MIDTERM EVALUATION

SMART THROTTLE CONTROL

NAME: AYUSH MISHRA

ROLL NUMBER: 250253

OVERALL SUMMARY

KEY CONCEPTS LEARNED

- The project helped me move beyond solving equations mechanically and develop an intuitive feel for how control systems behave in the real world.
- Transfer functions in the Laplace (s) domain began to feel like meaningful descriptions of system dynamics, not just mathematical expressions used for calculations.
- I developed a clear understanding of how poles and zeros shape system behavior, especially in terms of stability, transient response, steady-state behavior, and phase effects.
- Classifying systems as Type 0, Type 1, and higher-order systems made steady-state error much easier to anticipate for standard inputs like step and ramp signals.
- Time-domain performance measures such as rise time, settling time, overshoot, and steady-state error became practical tools for judging control quality rather than abstract definitions.
- The sessions consistently highlighted that control design is about trade-offs, particularly between speed and stability, which helped build a more realistic design mindset.
- Frequency-domain analysis using Bode magnitude and phase plots provided strong intuition about robustness, bandwidth, and stability margins.
- Studying P, PI, and PID controllers clarified why they are widely used, while also revealing their limitations in more demanding systems.
- Learning about lead and lag compensators, feedforward control, and basic MIMO concepts showed how system performance can be improved through thoughtful structural design rather than repeated gain tuning.

Tools and Techniques Applied

- MATLAB was used primarily as a tool for verification and insight, not as a shortcut to answers.
- Functions such as tf, step, stepinfo, and feedback helped in modeling systems and studying closed-loop behavior.
- Commands like bode and lsim were used to analyze frequency response and system behavior for non-step inputs.
- A strong emphasis on manual sketching of Bode plots helped build intuition about how poles, zeros, and controllers affect magnitude and phase.
- The Final Value Theorem was frequently used to estimate steady-state behavior analytically before running simulations.
- I developed the habit of predicting system behavior before simulation, using MATLAB results to confirm understanding.
- PI and PID controllers, along with lead and lag compensators, were analyzed both on paper and through simulation to see their real impact on performance.
- Feedforward control was studied as a way to handle known disturbances proactively instead of reacting only after errors appear.
- Basic exposure to MIMO systems, including a 2×2 transfer function matrix, helped broaden understanding beyond simple SISO systems.

Major Challenges Faced

- One of the main challenges was learning to predict system response analytically instead of immediately relying on MATLAB simulations.
- Understanding system behavior for non-step inputs such as ramp and sinusoidal signals required careful reasoning and repeated validation.

- Gaining confidence in how poles and zeros affect phase, gain, and steady-state behavior took time, especially when designing lead and lag compensators.
- Clearly distinguishing between feedback and feedforward control required a shift from reactive thinking to more anticipatory system design.
- Working with MIMO systems was challenging due to interactions between control channels that are not present in SISO systems.
- Managing complex MATLAB simulations with multiple controllers and plots required better organization and discipline.
- The most subtle challenge was learning how to apply theoretical analysis to improve real physical systems, not just achieve mathematically correct results.

My Contribution to the Project

- Focused on analysis-driven understanding and system-level reasoning throughout the project.
- Analyzed plant models by identifying system type, pole-zero locations, and expected steady-state behavior.
- Used analytical tools, including closed-form calculations and the Final Value Theorem, to predict responses before simulation.
- Manually sketched and interpreted Bode plots to evaluate stability margins and performance trade-offs.
- Designed simple controllers and compensators to meet specific performance requirements.
- Compared different control strategies to understand their strengths, limitations, and practical suitability.
- Consistently worked to connect theoretical concepts with real-world behavior, ensuring the work remained relevant from an engineering perspective.

IMPLEMENTATION

DRONE ALTITUDE/SPEED CONTROL SYSTEM

Overall Workflow / Architecture

The drone altitude and speed control system is structured as a **closed-loop feedback system**, since stable flight cannot be achieved using fixed throttle values alone. The key idea is to continuously adjust motor thrust based on the actual motion of the drone.

The overall workflow of the system is:

Sensors → Error Calculation → Control Logic → Throttle Adjustment → Motors → Drone Motion → Feedback

In this structure, sensors measure the drone's altitude and vertical speed. These measurements are compared with the desired altitude or climb rate, and the resulting error is used to determine how the throttle should be adjusted. The motors then generate thrust accordingly, and the updated motion is sensed again. This continuous feedback loop allows the drone to automatically correct disturbances such as wind, payload variation, or battery voltage changes.

Key Components / Blocks

1. Sensors and Feedback

Altitude and vertical speed measurements are essential for effective control. These signals describe the drone's vertical motion and provide real-time feedback to the control system. Without accurate feedback, reliable altitude regulation would not be possible.

2. Reference and Error Computation

The reference block defines the desired altitude or vertical speed. By comparing this reference with the measured values, an error signal is generated. This error represents how far the drone is from the target condition and serves as the primary input to the controller.

