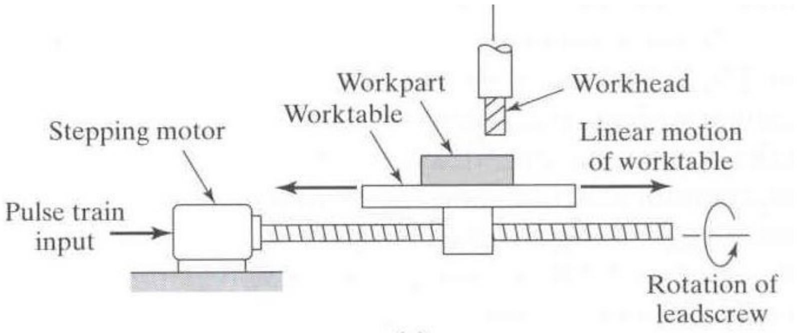
Microcontroller Application Series
PID Control 1/3

Credit : Shanying Zhu

Department of Automation, SJTU

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

- Control systems



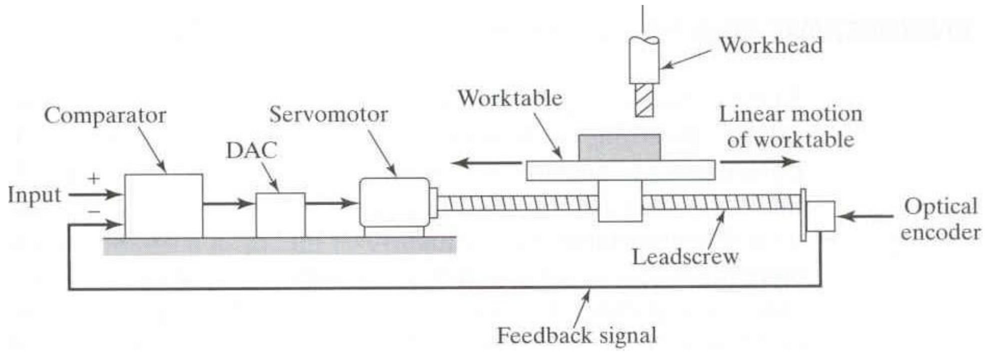
Open loop CNC

Problems with open loop systems

- They fly "blind"
- Cannot respond to disturbances
- Cannot adjust to different plants
- Models may be difficult or impossible to derive

VS

- Add feedback so controller knows results of actions.
- But, how to utilize these information to design a desirable controller?
- PID controller!

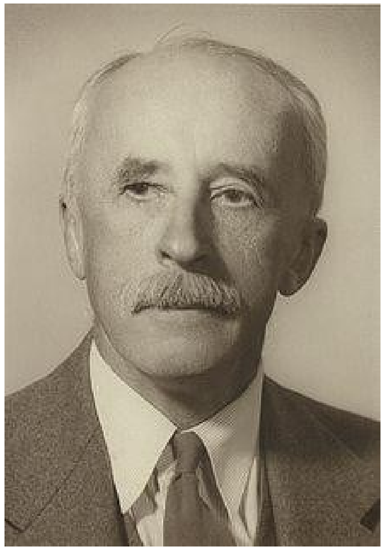


Closed-loop CNC

Outline

A Brief History of PID Control

- **1890's**, PID (**P**roportional - **I**ntegral – **D**erivative) Control, originally developed in the form of motor governors, which were manually adjusted
- **1922**, the first theory of PID Control was published by Nicolas Minorsky, who was working for the US Navy
- **1940's**, the first papers regarding PID tuning appeared
  - there are several hundred different rules for tuning PID controllers (See Dwyer, 2009)
- **Nowadays**, 97% of regulatory controllers utilize PID feedback
  - based on a survey of over eleven thousand controllers in the refining, chemicals and pulp and paper industries (see Desborough and Miller, 2002).


Nicolas Minorsky
(1885-1970)
a Russian American control theory mathematician, engineer and applied scientist

PID Control

Pros:

- Process independent
- The best controller where the specifics of the process can not be modeled
- Leads to a “reasonable” solution when tuned for most situations
- Inexpensive: Most of the modern controllers are PID
- Can be tuned without a great amount of experience required

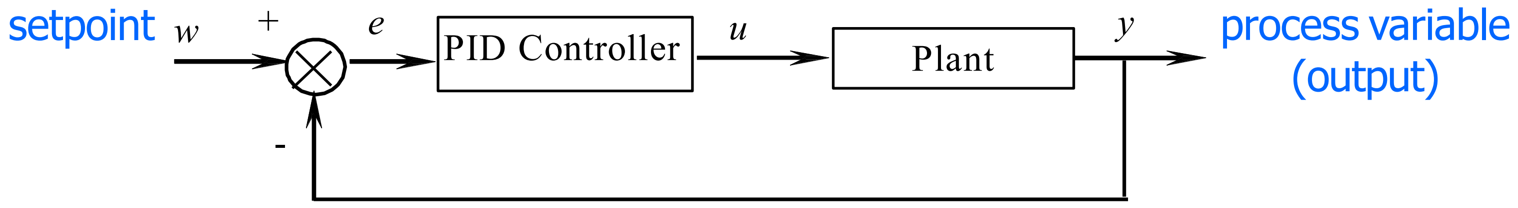
Cons:

