: 50% Error Luenberger Observer Convergence

The convergence rate is quantified in Fig. 5, which plots the Euclidean norm $\|e(t)\|_2$ for both zero and non–zero initial observer error. With zero initial error, the norm remains essentially at machine precision. With 50% initial error, the norm peaks at approximately 4.5 in mixed (km, rad) units before decaying by more than two orders of magnitude, consistent with the real parts of the observer poles in Eq. (25). This confirms that the observer behaves as designed.

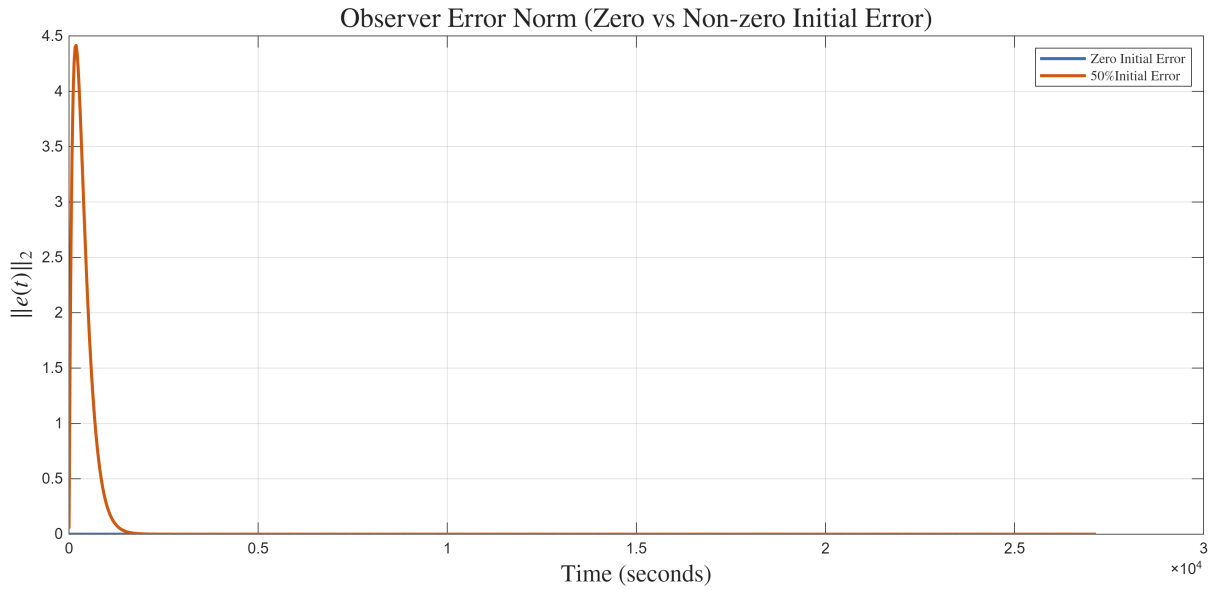

Figure 5: Luenberger Observer Error Magnitude Convergence

To assess the impact on closed–loop performance, Fig. 6 compares the manual full–state feedback response (using the true states) with the manual+observer configuration (using $\hat{x}$) for the same initial condition. The two responses are similar, both satisfy the settling and overshoot constraints defined in Part A.2. This suggests that the observer does not degrade the performance when its poles are chosen moderately faster than the controller poles.

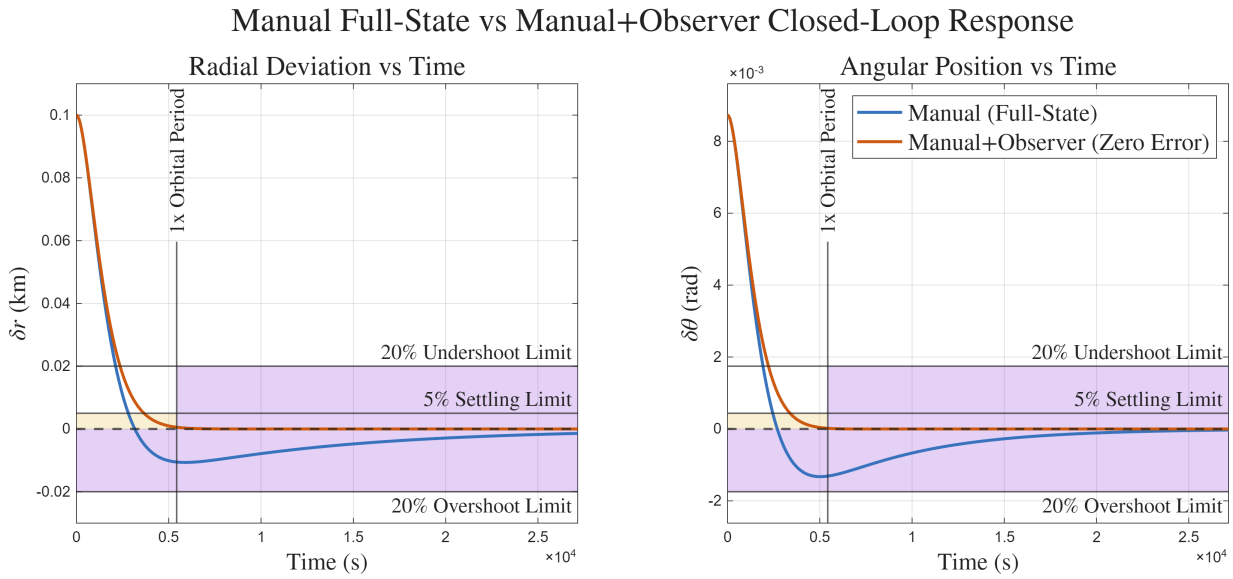

Figure 6: Manual Pole Placement Response Versus Observer Response

In a realistic setting with sensor noise, the choice of fast observer poles would trade estimation speed against noise amplification. Because L multiplies the output residual $y - C\hat{x}$, high–frequency measurement noise would be injected directly into the estimated states and, through the state feedback, into the control input. For this project, we neglect measurement noise and so the significance of this is not relevant. However, the current observer design would likely need to be slowed down in a noisy environment or replaced by a Kalman filter to avoid excessive control chattering.

3.3 Infinite-Horizon LQR Design with Observer and Comparison

Finally, we design an optimal state feedback law using the infinite–horizon quadratic cost

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right)\, dt, \tag{26}$$

where Q penalizes state deviations and R penalizes control effort. We select Q and R using Bryson's rule as outlined in the lecture so that each term in the cost is scaled by its maximum allowable magnitude. Let

$$x_{\max} = \begin{bmatrix} 0.2\,\delta r_0 \\ 1.0 \\ 0.2\,\delta\theta_0 \\ 1.0 \end{bmatrix}, \qquad \bar{u}_{\max} = 0.01\, g_0 = 0.01 \times \frac{9.80665}{1000}\ \text{km/s}^2, \qquad u_{\max} = \begin{bmatrix} \bar{u}_{\max} \\ \bar{u}_{\max} \end{bmatrix}, \tag{27}$$

where δr_0 and $\delta\theta_0$ are the initial perturbations and g_0 is standard gravity. The components of $x_{\max}$ correspond to the 20% overshoot/undershoot limits for the position states and conservative bounds for the velocity states.The components of $u_{\max}$ correspond to a 0.01 G maximum acceleration as previously discussed.

