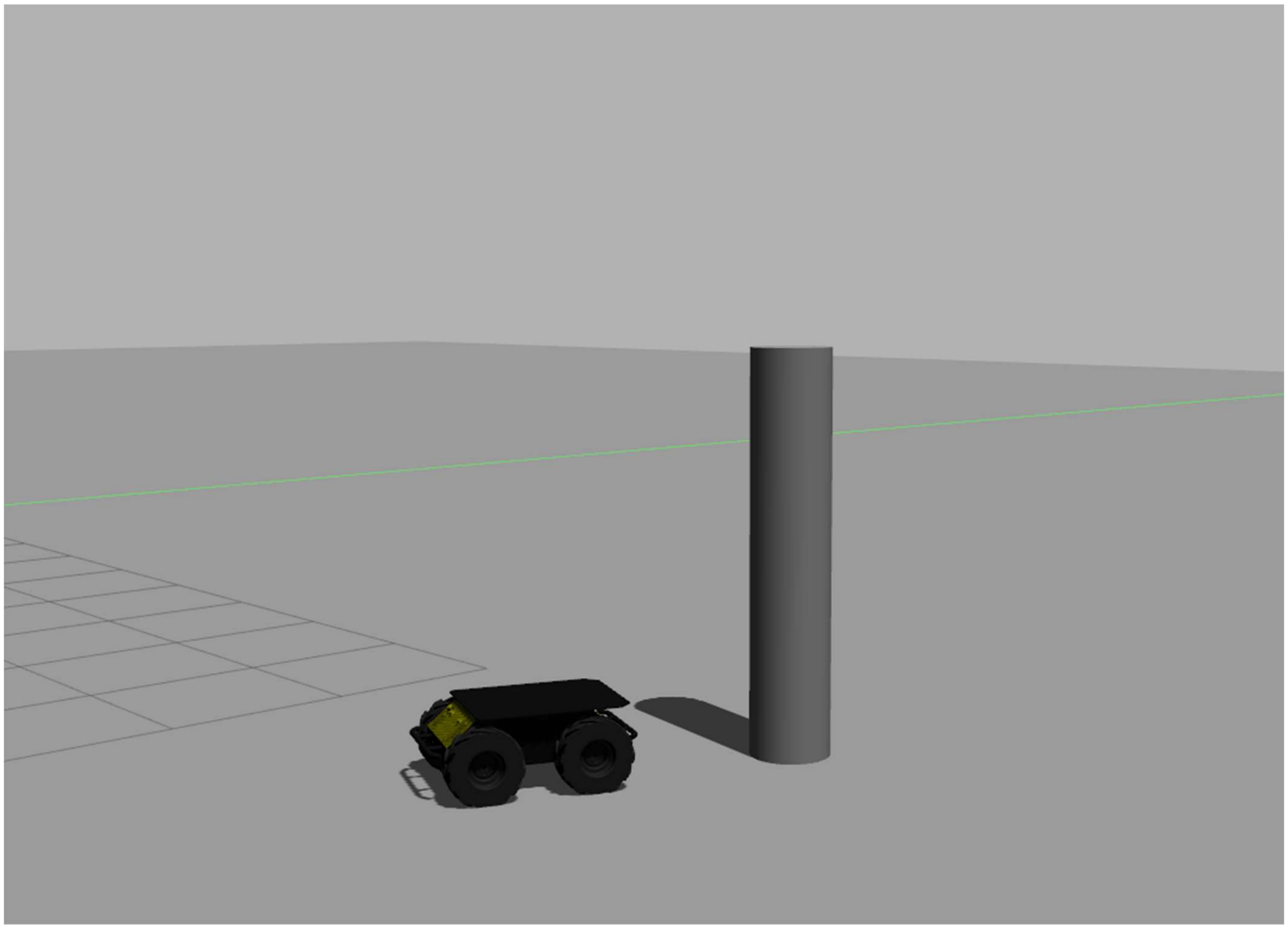
# PID CONTROL REPORT

*A0266505A*

**Arun Gandhi Shyam Krishna**

## OBJECTIVE

The main objective of this project is to design a PID controller, tune it and makes use of the same to control the motion of a Husky robot – efficiently – in a virtual environment. The Husky robot has two completely decoupled motions – linear and steering. So, two independent PID controllers should be designed to control this system. The simulation environment consists of a pillar which is the target destination for the Husky robot, which starts from the origin. The Husky Bot has to stop infront of the pillar within a predetermined distance.