- Not optimal for the problems
- Can be unstable unless tuned properly
- Not dependent on the process
- Hunting (oscillation about an operating point)
- Derivative noise amplification

Ways to Implement PID Control

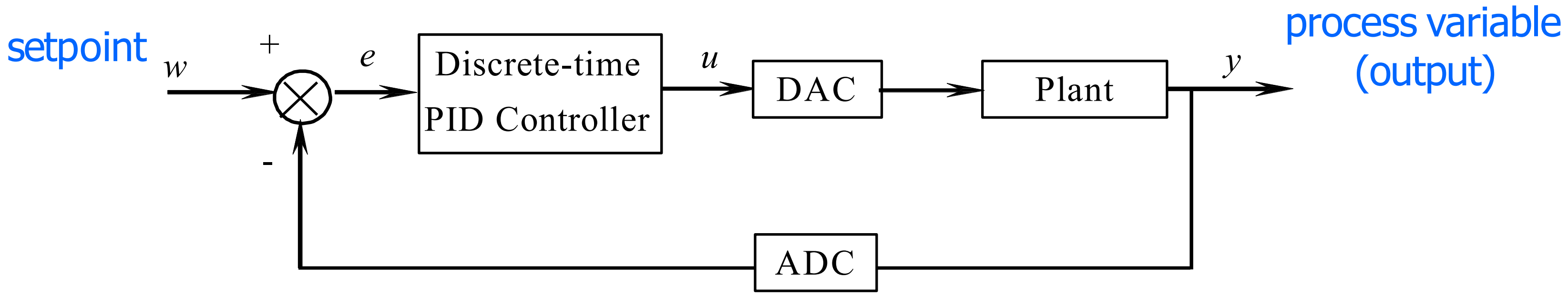
Analog PID:

- Receives a measured process variable ***y*(*t*)** using an electronic controller;
- Compares this value with that of a desired setpoint signal;
- Calculates an error value ***e*(*t*)** as the difference between the setpoint signal and process variable in a PID control circuit;
- The correction signal ***u*(t)** is then sent to the actuator to apply a correction.



Digital PID:

- Computer/Microcontroller aided;
- The computer registers the process variable ***y*(*t*)** via an AD converter, and produces a numerical value ***y*(*k*)**;
- Calculates an error value ***e*(*k*)** as the difference between the setpoint signal and process variable in a discrete-time PID control circuit;
- The correction signal ***u*(*k*)** is then sent to the DA converter producing ***u*(*t*)**, followed by the actuator to apply a correction.



Today's Lecture- PID/1

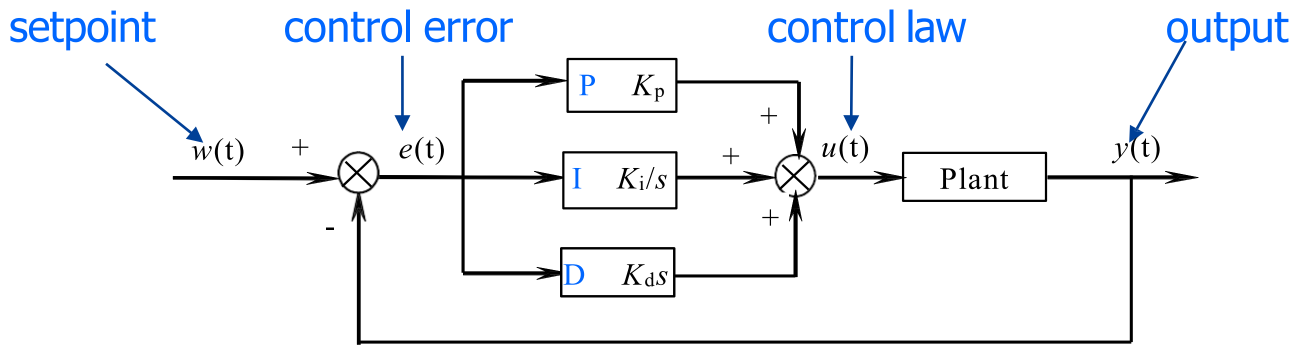
Today's Lecture- PID/1

2 Analog PID Controller

- PID control overview
- P control
- PI control
- PD control
- Simulation results
- Summary

Analog PID Controller

- **Block diagram of a PID controller**



- **Textbook form**

$$e(t) = w(t) - y(t)$$

$$u(t) = K_p e(t) + K_i \int_0^t e(s)\,\mathrm{d}s + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

$$= K_p \left(\underbrace{e(t)}_{\text{P}} + \underbrace{\frac{1}{T_i} \int_0^t e(s)\,\mathrm{d}s}_{\text{I}} + \underbrace{T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}}_{\text{D}} \right)$$