We then specify relative importance weights

$$\alpha_{\text{states}} = \begin{bmatrix} 0.10 \\ 0.05 \\ 0.80 \\ 0.05 \end{bmatrix}, \qquad \beta_{\text{inputs}} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix},$$

assigning the largest weight to angular position (which directly affects along–track error) while giving smaller but non–zero weight to the remaining states. It was found through experimentation that the larger weight on angular position was required to get the response to meet the design guidelines previously stated. The control weights are chosen equal because both thrusters are equally limited. Following Bryson's rule, we define

$$Q = \mathrm{diag}\left(\left(\frac{\alpha_{\text{states}}}{x_{\max}}\right)^2\right), \qquad R = \rho\,\mathrm{diag}\left(\left(\frac{\beta_{\text{inputs}}}{u_{\max}}\right)^2\right), \tag{28}$$

where the scalar $\rho > 0$ tunes the overall trade–off between performance and control effort. Through trial–and–error simulations, we found that $\rho = 80$ resulted in a balance that met our design requirements.

With these matrices, the MATLAB command `lqr(A,B,Q,R)` returns the optimal gain K_{LQR} and the corresponding closed–loop poles of $A - BK_{\text{LQR}}$. All LQR poles lie in the open left–half plane with real parts of roughly the same order of magnitude as the manual design. Parameters were adjusted until this was the case in order to get a similar response to what we manually achieved in order to ensure a fair "energy" comparison. Compared with the manually placed poles, the LQR poles are more evenly distributed, leading to a smoother, slightly less oscillatory response.

To compare fairly with the previous sections, we retain the same observer gain L from Part B.5 and use the estimated state in the LQR control law:

$$u = -K_{\text{LQR}}\hat{x}. \tag{29}$$

The resulting closed–loop perturbation response is shown in Fig. 7. Relative to the manual design, the LQR controller produces a slightly faster and more monotone return to the origin with reduced (almost zero) undershoot, while still meeting the 5% settling and $\pm 20\%$ overshoot limits within one orbital period.

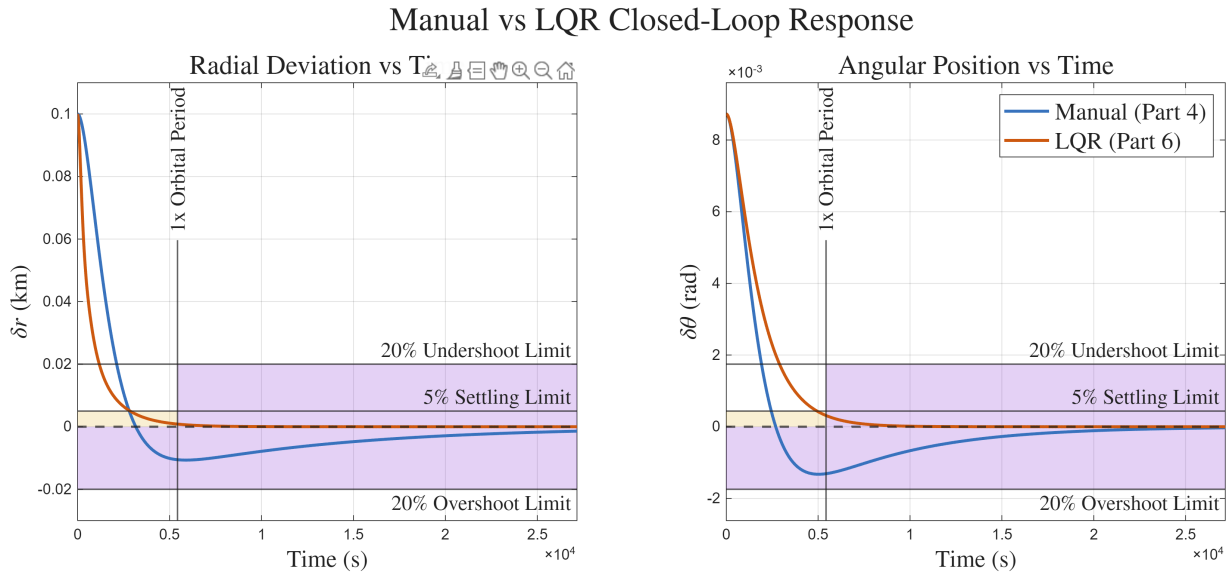

Figure 7: Manual Pole Placement Response Versus LQR Selected Response

The corresponding thrust accelerations are plotted in Fig. 8.

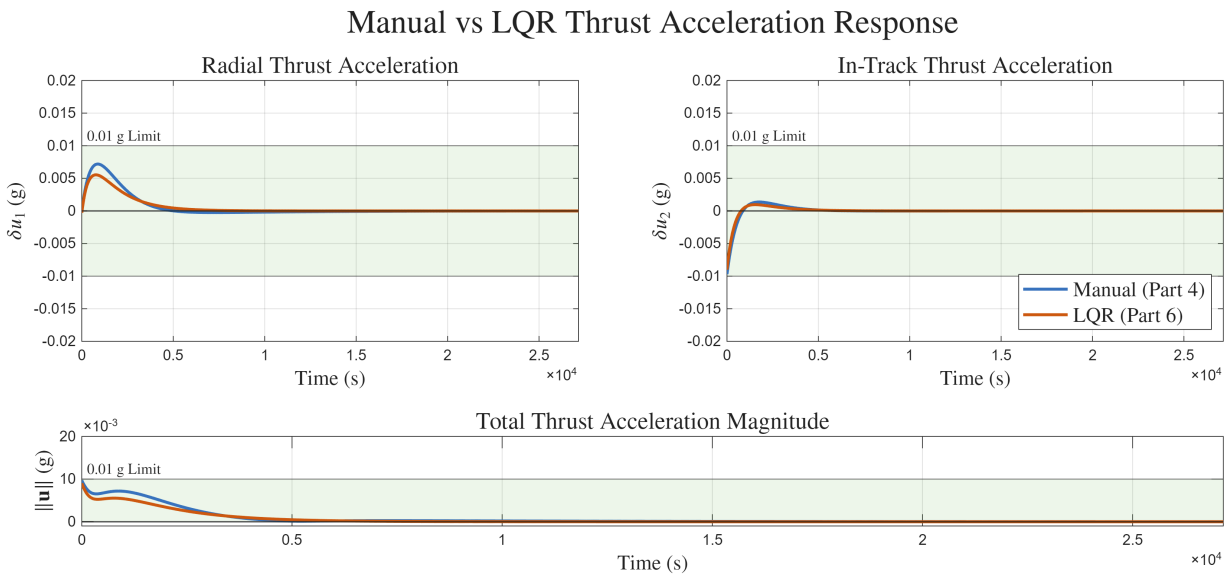

Figure 8: Manual Pole Placement Response Versus LQR Selected Response

Despite the faster settling performance, shown in Fig. 7, the LQR controller commands slightly lower peak radial and in–track thrusts than the manual design and maintains a smaller total thrust magnitude at all times while satisfying the same $\pm 0.01\ g$ constraint. The integrated LQR actuator "energy"

$$E_{\mathrm{LQR}} = \int_0^{T_{\mathrm{sim}}} \|u_{\mathrm{LQR}}(t)\|^2\, dt \tag{30}$$

is 38.01% smaller than E_{man} from Eq. (21).

$$E_{\mathrm{man}} = 9.7806 \times 10^{-6}\ \mathrm{km}^2/\mathrm{s}^3, \qquad E_{\mathrm{LQR}} = 6.0633 \times 10^{-6}\ \mathrm{km}^2/\mathrm{s}^3.$$

Therefore, the LQR design achieves at least comparable (if not better) disturbance-rejection performance while also requiring less control effort.