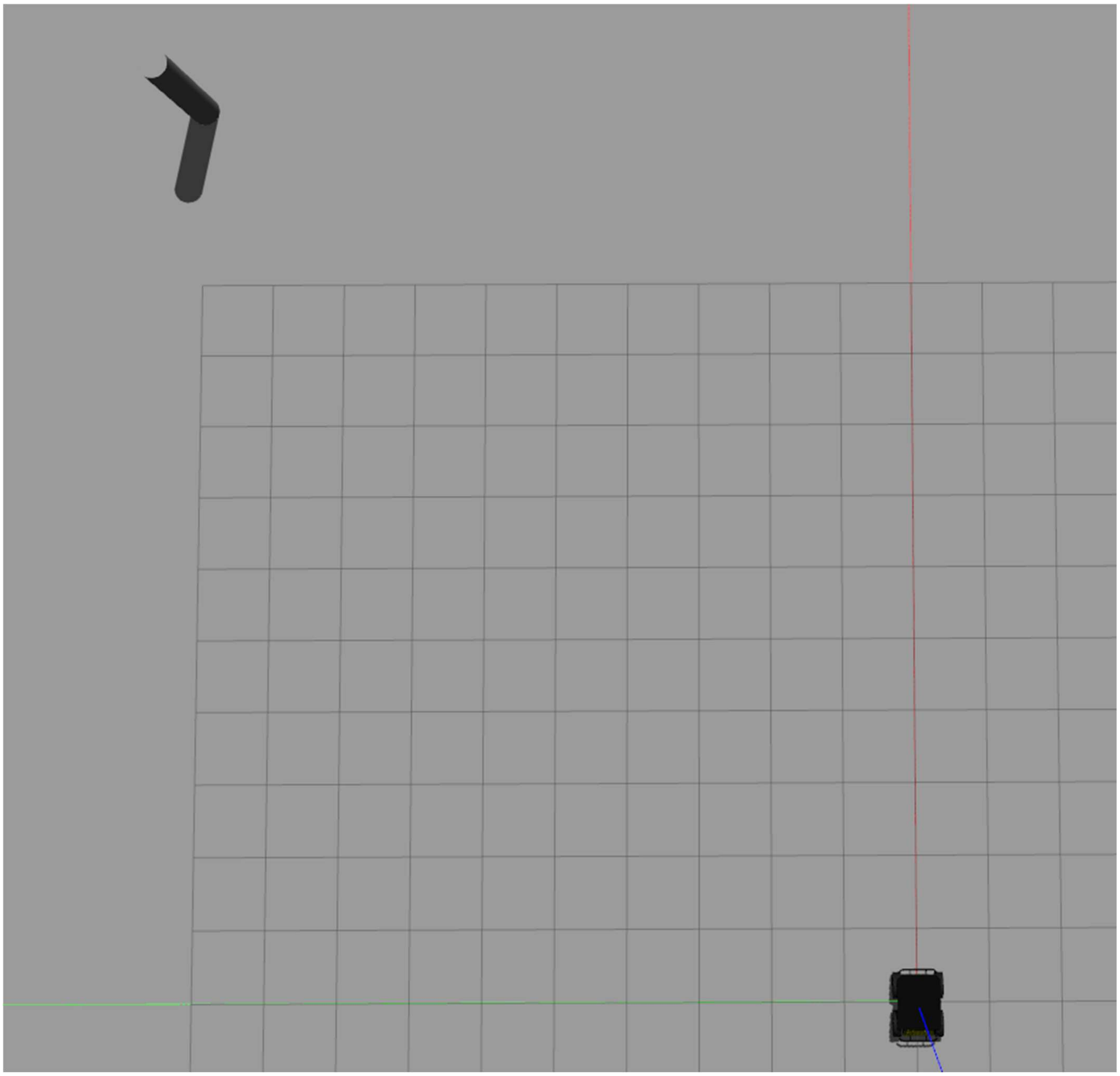


## INITIAL CONDITIONS

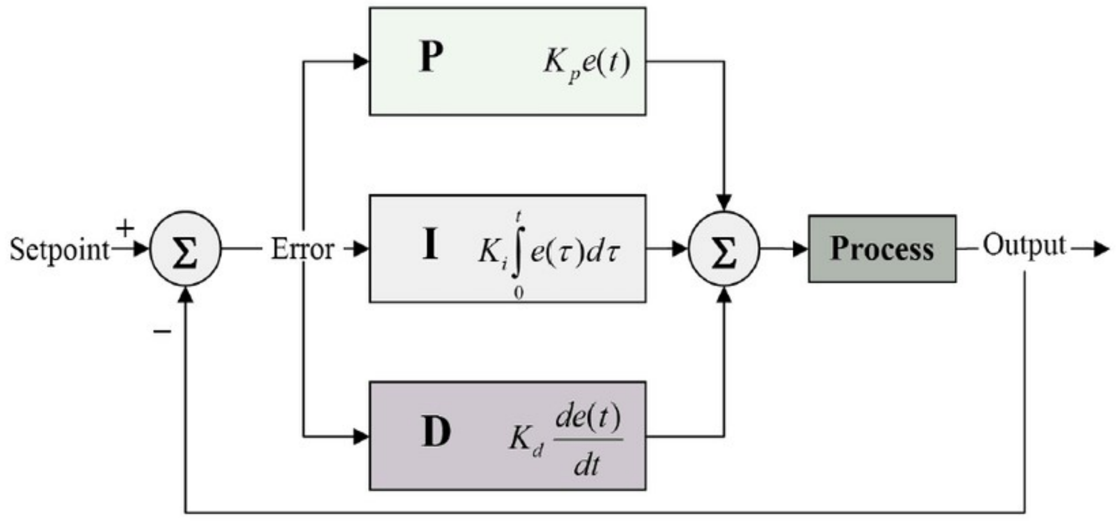
**Position of Husky Robot:** (0,0) – Origin

**Position of Pillar:** (12.5, 10)  **Angle:** 38.66$^0$  **Distance:** 16 units

**Mass of Husky:** 450 Kg



## OVERVIEW OF PID CONTROL



- **Proportional Term(P) :** The proportional term yields an output value that is proportionate to the current error value. By multiplying the error by a factor Kp, referred to as the proportional gain constant, the proportional response can be changed. For a given change in the error, a high proportional gain causes a significant change in the output. A high proportional gain can cause the system to lose stability. A tiny gain, on the other hand, results in a controller that is less sensitive or responsive and has a small output reaction to a significant input error. When responding