Introduction to PID Controller

M. H. M. Ramli

What is PID Control?

Let’s take a step back… What is **control**?

Control is just making a dynamic process behave in the way we want

We need 3 things to do this:

- A way to **influence** the process
- A way to see how the process **behaves**
- A way to **define** how we want it to behave

Defining behavior

We usually specify a **value** we want some output of the system to have

- Usually called the **Setpoint (SP)**

Can be the temperature of a room, the level in a tank, the flow rate in a pipe…

The value can be fixed, or may change with time…

Influencing the process

We need some kind of **control input** which can create changes in the behavior of the process

Can be a heater, a valve, a pump…

Typically it is **not** the same physical quantity as what we are controlling

Observing behavior

If we knew exactly how the process worked, we would know what the output would be for a given control input…

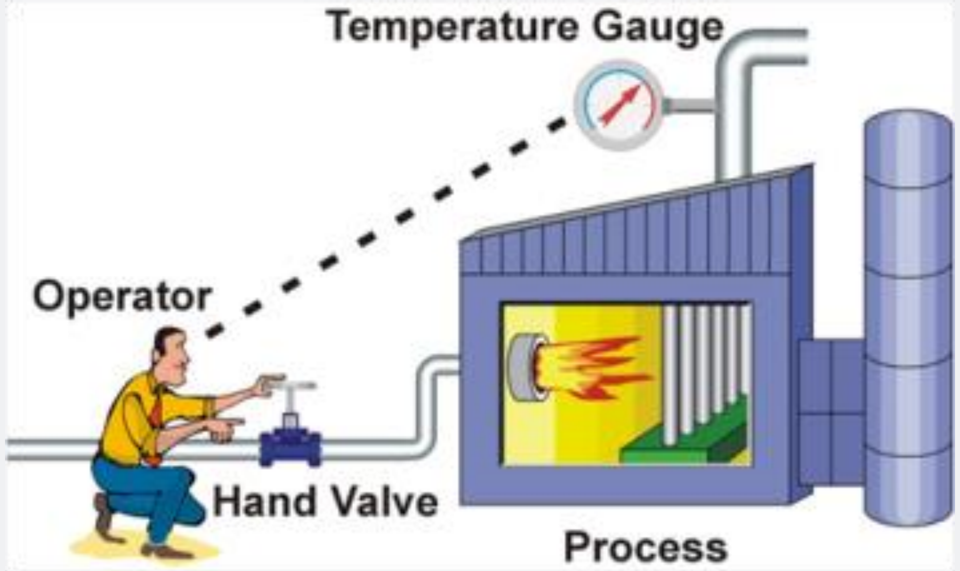
Most of the time we don’t know exactly, so we need to **measure** what the process does

- Usually called **Measured Variable (MV)** or **Process Variable (PV)**

Feedback Control

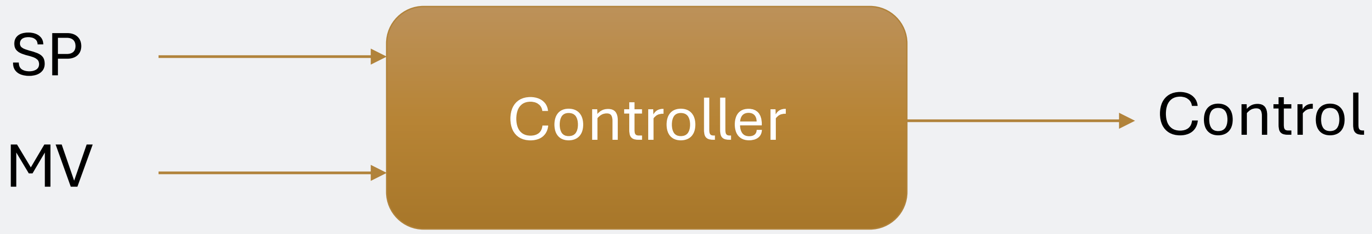
Now we have a measurement (**MV**), some value that we want it to be (**SP**), and some way to make changes to the process (**control input**)

We can ‘**close the loop**’



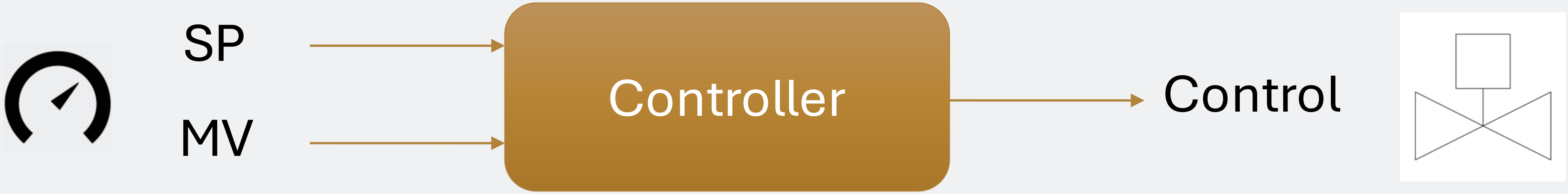
The Controller as a System

Now we can see that any controller can be thought of as a system that takes a **setpoint** and a **measured value** as inputs, and gives a **control** signal as an output



The Controller as a System

The controller needs to convert two signals of one physical quantity (such as temperature) into one signal of another (such as valve position)



The Controller as a System

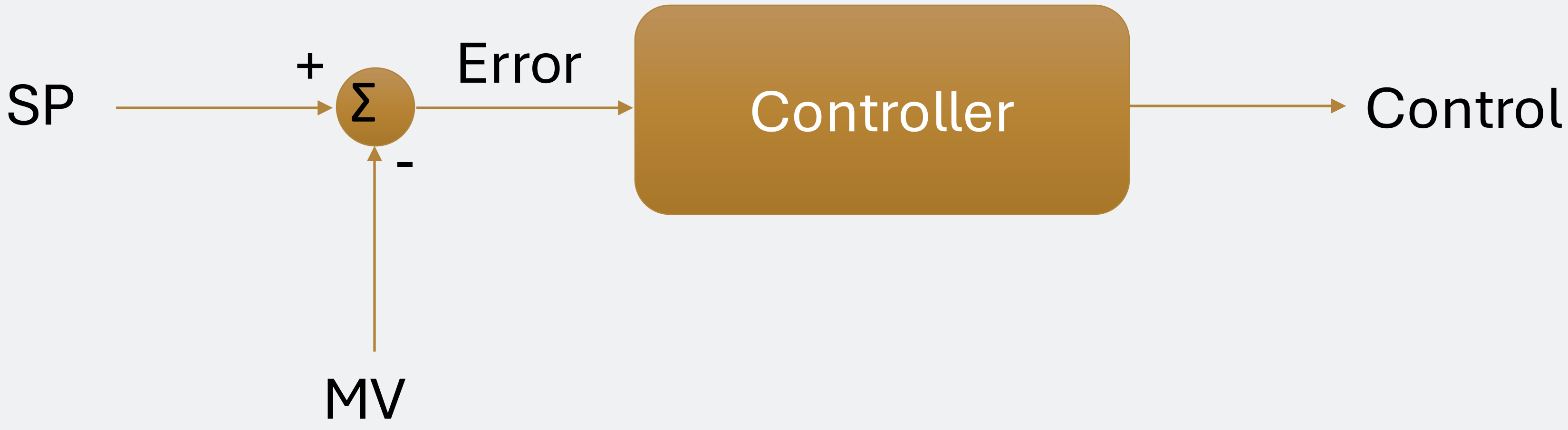
We know that the process is a **dynamic** system:

- Its outputs depend on **current** inputs as well as its **past** state

For the controller to deal with this, it makes sense that it should be a **dynamic** system too

