Quick Intro to PID Controllers

Proportional, Derivative and Integral Control

Dr. ASHRAF E. SUYYAGH

THE UNIVERSITY OF JORDAN

DEPARTMENT OF COMPUTER ENGINEERING

FALL 2022

PID Control – Intuitive Explanation

https://www.youtube.com/watch?v=wkfEZmsQqiA&list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y
Or in ***presentation mode***, click on the video below

Integral Windup Problem– Intuitive Explanation

https://www.youtube.com/watch?v=NVLXCwc8HzM&list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y&index=2

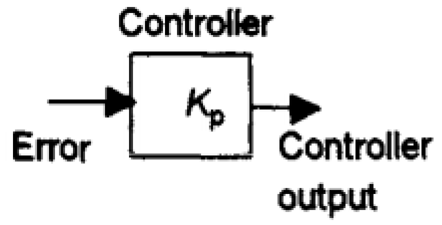
Or in ***presentation mode***, click on the video below

Noise Filtering in PID Control– Intuitive Explanation

https://www.youtube.com/watch?v=7dUVdrs1e18&list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y&index=3
Or in ***presentation mode***, click on the video below

P-Controller (Proportional Control)

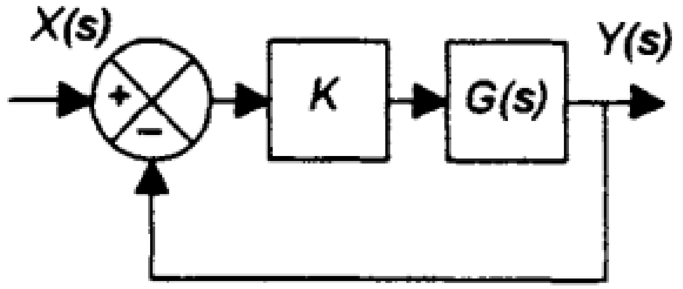
In proportional control, the controller produces a control action that is proportional to the error. There is a constant gain K_p acting on the error signal *e* and so:



$$\text{controller output} = K_p e$$

We have steady-state error! How the P-control gain K_p affects the stead-state errors?

For a closed loop-system with a process transfer function G(s), and unity feedback:



$$Y(s) = \frac{G(s)K_P}{1 + G(s)K_P} X(s)$$

We need to determining the value of the output as the time *t* tends to an infinite value. To do this in the s-domain, we use the *final value theorem (which if the limit exists):*

$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} sF(s)$$

P-Controller (Proportional Control)

$$Y(s) = \frac{G(s)K_P}{1+G(s)K_P}X(s) \rightarrow y_{ss} = \lim_{s\to 0} s\frac{G(s)K_P}{1+G(s)K_P}X(s)$$

If x(t) = u(t), then X(s) = 1/s

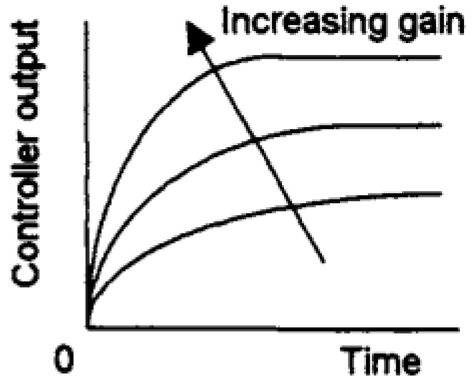
$$y_{ss} = \lim_{s\to 0} s\frac{K_P/(s+1)}{1+K_P/(s+1)}\frac{1}{s}$$

$$= \lim_{s\to 0} \frac{K_P}{s+1+K_P}$$

Suppose G(s) = 1/(1+s) for this demo example

As the limit goes to zero → $y_{ss} = K_P/(1+K_P)$

Kp	Yss	Offset
1	0.5	0.5
4	0.8	0.2
10	0.91	0.09

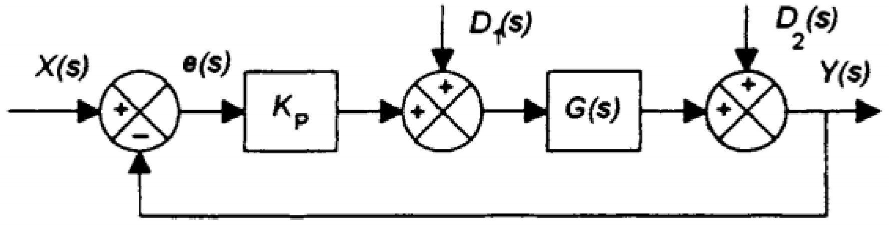


Increasing the proportional gain decreases the steady state error, but it does not eliminate it to zero

