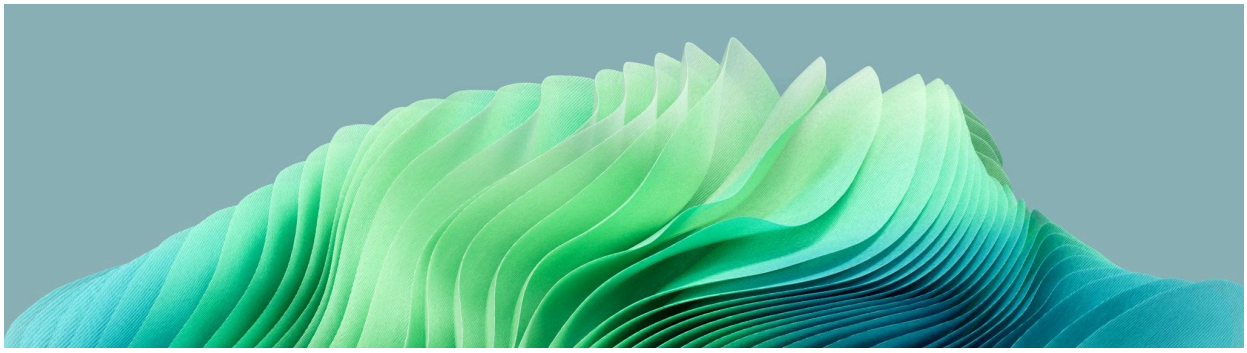

Smart Throttle Control Report

Summary

- **Control System** :- A system that directs the flow, manages, commands and regulates the behaviour of a plant to achieve a result.

- **Laplace Transform :-** Helps transform big differential equations in algebraic equations.

- Two types of control systems exist: open loop and closed loop system.We use Controllers to improve the stability and reject disturbances of a system.

- Transfer Function is Output by input in frequency domain.

- Order of the Transfer Function depends on the number of integrators in the function.
  Type 0 fails to track step and ramp response completely.
  Type 1 tracks step perfectly but not ramp.
  Type 2 tracks both perfectly.

- In the time domain if there is unbounded growth and in the frequency domain the pole is in the right hand plane then it is an unstable system.

- **Controllers**
    - **P Controller** decreases the rise time but increases the overshoot and does not affect the steady state error.
    - **I Controller** eliminates the steady state error but increases overshoot and settling time thus system instability Integral Windup: In systems with actuator saturation, the integral term can accumulate excessively, causing delayed or poor response. Therefore,anti-windup techniques are often required.
    - **D Controller** introduces damping into the system, which reduces oscillations and overshoot, making the system more stable—especially for systems with high inertia.By anticipating changes in error, reduces the settling time.The derivative action is highly sensitive to noise in the error signal because it amplifies high-frequency components. This can cause erratic control action, so filtering is usually necessary.
- The P, I, and D terms are not independent. Adjusting one to fix a specific issue (like rise time) often degrades another (like stability).
  In complex real-world systems, this interdependency makes ”perfect” tuning difficult (and sometimes impossible) using only a PID structure
  To achieve superior performance, we not only use PID but decouple these requirements using Compensators, Feedforward and MIMO
  Strategies.

- Compensators are like fine tuning the response after the coarse adjustment is
  done through PID
  There are two types of Compensators
- Lead Compensator : (|Zo>Po|) It increases the speed and reduces the settling
  time of the system
- Lag Compensator : (|ZO<Po|) It eliminates steady-state error with much higher
  Precision

- Feedforward Controllers are proactive. Instead of waiting for an error to occur (as feedback control does), it measures disturbances entering the

system and adjusts the control variable *before* the system's output is affected.

- Modern Systems usually use MIMO systems and there are various ways to implement them

Decentralized Controller (Ignore It)

Breaks the MIMO system into multiple independent Single-Input Single-Output (SISO) loops.

- **Method:** Pairs one input to one output (e.g., Loop A controls Temp, Loop B controls Pressure).
- **Handling Coupling:** It **ignores** interactions, treating them as external disturbances. Fails if interactions are strong.

Decoupling Control (Cancel It)

Mathematically untangles the system so inputs don't interfere with each other.

