Control Techniques Beyond Basic PID

Assignment 3

Submission Guidelines

Format: All work must be submitted as a **MATLAB Live Script (.mlx file)**.
Your submission should include:

- Section headers for each problem
- MATLAB code with clear comments
- Text explanations and observations after each task
- Performance comparison tables

File naming: `SmartThrottle_Ass3_YourName_rollno.mlx`

Learning Resources

Watch these videos before starting:

1. **Lead and Lag Compensators:** `https://www.youtube.com/watch?v=xLhvil5sDcU` (Brian Douglas)
2. **Understanding Compensators:** `https://www.youtube.com/watch?v=rH44ttR3G4Q` (Designing Lead Compensator - Brian Douglas)
3. **Feedforward Control:** `https://www.youtube.com/watch?v=FW_ay7K4jPE` (MATLAB Tech Talk)

1 Problem 1: Lead Compensator for Speed Improvement

Consider a sluggish DC motor position control system:

$$G(s) = \frac{20}{s(s+2)(s+5)}$$

This system has slow response. Your goal is to speed it up using a lead compensator.

Tasks:

a) **Baseline PID Performance:**

- Design a PID controller to achieve zero steady-state error
- Use MATLAB's PID Tuner or manual tuning
- Plot the step response and record:
  - Rise time
  - Settling time (2% criterion)
  - Overshoot percentage

b) **Lead Compensator Design:**

- Design a lead compensator with the form:

$$C_{lead}(s) = K_c \frac{s+z}{s+p} \quad \text{where } |z| < |p|$$

- Try: $z = 3$, $p = 15$, and tune K_c for good response
- Combine with your PID: $C_{total}(s) = C_{PID}(s) \times C_{lead}(s)$
- Plot the step response

c) **Performance Comparison:**

- Create a comparison plot showing:
  - PID only response
  - PID + Lead compensator response
- Create a table comparing metrics:
  - Rise time improvement
  - Settling time improvement
  - Effect on overshoot
- Explain how the lead compensator improved the transient response

d) **Parameter Experimentation:**

- Try different zero-pole combinations:
  - $z = 2, p = 10$
  - $z = 5, p = 20$
  - $z = 4, p = 12$
- Plot all responses on one figure
- Discuss how the zero-pole placement affects the speed of response

Hint: Use `tf()` *to create the compensator, then multiply with PID controller:* `C_total = C_pid * C_lead`

2 Problem 2: Lag Compensator for Steady-State Accuracy

Consider a temperature control system with steady-state error issues:

$$G(s) = \frac{50}{(s+1)(s+3)(s+8)}$$

Tasks:

a) **Proportional Controller Analysis:**

- Implement a P controller with $K_p = 2$
- Apply a unit step reference
- Measure and report the steady-state error
- Plot the response showing the error clearly

b) **Lag Compensator Design:**

- Design a lag compensator:

$$C_{lag}(s) = K_c \frac{s+z}{s+p} \quad \text{where } |z| > |p|$$

- Try: $z = 1$, $p = 0.1$, $K_c = 1$
- Combine with P controller: $C(s) = K_p \times C_{lag}(s)$
- Plot the step response

c) **Steady-State Error Reduction:**

- Compare steady-state errors:
  - P controller only
  - P + Lag compensator
- Create a table showing:
  - Steady-state error (calculate as: $e_{ss} = |r_{final} - y_{final}|$)
  - Error reduction percentage
  - Settling time for both cases
- Explain why the lag compensator reduces steady-state error

d) **Trade-off Analysis:**

- Observe and document any negative effects of the lag compensator:
  - Does settling time increase?
  - Is the response slower?
- Try reducing the pole value to $p = 0.01$ and compare
- Discuss the trade-off between accuracy and speed

3 Problem 3: Feedforward + Feedback Control

Consider a cruise control system where the vehicle encounters disturbances (hills, wind) while trying to maintain speed.

Plant model (vehicle dynamics):

$$G(s) = \frac{1}{s + 0.5}$$

Scenario:

The vehicle faces a known disturbance (uphill slope) that can be predicted.