APPENDIX

```
%Clear workspace
clear; close all; clc;

%Define Orbital Constants
mu = 398600; %km3/s2
r0 = 6678; %km

%Set Initial Condition (small perturbations about nominal orbit)
%x = [delta r; delta rdot; delta theta; delta thetadot]
deltar0 = 0.1; %0.1 km = 100 m radial deviation
deltardot0 = 0.0; %km/s
deltatheta0 = deg2rad(0.5); %rad (0.5 degrees)
deltathetad0 = 0.0; %rad/s

xnom = [deltar0; deltardot0; deltatheta0; deltathetad0];

%Define A, B, C, D matrices (At nominal Orbit)
A = [0      1              0 0;
    3*mu/r03 0             0 2*sqrt(mu/r0);
    0        0             0 1;
    0        -2*sqrt(mu/(r05)) 0 0];

B = [0 0;
    1 0;
    0 0;
    0 1/r0];

C = [1 0 0 0;
    0 0 1 0];

D = zeros(2);
%% Part 1, Steps 1-3 - Linear System Analysis
%Find eigenvalues of A matrix (open-loop poles)
lambda = eig(A);
disp("Open Loop A Matrix Eigenvalues: ");
disp(lambda);

%Determine Reachability
P = ctrb(A, B);
rp = rank(P); %Full Rank == Fully Reachable
disp("Reachability Rank: " + rp + " (Should be 4)");

%Determine Observability
O = obsv(A, C);
ro = rank(O); %Full Rank == Fully Observable
disp("Observability Rank: " + ro + " (Should be 4)");

%Set up Linear System
sys = ss(A, B, C, D);

%Determine time vector for 2 orbits
Torbit = 2*pi/sqrt(mu/r03); %orbital period [s]
numOrbits = 5;
t = linspace(0, numOrbits*Torbit, 1000); %Simulation time vector [s]

%Simulate zero-input response from perturbation initial condition
```

```
[y, tOut] = initial(sys, xnom, t);


%% Part 2, Step 4 - Closed Loop Control
%Desired reference values
rhist = zeros(length(t),2);

%Define augmented matrices
Aaug = [ A zeros(4,2);
 -C zeros(2,2)];

Baug = [ B; zeros(2,2)];

Faug = [zeros(size(B)); eye(2)];

Caug = [ C zeros(2)];

Daug = zeros(size(Caug,1),size(Baug,2));

%Choose poles to satisfy requirements
poles = [-0.00123, -0.001, -0.0011, -0.00115, -0.00018, -0.0001];
Kaug = place(Aaug, Baug, poles);

%Define augmented closed loop system
AaugCL = Aaug - Baug*Kaug;
BaugCL = Faug;
sysCL = ss(AaugCL, BaugCL, Caug, Daug);

%Get response from reference profile
[yCLaug,,xCLaug] = lsim(sysCL, rhist, t, [xnom; zeros(2,1)]);

%Compute actuator efforts in each case, where uclaug = -K * xclaug
uCLaug = -(Kaug * xCLaug'); %units: km/s2

%Convert control accelerations from km/s2 to g's
g0 = 9.80665; %m/s2
kmps2tog = 1000 / g0; %(km/s2) * (1000 m/km) / (9.80665 m/s2)
uCLaugg = uCLaug * kmps2tog; %now in units of g

%% Part 2, Step 5 - Luenberger Observer
%Select observer poles
polesluen = 5 * real(poles(1:4));

%Compute Observer Gain
L = place(A', C', polesluen)';

%Extract feedback part from augmented gain
Kx = Kaug(:, 1:4); %Feedback gains
Ki = Kaug(:, 5:6); %Integrator gains

%Form Augmented Matrices
Aclaug = [A - B*Kx, B*Kx ;...
        zeros(4), A - L*C];
Bclaug = [B*Ki; zeros(4, size(Ki, 2))];
Cclaug = [C, zeros(size(C, 1), 4)];
Dclaug = zeros(size(Cclaug, 1), size(Bclaug, 2));

%Case 1 - Zero initial observer error
z0case1 = [xnom; zeros(4, 1)];
```

```
116
117 %Case 2: Non-zero observer error
118 percentError = 0.5;
119 z0case2 = [xnom; xnom*percentError];
120
121 %Simulate Case 1
122 sysObs1 = ss(Aclaug, Bclaug, Cclaug, Dclaug);
123 [, , zObs1] = lsim(sysObs1, rhist, t, z0case1);
124 xtruecase1 = zObs1(:,1:4);
125 ehistcase1 = zObs1(:,5:8);
126 xhatcase1 = xtruecase1 - ehistcase1;
127 esterrcase1 = xtruecase1 - xhatcase1;
128
129 %Simulate Case 2
130 sysObs2 = ss(Aclaug, Bclaug, Cclaug, Dclaug);
131 [, , zObs2] = lsim(sysObs2, rhist, t, z0case2);
132 xtruecase2 = zObs2(:,1:4);
133 ehistcase2 = zObs2(:,5:8);
134 xhatcase2 = xtruecase2 - ehistcase2;
135 esterrcase2 = xtruecase2 - xhatcase2;
136
137 %Display Observer Poles
138 eigobs = eig(A - L*C);
139 disp("Observer poles (A - L*C):");
140 disp(eigobs);
141
142 %% Part 2, Step 6 - Infinite-Horizon Cost Function
143 %Define Max Allowable Deviations
144 xmaxdeltar = 0.20 * deltar0; %km (20% of initial radial offset)
145 xmaxdeltardot = 1.0; %km/s
146 xmaxdeltatheta = 0.20 * deltatheta0; %rad (20% of initial angle offset)
147 xmaxdeltathetadot = 1e0; %rad/s
148
149 xmaxvec = [xmaxdeltar; xmaxdeltardot; xmaxdeltatheta; xmaxdeltathetadot];
150
151 %Define Max allowable control accelerations
152 umaxg = 0.01; %Max acceleration = 0.01 g
153 umaxkmps2 = umaxg * g0 / 1000; %Convert to km/s2
154 umaxvec = umaxkmps2 * [1; 1];
155
156 %Define relative importance of each state (sum to 1)
157 alphastates = [0.10; 0.05; 0.80; 0.05];
158
159 %Define relative importance of each input (sum to 1)
160 betainputs = [0.5; 0.5]; %Equal weight thrusters
161
162 %Define overall tradeoff between state deviations and control effort
163 rho = 80;
164
165 %Build Q and R via Bryson's Rule
166 Qlqr = diag((alphastates ./ xmaxvec).2);
167 Rlqr = rho * diag((betainputs ./ umaxvec).2);
168
169 %Calculate optimal gains
170 [Klqr, Slqr, ] = lqr(A, B, Qlqr, Rlqr);
171
172 %Compare closed-loop poles with manual design
173 eigmanual = eig(A - B*Kx);
174 eiglqr = eig(A - B*Klqr);
```

```
175
176 disp("Manual closed-loop poles (A - B*Kx):");
177 disp(eigmanual);
178
179 disp("LQR closed-loop poles (A - B*Klqr):");
180 disp(eiglqr);
181
182 %Create Luenberger Observer Augmented Dynamics
183 Acllqrobs = [A - B*Klqr, B * Klqr;...
184            zeros(4), A - L*C];
185 Bcllqrobs = zeros(8, size(rhist, 2));
186 Ccllqrobs = eye(8);
187 Dcllqrobs = zeros(8, size(rhist,2));
188
189 %Create State Space Model Object
190 syslqrobs = ss(Acllqrobs, Bcllqrobs, Ccllqrobs, Dcllqrobs);
191
192 %Create 0% initial state error in observer for comparing response
193 percentError = 0.0;
194 z0lqr = [xnom; xnom * percentError];
195
196 %Simulate LQR and observer closed loop
197 [, , zlqr] = lsim(syslqrobs, rhist, t, z0lqr);
198
199 %Pull out true and estimated states
200 xtruelqr = zlqr(:, 1:4); %True States
201 ehistlqr = zlqr(:, 5:8); %Error States
202 xhatlqr = xtruelqr - ehistlqr; %Estimated = True - Error
203
204 %Compute LQR control forcing based on estimated state [Acceleration = km/s2]
205 ulqr = -(Klqr * xhatlqr')';
206
207 %Covert to Gs of acceleration
208 ulqrg = ulqr * kmps2tog;
209