to system disturbances, if the proportional gain is too low, the control action may be inappropriate. This is unable to eliminate **steady-state error**.

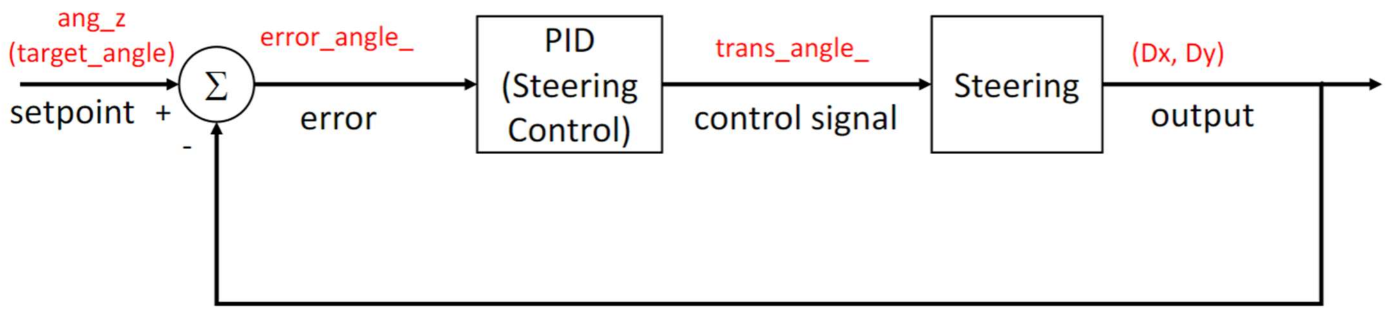
- **Integral Term(I) :** The integral term's contribution is directly correlated with the size and duration of the error. The integral in a PID controller calculates the cumulative offset that needs to be rectified earlier by adding the instantaneous error over time. The integral gain (Ki) is then multiplied by the cumulative error before being added to the controller output. The integral term reduces the **residual steady-state** error that results from a pure proportional controller and speeds up the process's progress towards the setpoint. The present value, however, may **overshoot** the setpoint value because the integral term reacts to past errors that have accumulated.