Tasks:

a) **Feedback-Only Control:**

- Design a PI controller: $C_{fb}(s) = K_p + \frac{K_i}{s}$ with $K_p = 2$, $K_i = 1$
- Create a disturbance signal: $d(t) = 0.3 \cdot \text{step}(t - 5)$ (uphill starts at t=5s)
- Simulate using `lsim()` with:
  - Reference speed: $r(t) = 1$ (constant)
  - Disturbance injected at plant output: $y = G(u) - d$
- Plot: reference, output, and error over time
- Measure: maximum error and recovery time when disturbance hits

b) **Adding Feedforward Control:**

- Design a feedforward controller that predicts the disturbance
- Feedforward gain: $K_{ff} = 1/\text{dcgain}(G)$ (inverse of plant DC gain)
- Total control signal: $u = u_{feedback} + K_{ff} \cdot d_{predicted}$
- Simulate with the same disturbance
- Plot: reference, output, and error

c) **Performance Comparison:**

- Create comparison plots:
  - Output response (feedback only vs feedforward+feedback)
  - Error signals for both cases
- Create a table comparing:
  - Maximum error when disturbance hits
  - Recovery time
  - Steady-state error after disturbance
- Explain why feedforward helps with disturbance rejection

d) **Unpredicted Disturbance Test:**

- Add an unpredicted disturbance at t=15s: $d_{unpredicted}(t) = 0.2 \cdot \text{step}(t - 15)$

- Keep feedforward compensating only the first disturbance
- Show that feedback control handles the unpredicted disturbance
- Discuss the complementary nature of feedforward and feedback

Hint: For disturbance injection, you can simulate the system in a for-loop or use the state-space representation with disturbance input.

MATLAB Implementation Tip:

```
% Feedforward + Feedback example structure
for i = 2:length(t)
    % Feedback controller
    e = r(i) - y(i-1);
    u_fb = Kp * e + Ki * e_int;
    e_int = e_int + e * dt;

    % Feedforward controller (if disturbance is known)
    if t(i) >= 5
        u_ff = Kff * 0.3;  % Compensate for known disturbance
    else
        u_ff = 0;
    end

    % Total control
    u_total = u_fb + u_ff;

    % Plant with disturbance
    % y(i) = G(u_total) - d(i)
    % (implement using discretization or lsim)
end
```

4 Problem 4: Simple MIMO System Control

Understanding MIMO Systems

What is MIMO? MIMO stands for Multi-Input Multi-Output. Unlike SISO (Single-Input Single-Output) systems where one controller controls one output, MIMO systems have multiple inputs and multiple outputs that interact with each other.

Real-World Example: Consider a simplified two-tank liquid level control system where both tanks interact:

- Tank 1 receives input flow u_1 and drains to Tank 2
- Tank 2 receives input flow u_2 and also receives overflow from Tank 1
- We want to control levels h_1 and h_2 independently
- The challenge: changing flow u_1 affects BOTH tank levels, not just Tank 1!

MIMO Transfer Function Matrix:
In SISO systems, we write: $Y(s) = G(s) \cdot U(s)$ (one output, one input)
In MIMO systems, we write it as a matrix:

$$\begin{bmatrix} H_1(s) \\ H_2(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}$$

This means:

$$H_1(s) = G_{11}(s) \cdot U_1(s) + G_{12}(s) \cdot U_2(s)$$
$$H_2(s) = G_{21}(s) \cdot U_1(s) + G_{22}(s) \cdot U_2(s)$$

Interpreting the Transfer Functions:

- $G_{11} = \frac{2}{s+1}$: How much does u_1 affect h_1? (Main effect - direct control)
- $G_{12} = \frac{0.5}{s+2}$: How much does u_2 affect h_1? (Cross-coupling)
- $G_{21} = \frac{1}{s+1.5}$: How much does u_1 affect h_2? (Cross-coupling)
- $G_{22} = \frac{3}{s+1}$: How much does u_2 affect h_2? (Main effect - direct control)

The Key Concept: Each input affects MULTIPLE outputs. This is called *coupling* or *interaction*.