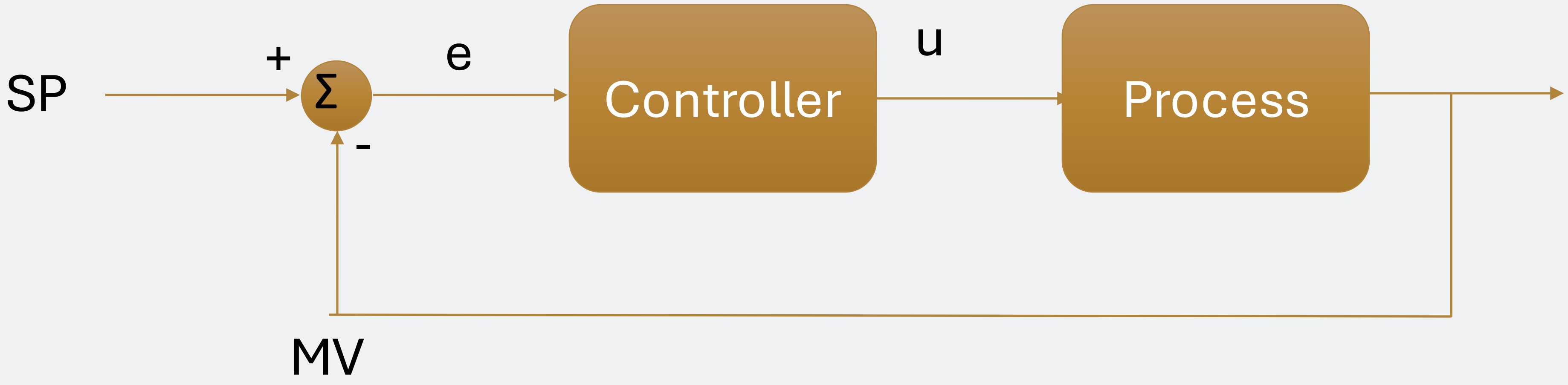
The Error Signal

Very often we can think of the controller acting on the **difference** between SP and MV:



The Closed Loop

This is the ‘classic’ closed loop block diagram representation of a control system



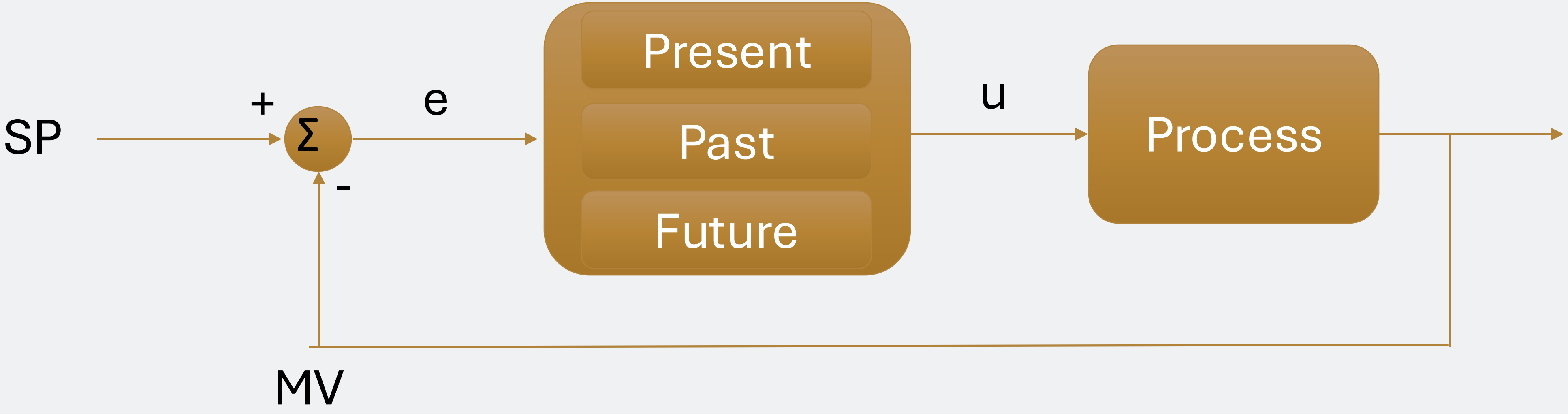
A Dynamic Controller

We said that since the process is dynamic (dependent on inputs made at different times), it makes sense that the controller should be too

How do we usually think of time?

- ‘Present’
- ‘Past’
- ‘Future’

Splitting the Controller



The ‘Present’

This part of the controller is only concerned with what the **error** is **now**

Let’s take a simple law: let the control signal be **proportional** to the error:

$$\boldsymbol{u = k_p \times e}$$

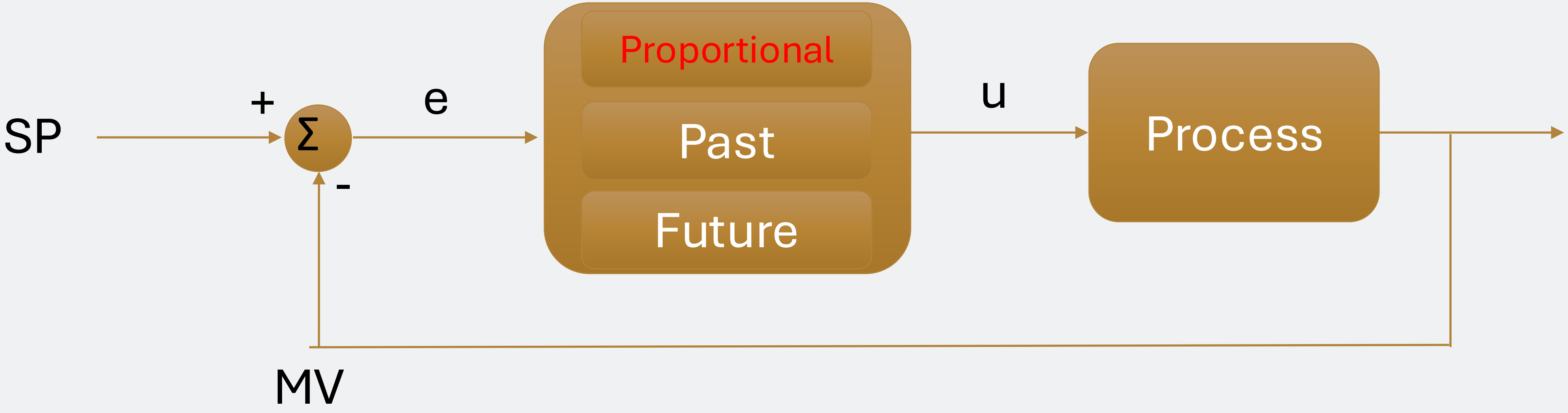
Proportional Control

This is what is referred to as **proportional control.** The control action at any instant is the same as a constant times the error at the same instant

The constant k_p is the **Proportional Gain,** and is the first of our controller's parameters

Proportional Control

$$u = k_p \times e$$

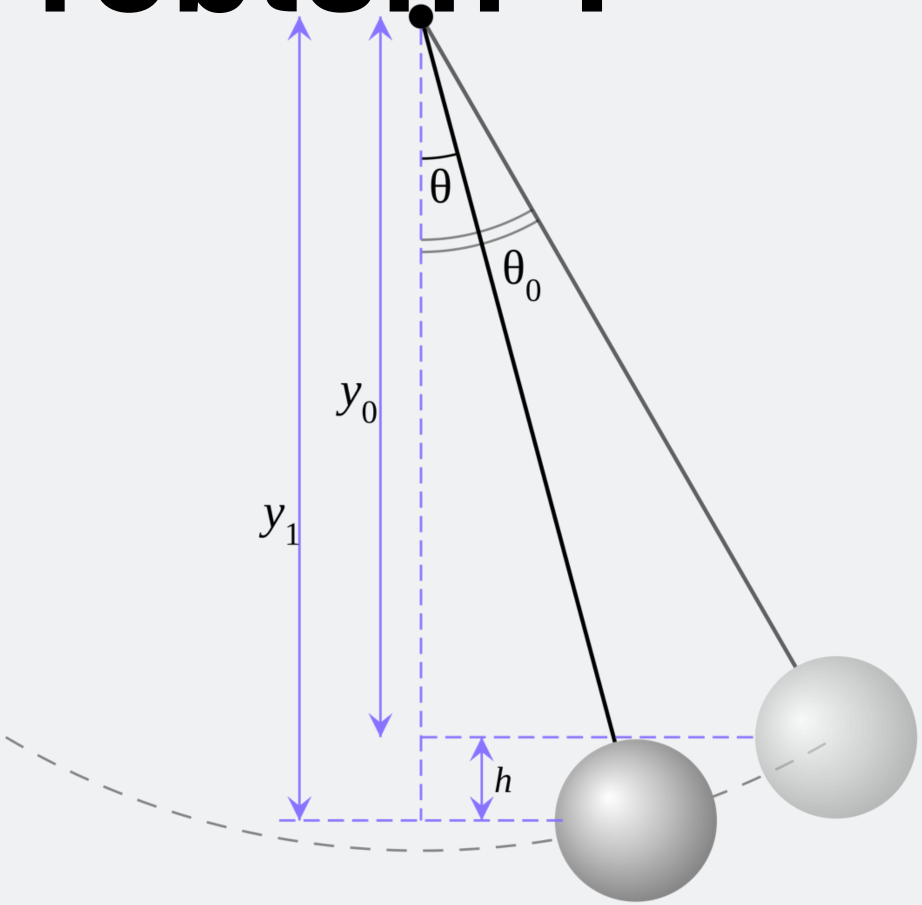


Is Proportional Control enough?