3. Controller (PID-Based Control Logic)

A PID-based control approach is suitable for drone altitude and speed control because it balances responsiveness, accuracy, and stability.

- **Proportional action** provides immediate correction when the drone deviates from the desired altitude or speed.
- **Integral action** compensates for constant effects such as gravity and payload weight, preventing long-term altitude drift.
- **Derivative action** improves damping and reduces overshoot during ascent and descent.

Instead of directly controlling altitude through thrust, the system also regulates **vertical speed**. Large altitude errors result in a controlled climb or descent, while smaller errors near the target reduce the commanded vertical speed gradually. This approach ensures smooth motion and minimizes overshoot.

Hovering is achieved by maintaining **zero vertical speed**, not zero thrust. The controller continuously adjusts throttle so that thrust balances gravity and external disturbances, allowing the drone to remain at a constant altitude.

4. Throttle Modulation / Allocation

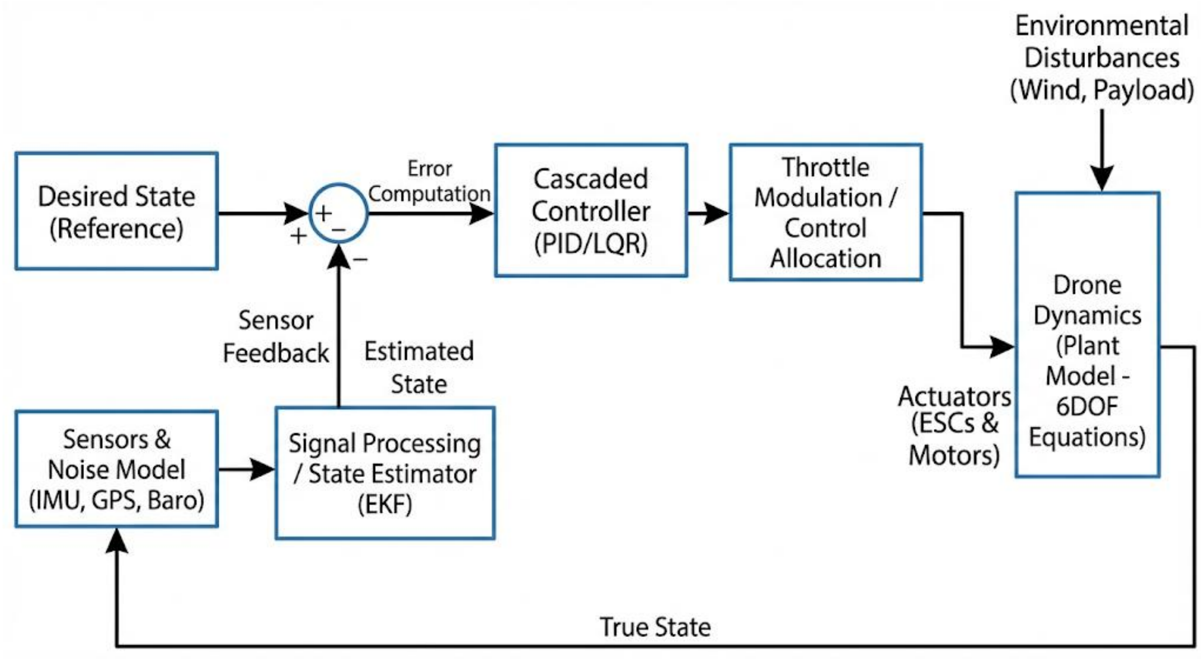
The throttle modulation block converts controller output into smooth and bounded throttle commands. This prevents abrupt thrust changes and ensures safe, stable operation of the motors.

5. Actuators (Motors and Propellers)

Motors and propellers act as actuators that convert throttle commands into thrust. Any change in throttle directly affects altitude and vertical speed, making this block critical to system performance.

6. Safety and Constraint Mechanisms

Safety mechanisms are included to limit maximum and minimum throttle values and to restrict climb and descent rates. These constraints help maintain stable and predictable behavior, even during sudden commands or disturbances.


Environmental Disturbances (Wind, Payload)
Desired State (Reference)
+
−
Error Computation
Cascaded Controller (PID/LQR)
Throttle Modulation / Control Allocation
Drone Dynamics (Plant Model - 6DOF Equations)
Actuators (ESCs & Motors)
Sensor Feedback
Estimated State
Sensors & Noise Model (IMU, GPS, Baro)
Signal Processing / State Estimator (EKF)
True State

Brief Research Note

In practical drone systems, altitude and speed control are usually implemented using a **layered control structure**. Fast inner loops stabilize the drone's attitude, while slower outer loops regulate altitude and vertical speed through throttle control. Although specific implementations vary, the fundamental feedback-based structure remains consistent.

Modern drones often enhance this framework using feedforward compensation and improved sensor processing. However, the underlying principle remains rooted in classical control theory: measuring system behavior, comparing it with a reference, and correcting errors through controlled actuation.

Stable altitude control and hovering are achieved by continuously regulating vertical speed and throttle through feedback, rather than relying on fixed thrust values.