P-Controller (Proportional Control and Disturbance Rejection)

Previously we considered the effect of a disturbance on the performance of a closed-loop control system. We have seen that closed-loop control systems are better at minimizing disturbances than an open-loop system.

Consider this closed-loop control system with two possible sources of disturbances, one being a disturbance affecting the input to the process and the other affecting its output.



$Y(s)$ is $K_P G(s)e(s) + G(s)D_1(s) + D_2(s)$

Error = $X(s) - Y(s)$.

$$Y(s) = K_P G(s)[X(s) - Y(s)] + G(s)D_1(s) + D_2(s)$$

$$Y(s)[1 + K_P G(s)] = K_P G(s)X(s) + G(s)D_1(s) + D_2(s)$$

$$Y(s) = \frac{K_P G(s)}{1 + K_P G(s)}X(s) + \frac{G(s)}{1 + K_P G(s)}D_1(s) + \frac{1}{1 + K_P G(s)}D_2(s)$$

Increasing the proportional gain reduces the effect of disturbances.

The first term is the normal expression for the closed-loop system with no disturbances. The other terms are the terms arising from the two disturbances. The factor $1/[1 + K_pG(s)]$ is thus a measure of how much the effects of the disturbances are modified by the closed-loop.

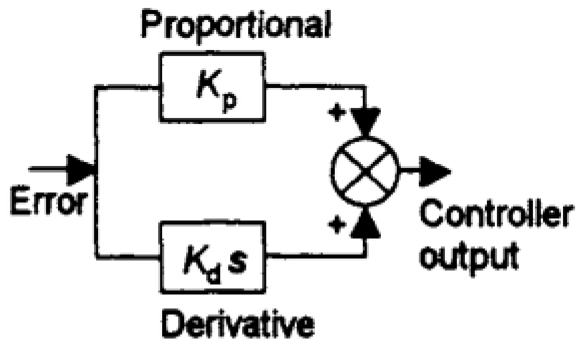
PD-Controller (Proportional and Derivative Control)

In derivative control, the controller produces a control action that is proportional to the rate at which the error is changing.

$$K_d \frac{de}{dt}$$ where K_d is the derivative gain

Derivative control is not used alone but always in conjunction with proportional control and, often, also integral control.

$$\text{controller output} = K_p e + K_d \frac{de}{dt}$$



The proportional element has an input of the error **e** and an output of K_p**e**. The derivative element has an input of **e** and an output which is proportional to the derivative of the error with time

In Laplace form: $\text{controller output } (s) = (K_p + K_d s)E(s)$
Can be rewritten as: $\text{controller output } (s) = K_P(1 + T_d s)E(s)$
where $T_d = K_d/K_p$ and is called the ***derivative time constant***.

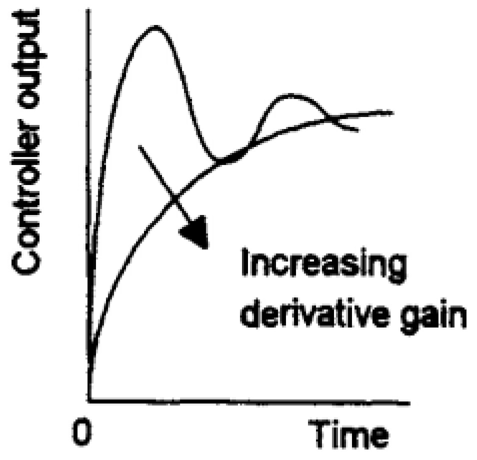
PD-Controller (Proportional and Derivative Control)

Derivative control has the controller effectively ***anticipating*** the way an error signal is growing and responding as the error signal begins to change (This is why we say that the derivative controller looks for the future).