210 %Manual controller (From Part 4) control based on estimated state (case 2 from
       part 5)
211 umanual = uCLaug';
212 umanualg = umanual * kmps2tog;
213
214 %Numerically integrate actuation acceleration to get "Total Energy"-ish
       comparison
215 Emanual = trapz(t, sum(umanual.2, 2)); %Manual state feedback
216 Elqr = trapz(t, sum(ulqr.2, 2)); %LQR state feedback
217
218 disp("Manual controller energy: " + Emanual);
219 disp("LQR controller energy: " + Elqr);
220
221 %Run LQR with 50% initial observer error
222 percentError = 0.5;
223 z0lqrerr = [xnom; xnom*percentError];
224 [,,zlqrerr] = lsim(syslqrobs,rhist,t,z0lqrerr);
225 xtruelqrerr = zlqrerr(:,1:4);
226
227
228 %% Produce Open Loop Response Figures
229 figure('Color','w');
230 tiles = tiledlayout('flow');
231
```

```
%Overall title for the tiled layout
axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
title(tiles, 'Open-Loop Perturbation Response Over Two Orbits', "Interpreter",
    "Latex", "FontSize", figureTitleFontSize);

%---RADIUS DEVIATION VERSUS TIME---
nexttile;
plot(tOut(tOut¡2*Torbit), y((tOut¡2*Torbit),1), 'LineWidth', 2);
grid on;
title('Radius Deviation vs Time', "Interpreter","Latex", "FontSize",
    plotTitleFontSize);
ylabel('$"delta r$ (km)', "Interpreter","Latex", "FontSize", axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
axis padded;
xlim([0, 2*Torbit]);

%---ANGLE DEVIATION VERSUS TIME---
nexttile;
plot(tOut(tOut¡2*Torbit), y((tOut¡2*Torbit),2), 'LineWidth', 2);
grid on;
title('Angle Deviation vs Time', "Interpreter","Latex", "FontSize",
    plotTitleFontSize);
ylabel('$"delta "theta$ (rad)', "Interpreter","Latex", "FontSize", axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
axis padded;
xlim([0, 2*Torbit]);

%% Produce Response Figure
figure('Color','w');
tiles = tiledlayout('flow');

%Overall title for the tiled layout
axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
title(tiles, 'Close-Loop Perturbation Response Over Two Orbits', "Interpreter",
    "Latex", "FontSize", figureTitleFontSize);

%-- RADIUS DEVIATION VERUS TIME PLOT --
nexttile;

%Add in Response Boundaries
radialStep = deltar0;
plotColors = orderedcolors("gem");
patch([0, Torbit, Torbit, 0], [0, 0, radialStep * 0.05, radialStep*0.05],
    plotColors(3, :), "FaceAlpha", 0.2, "EdgeColor", "none");
hold on; box on;
patch([0, tOut(end), tOut(end), Torbit, Torbit, 0], [-radialStep*0.2,
    -radialStep*0.2, radialStep*0.2, radialStep*0.2, 0, 0], plotColors(4, :),
    "FaceAlpha", 0.2, "EdgeColor", "none")
yline(0, '--', 'LineWidth', 1.5, "FontSize", 14);
xline(Torbit, '-', "1x Orbital Period", "LabelHorizontalAlignment","center",
    "LabelVerticalAlignment","top", "Interpreter", "latex", "LineWidth", 1,
    "FontSize", 14);
%xline(2*Torbit, '-', "2x Orbital Period", "LabelHorizontalAlignment","left",
    "LabelVerticalAlignment","top", "Interpreter", "latex", "LineWidth", 1,
    "FontSize", 14);
```

```
yline(radialStep * 0.05, '-', "$5“%$ Settling Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter", "latex", "LineWidth", 1, "FontSize", 14);
yline(radialStep * 0.20, '-', "$20“%$ Undershoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter", "latex", "LineWidth", 1, "FontSize", 14);
yline(radialStep *-0.20, '-', "$20“%$ Overshoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","bottom",
    "Interpreter", "latex", "LineWidth", 1, "FontSize", 14);

%Plot Response Line
plot(tOut, yCLaug(:,1), 'LineWidth', 2);

%Plot Styling
grid on;
title('Radius Deviation vs Time', "Interpreter","Latex", "FontSize",
    plotTitleFontSize);
ylabel('$“delta r$ (km)', "Interpreter","Latex", "FontSize", axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
xlim([tOut(1), tOut(end)]);
ylim([radialStep *-0.30, 1.1*abs(radialStep)]);

%-- ANGLE DEVIATION VERSUS TIME PLOT --
nexttile;

%Add in Response Boundaries
angularStep = deltatheta0;
plotColors = orderedcolors("gem");
patch([0, Torbit, Torbit, 0], [0, 0, angularStep * 0.05, angularStep*0.05],
    plotColors(3, :), "FaceAlpha", 0.2, "EdgeColor", "none");
hold on; box on;
patch([0, tOut(end), tOut(end), Torbit, Torbit, 0], [-angularStep*0.2,
    -angularStep*0.2, angularStep*0.2, angularStep*0.2, 0, 0], plotColors(4, :),
    "FaceAlpha", 0.2, "EdgeColor", "none")
yline(0, '--', 'LineWidth', 1.5, "FontSize", 14);
xline(Torbit, '-', "1x Orbital Period", "LabelHorizontalAlignment","center",
    "LabelVerticalAlignment","top", "Interpreter", "latex", "LineWidth", 1,
    "FontSize", 14);
%xline(2*Torbit, '-', "2x Orbital Period", "LabelHorizontalAlignment","left",
    "LabelVerticalAlignment","top", "Interpreter", "latex", "LineWidth", 1,
    "FontSize", 14);
yline(angularStep * 0.05, '-', "$5“%$ Settling Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter", "latex", "LineWidth", 1, "FontSize", 14);
yline(angularStep * 0.20, '-', "$20“%$ Undershoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter", "latex", "LineWidth", 1, "FontSize", 14);
yline(angularStep *-0.20, '-', "$20“%$ Overshoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","bottom",
    "Interpreter", "latex", "LineWidth", 1, "FontSize", 14);

%Plot Response Line
plot(tOut, yCLaug(:,2), 'LineWidth', 2);

%Plot Styling
grid on;
title('Angle Deviation vs Time', "Interpreter","Latex", "FontSize",
    plotTitleFontSize);
ylabel('$“delta “theta$ (radians)', "Interpreter","Latex", "FontSize",
```

```
    axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
xlim([tOut(1), tOut(end)]);
ylim([angularStep *-0.30, 1.1*abs(angularStep)]);

%% Produce Output Force Figure
figure('Color','w');
tiles = tiledlayout('flow');

%Overall title for the tiled layout
axisFontSize = 14;
plotTitleFontSize = 16;
figureTitleFontSize = 22;
title(tiles, 'Close-Loop Thrust Acceleration Response Over Two Orbits',
    "Interpreter", "Latex", "FontSize", figureTitleFontSize);

%---RADIAL FORCING ACCELERATION VERSUS TIME---
nexttile;
patch([0, tOut(end), tOut(end), 0], [-0.01 -0.01 0.01 0.01], plotColors(5, :),
    'FaceAlpha', 0.1, 'EdgeColor', 'none');
hold on;
plot(tOut, uCLaugg(1,:), 'LineWidth', 2.0); hold on
yline(0.01);
yline(-0.01);