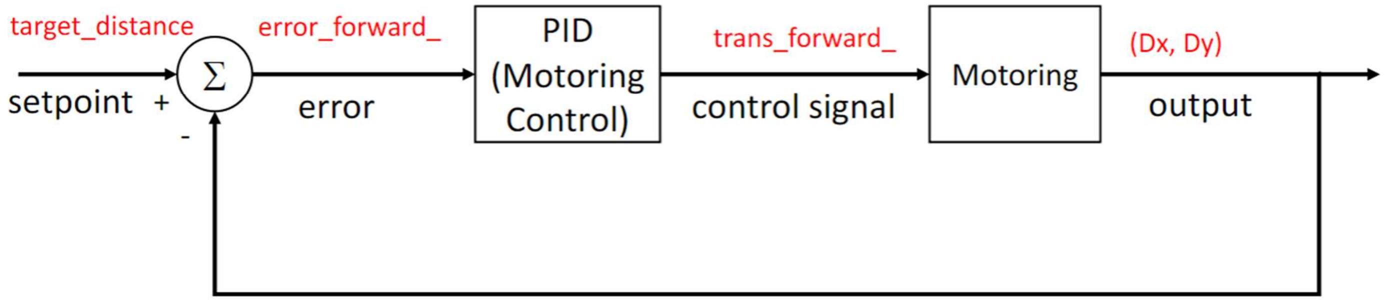
- **Derivative Term(D) :** The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd.

# IMPLEMENTATION OF PID CONTROL IN ROS-NOETIC

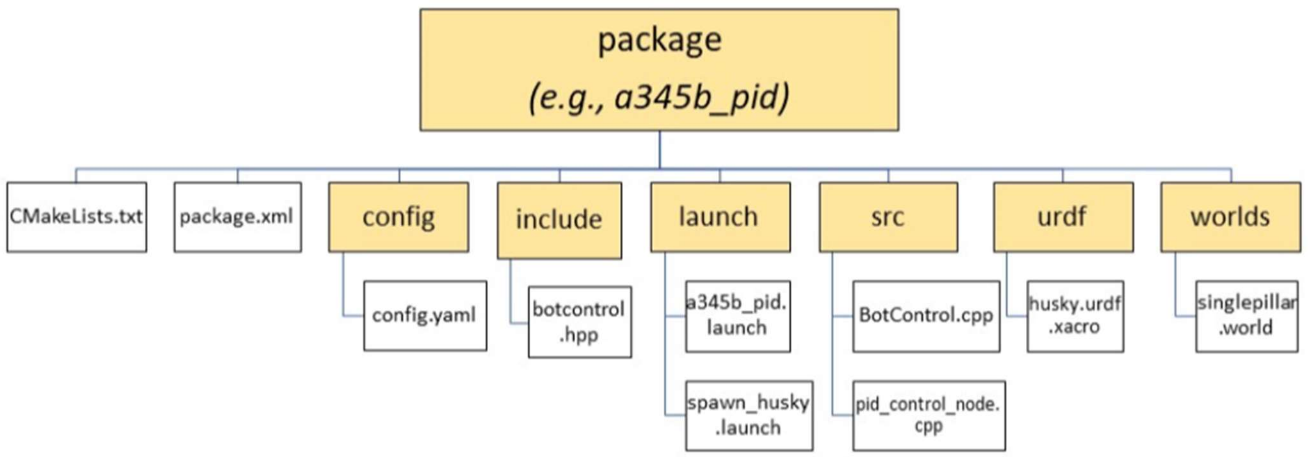
**ASSUMPTION**: The steering mechanism of the HUSKY is completely decoupled with its rectilinear motion.

So, two independent PID controls have been designed to control the DOFs individually.





## STRUCTURE AND RATIONALE OF THE PACKAGE



### config.yaml

This file declares the values of all PID gain constants and some other constants used in the code to control the robot. This includes,

- Kp_d, Ki_d, Kd_d – The gains of the first controller (for the translatory motion).
- Kp_a, Ki_a, Kd_a – The gains of the second controller (for steering mechanism).
- dt – The time frame under consideration.
- X

### botcontrol.hpp

This file defines how the Husky robot receives inputs and provides outputs. Variables are classified into private variables and public variables.