T_i : integration/reset time

T_d : derivative time

P Control

- **Proportional control (P):** accounts for present values of the error

$$u_\mathrm{p}(t) = K_\mathrm{p} e(t)$$

u_p — control signal

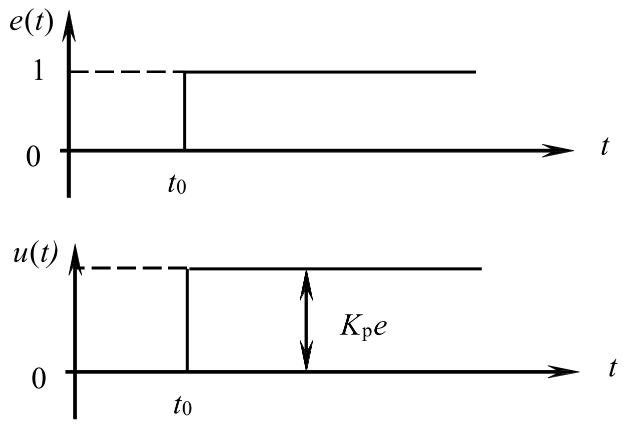
K_p — proportional gain

e — error signal

- In the Laplace domain

$$U_\mathrm{p}(s) = K_\mathrm{p} E(s)$$

- Pros&Cons
  - Rapid response to track the error signal
  - Steady-state error
  - Prone to be unstable for large K_p



Step response for P control

- Proportional control is always present, either by itself, or allied with derivative and/or integral control

- **Integral control (I):** accounts for past values of the error

$$u_{\mathrm{i}}(t) = K_{\mathrm{i}} \int_0^t e(s)\,\mathrm{d}s$$

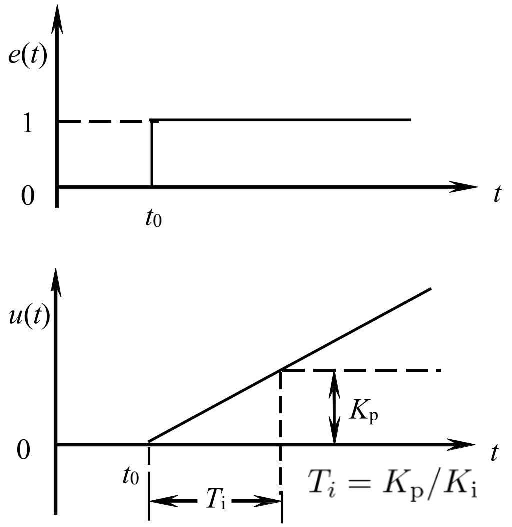
u_{i} — control signal

K_{i} — integral gain

e — error signal

- In the Laplace domain

$$U_{\mathrm{i}}(s) = \frac{K_{\mathrm{i}} E(s)}{s}$$



Step response for I control

- Pros&Cons
  - Eliminates the steady-state error that occurs with pure P control
  - Prone to cause the present value to overshoot the setpoint (responds to accumulated errors from the past)

PI Control

- The I control action is rarely used by itself, but is coupled with proportional (P) action for PI controller.
- **Proportional-Integral control (PI):** a combination of P and Icontrol

$$u_{\mathrm{pi}}(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(s)\,\mathrm{d}s$$
$$= K_{\mathrm{p}} \left(e(t) + \frac{1}{T_{\mathrm{i}}} \int_0^t e(s)\,\mathrm{d}s \right)$$

- In the Laplace domain

$$U_{\mathrm{pi}}(s) = K_{\mathrm{p}} \left(1 + \frac{1}{sT_{\mathrm{i}}} \right) E(s)$$

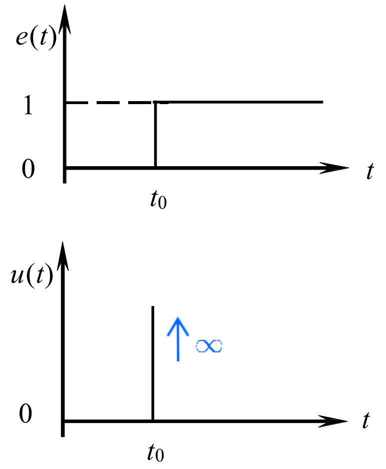
- **Derivative control (D):** accounts for possible future trends of the error

$$u_{\mathrm{d}}(t) = K_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

u_{d}	— control signal
K_{d}	— derivative gain
e	— error signal

- In the Laplace domain

$$U_{\mathrm{d}}(s) = sK_{\mathrm{d}}E(s)$$



Step response for D control

- Pros&Cons
  - Predicts system behavior and thus improves settling time/transient response and stability of the system
  - Helps reduce overshoot, but amplifies noise (derivative kick)
  - Seldom used in practice, 80% of the employed PID controllers have the D part switched-off (see Ang et al., 2005)

PD Control

- **Proportional-Derivative control (PD):** a combination of P and Icontrol

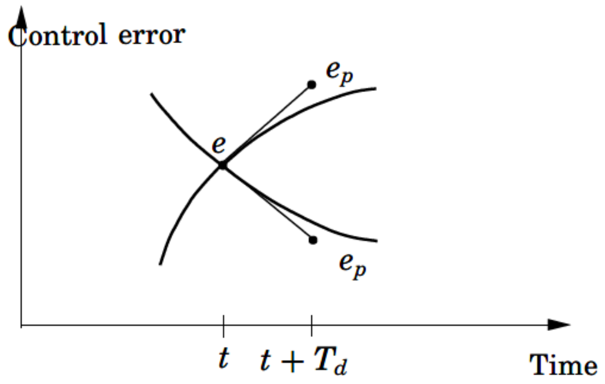
$$u_{\mathrm{pd}}(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{d}} \frac{\mathrm{d}e(s)}{\mathrm{d}s}$$
$$= K_{\mathrm{p}} \left(e(t) + T_{\mathrm{d}} \frac{\mathrm{d}e(s)}{\mathrm{d}s} \right)$$