ylim([-0.02 0.02]);

%Plot Styling and Labels
grid on;
title('Radial Thrust-Generated Acceleration vs Time', "Interpreter","Latex",
    "FontSize", plotTitleFontSize);
ylabel('Radial Thrust $"delta u-1$ (g)', "Interpreter","Latex", "FontSize",
    axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
xlim([tOut(1), tOut(end)]);

%---IN-TRACK FORCING ACCLERATION VERSUS TIME---
nexttile;
patch([0, tOut(end), tOut(end), 0], [-0.01 -0.01 0.01 0.01], plotColors(5, :),
    'FaceAlpha', 0.1, 'EdgeColor', 'none');
hold on;
plot(tOut, uCLaugg(2,:), 'LineWidth', 2.0); hold on
yline(0.01);
yline(-0.01);
ylim([-0.02 0.02]);

%Plot Styling and Labels
grid on;
title('In-Track Thrust-Generated Acceleration vs Time', "Interpreter","Latex",
    "FontSize", plotTitleFontSize);
ylabel('In-Track Thrust $"delta u-2$ (g)', "Interpreter","Latex", "FontSize",
    axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
xlim([tOut(1), tOut(end)]);

%---TOTAL FORCING ACCELERATION VERSUS TIME---
nexttile(tiles, "south", [1, 2]);
umagg = sqrt(uCLaugg(1,:).2 + uCLaugg(2,:).2); %Forcing magnitude in g

%Plot Bounding Box
```

```
plotColors = orderedcolors("gem");
patch([0, tOut(end), tOut(end), 0], [0 0 0.01 0.01], plotColors(5, :), 'FaceAlpha', 0.1, 'EdgeColor', 'none');
hold on;

%Plot Line
plot(t, umagg, 'LineWidth', 2.0);

%Add in Boundaries
yline(0.01, '-', "Maximum Acceleration = 0.01G", "Interpreter", "Latex", 'LabelHorizontalAlignment','left', 'LabelVerticalAlignment','top');
yline(0, '-');

%Plot Styling
grid on;
title('Total Thrust Acceleration Magnitude vs Time', "Interpreter","Latex", "FontSize", plotTitleFontSize);
ylabel('Total Thrust $"—"mathbf-u"—$ (g)', "Interpreter","Latex", "FontSize", axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
ylim([-0.001, 0.02]);
xlim([tOut(1), tOut(end)]);

%% Observer Performance Plots
figure('Color','w');
tiles = tiledlayout('flow', 'TileSpacing','compact','Padding','compact');
axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
timeEnd = Torbit * 2;

%Set Figure Window Title
title(tiles, 'Observer Convergence from Non-zero Initial Error', "Interpreter","Latex","FontSize",figureTitleFontSize);
plotColors = orderedcolors("gem");

%---Delta R Plot---
nexttile;

%Plot the True State
plot(t, xtruecase2(:, 1), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
    hold on;

%Plot Estimated Case
plot(t, xhatcase2(:, 1), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);

%Plot Styling
ylabel("$"delta r$ (km)", "Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
title('Radial Deviation State', "Interpreter","Latex", "FontSize",plotTitleFontSize);
box on; grid on; axis padded;
xlim([0, timeEnd]);

%---Delta Rdot Plot---
nexttile;

%Plot the True State
plot(t, xtruecase2(:, 2), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
```

```
    hold on;

%Plot Estimated Case
plot(t, xhatcase2(:, 2), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);

%Plot Styling
ylabel("$"delta "dot r$ (km/s)", "Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
title('Radial Velocity State', "Interpreter","Latex",
    "FontSize",plotTitleFontSize);
box on; grid on; axis padded;
xlim([0, timeEnd]);

%---Delta Theta Plot---
nexttile;

%Plot the True State
plot(t, xtruecase2(:, 3), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
    hold on;

%Plot Estimated Case
plot(t, xhatcase2(:, 3), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);

%Plot Styling
ylabel("$"delta "theta$ (rad)", "Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
title('Angular Position State', "Interpreter","Latex",
    "FontSize",plotTitleFontSize);
box on; grid on; axis padded;
xlim([0, timeEnd]);

%---Delta Thetadot Plot---
nexttile;

%Plot the True State
plot(t, xtruecase2(:, 4), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
    hold on;

%Plot Estimated Case
plot(t, xhatcase2(:, 4), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);

%Plot Styling
ylabel("$"delta "dot "theta$ (rad/s)",
    "Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
title('Angular Velocity State', "Interpreter","Latex",
    "FontSize",plotTitleFontSize);
box on; grid on; axis padded;
xlim([0, timeEnd]);
legend(["True State","Estimated State"],"Interpreter","Latex",'Location','best',
    "FontSize", axisFontSize);

%% Observer Performance Plots (Zero Initial Error)
figure('Color','w');
tiles = tiledlayout('flow', 'TileSpacing','compact','Padding','compact');
axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
timeEnd = Torbit * 2;
```

```
472
473 title(tiles, 'Observer Convergence from Zero Initial Error',
        "Interpreter","Latex","FontSize",figureTitleFontSize);
474 plotColors = orderedcolors("gem");
475
476 %---Delta R Plot---
477 nexttile;
478 plot(t, xtruecase1(:, 1), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
        hold on;
479 plot(t, xhatcase1(:, 1), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);
480 ylabel("$"delta r$ (km)", "Interpreter","Latex","FontSize",axisFontSize);
481 xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
482 title('Radial Deviation State',
        "Interpreter","Latex","FontSize",plotTitleFontSize);
483 box on; grid on; axis padded;
484 xlim([0, timeEnd]);
485
486 %---Delta rdot Plot---
487 nexttile;
488 plot(t, xtruecase1(:, 2), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
        hold on;
489 plot(t, xhatcase1(:, 2), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);
490 ylabel("$"delta "dot r$ (km/s)", "Interpreter","Latex","FontSize",axisFontSize);
491 xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
492 title('Radial Velocity State',
        "Interpreter","Latex","FontSize",plotTitleFontSize);
493 box on; grid on; axis padded;
494 xlim([0, timeEnd]);
495
496 %---Delta theta Plot---
497 nexttile;
498 plot(t, xtruecase1(:, 3), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
        hold on;
499 plot(t, xhatcase1(:, 3), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);
500 ylabel("$"delta "theta$ (rad)", "Interpreter","Latex","FontSize",axisFontSize);
501 xlabel('Time (sec)', "Interpreter","Latex","FontSize",axisFontSize);