Intuitively it seems like it should be fine on its own: when the error is big, the control input is big to correct it. As the error reduced so does the control input.

But there are problems…

Problem 1

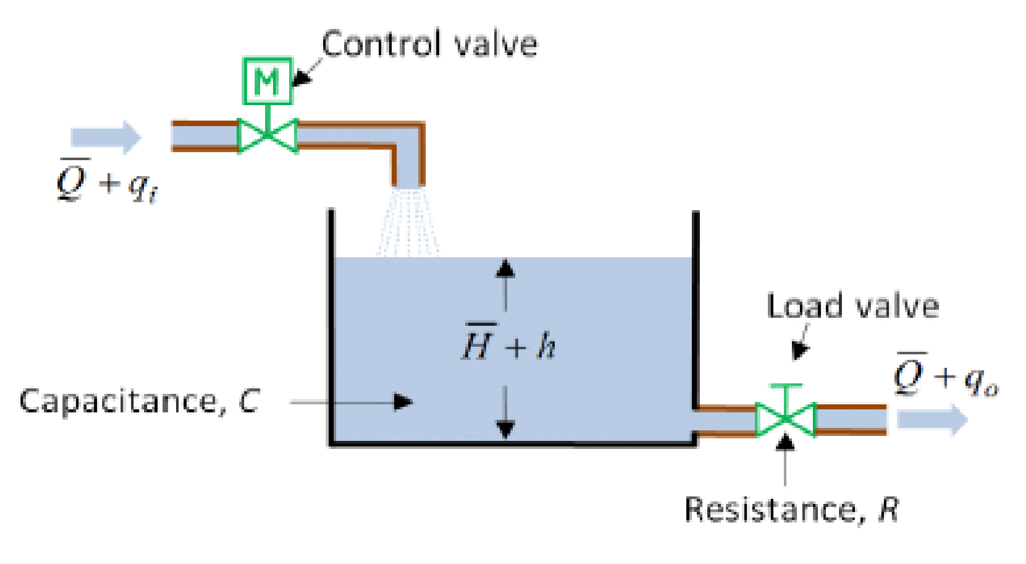


Think of a pendulum.

If the setpoint is hanging straight down, then gravity acts as a proportional controller for the position…

Pendulum will **oscillate**!

Problem 2



What happens when the error is zero?

Control input is zero.

Causes problems if we need to have a nonzero control value while at our setpoint

Problem 2: steady state error

This problem is normally called **steady state error**

It’s a confusing name. The issue is just that the controller can’t produce any output when the error is zero.

Easiest to see for tank level control: If there is a constant flow out of the tank, the controller must provide the same flow in, while the level is at the setpoint.

Problem 2: steady state error

With P control, once the error has reached a value where $k_p \times e$ is equal to the flow out, the level will stabilize. But it will be different to the setpoint

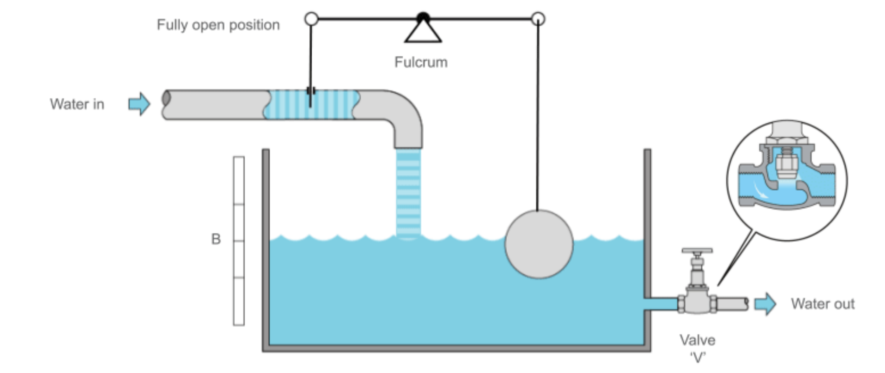


Fig. 5.2.6 Valve open

P control problem summary

Problem 1: oscillations

- P control will give us oscillations in some processes, regardless of the value of the gain parameter.

Problem 2: steady state error

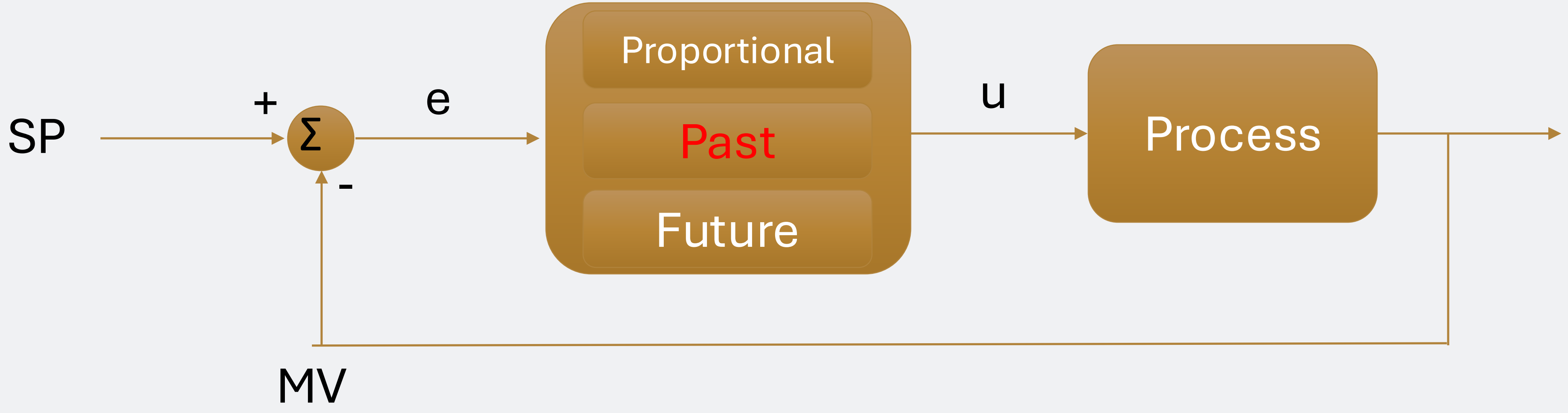
- P control cannot give us a nonzero value of the control at zero error for some types of process

Solving P control’s problems

How to get rid of steady state error?

Let’s ignore the present for the moment and concentrate on what has happened in the **past**

Solving steady state error: ‘the Past’

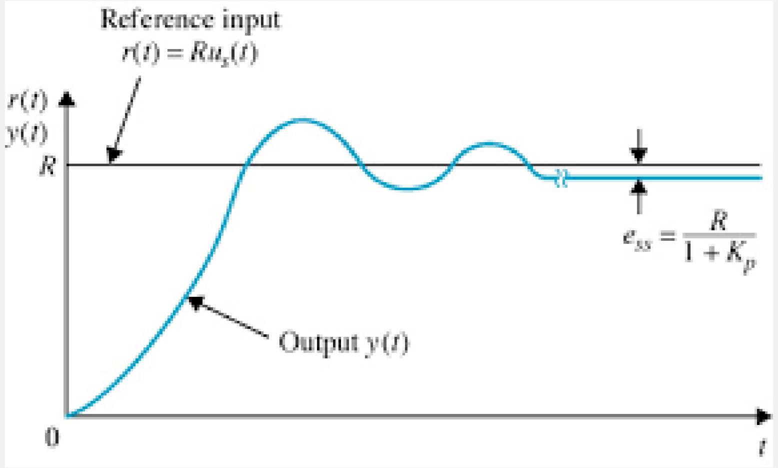


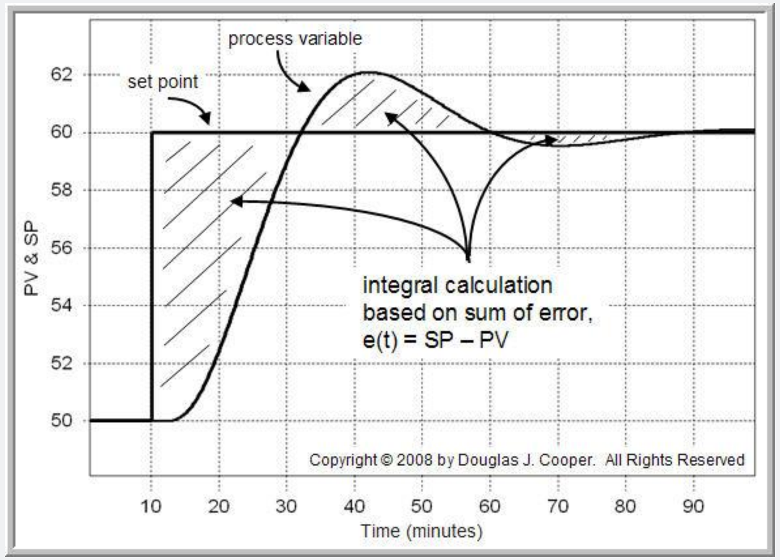
Let’s look at the error in the **past**

Solving steady state error

We can examine how the controller error has evolved in the **past**

If we sum up the past values of the error, we can get a value that increases when there is a constant error