A problem with this is that noise can lead to quite large responses.

Adding derivative control to **only** proportional control still leaves the output steady-state error and does not eliminate it.

Changing the amount of derivative control in a closed-loop system will change the damping ratio since increasing K_d increases the damping ratio.



PI-Controller (Proportional and Integral Control)

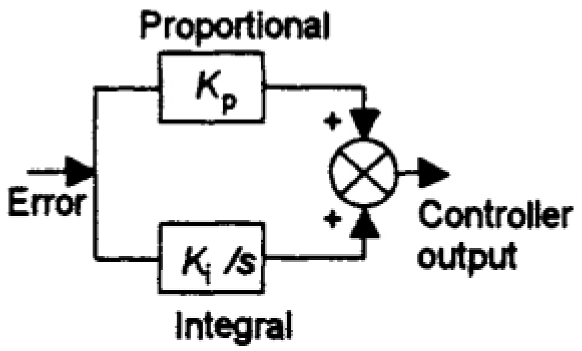
In integral control, the controller produces a control action that is proportional to the integral of the error with time.

$$K_i \int e \, dt$$ where K_i is the integrating gain

Integral control is not used alone but always in conjunction with proportional control and, often, also derivatve control.

$$\text{controller output} = K_p e + K_i \int e \, dt$$

The proportional element has an input of the error **e** and an output of K_p**e**. The integral element has an input of **e** and an output which is proportional to the integral of the error with time



In Laplace form: $\text{controller output }(s) = \left(K_p + \frac{K_i}{s}\right)E(s)$

Can be rewritten as: $\text{controller output }(s) = \frac{K_p}{s}\left(s + \frac{1}{T_i}\right)E(s)$

where $T_i = K_p/K_i$ and is called the ***integral time constant.***

The presence of integral control **eliminates** steady-state errors and this is generally an important feature required in a control system.

PID-Controller (Proportional, Derivative and Integral Control)

The basic form is a ***three-term controller.***

$$\text{output} = K_p e + K_i \int e\,dt + K_d \frac{de}{dt}$$

In Laplace form:

$$\text{output}(s) = K_p\left(1 + \frac{K_i}{K_p s} + \frac{K_d}{K_p} s\right)E(s)$$

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

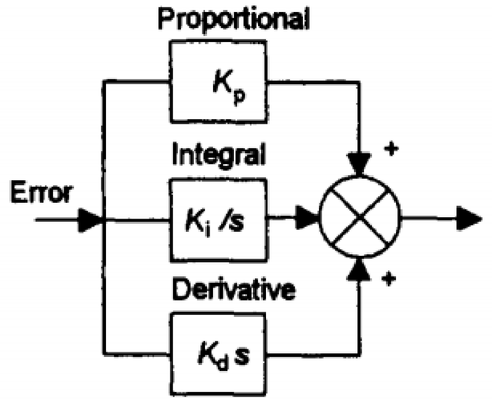


Table 7.4 Effect of Increasing the PID Gains K_P, K_D, and K_I on the Step Response

PID Gain	Percent Overshoot	Settling Time	Steady-State Error
Increasing K_P	Increases	Minimal impact	Decreases
Increasing K_I	Increases	Increases	Zero steady-state error
Increasing K_D	Decreases	Decreases	No impact

PID-Controller (Proportional, Derivative and Integral Control)