- Take T_{d} as a step size, then

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t} \approx \frac{e(t+T_{\mathrm{d}}) - e(t)}{T_{\mathrm{d}}}$$

$$\Downarrow$$

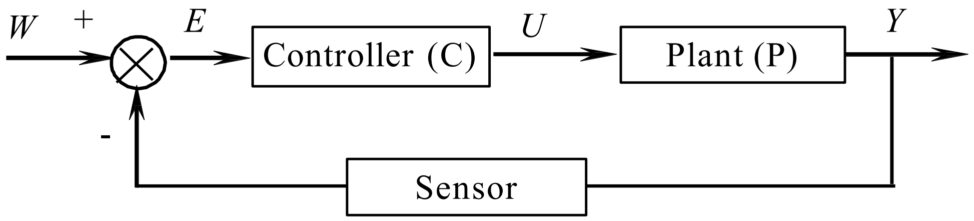
$$\boxed{u_{\mathrm{pd}}(t) \approx K_{\mathrm{p}} e(t+T_{\mathrm{d}})}$$



- D control action is able to predict system behavior and thus improving settling time/transient response.

Effect of P-I-D Control

- We will examine effect of PID control on a canonical 2nd order system to gain insight.



Setpoint: unit step signal

$$w = 1_{\geq 0} \Rightarrow W(s) = \frac{1}{s}$$

Plant: 2nd order system

$$P(s) = \frac{b}{s^2 + as + b}$$

Controller: P-I-D

$$C(s) = \begin{cases} K_\mathrm{p}, & \text{P control} \\ K_\mathrm{p} + K_\mathrm{i}/s, & \text{PI control} \\ K_\mathrm{p} + K_\mathrm{d}s, & \text{PD control} \end{cases}$$

Transfer function:

$$G(s) = \frac{Y(s)}{W(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)}$$

Error signal:

$$E(s) = W(s) - Y(s)$$

$$= \boxed{\left(1 - \frac{bC(s)}{s^2 + as + b(1 + C(s))}\right)\frac{1}{s}}$$

P Control

- <u>Effect on steady-state performance</u>

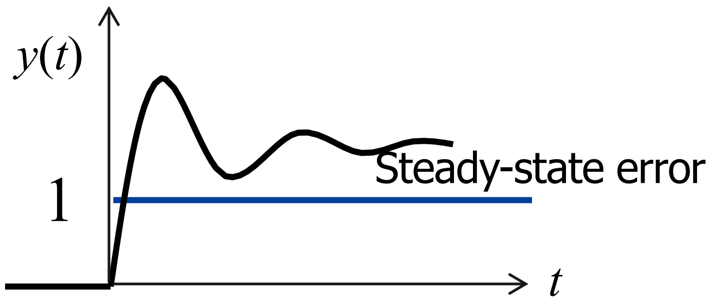
Steady-state error for a unit step reference

$$e(\infty) = \lim_{t\to\infty} e(t)$$

- Final-value theorem

$C(s) = K_p$, P control

$$\begin{aligned} e(\infty) &= \lim_{s\to 0} sE(s) \\ &= \lim_{s\to 0} s\left(1 - \frac{bK_p}{s^2 + as + b(1+K_p)}\right)\frac{1}{s} = \frac{1}{1+K_p} \neq 0 \end{aligned}$$



- Steady-state error always occurs;
- Larger K_p makes steady state error goes to zero

PI Control

- Effect on steady-state performance

  Steady-state value for a unit step reference

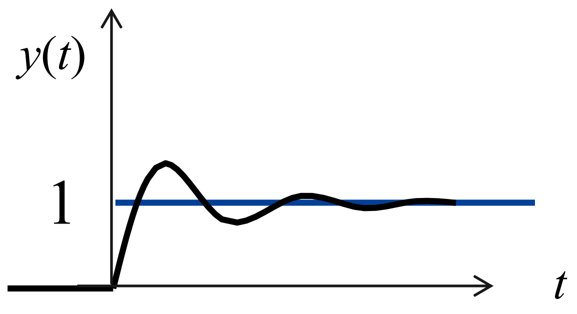
- Final-value theorem

$C(s) = K_\mathrm{p} + K_\mathrm{i}/s$, PI control

$$e(\infty) = \lim_{s\to 0} sE(s)$$

$$= \lim_{s\to 0} s\left(1 - \frac{b(K_\mathrm{p}s + K_\mathrm{i})}{s^3 + as^2 + b(1+K_\mathrm{p})s + bK_\mathrm{i}}\right)\frac{1}{s} = 0$$

- Steady-state error is zero for a step reference, even for small K_i (just takes longer to reach steady state).



