Technical Report: PID Controller Implementation in C

# 1. Introduction

This technical report outlines the implementation of a **PID Controller** in the C programming language. The PID (Proportional-Integral-Derivative) controller is a widely used mechanism in control systems for regulating processes to achieve a desired setpoint. In this project, the PID controller is used to regulate motor speed, which has various applications in industries such as robotics, automotive systems, and industrial automation.

# 2. Objective

The primary objective of this project is to:

Simulate a PID controller for motor speed regulation.

Demonstrate how the proportional, integral, and derivative components collectively minimize errors.

Provide insights into the performance of the PID controller under different conditions.

# 3. PID Controller Basics

## 3.1. What is a PID Controller?

A PID controller is a control loop feedback mechanism that computes an error value as the difference between a desired setpoint and the current process variable. The controller minimizes the error by adjusting the process input through three components:

**Proportional (P):** Provides a correction proportional to the current error.

**Integral (I):** Addresses accumulated errors over time.

**Derivative (D):** Predicts future errors by considering the rate of change of the error.

## 3.2. PID Formula

The control output is calculated as:

Where:

: Proportional gain

: Integral gain

: Derivative gain

: Difference between the setpoint and the current process value

# 4. Implementation

## 4.1. Tools and Environment

**Language:** C

**IDE:** Visual Studio

**Compiler:** GCC

## 4.2. Workflow

The program accepts two inputs: the desired speed (setpoint) and the current speed.

It calculates the error and computes the control signal using the PID formula.

Over multiple iterations, the system adjusts the motor speed to minimize the error.