Watch this for visual understanding:

- **MIMO Control Introduction:** `https://www.youtube.com/watch?v=qhXl55EZ5jg` (Understanding MIMO Systems)

Tasks:

a) **Open-Loop MIMO Analysis:**

- Create the MIMO transfer function in MATLAB
- Apply step inputs: $u_1 = 1$, $u_2 = 0$
- Plot both outputs h_1 and h_2 to see the coupling effect
- Repeat with: $u_1 = 0$, $u_2 = 1$
- Explain how changing one input affects both outputs

b) **Decentralized PI Control:**

- Design two independent PI controllers:
  - C_1 controls h_1 using u_1 (ignoring h_2)
  - C_2 controls h_2 using u_2 (ignoring h_1)
- Try: $K_{p1} = 1.5$, $K_{i1} = 0.8$ and $K_{p2} = 1.2$, $K_{i2} = 0.6$
- Set references: $r_1 = 1.0$, $r_2 = 1.5$
- Simulate the closed-loop MIMO system
- Plot both tank levels and control signals

c) **Analyzing Interaction Effects:**

- Observe the coupling effects in the response:
  - Does controlling h_1 affect h_2?
  - Do the outputs interfere with each other?
  - Is there overshoot or oscillation due to interaction?
- Create a table showing final settling values and settling times for both outputs

d) **Sequential Setpoint Changes:**

- Test the system with time-varying references:
  - $r_1 = 1.0$ for all time
  - $r_2 = 1.0$ for $t < 10$s, then $r_2 = 1.5$ for $t \geq 10$s
- Plot the response showing how changing r_2 affects h_1
- Discuss why MIMO systems are more challenging than SISO systems
- Suggest how the controllers could be improved (conceptually, no implementation needed)

Hint: Create MIMO system using: `G = [G11 G12; G21 G22]` *where each element is a transfer function.*

MATLAB Implementation Tips:

```
% Creating MIMO transfer function - Step by step
% Step 1: Define each transfer function element
G11 = tf(2, [1 1]);       % Effect of u1 on h1
G12 = tf(0.5, [1 2]);     % Effect of u2 on h1 (coupling!)
G21 = tf(1, [1 1.5]);     % Effect of u1 on h2 (coupling!)
G22 = tf(3, [1 1]);       % Effect of u2 on h2

% Step 2: Create MIMO system as a matrix
G = [G11 G12; G21 G22];  % 2x2 MIMO system

% Step 3: To see how it responds to inputs, use step()
% Example: What happens if u1=1 and u2=0?
u = [1; 0];  % Input vector
step(G * u);  % This won't work directly, see below

% Better approach: Use lsim() for MIMO
t = 0:0.01:20;
u1 = ones(size(t));    % Step input to u1
u2 = zeros(size(t));   % No input to u2
u = [u1; u2]';         % Combine inputs (note the transpose!)

[y, t] = lsim(G, u, t);
% y(:,1) is h1 (tank 1 level)
% y(:,2) is h2 (tank 2 level)

```

```
plot(t, y(:,1), 'b-', t, y(:,2), 'r--');
legend('h1 (Tank 1)', 'h2 (Tank 2)');

% For decentralized control with feedback, simulate in loop:
h1 = zeros(size(t));
h2 = zeros(size(t));
e1_int = 0; e2_int = 0;

for i = 2:length(t)
    % Calculate errors
    e1 = r1(i) - h1(i-1);
    e2 = r2(i) - h2(i-1);

    % Two independent PI controllers
    u1(i) = Kp1*e1 + Ki1*e1_int;
    u2(i) = Kp2*e2 + Ki2*e2_int;

    % Update integrals
    e1_int = e1_int + e1*dt;
    e2_int = e2_int + e2*dt;

    % Apply inputs to MIMO plant using lsim for one time step
    % Or use state-space discretization
    u_vec = [u1(i); u2(i)];

    % For simplicity, you can use lsim for small segments
    % or discretize the MIMO system using c2d()
end

% Key point: Each controller only looks at "its" output
% but both inputs affect both outputs through G matrix!
```

Summary Questions

After completing all problems, answer these briefly:

1. When would you choose a lead compensator over increasing PID gains?
2. What are the main advantages and limitations of lag compensators?
3. Why is feedforward control called "proactive" compared to feedback?
4. What makes MIMO systems more difficult to control than SISO systems?