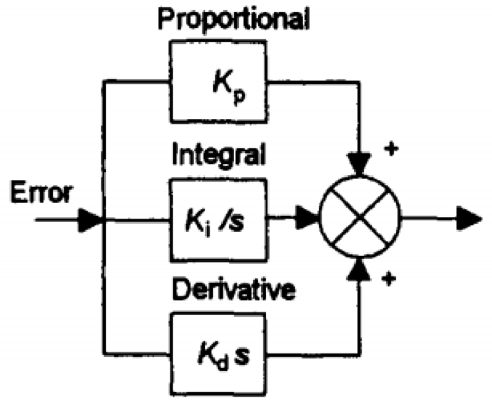
The basic form is a ***three-term controller.***

$$\text{output} = K_p e + K_i \int e\,dt + K_d \frac{de}{dt}$$

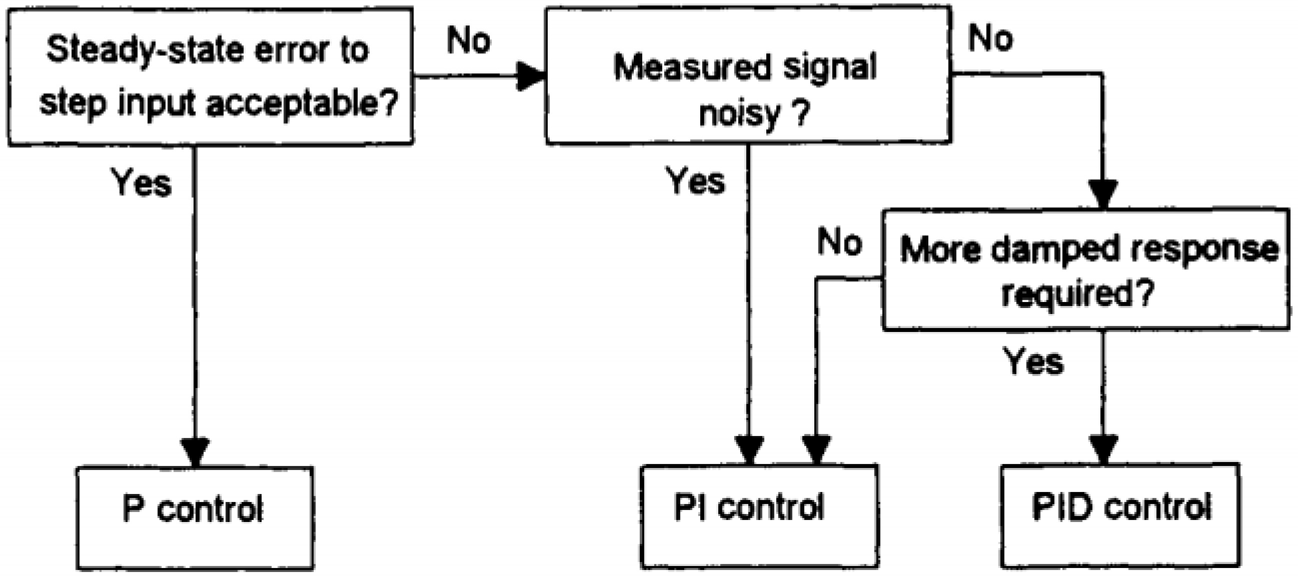
In Laplace form:

$$\text{output}\ (s) = K_p\left(1 + \frac{K_i}{K_p s} + \frac{K_d}{K_p} s\right)E(s)$$

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

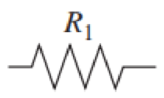
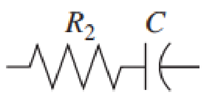
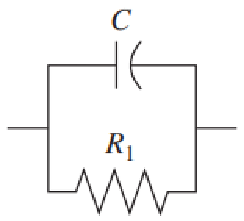
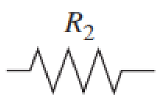
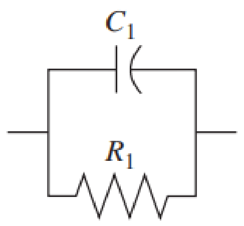
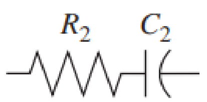


