PID Controller Optimization System Documentation

Model Structure

The PID optimization system consists of four interconnected subsystems that work together to optimize controller performance for a rotational mechanical system:

DOF Rotational Drone Attitude Axis

Modelling the rotation of a drone around one axis (e.g. pitch), where:

- Input = control torque (from PID)
- Output = angular position (θ), which is fed back
- System has rotational inertia and damping, like a real drone body reacting to torque

Main System Architecture:

- Set point signal generator for reference input
- Summing junction for error calculation (setpoint - feedback)
- PID controller block with adjustable P, I, D gains
- Actuator subsystem that models physical dynamics
- Performance metrics subsystem for quantitative evaluation
- System response analysis for visualization and data export

Actuator Subsystem:

- Physical model of a rotational mechanical system with:
    - Torque source (control input from PID)
    - Rotational spring element
    - Rotational damper
    - Rotational inertia
    - Mechanical rotational reference
    - Motion and torque sensors for feedback

Performance Metrics Subsystem:

- Parallel computation of two error metrics:
    - IAE (Integral Absolute Error): Absolute value block → Integrator
    - ISE (Integral Squared Error): Square block → Integrator

System Response Analysis:

- Captures error signal and feedback loop signals
- Provides visualization capabilities
- Exports time-domain signals to workspace

Mathematical Explanation

Cost Function Definition

The optimization uses a weighted combination of two standard error metrics

cost = iae + 0.1 * ise

Where:

- **IAE (Integral Absolute Error)**: Mathematically expressed as:

  $IAE = \int |e(t)| dt$

  This integrates the absolute value of error over time, penalizing all deviations equally regardless of sign.

- **ISE (Integral Squared Error)**: Mathematically expressed as:

  $ISE = \int e^2(t) dt$

This integrates the squared error over time, giving greater weight to larger errors

Optimization Algorithm and Error Signal Utilization

The system implements Particle Swarm Optimization (PSO) to find optimal PID gains

1. **Parameter Space Exploration**:

- The algorithm initializes 50 particles (parameter sets) randomly distributed across the search space.
- Each particle represents a candidate PID parameter set: [Kp, Ki, Kd].
- Bounded search space: [0-100] for Kp and Ki, [0-10] for Kd.

2. **Fitness Evaluation Process**:
    - For each parameter set, the system:
        - Configures PID controller with candidate gains
        - Runs simulation with the "unknown" system dynamics
        - Collects error signal between setpoint and feedback
        - Calculates cost function value from IAE and ISE metrics.
3. **Parameter Adjustment Mechanism**:
    - Each particle maintains memory of its personal best position (lowest cost)
    - The swarm shares information about global best position
    - Particles update velocities based on:

      v_new = w*v_old + c1*r1*(p_best - current) + c2*r2*(g_best - current)

Where w is inertia weight, c1/c2 are acceleration coefficients, and r1/r2 are random values

Black-Box Optimization Approach

The algorithm works without requiring explicit system dynamics knowledge through

Model-Free Evaluation:

- Treats the controlled system as a black box
- No mathematical model of the plant dynamics needed
- Relies solely on input-output behavior[

Iterative Performance Assessment:

1. PSO generates candidate PID parameter sets

2. Each parameter set is tested on the actual system
3. System responds to standard input (setpoint signal)
4. Response quality measured purely through error metrics (IAE, ISE)
5. Parameter sets ranked based on weighted cost function values
6. Swarm converges toward optimal values by sharing information.

Implementation Workflow:

1. Set initial PID parameters and optimization constraints
2. For each iteration:
    - Apply control parameters to the system
    - Measure system response (without knowledge of dynamics)
    - Calculate error metrics and cost function
    - Update particle positions and velocities
3. Terminate when maximum iterations reached or convergence criteria met
4. Apply best found PID parameters for final validation

This black-box approach enables effective controller tuning for systems with unknown or complex dynamics, relying purely on performance feedback without requiring mathematical models of the system being controlled.

Run the system in the order:

- PID_Optimization-Model (Simulink model)
- optimizePID.m file
- validatePIDPerformance.m fil

- NOTE:While running validatePIDPerformance use validatePIDPerformance(x_opt) in the command window and do not press the run button in matlab directly