Single Input Single Output system control

Elena VANNEAUX

elena.vanneaux@ensta.fr

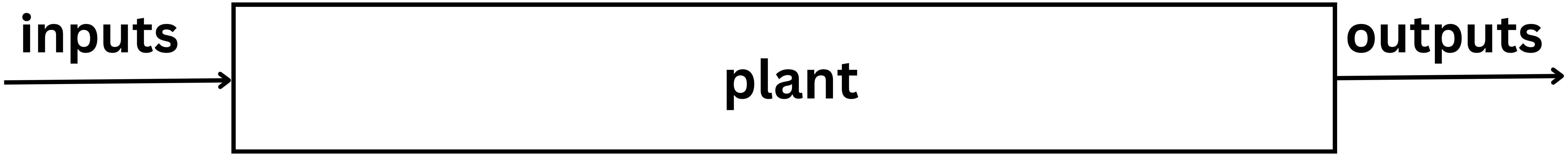
Course grade breakdowns

Labs - 40%

Final test - 30%

Final project - 30 %

What is a control system?



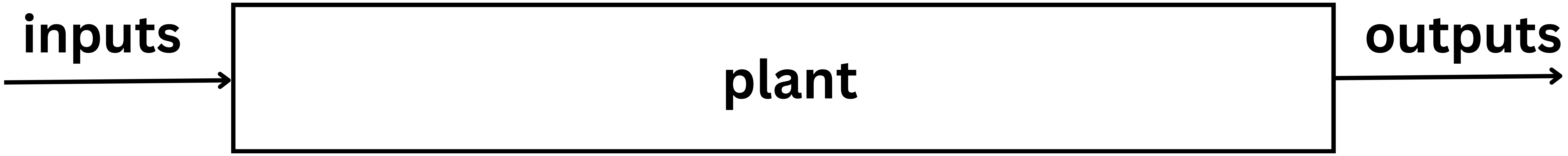
how do I change this ?

to get what I want?

Control system =

mechanism that alters the future state of the system

What is a control theory?



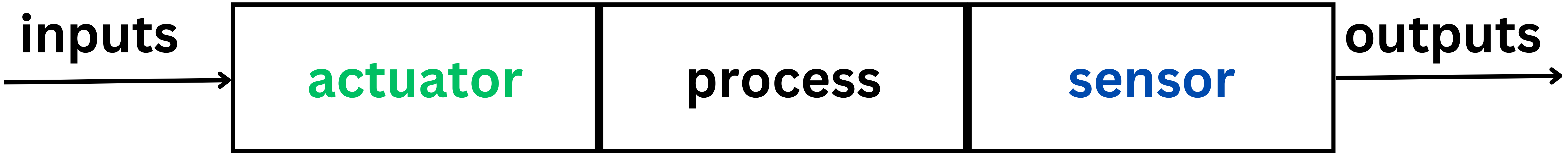
how do I change this ?

to get what I want?

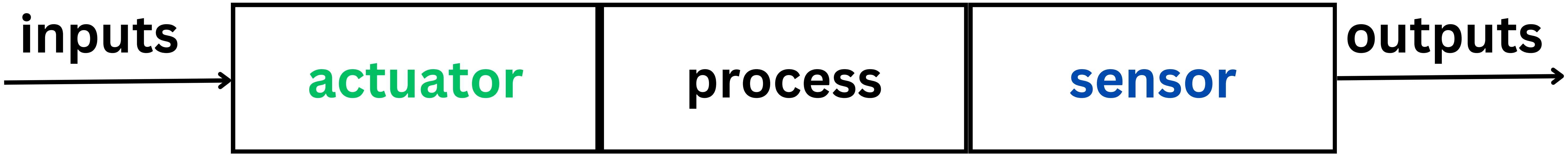
Control system =
mechanism that alters the future state of the system