- **Packages used:** 1) `sensor_msgs` 2) `nav_msgs` `3) geometry_msgs`
  4) `std_msgs` 5) `gazebo_msgs`

- **Private variables:**
The variables used for localisation and for PID control.
2) The topics that the simulations subscribes and the topics that it publishes are defined.

- **Public variables:**

  1)The variables used to tune the PID control.
  2)The variables to assign the targeted distance of the Husky robot from the pillar and the location of the pillar.

**Botcontrol.cpp**

This file defines the PID control.

## CODE ANALOGY

### PID 1:

```
trans_forward = error_forward * Kp_d + I_forward * Ki_d

                + D_forward * Kd_d
```

### PID 2:

```
trans_angle = error_angle * Kp_a + I_angle * Ki_a

                + D_angle * Kd_a
```

- Here `trans_forward` and `trans_angle` are the control signals provided to the actuators to respond to these specific conditions.
- `error_forward and error_angle` are the instantaneous errors of those specific parameters at that particular instant.
- `I_forward and I_angle` are the summation of instantaneous errors of those parameters over a period of time.
- `D_forward and D_angle` are the rate of change of these errors at that instant.
- Eventually, these quantities are multiplied with their respective gains to yield the desired control signal.

# TUNING OF PID CONTROLLER

## POINTS TO CONSIDER

### Settling Time

Essentially Settling Time is a measure of the time needed by a process to return to the Set Point – in response to a disturbance. For some PIDs it's essential for the loop to settle within a fixed period of time as longer periods of variability could have a negative impact on one or more processes. A shorter Settling Time is generally a good thing as it is indicative of a control loop that is performing more efficiently.

### Percent Overshoot

It is the degree to which the Process Variable exceeds the associated Set Point in response to a Set Point change. Percent Overshoot can be an especially important performance assessment tool when tuning loops that regulate highly sensitive processes.

### Output Travel

PID controllers that are tuned too aggressively can accelerate the mean time to failure of the valve. The unplanned downtime that results from failure can be costly. Knowing the amount of Output Travel can help practitioners to tune loops so as to limit wear and tear and avoid unexpected downtime.

### Stability

Stability – or Robustness – is one more performance measure that many practitioners consider when tuning PIDs. It's well understood that instability cripples the control of a given loop. Unstable loop can cascade downstream and negatively impact other production processes.

## RISE TIME

Different applications differ from their requirements. Some require a quick rise and some may not. But there is a trade off between rise time, settling time and stability. The controller should be designed according to the needs of the environment.

## OBJECTIVE OF TUNING PIDs

It is highly necessary to tune a PID controller to implement it in any industrial applications, in robots, in machineries or any other places, because the requirements and constraints that we are bound into are unique. As a designer of a PID control, it is our duty to consider all parameters and possibilities into our considerations and make the control most efficient and effective.

**Objective in Tuning the PID control for the Husky bot:**

- Minimize the time required for reaching the desired location.
- Precision coupled with accuracy.
- Stability – with minimum noise in the error flow.
- Decrease the settling time.
- Decrease percentage overshoot.

**Trade – offs faced:**

- Rise time vs Percentage Overshoot.
- Rise time vs Settling time.
- Performance vs Robustness

## HOW I TUNED THE PID CONTROL?

A PID controller's gains can be discovered through trial and error. This procedure becomes comparatively simple once we are aware of the importance of each gain parameter. In this technique, the I and D terms are initially set to zero and the proportional gain is raised until the output of the loop oscillates. As we increase the proportional gain, the system becomes faster, but care must be taken not make the system unstable. The integral gain is increased to eliminate the oscillations after P has been set to produce the desired quick response. The integral term increases overshoot while lowering steady state error. The integral term is adjusted to achieve a minimal steady state error. When we are satisfied with the P and I gain, the derivative term is increased to get an optimal settling time. Increasing the derivative term reduces overshoot and results in bigger gains with stability, but it makes the system extremely noise sensitive.