- **Method:** Adds a "Decoupler" block that generates signals to **cancel out** the cross-interactions.
- **Result:** The complex system acts like a set of simple, independent loops (Input 1 *only* affects Output 1).

Model Predictive Control / MPC (Optimize It)

Uses a computer model to predict future behavior and solve an optimization problem in real-time.

- **Method:** Adjusts **all inputs simultaneously**.
- **Strength:** Naturally handles interactions and physical constraints (e.g., valves maxing out) better than any other me

- How the location of poles and zeroes affect the rise time, settling time and overshoot.(Performance metrics)
- Using lsim() for generating varying responses
- Actuator Saturation (Physical Limits): Real-world hardware (like motors) has physical limits (e.g., voltage between -10V and 10V). Simulating this by clamping the control signal in your code using logic like max(min(u, 10), -10).

- Integral Windup:
  When the Actuator saturates (hits its limit), the Integral (I) term keeps accumulating error (winding up) even though the system cannot respond faster. This causes the system to overshoot significantly and take a long time to settle once the target is finally reached.
- Anti-Windup (Back-Calculation Method):
  To fix windup, implementing a Back-Calculation strategy. This involves calculating the difference between the calculated control signal (u) and the saturated actual output (u_sat) and feeding this difference back to reduce the integral term. This stops the integrator from growing uselessly during saturation.
- Derivative Noise Amplification:
  Derivative (D) term is highly sensitive to sensor noise. By adding random noise to the measurement (y_noisy),we see that the D-term amplifies this noise, causing high-frequency oscillations in the control signal (u) even if the output (y) looks relatively smooth.
- Low-Pass Filtering:
  To solve the noise issue, implementing a First-Order Low-Pass Filter on the derivative term. This smooths out the jagged noise spikes while preserving the true trend of the data, allowing the D-term to provide damping without erratic behavior.
- Discrete Implementation (Coding the Loop):
  How to write a Discrete Time Simulation using a for loop. Manually calculating P, I, and D terms at every time step (dt), which is how digital controllers work in real embedded systems.
-

Project: Smart HVAC Temperature Control System

1. Overall Workflow / Architecture

The system follows a classic Closed-Loop Feedback architecture. It constantly monitors the environment and adjusts the output to match a desired state (Set Point).

The Flow:

1. User Input (Set Point): You set the target temperature (e.g., 24 degree celsius).

2. Sensing (Feedback): A sensor reads the actual room temperature (e.g. 28 degree celsius).
3. Processing (The Error Calculation): The controller compares the two values. Error = Desired Temp - Actual Temp.
4. Decision: The controller decides the action. Since the room is too hot, it commands the cooling unit to turn ON or increase speed.
5. Actuation: The fan/AC spins up, blowing cold air.
6. Loop: The sensor detects the temperature dropping (27 degree celsius... 26 degree celsius), and the process repeats until the error is zero.

2. Key Components / Blocks

To build this physically or simulate it, you need these distinct modules:

- The Plant (The System to be Controlled):
    - This is the Room or the specific space you are heating/cooling.

      Feedback Sensor

      Function: Converts physical heat into an electrical signal the controller can read.

- The Controller (The Brain):
    - Component: A microcontroller
    - Function: Runs the control algorithm.
- Actuator Driver:
    - Component: A Relay Module or Transistor/MOSFET.
    - Function: Microcontrollers operate at low power (3.3V/5V). Fans/Heaters often need 12V or AC power.
- The Processor:
    - Component: A DC Fan (for cooling) or a Heating Element/Resistor (for heating).
- Safety Mechanism (Critical):
    - Function: A hard limit in the code or a thermal fuse. If the temperature exceeds a dangerous limit (e.g., 60 degree celsius), the system must cut power immediately to prevent fire or melting components.

3. Brief Research Note

- Thermal Inertia (Lag): Unlike a lightbulb that turns on instantly, a heater takes time to warm up a room, and the room stays warm even after the heater turns off. Real HVAC systems must account for this delay so they don't overshoot the target temperature massively.

- Hysteresis (The "Deadband"): If you set your AC to 24 degree celsius, you don't want it turning on at 24.1 degree celsius and 23.9 degree celsius every second. That would destroy the motor. Real systems use a "deadband" (e.g., turn on at 25 degree celsius, turn off at 23 degree celsius).