Copyright © 2008 by Douglas J. Cooper. All Rights Reserved

Integral Action

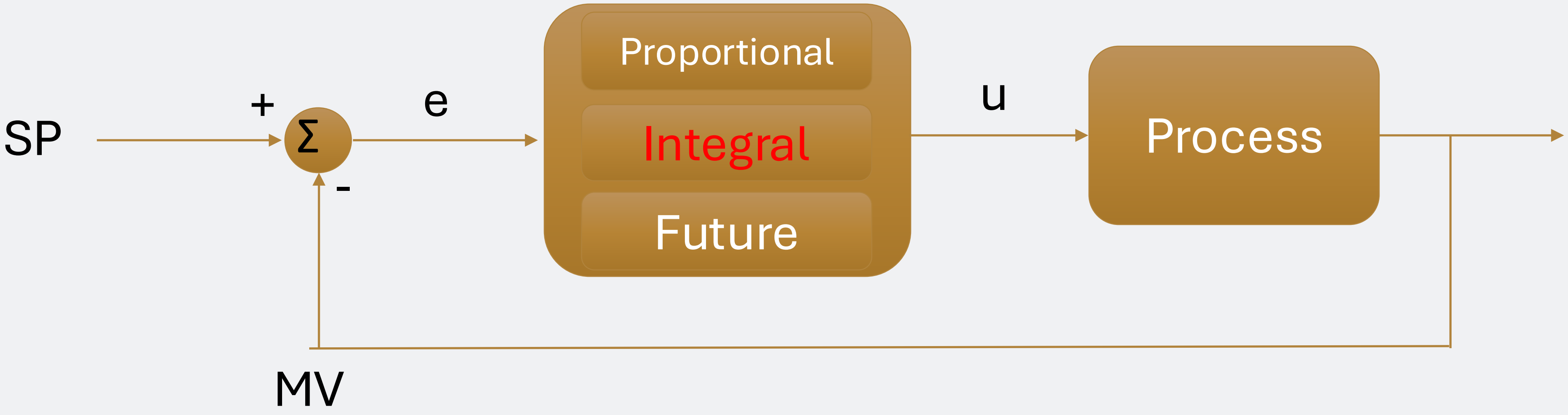
We can let the control be given by the sum of past values of the error, scaled by some gain.

In continuous time the sum is an **integral**:

$$\boldsymbol{u = k_i \int_0^t e(\tau) d\tau}$$

Integral Action

$$u = k_i \int_0^t e(\tau) d\tau$$



Integral gain, Integral time

Here is where confusion can start…

We have an **integral gain** $\boldsymbol{k_i}$ which converts an integrated error to a control signal

We would actually like to parameterize this as a **time**, as in how fast we can remove a steady state error

Integral gain, Integral time

Let’s rewrite:

$$\boldsymbol{u = \frac{k_p}{T_i}\int_0^t e(\tau)d\tau}$$

As:

is our Proportional Gain, and $\boldsymbol{T_i}$ is our **Integral Time**

The Integral time

Why do we use both $\boldsymbol{k_p}$ and $\boldsymbol{T_i}$ here?

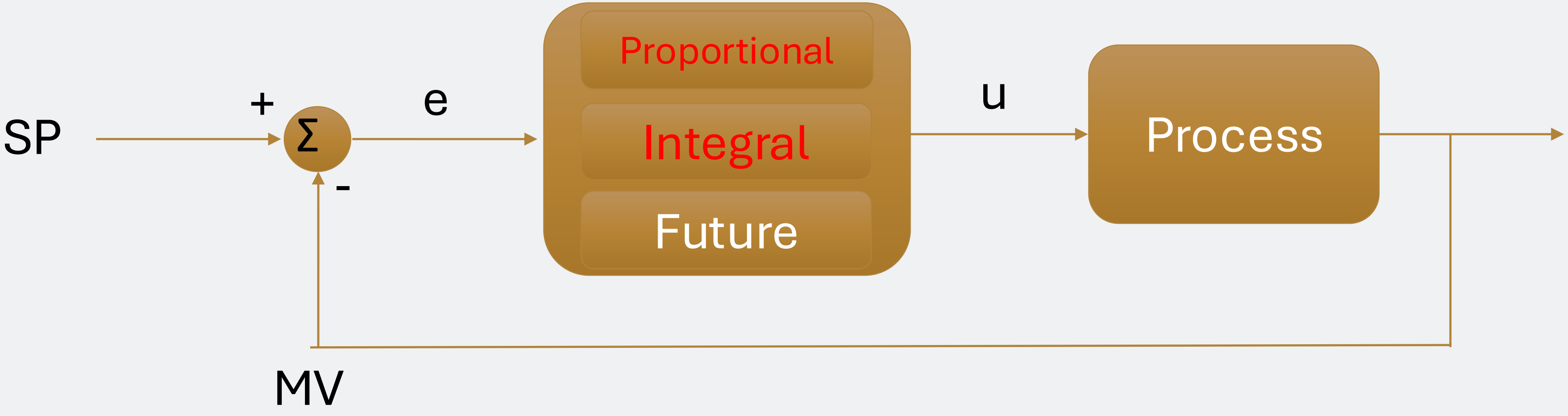
$$\boldsymbol{u = \frac{k_p}{T_i}\int_0^t e(\tau)d\tau}$$

Let’s add the proportional and integral parts:

$$\boldsymbol{u = k_p e(t) + \frac{k_p}{T_i}\int_0^t e(\tau)d\tau}$$

Proportional and Integral Controller

$$u = k_p e(t) + \frac{k_p}{T_i}\int_0^t e(\tau)d\tau$$



Proportional and Integral Controller = PI Controller

We can rewrite the control law:

$$\boldsymbol{u = k_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau\right)}$$

$\boldsymbol{k_p}$ is the gain parameter, $\boldsymbol{T_i}$ is the time it takes to fix a steady state error

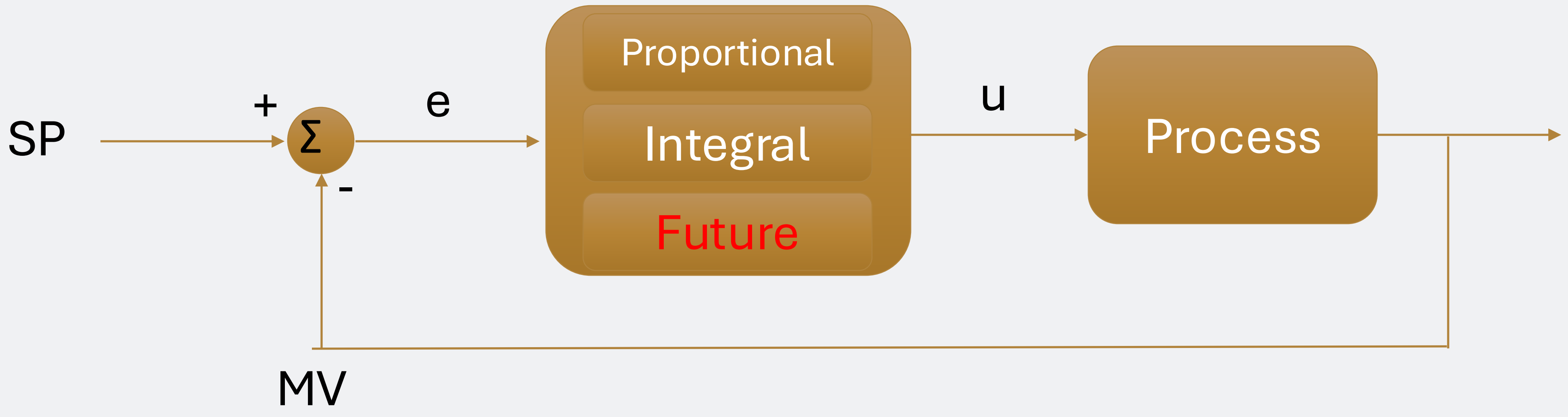
Solving P control’s problems revisited

We solved the steady state error by adding **integral action** (summing the past)

How can we solve the oscillation problem?

