**5.15 PID Controller**

A **Proportional-Integral-Derivative (PID) controller** is a feedback-based control loop mechanism widely used in industrial and automation systems to regulate processes requiring continuous control and automatic adjustments. It operates by comparing a desired target value (**setpoint** or **SP**) with the actual system output (**process variable** or **PV**). The difference between these values, known as the **error (e(t))**, is used to apply corrective actions through three control terms: **Proportional (P)**, **Integral (I)**, and **Derivative (D)**.

**Key Components of a PID Controller:**

1. **Proportional (P)**: Responds to the current error magnitude, providing immediate correction proportional to the error.
2. **Integral (I)**: Addresses residual steady-state errors by integrating past errors over time.
3. **Derivative (D)**: Predicts future error trends based on the rate of change, reducing overshoot and improving stability.

PID controllers enhance automation by minimizing human intervention and errors, ensuring precise control.

**5.15.1 Fundamental Operation**

The PID controller continuously calculates the error **e(t) = SP – PV** and applies corrections using a weighted sum of the P, I, and D terms. The control variable **u(t)** (e.g., valve position) is adjusted to minimize the error over time.

* **Proportional Term (P)**: Directly proportional to the current error. A high proportional gain (**Kp**) results in a stronger response but may cause instability if too high.
* **Integral Term (I)**: Eliminates steady-state errors by integrating past errors. However, excessive integral gain (**Ki**) can cause overshoot.
* **Derivative Term (D)**: Dampens system response by predicting future errors based on the error rate of change. High derivative gain (**Kd**) improves stability but can amplify noise.

**5.15.2 Tuning**

PID controllers require **tuning** to balance the P, I, and D terms for optimal performance. Tuning involves adjusting the gains (**Kp**, **Ki**, **Kd**) based on the system's response characteristics. Common tuning methods include:

1. **Manual Tuning**: Adjust gains iteratively to achieve desired performance.
2. **Ziegler-Nichols Method**: A systematic approach to determine initial gain values.
3. **Software-Based Tuning**: Advanced tools automate tuning by analyzing system responses and suggesting optimal parameters.

**5.15.3 Mathematical Form**

The PID control function is expressed as:

Where:

* Kp*Kp*​: Proportional gain
* Ki*Ki*​: Integral gain
* Kd*Kd*​: Derivative gain
* e(t)*e*(*t*): Error (SP – PV)

In the Laplace domain, the transfer function is:

**5.15.4 Selective Use of Control Terms**

Not all applications require all three terms. Common configurations include:

* **PI Controller**: Used when derivative action is sensitive to noise.
* **PD Controller**: Applied where integral action is unnecessary.
* **P or I Controller**: Simplified controllers for specific use cases.

**5.15.5 Controller Theory**

The PID controller output is the sum of the P, I, and D terms. The manipulated variable (**MV**) is calculated as:

**5.15.6 Detailed Control Terms**

**5.15.6.1 Proportional Term**

* Output: Pout=Kp⋅e(t)*Pout*​=*Kp*​⋅*e*(*t*)
* High Kp*Kp*​ improves responsiveness but risks instability.
* Low Kp*Kp*​ results in sluggish control.

**5.15.6.2 Integral Term**

* Output: Iout=Ki∫0te(τ) dτ*Iout*​=*Ki*​∫0*t*​*e*(*τ*)*dτ*
* Eliminates steady-state errors but can cause overshoot.

**5.15.6.3 Derivative Term**

* Output: Dout=Kdde(t)dt*Dout*​=*Kd*​*dtde*(*t*)​
* Improves stability by damping oscillations but is sensitive to noise.

**5.15.7 Loop Tuning**

Tuning ensures stability and optimal performance. Key considerations:

* **Stability**: Avoid excessive gains to prevent oscillations.
* **Manual Tuning**: Start with Kp*Kp*​, then adjust Ki*Ki*​ and Kd*Kd*​.
* **Software Tools**: Automate tuning for complex systems.

**5.15.8 Common Issues and Solutions**

**5.15.8.1 Integral Windup**

* **Cause**: Integral term accumulates excessive error during large setpoint changes.
* **Solution**: Disable integration or limit integral term bounds.

**5.15.8.2 Overshooting**

* **Cause**: Rapid changes in setpoint or disturbances.
* **Solution**: Use setpoint ramping or derivative of the process variable.

**5.15.9 Simulation and Practical Example**

**Simulation Setup**

* **Plant**: Brushless DC Motor (BLDC) with transfer function G(s)=1s3+3s2+2s*G*(*s*)=*s*3+3*s*2+2*s*1​.
* **Components**: Setpoint, Sum Block, PID Controller, Transfer Function, Scope.

**Tuning Process**

1. Start with Kp=Ki=Kd=0*Kp*​=*Ki*​=*Kd*​=0.
2. Increase Kp*Kp*​ until steady-state error is minimized.
3. Adjust Ki*Ki*​ to eliminate residual error.
4. Tune Kd*Kd*​ to reduce overshoot and oscillations.

**Optimal Parameters**

* Kp=10*Kp*​=10
* Ki=10*Ki*​=10
* Kd=3*Kd*​=3