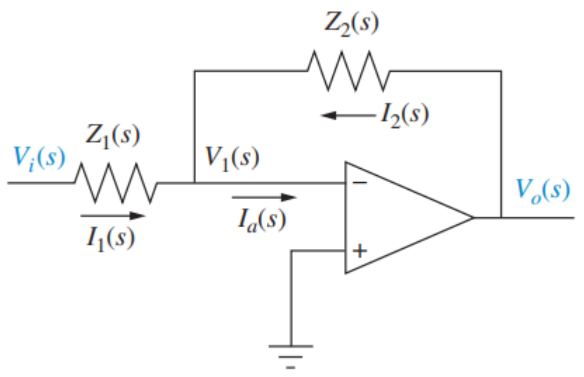
Controller Selection →


Steady-state error to step input acceptable?
No
Measured signal noisy ?
No
Yes
Yes
No
More damped response required?
Yes
P control
PI control
PID control

How are PID Controllers Realized (Classical Techniques)

This slide is just to have a quick idea, do not memorize the shapes or equations!
There are equations used to get the values for the resistors and capacitors from K_p, K_i, and K_d

Function	$Z_1(s)$	$Z_2(s)$	$G_c(s) = -\dfrac{Z_2(s)}{Z_1(s)}$
PI controller			$-\dfrac{R_2}{R_1}\dfrac{\left(s+\dfrac{1}{R_2C}\right)}{s}$
PD controller			$-R_2C\left(s+\dfrac{1}{R_1C}\right)$
PID controller			$-\left[\left(\dfrac{R_2}{R_1}+\dfrac{C_1}{C_2}\right)+R_2C_1s+\dfrac{\dfrac{1}{R_1C_2}}{s}\right]$



How are PID Controllers Realized (Computer Control)

We use libraries and functions, because the controller and system equations are all realized in software, so is the PID controller

Example, ARM DSP library has a function for PID Controller

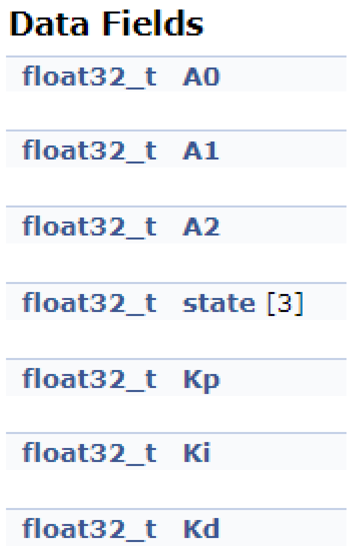
https://www.keil.com/pack/doc/CMSIS/DSP/html/group__PID.html

The function takes as input two parameters S and in:
__STATIC_FORCEINLINE float32_t arm_pid_f32 (arm_pid_instance_f32 * S, float32_t in)

in is the input signal
And S is a struct which has the gains and the values of Kp, Kd, Ki
https://www.keil.com/pack/doc/CMSIS/DSP/html/structarm__pid__instance__f32.html

But where and how do we get the values of Kp, Kd, Ki?

MATLAB PID Tuner (DEMO in class)

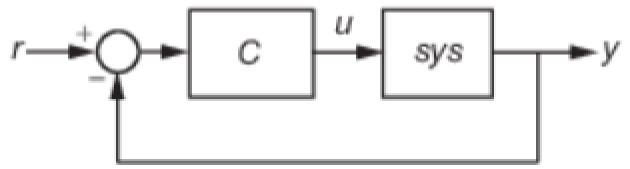


Data Fields

float32_t	**A0**
float32_t	**A1**
float32_t	**A2**
float32_t	**state** [3]
float32_t	**Kp**
float32_t	**Ki**
float32_t	**Kd**

Initial Automatic PID Tuning

MATLAB has the ***pidtune*** command which takes in as input the plant system as input, and the type of controller you want to use. Notice, that MATLAB assumes a unity feedback design, so, if need be, transform your system to match a unity feedback design.



sys is the transfer function, it could be constructed using any of the techniques we have used before (tf, zpk)
C could be any of the following controllers (in this course, we only cover 1-DOF types, and only those in red)

- 'P' — Proportional only
- 'I' — Integral only
- 'PI' — Proportional and integral
- 'PD' — Proportional and derivative
- 'PDF' — Proportional and derivative with first-order filter on derivative term
- 'PID' — Proportional, integral, and derivative
- 'PIDF' — Proportional, integral, and derivative with first-order filter on derivative term