Let’s look at the future!

Solving oscillations: ‘the Future’



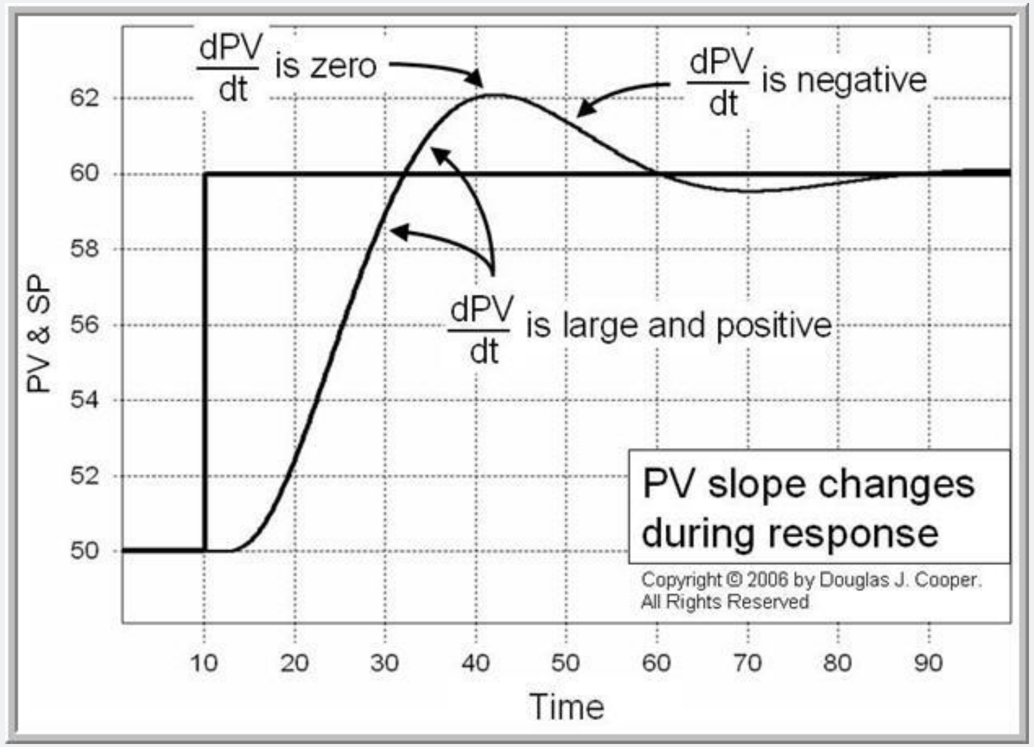
Let’s look at the error in the **future**

Solving oscillations: ‘the Future’

How do we predict the future of the error?

Look at its **gradient**!

If the gradient (the time derivative) of the error is in a direction that makes the error smaller, we can reduce the control input

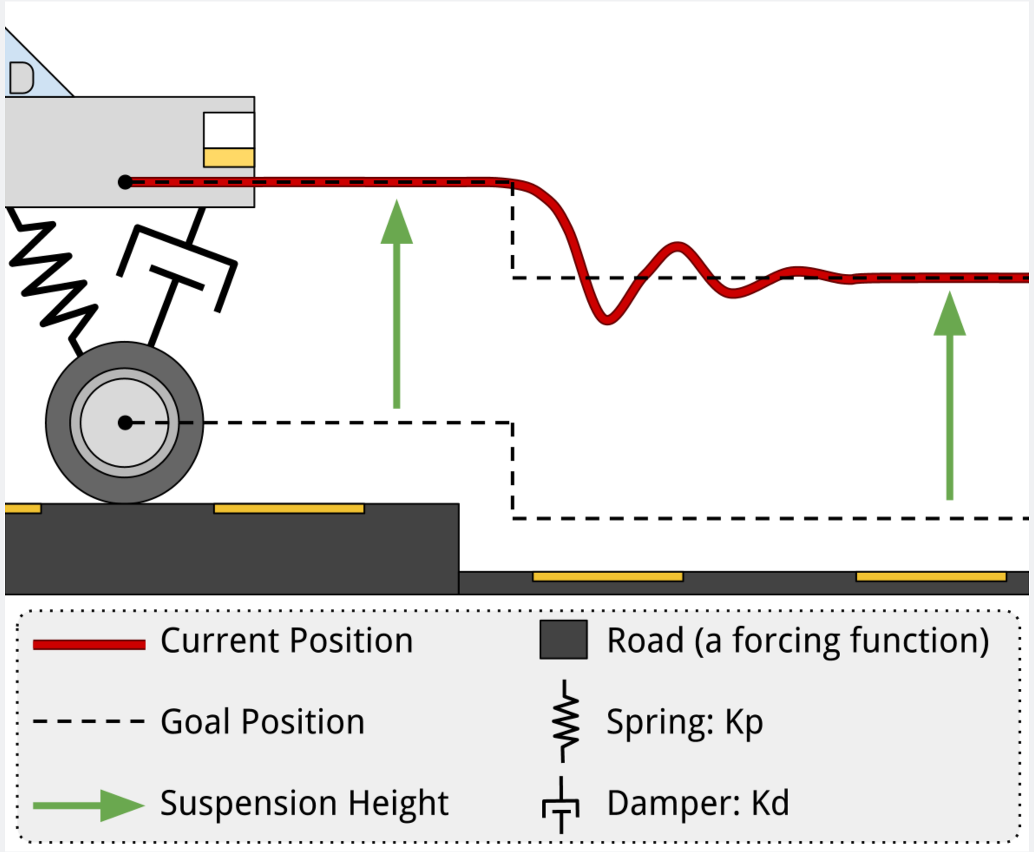


Copyright © 2006 by Douglas J. Cooper. All Rights Reserved

Damping

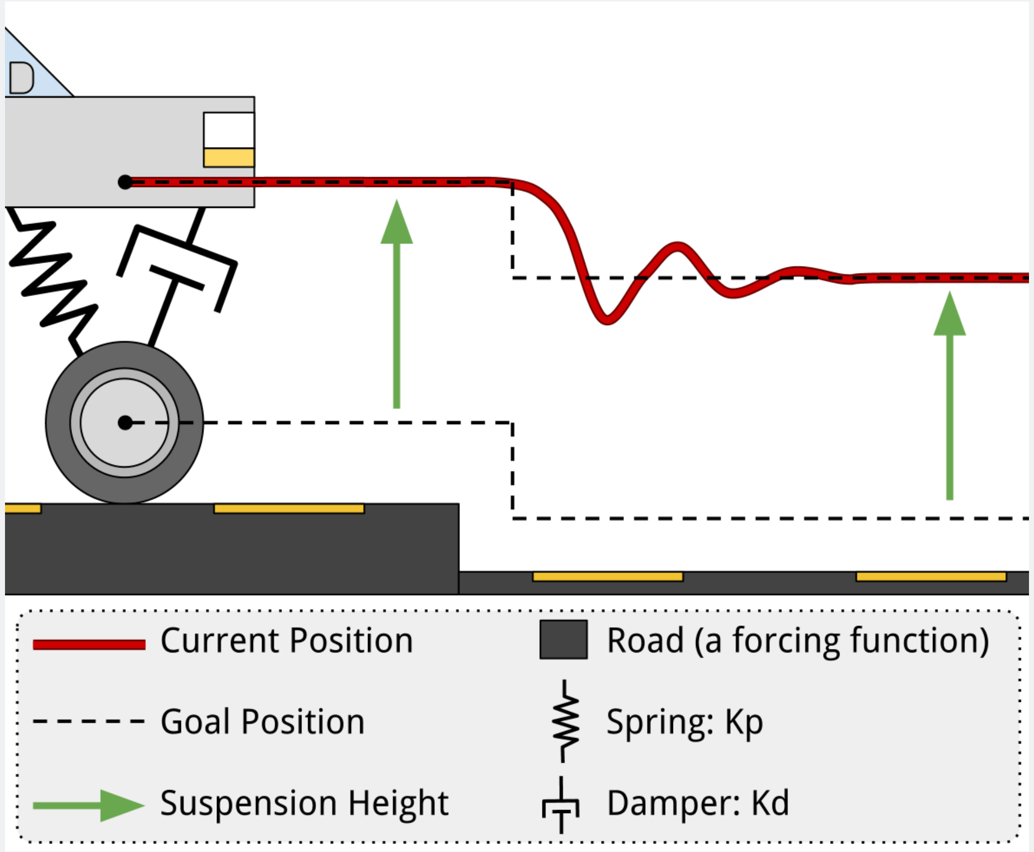
It can be easier to think of this as **damping,** something that resists velocity