Control theory =
a strategy to select appropriate input

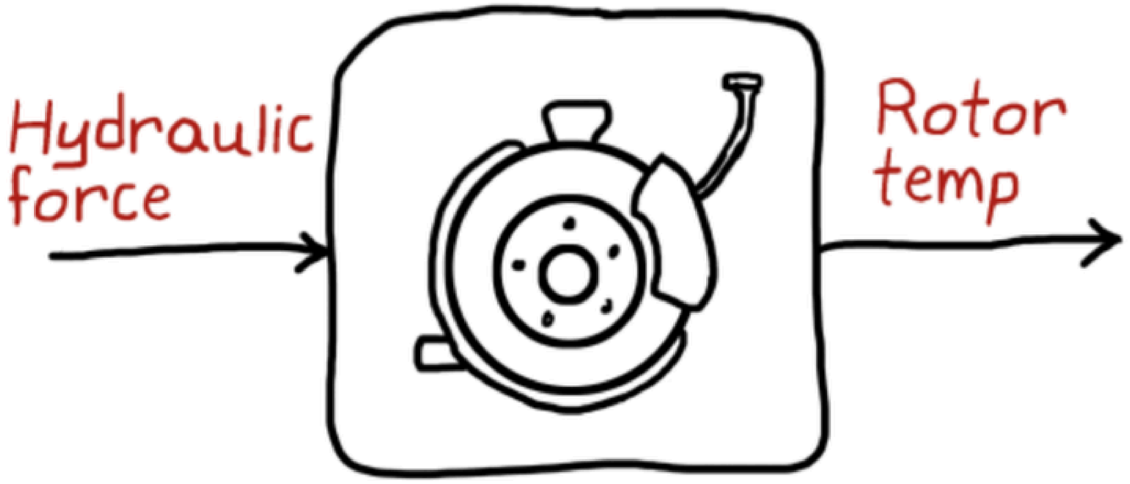
SISO vs MIMO



SISO vs MIMO

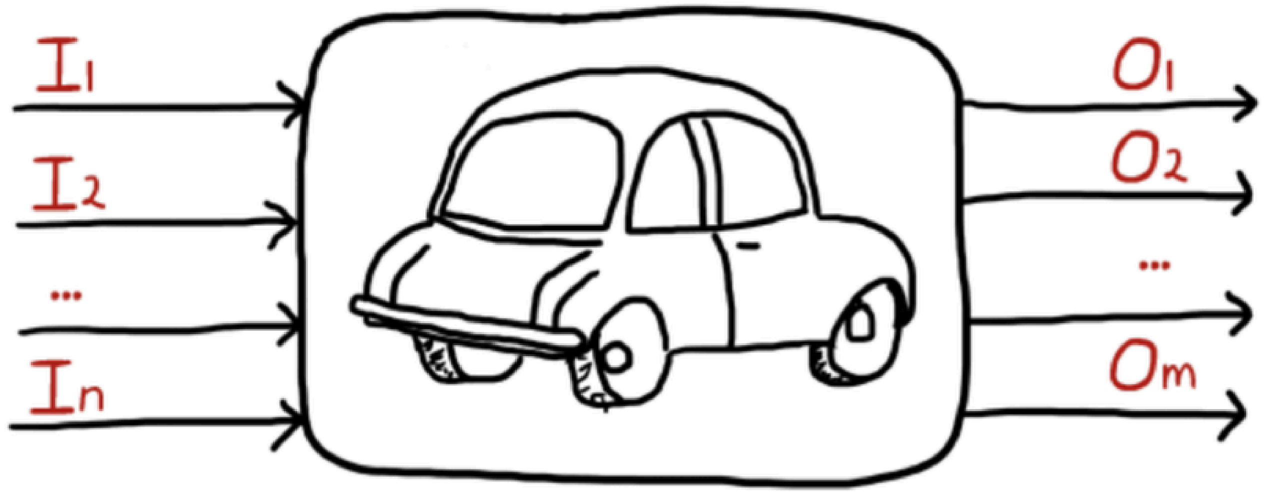


SISO



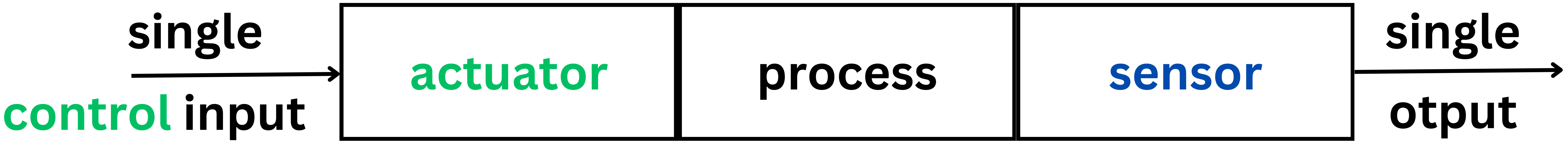
Single Input Single Output

MIMO



Multiple Inputs Multiple Outputs

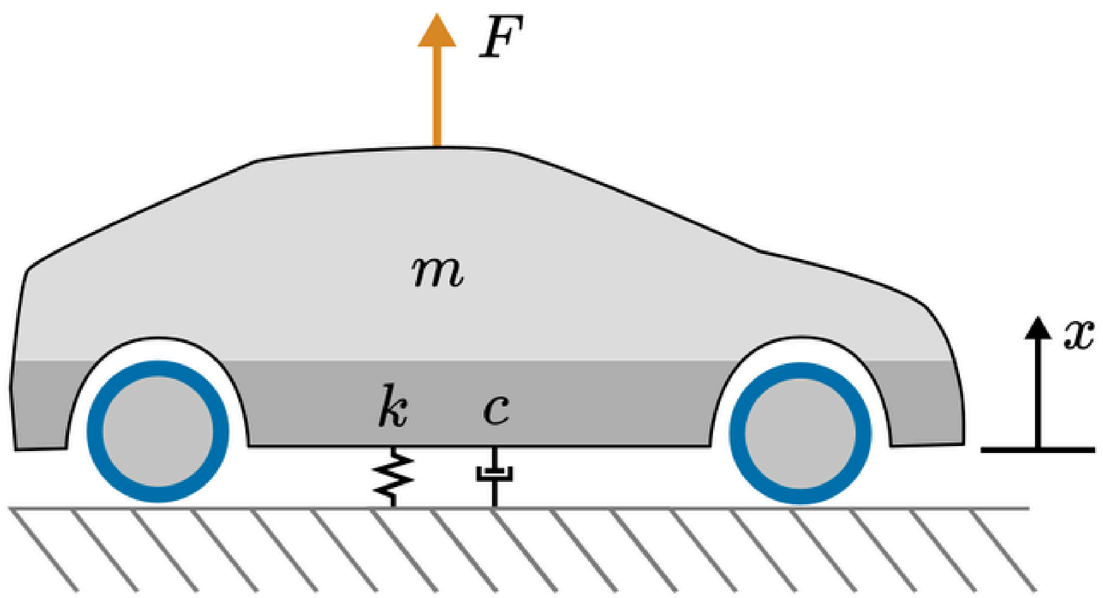
SISO control system



This lecture we will focus on **single input single output (SISO)** systems

Ex.1: . Vehicle Suspension System