PD Control

- <u>Effect on steady-state performance</u>

  Steady-state error for a unit step reference

- Final-value theorem

$$e(\infty) = \frac{1}{1+K_{\mathrm{p}}}, \text{ P control}$$

$$e(\infty) = \lim_{s\to 0} sE(s)$$
$$= \lim_{s\to 0} s\left(1 - \frac{b(K_{\mathrm{p}}+K_{\mathrm{d}}s)}{s^2+as+b(1+K_{\mathrm{p}}+K_{\mathrm{d}}s)}\right)\frac{1}{s} = \frac{1}{1+K_{\mathrm{p}}} \neq 0$$

- Steady-state error remains the same as the steady-state error with

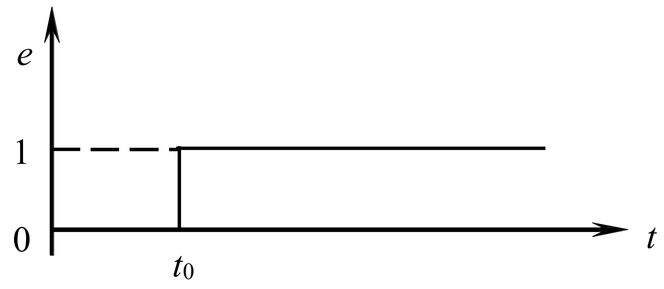
  pure P control. Indeed, D control does not track error, only the rate of change of it.

- No significant value added by including the D control with respect to the steady-state performance (transient performance probably differs).

PID Control

- **Proportional integral derivative control (PID):** a combination of P, I and D control

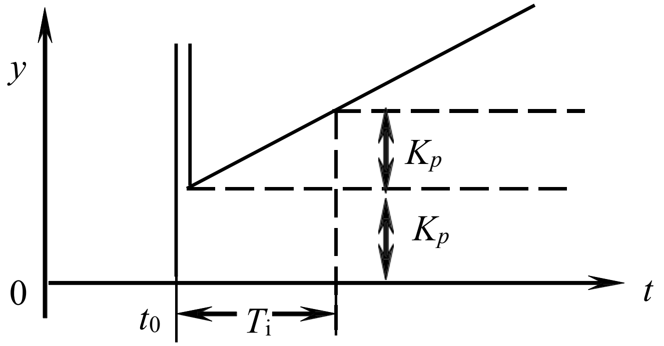
$$u(t) = K_p e(t) + K_i \int_0^t e(s)\,\mathrm{d}s + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$
$$= K_p \left(e(t) + \frac{1}{T_i}\int_0^t e(s)\,\mathrm{d}s + T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right)$$



- Effect on steady-state performance

$$e(\infty) = \lim_{s\to 0} sE(s) = 0$$

Therefore, steady-state error is zero for a step reference. It can be used to control the response characteristics better than the other types of controllers, e.g., P, PI, PD. Nevertheless, more complex to tune the parameters.



Step response for PID control

Simulation using MATLAB

- **A canonical 2nd order system**

Setpoint: unit step signal

$$w = 1_{\geq 0} \Rightarrow W(s) = \frac{1}{s}$$

Plant: 2nd order system

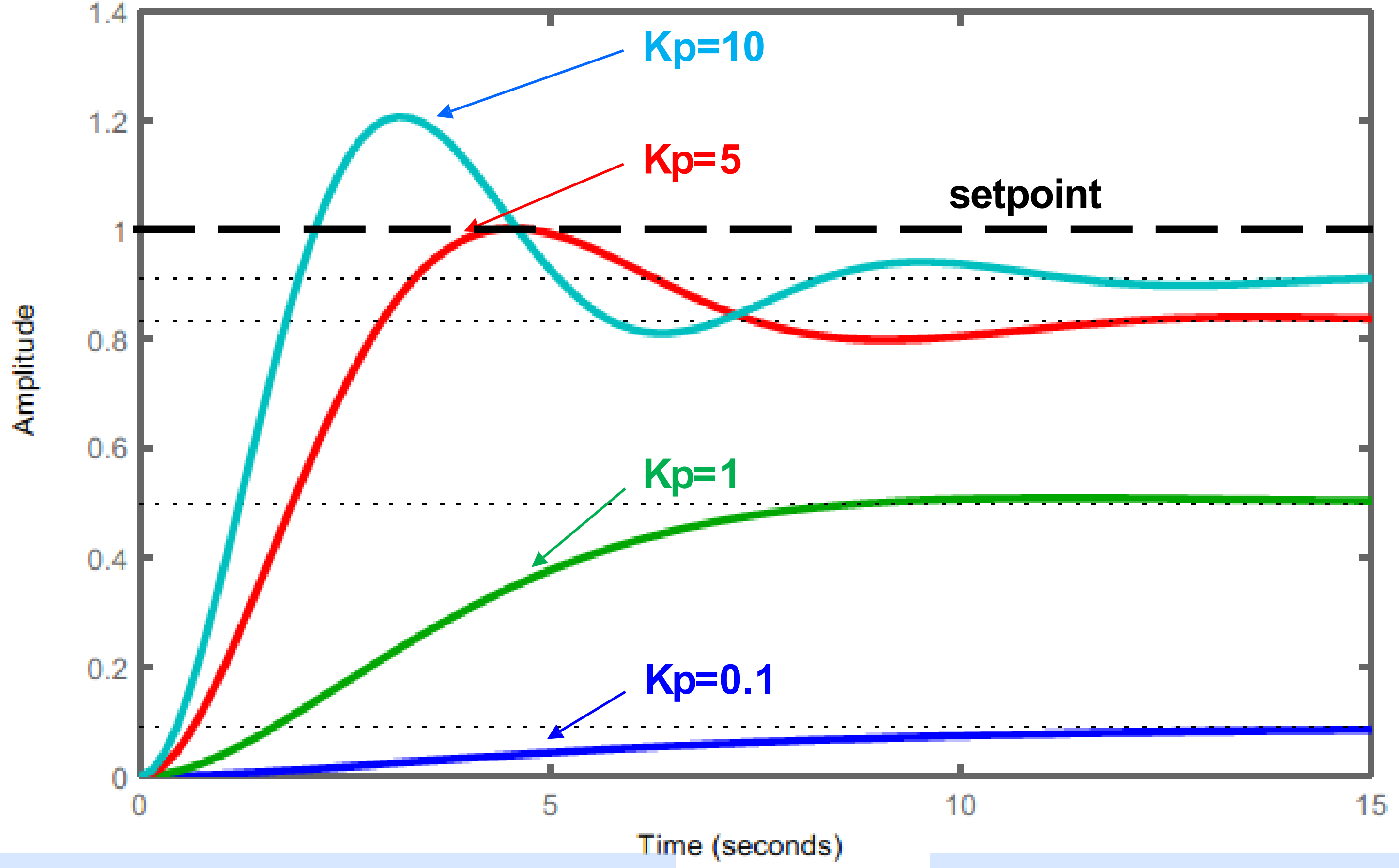
$$a = 0.7,\ \ b = 0.1 \Rightarrow P(s) = \frac{1}{10s^2 + 7s + 1}$$