502 title('Angular Position State',
        "Interpreter","Latex","FontSize",plotTitleFontSize);
503 box on; grid on; axis padded;
504 xlim([0, timeEnd]);
505
506 %---Delta thetadot Plot---
507 nexttile;
508 plot(t, xtruecase1(:, 4), '-', 'Color', plotColors(1, :), 'LineWidth', 2.0);
        hold on;
509 plot(t, xhatcase1(:, 4), '-', 'Color', plotColors(2, :), 'LineWidth', 2.0);
510 ylabel("$"delta "dot "theta$ (rad/s)",
        "Interpreter","Latex","FontSize",axisFontSize);
511 xlabel('Time (sec)',   "Interpreter","Latex","FontSize",axisFontSize);
512 title('Angular Velocity State',
        "Interpreter","Latex","FontSize",plotTitleFontSize);
513 box on; grid on; axis padded;
514 xlim([0, timeEnd]);
515 legend(["True State","Estimated State"],
        "Interpreter","Latex",'Location','best', "FontSize", axisFontSize);
516
517
518 %% Estimation Error Norms Plots
519 enormcase1 = vecnorm(esterrcase1,2,2); %Case 1 Total Error Magnitude
```

```
enormcase2 = vecnorm(esterrcase2,2,2); %Case 2 Total Error Magnitude

figure('Color','w');

axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
title("State Estimation Error Norms", "Interpreter","latex", "FontSize",
    figureTitleFontSize);

%Add Error Lines
plot(t, enormcase1, "-", "Color", plotColors(1,:), "LineWidth", 2.0); hold on;
plot(t, enormcase2, "-", "Color", plotColors(2,:), "LineWidth", 2.0);

%Plot Styling
xlabel("Time (seconds)", "Interpreter","Latex","FontSize",axisFontSize);
ylabel("$"—e(t)"—2$", "Interpreter","Latex","FontSize",axisFontSize);
title("Observer Error Norm (Zero vs Non-zero Initial Error)",
    "Interpreter","Latex","FontSize",plotTitleFontSize);
legend(["Zero Initial Error", "50"%Initial Error"],
    "Interpreter","Latex",'Location','NorthEast');
box on; grid on;

%% Manual vs LQR Closed-Loop Response (Radial & Angular Positions) Plots
figure('Color','w');
tiles = tiledlayout('flow');

axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
title(tiles, 'Manual vs LQR Closed-Loop Response',
    "Interpreter","Latex","FontSize",figureTitleFontSize);

plotColors = orderedcolors("gem");

%---------------- RADIAL DEVIATION ----------------%
nexttile;

radialStep = deltar0;

%Settling / overshoot bands
patch([0, Torbit, Torbit, 0], [0, 0, radialStep * 0.05, radialStep * 0.05],
    plotColors(3, :), "FaceAlpha", 0.2, "EdgeColor", "none");
hold on; box on;
patch([0, t(end), t(end), Torbit, Torbit, 0], [-radialStep * 0.2, -radialStep *
    0.2, radialStep * 0.2, radialStep * 0.2, 0, 0], plotColors(4, :),
    "FaceAlpha", 0.2, "EdgeColor", "none");

yline(0, '--', 'LineWidth', 1.5, "FontSize", 14);
xline(Torbit, '-', "1x Orbital Period", "LabelHorizontalAlignment","center",
    "LabelVerticalAlignment","top", "Interpreter","latex", "LineWidth",1,
    "FontSize",14);
yline(radialStep * 0.05, '-', "$5"%$ Settling Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex", "LineWidth",1, "FontSize",14);
yline(radialStep * 0.20, '-', "$20"%$ Undershoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex", "LineWidth",1, "FontSize",14);
yline(-radialStep * 0.20, '-', "$20"%$ Overshoot Limit",
```

```
    ”LabelHorizontalAlignment”,”right”, ”LabelVerticalAlignment”,”bottom”,
    ”Interpreter”,”latex”, ”LineWidth”,1, ”FontSize”,14);

%Plot Response Lines
plot(t, yCLaug(:,1), 'LineWidth', 2.0, 'Color', plotColors(1,:)); hold on;
plot(t, xtruelqr(:,1), '-', 'LineWidth', 2.0, 'Color', plotColors(2,:));

%Plot Styling
grid on;
title('Radial Deviation vs Time', ”Interpreter”,”Latex”,
    ”FontSize”,plotTitleFontSize);
ylabel('$“delta r$ (km)', ”Interpreter”,”Latex”, ”FontSize”,axisFontSize);
xlabel('Time (s)', ”Interpreter”,”Latex”, ”FontSize”,axisFontSize);
xlim([t(1), t(end)]);
ylim([radialStep * -0.30, 1.1 * abs(radialStep)]);

%---------------- ANGULAR POSITION ----------------%
nexttile;

angularStep = deltatheta0;

%Settling / overshoot bands (same as previous plots)
patch([0, Torbit, Torbit, 0], [0, 0, angularStep * 0.05, angularStep * 0.05],
    plotColors(3, :), ”FaceAlpha”, 0.2, ”EdgeColor”, ”none”);
hold on; box on;
patch([0, t(end), t(end), Torbit, Torbit, 0], [-angularStep * 0.2, -angularStep
    * 0.2, angularStep * 0.2, angularStep * 0.2, 0, 0], plotColors(4, :),
    ”FaceAlpha”, 0.2, ”EdgeColor”, ”none”);

yline(0, '--', 'LineWidth', 1.5, ”FontSize”,14);
xline(Torbit, '-', ”1x Orbital Period”, ”LabelHorizontalAlignment”,”center”,
    ”LabelVerticalAlignment”,”top”, ”Interpreter”,”latex”, ”LineWidth”,1,
    ”FontSize”,14);
yline(angularStep * 0.05, '-', ”$5“%$ Settling Limit”,
    ”LabelHorizontalAlignment”,”right”, ”LabelVerticalAlignment”,”top”,
    ”Interpreter”,”latex”, ”LineWidth”,1, ”FontSize”,14);
yline(angularStep * 0.20, '-', ”$20“%$ Undershoot Limit”,
    ”LabelHorizontalAlignment”,”right”, ”LabelVerticalAlignment”,”top”,
    ”Interpreter”,”latex”, ”LineWidth”,1, ”FontSize”,14);
yline(-angularStep * 0.20, '-', ”$20“%$ Overshoot Limit”,
    ”LabelHorizontalAlignment”,”right”, ”LabelVerticalAlignment”,”bottom”,
    ”Interpreter”,”latex”, ”LineWidth”,1, ”FontSize”,14);

%Plot Response Lines
h(1) = plot(t, yCLaug(:,2), 'LineWidth', 2.0, 'Color', plotColors(1,:)); hold
    on; %Manual Placement
h(2) = plot(t, xtruelqr(:,3), '-', 'LineWidth', 2.0, 'Color', plotColors(2,:));
    %LQR Placement

%PLot Styling
grid on;
title('Angular Position vs Time', ”Interpreter”,”Latex”,
    ”FontSize”,plotTitleFontSize);
ylabel('$“delta “theta$ (rad)', ”Interpreter”,”Latex”, ”FontSize”,axisFontSize);
xlabel('Time (s)', ”Interpreter”,”Latex”, ”FontSize”,axisFontSize);
xlim([t(1), t(end)]);
ylim([angularStep * -0.30, 1.1 * abs(angularStep)]);
legend(h, [”Manual (Part 4)”, ”LQR (Part 6)”],
    ”Interpreter”,”Latex”,'Location','NorthEast', ”FontSize”, axisFontSize);
```

```
607
608 %% Manual vs LQR Closed-Loop Acceleration Plots
609 figure('Color','w');
610 tiles = tiledlayout('flow');
611 axisFontSize = 14;
612 plotTitleFontSize = 16;
613 figureTitleFontSize = 22;
614
615 title(tiles, 'Manual vs LQR Thrust Acceleration Response',
        "Interpreter","Latex", "FontSize", figureTitleFontSize);