## 4.3. Code Structure

**src/PID\_CONTROLLER.c:** Contains the main PID logic and simulation.

**Inputs:** Setpoint and current speed values.

**Outputs:** Adjusted speed over iterations.

# 5. Results

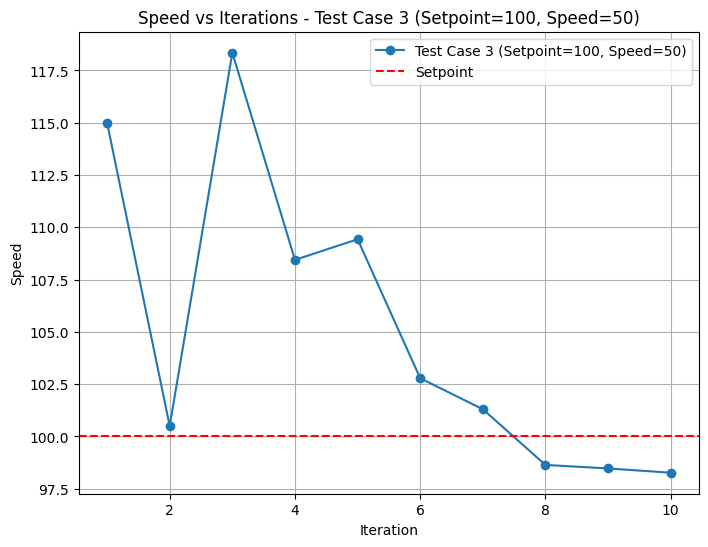
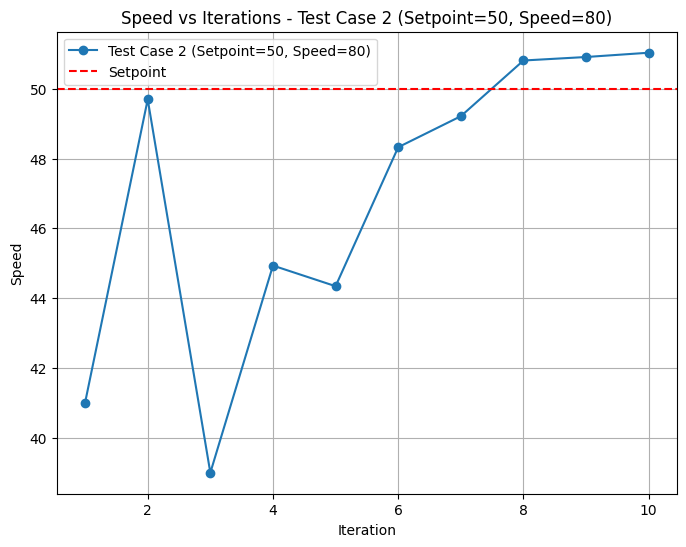
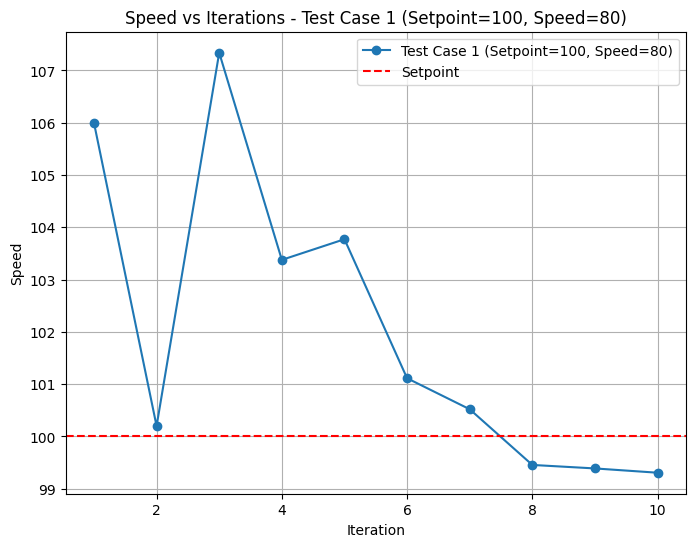
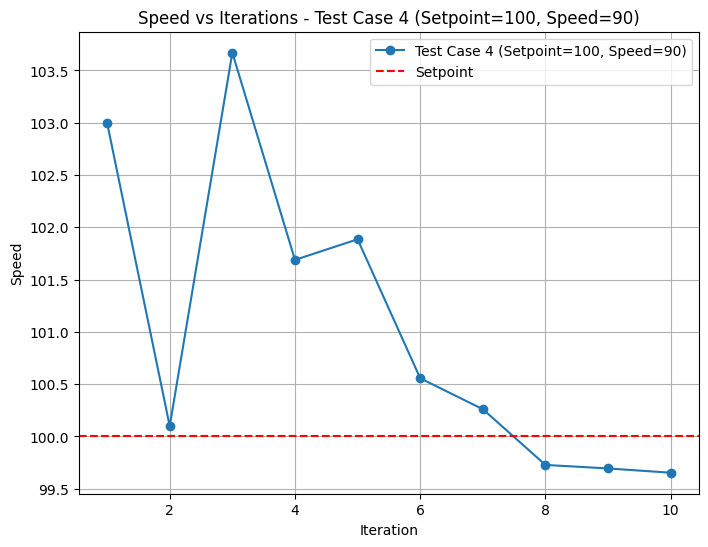
## 5.1. Test Cases

The following test cases were used to evaluate the PID controller:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Set Point** | **Initial Speed** | **Observations** |
| 1 | 100 | 80 | Converged to setpoint within 10 iterations. |
| 2 | 50 | 80 | Gradually decreased speed to setpoint. |
| 3 | 100 | 50 | Increased speed to match setpoint. |
| 4 | 100 | 90 | Fine-tuned speed close to setpoint. |

## 5.2. Graphs

The graphs below illustrate the convergence of speed over iterations for each test case:



# 6. Conclusion

The implementation of the PID controller demonstrated its ability to regulate motor speed efficiently. The system adjusted the speed to match the setpoint within a few iterations, showcasing the effectiveness of the proportional, integral, and derivative components. Future work may involve:

Testing under dynamic setpoints.

Incorporating noise to simulate real-world disturbances.

Comparing performance with alternative control algorithms.

# 7. References

MathWorks, "PID Controller Basics": <https://www.mathworks.com/help/simulink/ug/pid-controllers.html>

Franklin, Gene F., et al., *Feedback Control of Dynamic Systems*, Pearson Education.

# 8. Author

**Name:** Vishnu PV

**GitHub:** [vishnupv7](https://github.com/vishnupv7)

**Email:** [pvvishnu444@gmail.com](mailto:pvvishnu444@gmail.com)