## MERITS AND DEMERITS OF MY DESIGN

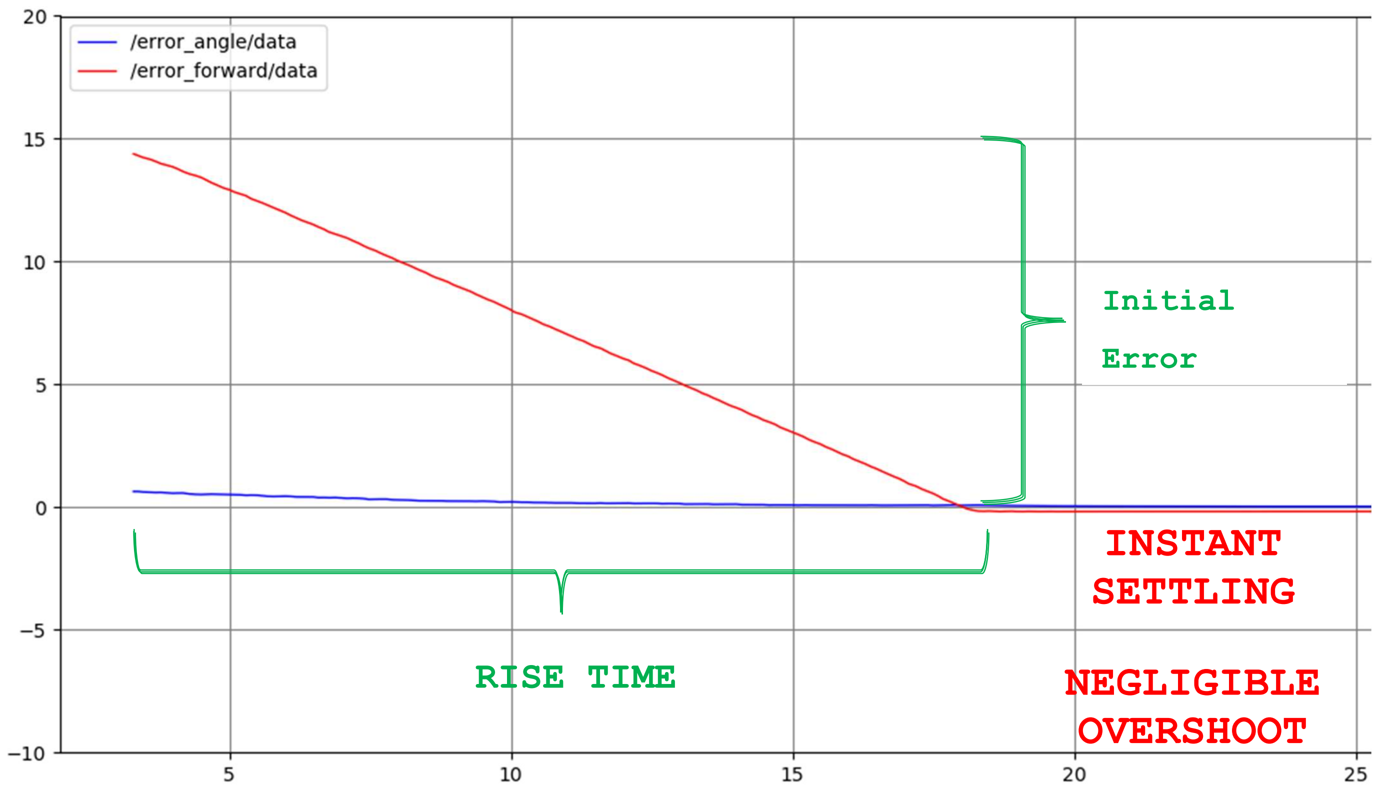
**MERITS:**

- Appropriate rise time (not too quick or slow).
- Insignificant overshooting.
- Instant settling.
- Smooth movement.

**DEMERITS:**

- Corrupted with slight noise.
- Could not produce the same level of efficiency in every environment.

# PERFORMANCE OF PID CONTROL



## CONCLUSION AND KEY LEARNINGS

- There are physical constraints which play a major role in designing, regardless of the theoretical constraints.
- The actuators have a limiting capacity to provide work. So, after a certain value there is no significant change in the output with increase in the gains.
- One can find a Kp, for which there is no steady state error. But that is not feasible in all the cases.
- The P function of a PID controller is absolutely necessary.
- The I function removes the steady-state error, but leads to overshoot.
- The D function takes care of sudden change in slope of the control signal.
- In some applications, the set point is not fixed and mobile. So it is required to design a controller which provides the same efficiency in most of the cases.
- Overshooting is extremely problematic in many cases, as ours. Many applications are highly sensitive towards overshooting.