Controller: PID

$$C(s) = K_\mathrm{p}\left(1 + \frac{1}{T_\mathrm{i}s} + T_\mathrm{d}s\right)$$

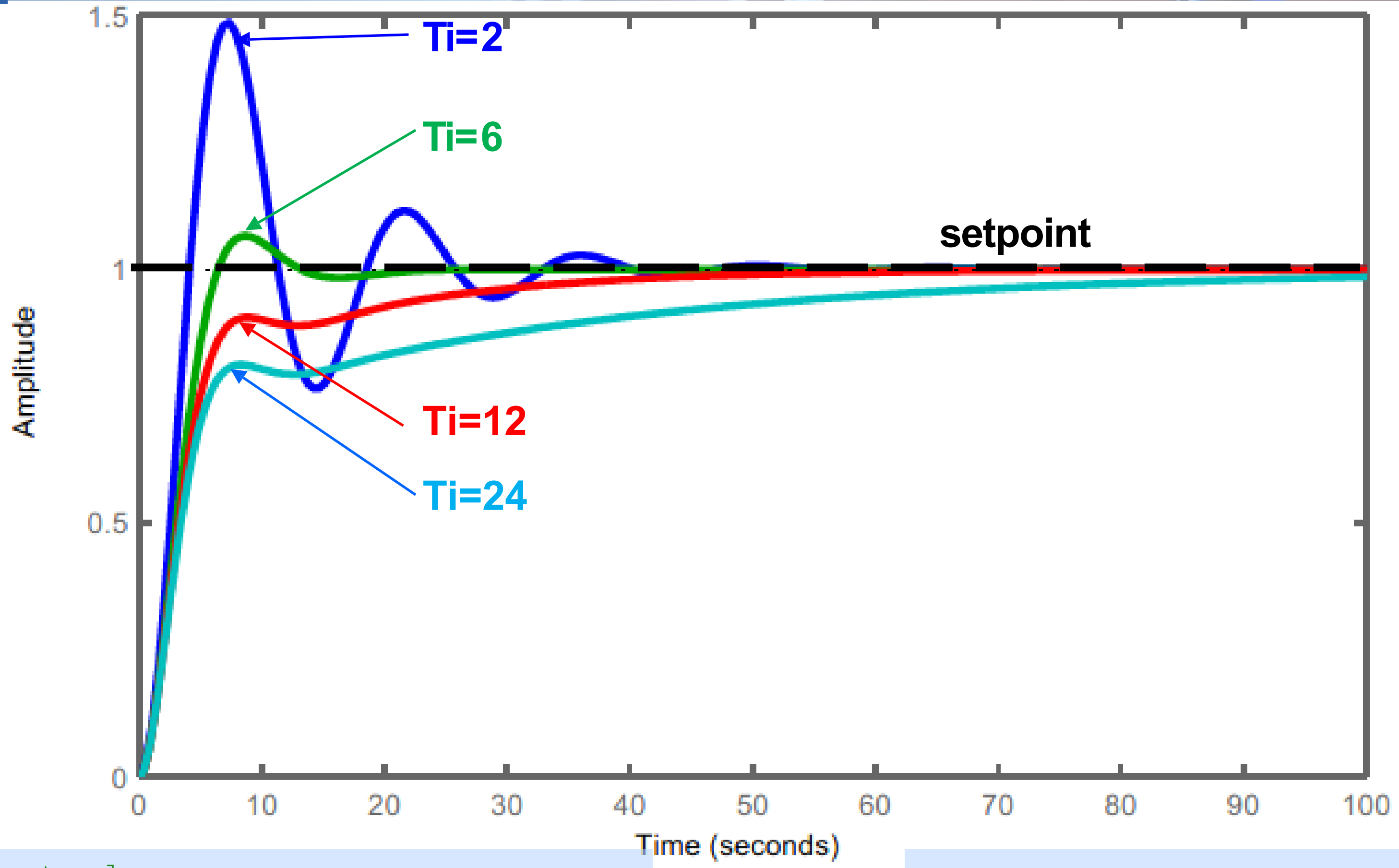
- **MATLAB code**

```
%plant
clc; clear all; close all;
Plt = tf(1,[10,7,1]); %transfer function
%PID control:
sys = feedback(C*Plt,1); %feedback
connection
step(sys); %unit step response
```