616 plotColors = orderedcolors("gem");
617
618 %---------------- RADIAL THRUST ACCELERATION ----------------%
619 nexttile;
620
621 %Shaded allowable band +/- 0.01 g
622 patch([0, t(end), t(end), 0], [-0.01, -0.01, 0.01, 0.01], plotColors(5, :),
        'FaceAlpha', 0.1, 'EdgeColor', 'none');
623 hold on;
624
625 %Plot Lines
626 plot(t, umanualg(:,1), 'LineWidth', 2.0, 'Color', plotColors(1,:)); %Manual Pole
        Placement Response
627 plot(t, ulqrg(:,1), 'LineWidth', 2.0, 'Color', plotColors(2,:)); %LQR Pole
        PLacement Response
628
629 %Add in Bounding Lines / Limits
630 yline(0.01, '-', "0.01 g Limit", "Interpreter","Latex",
        "LabelHorizontalAlignment","left", "LabelVerticalAlignment","top");
631 yline(-0.01, '-', "Interpreter","Latex");
632 yline(0, '-', "LineWidth", 1.0);
633
634 %Plot Styling
635 ylim([-0.02, 0.02]);
636 xlim([t(1), t(end)]);
637 grid on; box on;
638 title('Radial Thrust Acceleration', "Interpreter","Latex", "FontSize",
        plotTitleFontSize);
639 ylabel('$"delta u-1$ (g)', "Interpreter","Latex", "FontSize", axisFontSize);
640 xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
641
642 %---------------- IN-TRACK THRUST ACCELERATION ----------------%
643 nexttile;
644
645 %Shaded allowable band +/- 0.01 g
646 patch([0, t(end), t(end), 0], [-0.01, -0.01, 0.01, 0.01], plotColors(5, :),
        'FaceAlpha', 0.1, 'EdgeColor', 'none');
647 hold on;
648
649 %Plot Lines
650 h(1) = plot(t, umanualg(:,2), 'LineWidth', 2.0, 'Color', plotColors(1,:));
651 h(2) = plot(t, ulqrg(:,2), 'LineWidth', 2.0, 'Color', plotColors(2,:));
652
653 %Add in Bounding Lines / Limits
654 yline(0.01, '-', "0.01 g Limit", "Interpreter","Latex",
        "LabelHorizontalAlignment","left", "LabelVerticalAlignment","top");
655 yline(-0.01, '-', "Interpreter","Latex");
656 yline(0, '-', "LineWidth", 1.0);
657
```

```
%Plot Styling
ylim([-0.02, 0.02]);
xlim([t(1), t(end)]);
grid on; box on;
title('In-Track Thrust Acceleration', "Interpreter","Latex", "FontSize", plotTitleFontSize);
ylabel('$"delta u-2$ (g)', "Interpreter","Latex", "FontSize", axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);
legend(h, ["Manual (Part 4)", "LQR (Part 6)"], "Interpreter","Latex", 'Location','SouthEast', "FontSize", axisFontSize);

%---------------- TOTAL THRUST ACCELERATION MAGNITUDE ----------------%
nexttile(tiles, "south", [1, 2]);

%Calculate Total Forcing Mangitude
umagmanualg = sqrt(umanualg(:,1).2 + umanualg(:,2).2);
umaglqrg   = sqrt(ulqrg(:,1).2 + ulqrg(:,2).2);

%Shaded allowable band +/- 0.01 g
patch([0, t(end), t(end), 0], [0, 0, 0.01, 0.01], plotColors(5, :), 'FaceAlpha', 0.1, 'EdgeColor', 'none');
hold on;

%Plot Lines
plot(t, umagmanualg, 'LineWidth', 2.0, 'Color', plotColors(1,:));
plot(t, umaglqrg, 'LineWidth', 2.0, 'Color', plotColors(2,:));

%Add in Bounding Lines / Limits
yline(0.01, '-', "0.01 g Limit", "Interpreter","Latex", "LabelHorizontalAlignment","left", "LabelVerticalAlignment","top");
yline(0, '-', "LineWidth", 1.0);

%Plot Styling
ylim([-0.001, 0.02]);
xlim([t(1), t(end)]);
grid on; box on;
title('Total Thrust Acceleration Magnitude', "Interpreter","Latex", "FontSize", plotTitleFontSize);
ylabel('$"—"mathbf-u"—$ (g)', "Interpreter","Latex", "FontSize", axisFontSize);
xlabel('Time (s)', "Interpreter","Latex", "FontSize", axisFontSize);

%% Manual Controller: Full-State vs Manual+Observer Closed-Loop Response
figure('Color','w');
tiles = tiledlayout('flow');
axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
title(tiles,'Manual Full-State vs Manual+Observer Closed-Loop Response', "Interpreter","Latex","FontSize",figureTitleFontSize);
plotColors = orderedcolors("gem");

%---RADIAL DEVIATION---%
nexttile;
radialStep = deltar0;

%Add in shaded allowance band
patch([0,Torbit,Torbit,0],[0,0,radialStep*0.05,radialStep*0.05], plotColors(3,:),"FaceAlpha",0.2,"EdgeColor","none");
hold on;
```

```
patch([0,t(end),t(end),Torbit,Torbit,0],
    [-radialStep*0.2,-radialStep*0.2,radialStep*0.2,radialStep*0.2,0,0],
    plotColors(4,:),"FaceAlpha",0.2,"EdgeColor","none");
yline(0,'--','LineWidth',1.5,"FontSize",14);
xline(Torbit,'-',"1x Orbital Period", "LabelHorizontalAlignment","center",
    "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(radialStep*0.05,'-',"$5“%$ Settling Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(radialStep*0.20,'-',"$20“%$ Undershoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(-radialStep*0.20,'-',"$20“%$ Overshoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","bottom",
    "Interpreter","latex","LineWidth",1,"FontSize",14);

%Plot Lines
plot(t,yCLaug(:,1),'LineWidth',2.0,"Color",plotColors(1,:));
plot(t,xtruecase1(:,1),'LineWidth',2.0,"Color",plotColors(2,:));

%Plot Styling
grid on; box on;
title('Radial Deviation vs
    Time',"Interpreter","Latex","FontSize",plotTitleFontSize);
ylabel('$“delta r$ (km)',"Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (s)',"Interpreter","Latex","FontSize",axisFontSize);
xlim([t(1),t(end)]);
ylim([radialStep*-0.30,1.1*abs(radialStep)]);

%---ANGULAR POSITION---%
nexttile;
angularStep = deltatheta0;

%Add in shaded allowance band
patch([0,Torbit,Torbit,0],[0,0,angularStep*0.05,angularStep*0.05],
    plotColors(3,:),"FaceAlpha",0.2,"EdgeColor","none");
hold on;
patch([0,t(end),t(end),Torbit,Torbit,0],
    [-angularStep*0.2,-angularStep*0.2,angularStep*0.2,angularStep*0.2,0,0],
    plotColors(4,:),"FaceAlpha",0.2,"EdgeColor","none");
yline(0,'--','LineWidth',1.5,"FontSize",14);
xline(Torbit,'-',"1x Orbital Period", "LabelHorizontalAlignment","center",
    "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(angularStep*0.05,'-',"$5“%$ Settling Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(angularStep*0.20,'-',"$20“%$ Undershoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(-angularStep*0.20,'-',"$20“%$ Overshoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","bottom",
    "Interpreter","latex","LineWidth",1,"FontSize",14);

%Plot Lines
h(1) = plot(t,yCLaug(:,2),'LineWidth',2.0,"Color",plotColors(1,:));
h(2) = plot(t,xtruecase1(:,3),'LineWidth',2.0,"Color",plotColors(2,:));