Think of the wheel on your car…



Damping

The spring is a proportional controller for the wheel position. The damper adds a **derivative action** by opposing the **velocity** of the wheel



Derivative action

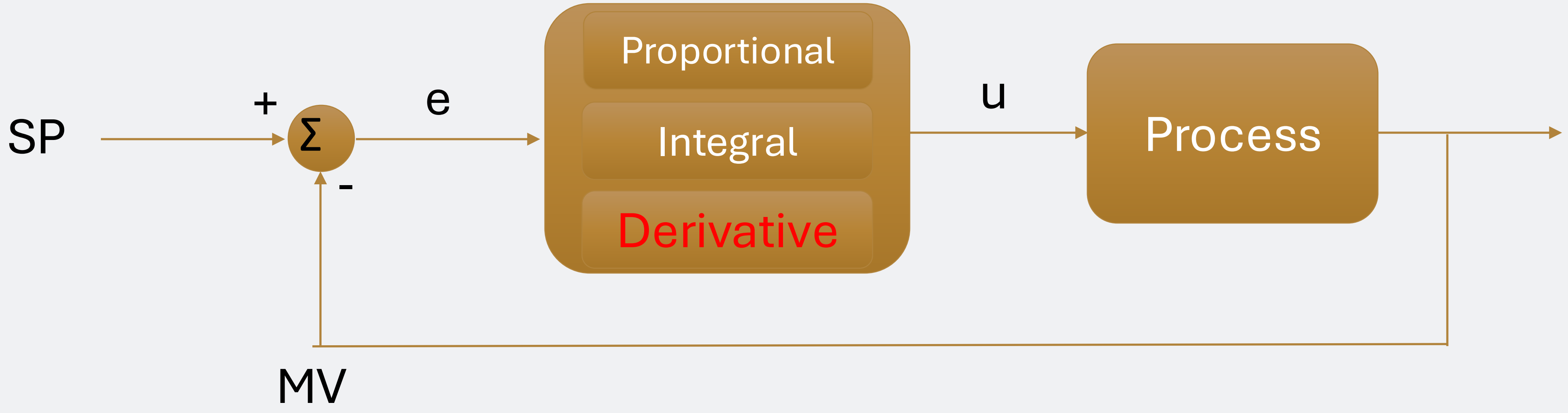
Let the control be dependent on the derivative of the error:

$$\boldsymbol{u = k_p T_d \frac{de(t)}{dt}}$$

Here k_d is the derivative gain. Let's again split this into k_p, T_d, where T_d is the **derivative time**

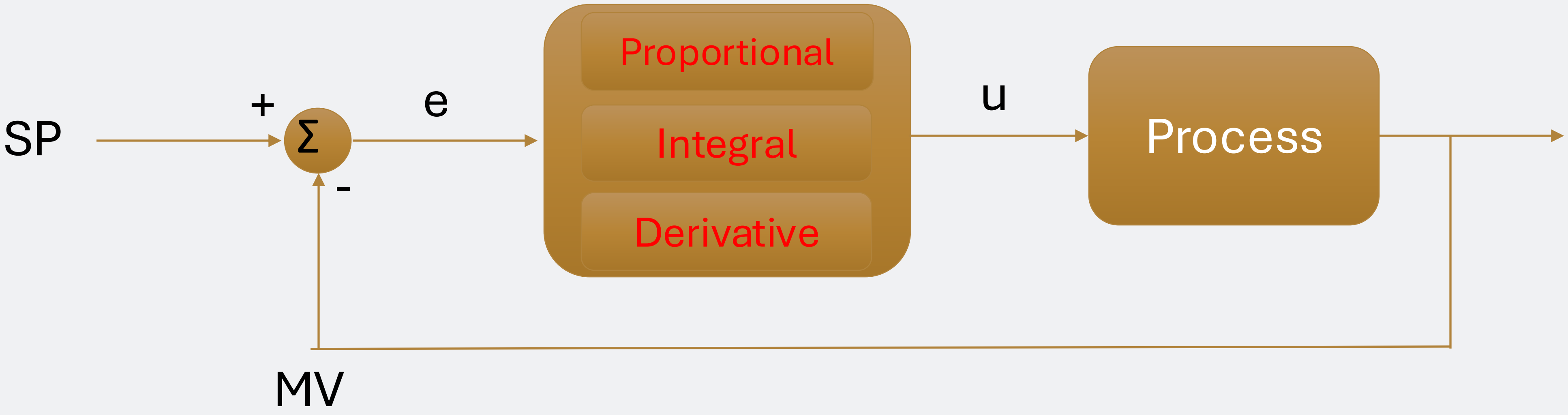
Derivative action

$$u = k_p T_d \frac{de(t)}{dt}$$



PID Controller!

$$u = k_p e(t) + \frac{k_p}{T_i}\int_0^t e(\tau)d\tau + k_p T_d \frac{de(t)}{dt}$$

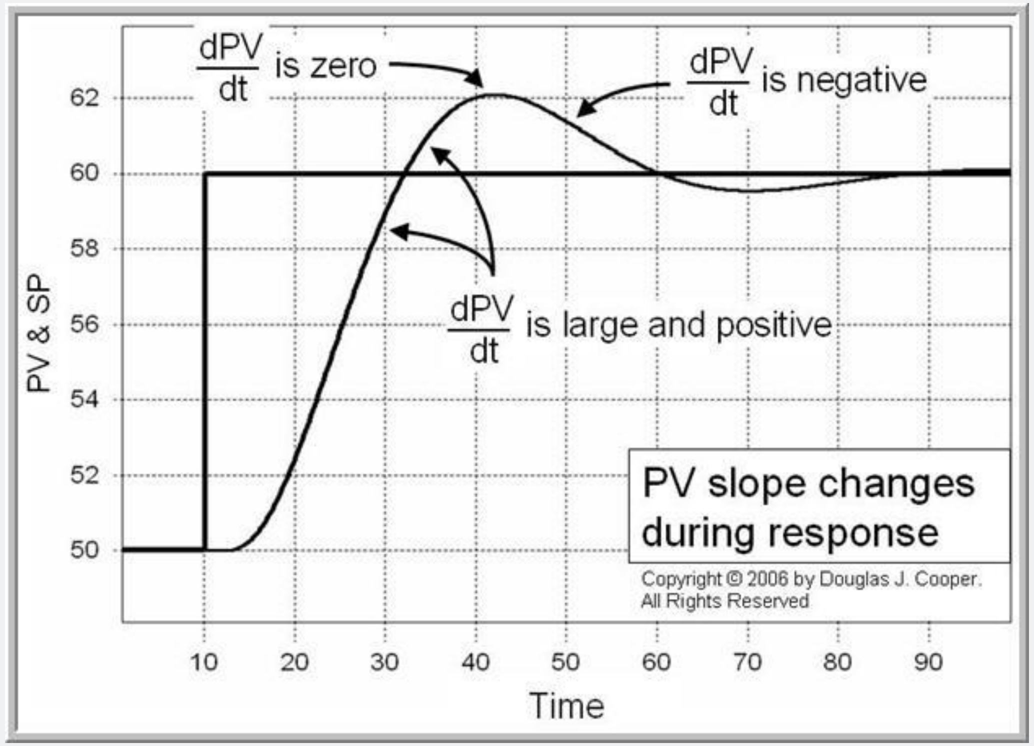


Derivative Time

Why do we want T_d as a parameter?

We can think of it as how far ahead we want to predict!