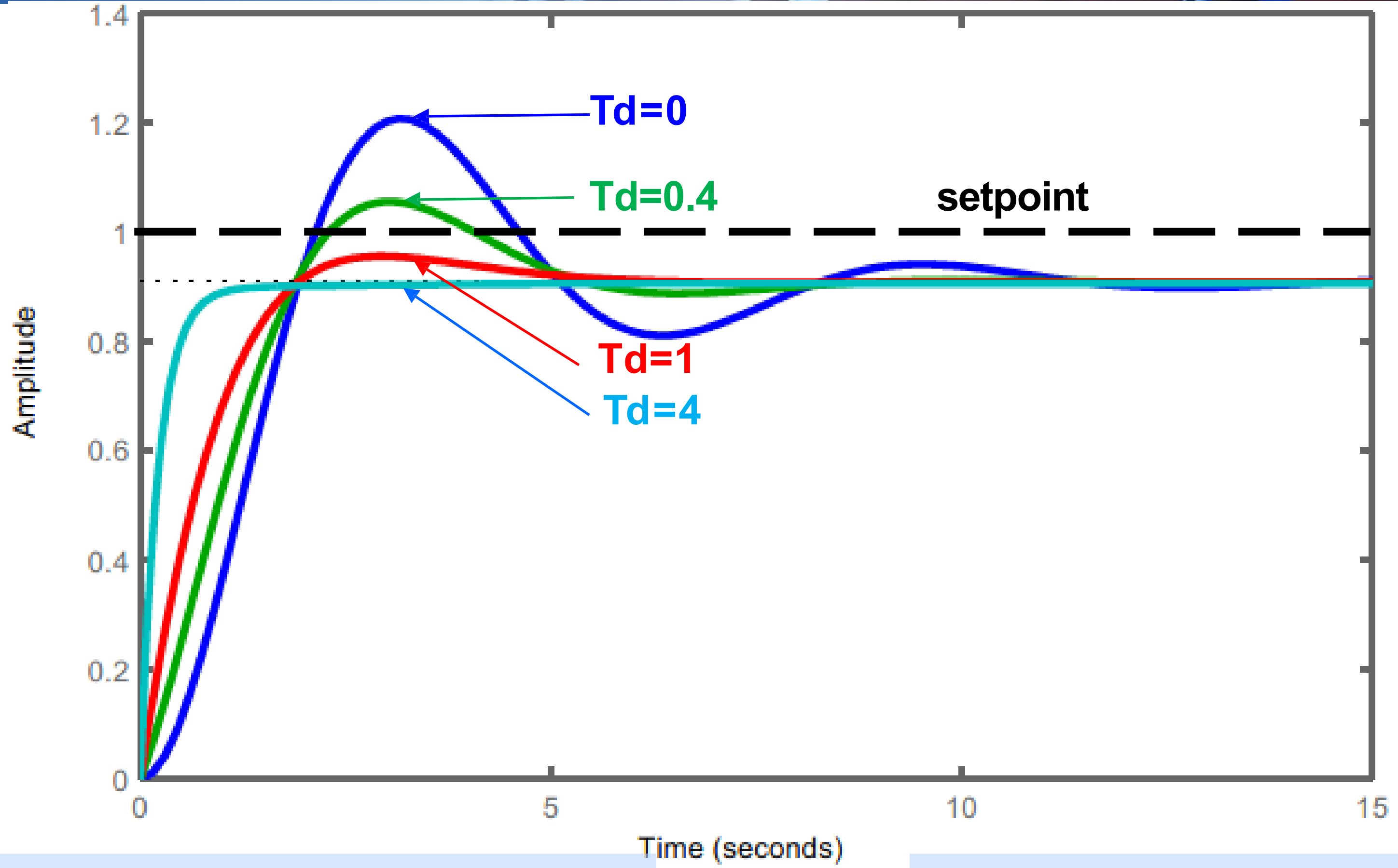
```
%P control
Plt = tf(1,[10,7,1]);
Kp = [0.1,1,5,10];
for k = 1:4
    sys = feedback(Kp(k)*Plt,1);
    step(sys),hold on
end
```

- ✓ Kp increases, the response speed of the system increases, the overshoot of the closed-loop system increases, and the steady-state error decreases.
- ✓ Kp large enough, the closed-loop system becomes unstable



```
%PI control:
Plt = tf(1,[10,7,1]);
Kp = 2; Ti = [2,6,12,24];
for m = 1:4
    Cpi = tf([Kp,Kp/Ti(m)],[1,0]);
    sys = feedback(Cpi*Plt,1);
    step(sys); hold on;
end
```

- ✓ No steady-state error in the step response
- ✓ Ti increases, the overshoot tends to be smaller, but the speed of response tends to be slower.