```

```
%Plot Styling
grid on; box on;
title('Angular Position vs
    Time',"Interpreter","Latex","FontSize",plotTitleFontSize);
ylabel('$"delta "theta$ (rad)',"Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (s)',"Interpreter","Latex","FontSize",axisFontSize);
xlim([t(1),t(end)]);
ylim([angularStep*-0.30,1.1*abs(angularStep)]);
legend(h,["Manual (Full-State)","Manual+Observer (Zero Error)"],
    "Interpreter","Latex","Location","NorthEast","FontSize",axisFontSize);


%% LQR Closed-Loop Response with Observer Error (0% vs 50% Initial Error)
figure('Color','w');
tiles = tiledlayout('flow');
axisFontSize = 16;
plotTitleFontSize = 18;
figureTitleFontSize = 22;
title(tiles,'LQR Closed-Loop Response with Observer Error',
    "Interpreter","Latex","FontSize",figureTitleFontSize);
plotColors = orderedcolors("gem");

%---RADIAL DEVIATION---%
nexttile;
radialStep = deltar0;

%Add in Allowable Tolerance
patch([0,Torbit,Torbit,0],[0,0,radialStep*0.05,radialStep*0.05],
    plotColors(3,:),"FaceAlpha",0.2,"EdgeColor","none");
hold on;
patch([0,t(end),t(end),Torbit,Torbit,0],
    [-radialStep*0.2,-radialStep*0.2,radialStep*0.2,radialStep*0.2,0,0],
    plotColors(4,:),"FaceAlpha",0.2,"EdgeColor","none");
yline(0,'--','LineWidth',1.5,"FontSize",14);
xline(Torbit,'-',"1x Orbital Period", "LabelHorizontalAlignment","center",
    "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(radialStep*0.05,'-',"$5"%$ Settling Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(radialStep*0.20,'-',"$20"%$ Undershoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
    "Interpreter","latex","LineWidth",1,"FontSize",14);
yline(-radialStep*0.20,'-',"$20"%$ Overshoot Limit",
    "LabelHorizontalAlignment","right", "LabelVerticalAlignment","bottom",
    "Interpreter","latex","LineWidth",1,"FontSize",14);

%Plot Lines
plot(t,xtruelqr(:,1),'LineWidth',2.0,"Color",plotColors(1,:));
plot(t,xtruelqrerr(:,1),'LineWidth',2.0,"Color",plotColors(2,:));

%Plot Styling
grid on; box on;
title('Radial Deviation vs Time
    (LQR)',"Interpreter","Latex","FontSize",plotTitleFontSize);
ylabel('$"delta r$ (km)',"Interpreter","Latex","FontSize",axisFontSize);
xlabel('Time (s)',"Interpreter","Latex","FontSize",axisFontSize);
xlim([t(1),t(end)]);
ylim([radialStep*-0.30,1.1*abs(radialStep)]);
```

```
791
792 %---ANGULAR POSITION---%
793 nexttile;
794 angularStep = deltatheta0;
795
796 %Add in shaded allowable bands
797 patch([0,Torbit,Torbit,0],[0,0,angularStep*0.05,angularStep*0.05],
        plotColors(3,:),"FaceAlpha",0.2,"EdgeColor","none");
798 hold on;
799 patch([0,t(end),t(end),Torbit,Torbit,0],
        [-angularStep*0.2,-angularStep*0.2,angularStep*0.2,angularStep*0.2,0,0],
        plotColors(4,:),"FaceAlpha",0.2,"EdgeColor","none");
800 yline(0,'--','LineWidth',1.5,"FontSize",14);
801 xline(Torbit,'-',"1x Orbital Period", "LabelHorizontalAlignment","center",
        "LabelVerticalAlignment","top",
        "Interpreter","latex","LineWidth",1,"FontSize",14);
802 yline(angularStep*0.05,'-',"$5“%$ Settling Limit",
        "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
        "Interpreter","latex","LineWidth",1,"FontSize",14);
803 yline(angularStep*0.20,'-',"$20“%$ Undershoot Limit",
        "LabelHorizontalAlignment","right", "LabelVerticalAlignment","top",
        "Interpreter","latex","LineWidth",1,"FontSize",14);
804 yline(-angularStep*0.20,'-',"$20“%$ Overshoot Limit",
        "LabelHorizontalAlignment","right", "LabelVerticalAlignment","bottom",
        "Interpreter","latex","LineWidth",1,"FontSize",14);
805 h(1) = plot(t,xtruelqr(:,3),'LineWidth',2.0,"Color",plotColors(1,:));
806 h(2) = plot(t,xtruelqrerr(:,3),'LineWidth',2.0,"Color",plotColors(2,:));
807
808 %Plot Styling
809 grid on; box on;
810 title('Angular Position vs Time
        (LQR)',"Interpreter","Latex","FontSize",plotTitleFontSize);
811 ylabel('$“delta “theta$ (rad)',"Interpreter","Latex","FontSize",axisFontSize);
812 xlabel('Time (s)',"Interpreter","Latex","FontSize",axisFontSize);
813 xlim([t(1),t(end)]);
814 ylim([angularStep*-0.30,1.1*abs(angularStep)]);
815 legend(h,["LQR (0“% Initial Error)","LQR (50“% Initial Error)"],
        "Interpreter","Latex","Location","NorthEast","FontSize",axisFontSize);
```