Easier to relate to process

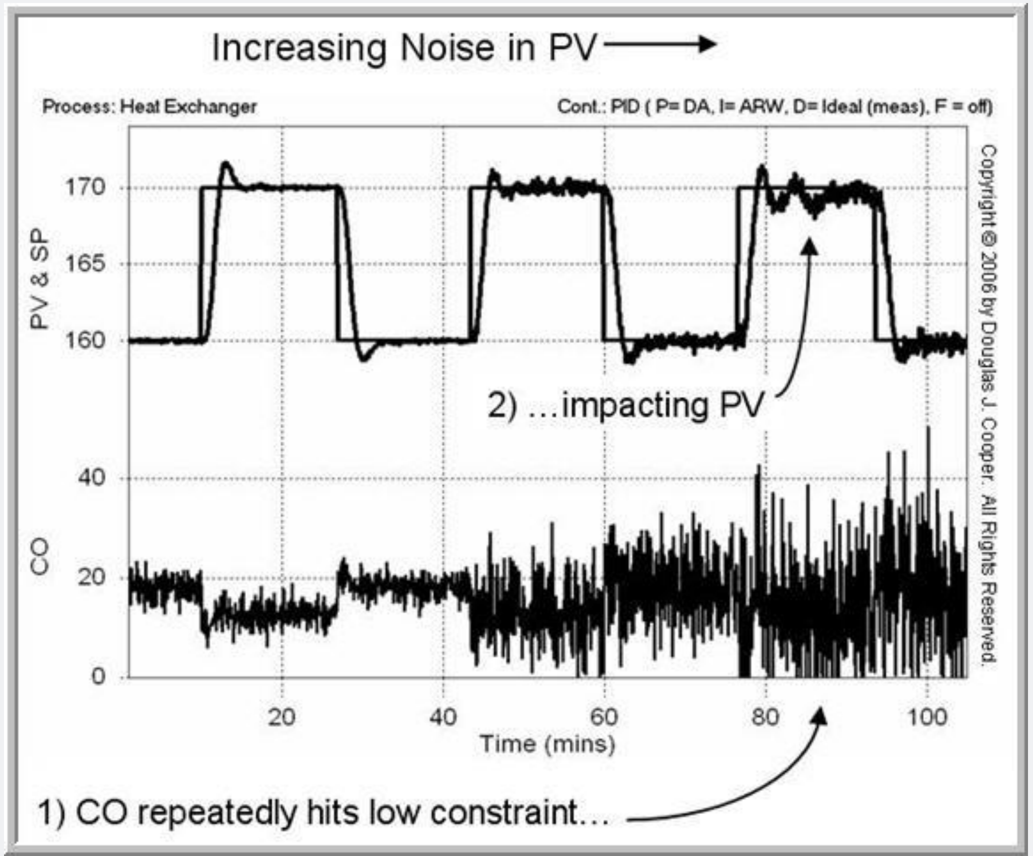


Problems with Derivative action

We know that a derivative amplifies quick changes.

We can get problems if MV is noisy.

Solution is to add low pass filter



Derivative with filter

We already have:

$$\boldsymbol{u = k_p T_d \frac{de(t)}{dt}}$$

Equations will get messy if we add a filter in time domain! Let’s use Laplace! Then we can use algebra instead of calculus.

Derivative with filter

Laplace transform frequency variable *s* is also an operator. Multiplication by *s* is derivation, and division is integration. So, the derivative part is now

$$U(s) = k_p T_d s E(s)$$

Derivative with filter

We add a low pass filter:

$$U(s) = \frac{k_p T_d s}{1 + \frac{T_d}{T_{ds}} s} E(s)$$

Now we have another parameter T_{ds}which is the filter bandwidth. So introducing derivative action requires two more parameters!

Full PID equation

The complete PID controller now looks like:

$$U(s) = k_p E(s) + \frac{k_p E(s)}{T_i s} + \frac{k_p T_d s}{1 + \frac{T_d}{T_{ds}} s} E(s)$$

Full PID equation

This simplifies to:

$$U(s) = k_p\left(1 + \frac{1}{T_i s} + \frac{T_d s}{1+\frac{T_d}{T_{ds}} s}\right) E(s)$$

ISA PID Form

This is the ISA ‘standard form’ for a PID

$$\frac{U(s)}{E(s)} = k_p\left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{T_{ds}} s}\right)$$

We have one gain, and three time constants

Introduction to PID Tuning: Ziegler-Nichols

Tuning: process of selecting the controller parameters to meet desired response

Analytical approach is not suitable for complicated plant

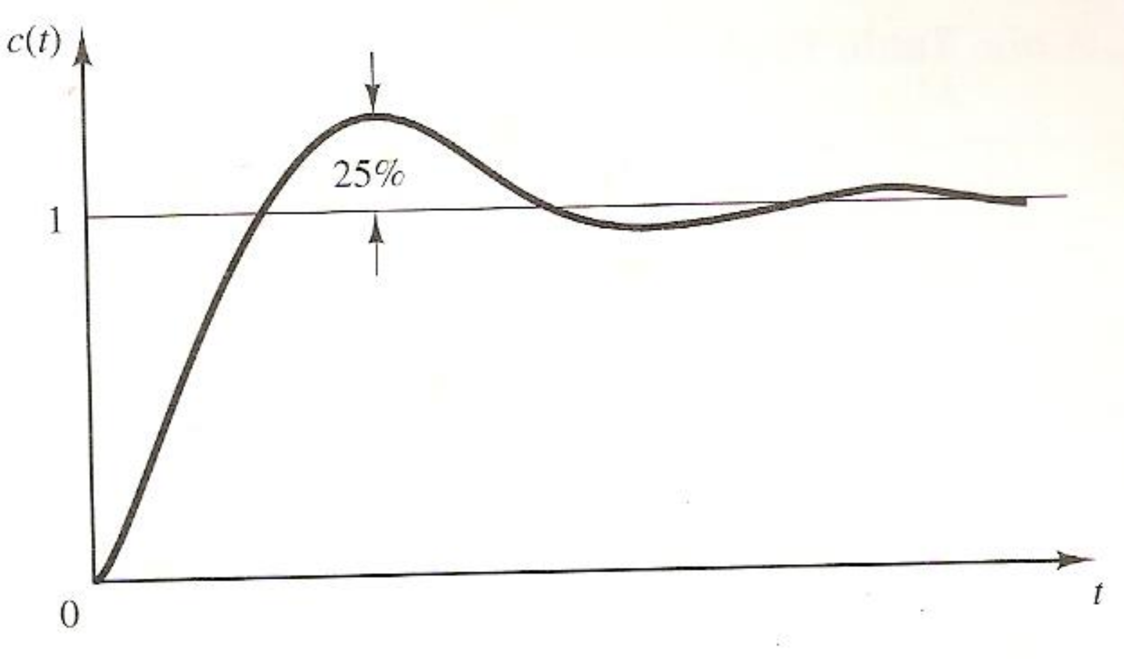
Experimental approaches are chosen for complicated system or system with unknown mathematical model

Zieglar-Nichols (Z-N) rules are very convenient

Z-N is based on experimental step response or on the value of Kp that results in marginal stability when only the proportional control action is used

Other tuning law: Chien, Hrones and Reswick method, Cohen Coon method

Z-N Method

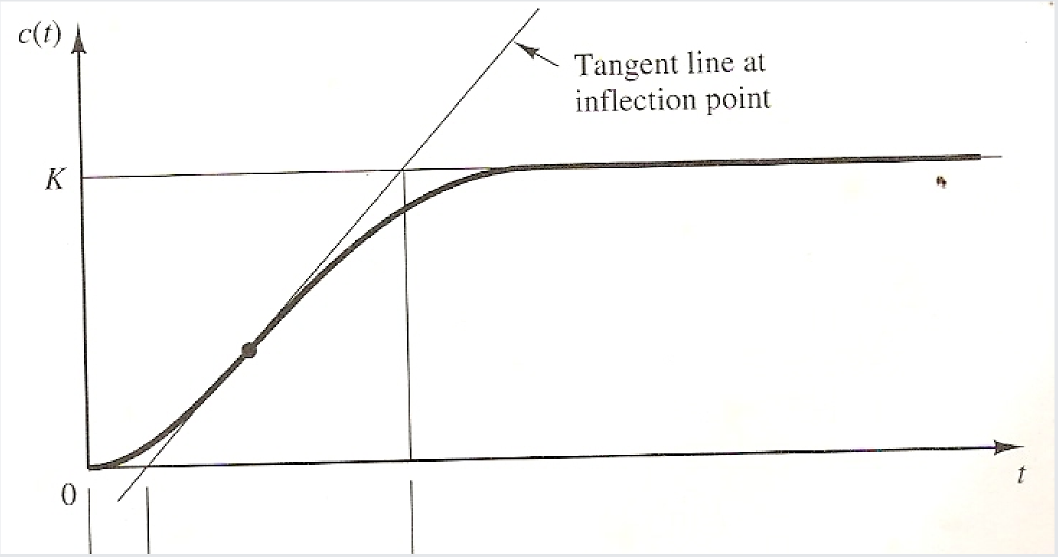


Two methods of Z-N tuning law:

1. 1st Method (for S-shaped step response curve)
2. 2nd Method (for output with sustained oscillation)

Z-N methods aim at obtaining 25% maximum overshoot in step response

Z-N Method (1st Method)



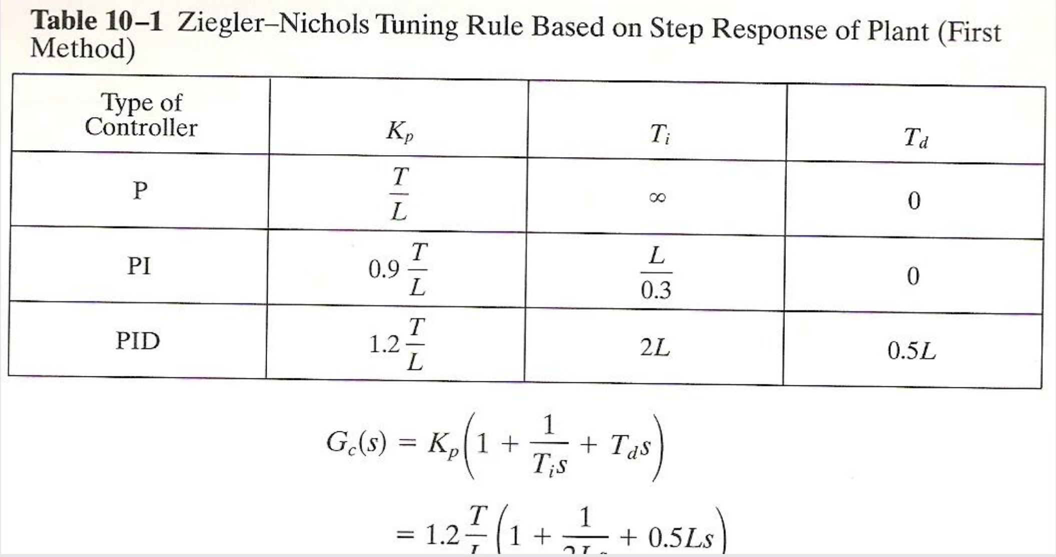


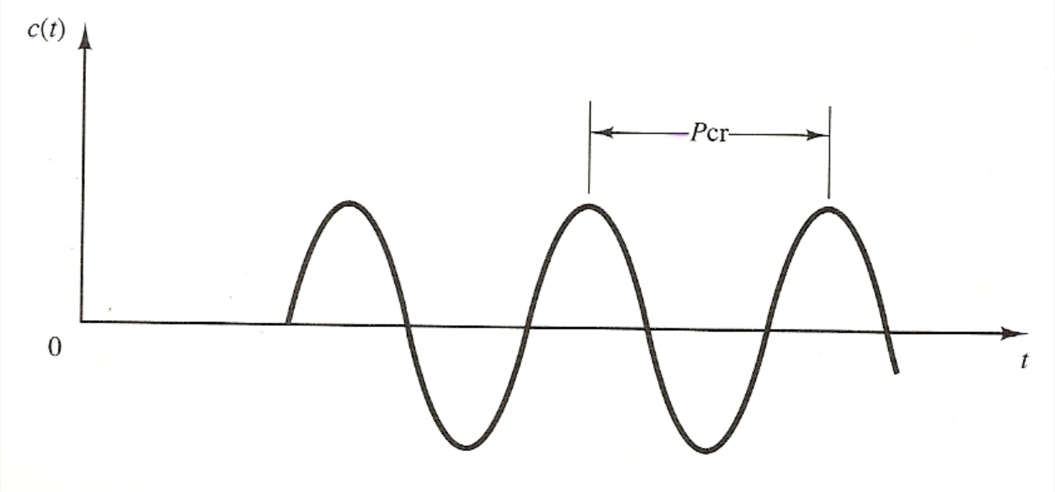
Table 10–1 Ziegler–Nichols Tuning Rule Based on Step Response of Plant (First Method)

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

$$= 1.2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0.5Ls\right)$$

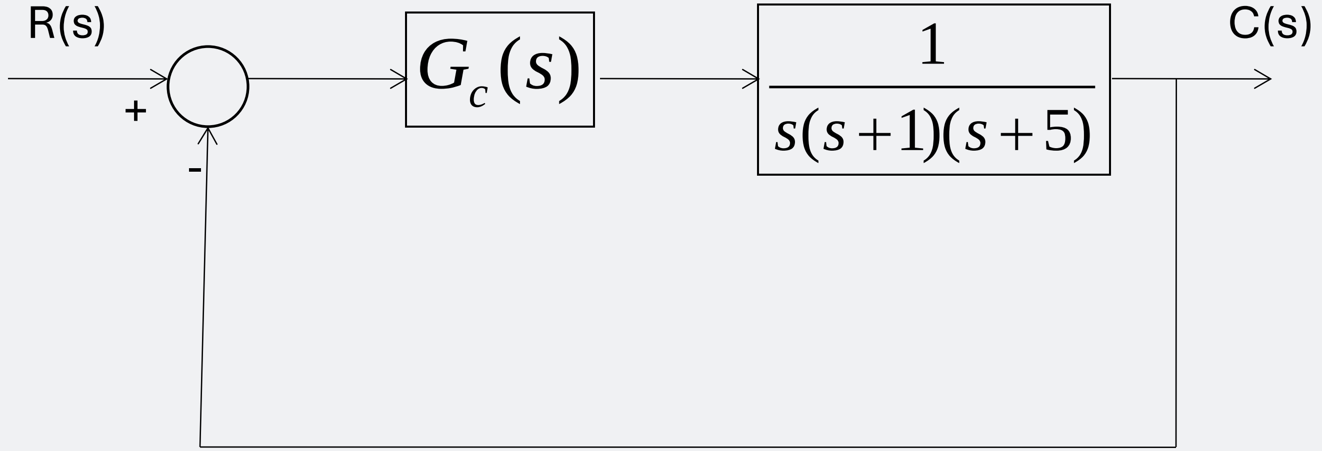
Z-N Method (2nd Method)



CONTROLLER	K	T_i	T_d	T_p
P	$0.5K_u$			T_u
PI	$0.4K_u$	$0.8T_u$		$1.4T_u$
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$0.85T_u$

Example 1

Applying Z-N tuning rule, determine the values of parameters K_p, T_i and T_d



Example 1

Solution:

-2nd method of Z-N tuning rules is used. By setting $T_i = \infty$ and $T_d = 0$, the CLTF:

$$\frac{C(s)}{R(s)} = \frac{K_p}{s(s+1)(s+5) + K_p}$$

-The value of K_p that makes the system marginally stable so that sustained oscillation occurs can be obtained by use of Routh's stability criterion

$\boldsymbol{s^3}$	1	5
$\boldsymbol{s^2}$	6	K_p
$\boldsymbol{s^1}$	$\frac{30 - K_p}{6}$	
$\boldsymbol{s^0}$	K_p	

Example 1

Solution (cont.)

-From s^1 row, sustained oscillation will occur if $K_p = 30$. Thus, $K_u = 30$

-Substituting $K = K_u$ in the characteristic equation yields

$$s^3 + 6s^2 + 5s + 30 = 0$$

-To obtain the freq of the sustained oscillation, substitute $s = j\omega$ into the characteristic equation

$$(j\omega)^3 + 6(j\omega)^2 + 5j\omega + 30 = 0$$

$$6(5 - \omega^2) + j\omega(5 - \omega^2) = 0$$

-The ultimate freq., ${\omega_u}^2 = 5$, hence $\omega_u = \sqrt{5}$

-Hence the T_u is

$$T_u = \frac{2\pi}{\omega_u} = 2.8099$$

Example 1

Solution (cont.)

-Thus, from the frequency tuning rule, the parameters can be determined

CONTROLLER	K	T_i	T_d	T_p
P	$0.5K_u$			T_u
PI	$0.4K_u$	$0.8T_u$		$1.4T_u$
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$0.85T_u$

-$K_p = 0.6K_u = 18,\ T_i = 0.5T_u = 1.405,\ T_d = 0.125T_u = 0.35124$

-The transfer function for the PID controller is

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) = 18\left(1 + \frac{1}{1.405s} + 0.35124s\right)$$

$$G_c(s) = \frac{6.3223(s + 1.4235)^2}{s}$$