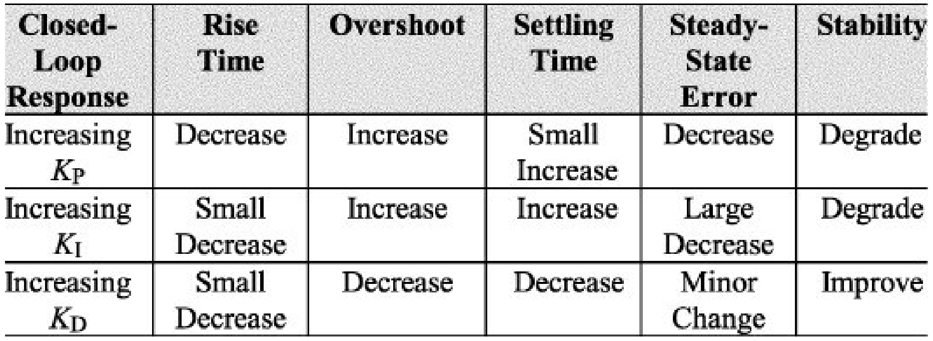


```
%PD control:
Plt = tf(1,[10,7,1]);
Kp = 10; Td = [0,0.4,1,4];
for m = 1:4
    Cpd = tf([Kp*Td(m),Kp],[0,1]);
    sys = feedback(Cpd*Plt,1);
    step(sys); hold on;
end
```

✓ Td increases, the response has a smaller overshoot with a slightly slower rise time but similar settling time

- **Some intuition about effects of the terms:**
  - **Increasing K_p:** Same amount of error generates a proportionally larger amount of control, makes system faster, but overshoot more (less stable)
  - **Increasing K_i:** Control effort builds as error is integrated over time, helps reduce steady state error, but can be slow to respond
  - **Increasing K_d:** Allows controller to anticipate an increase in error, adds damping to the system (reduces overshoot), can amplify noise

TABLE I
EFFECTS OF INDEPENDENT P, I, AND D TUNING



Closed-Loop Response	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Increasing K_P	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing K_I	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_D	Small Decrease	Decrease	Decrease	Minor Change	Improve

- These guidelines do not hold for all situations.
- For systems that are not canonical first or second order, need to use trial and error.

Practical Modifications of PID controllers

- **Textbook form**

$$
\begin{aligned}
e(t) &= w(t) - y(t) \\
u(t) &= K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(s)\,\mathrm{d}s + K_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t} \\
&= K_{\mathrm{p}} \left(\underset{\underset{\text{P}}{\uparrow}}{e(t)} + \underset{\underset{\text{I}}{\uparrow}}{\frac{1}{T_{\mathrm{i}}} \int_0^t e(s)\,\mathrm{d}s} + \underset{\underset{\text{D}}{\uparrow}}{T_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t}} \right)
\end{aligned}
$$

- Seldom used in practice because of a few problems arise leading to poor practical performance.
- Modifications:
  - P part: setpoint weighting
  - Ipart: anti-windup
  - D part: setpoint weighting and limited gain

Summary

- The controller performs the PID mathematical functions on the error and applies their sum to a process.
- We can build a PID controller that works well in practice in most situations without knowing control theory.

	Math Function	Effect on Control System
P Proportional	$K_\mathrm{p} e(t)$	Typically the main drive in a control loop, K_p reduces a large part of the overall error.
I Integral	$K_\mathrm{i} \int_0^t e(s)\,\mathrm{d}s$	Reduces the final error in a system. Summing even a small error over time produces a drive signal large enough to move the system toward a smaller error.
D Derivative	$K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$	Counteracts the K_p and K_i terms when the output changes quickly. This helps reduce overshoot and ringing. No effect on final error.

References

- Genke Yang, and Jianying Xie, **Micro-Computer Control Technology**, 4th ed. Changsha: National Defense Industry Press, 2016 (in Chinese).
- K. J. Åström, and T. Hägglund, **PID Controllers: Theory, Design, and Tuning**, 2nd ed. Research Triangle Park, NC: Instrument Society of America, 1995.
- A. O'Dwyer, **Handbook of PI and PID Controller Tuning Rules**, 3rd ed. London: Imperial College Press, 2009.
- L. Desborough, and R. Miller, **Increasing customer value of industrial control performance monitoring—Honeywell's experience.** AIChE Symposium Series, vol. 326, pp.158-186, 2002.
- K. H. Ang, G. Chong, and Y. Li, **PID control system analysis, design, and technology**, IEEE Transactions on Control Systems Technology, vol. 13, no. 4, pp.559-576, 2005.

Thanks for your attention!

E-mail: shyzhu@sjtu.edu.cn

Wechat ID: S14528707