https://www.youtube.com/watch?v=lPg695IXbPo



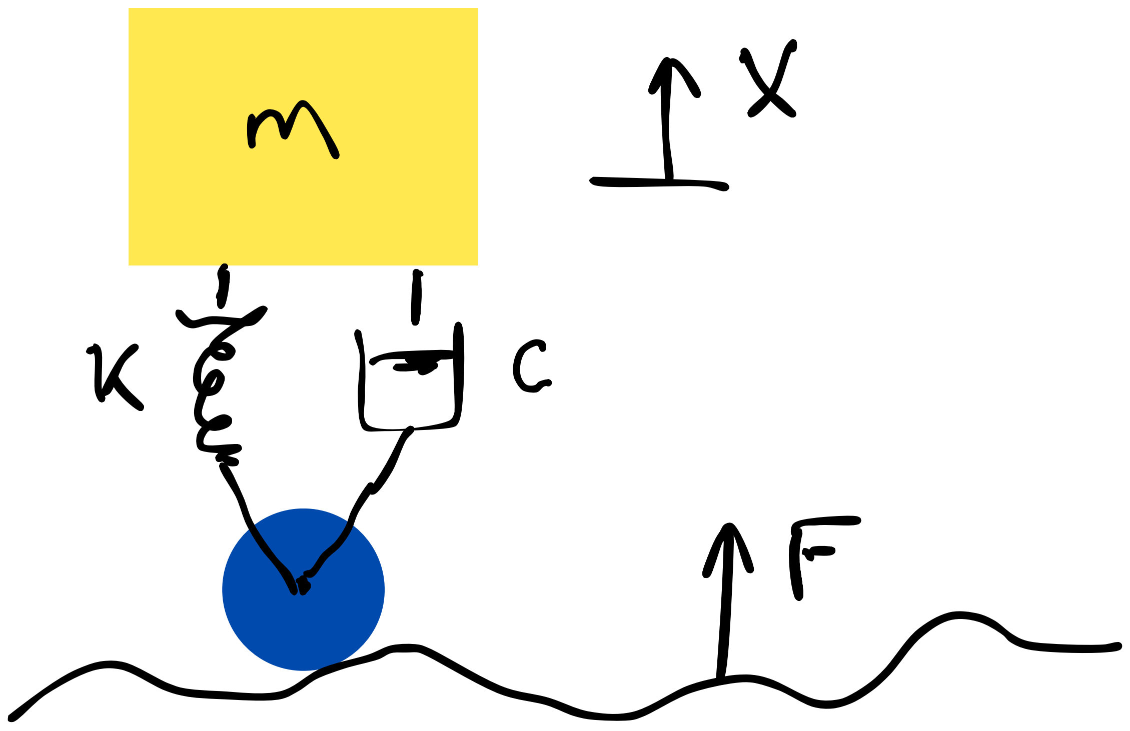
The suspension system in cars, trucks, and other vehicles uses springs and dampers to absorb shocks from the road and provide a smooth ride.

Newton's second law (translational motion):

$$m\ddot{x} = F_{total} = \underbrace{-kx}_{\text{spring force}} - \underbrace{c\dot{x}}_{\text{friction force}} + \underbrace{F}_{\text{external force}}$$

Hooke's law Stokes' law

Ex.1: . Vehicle Suspension System



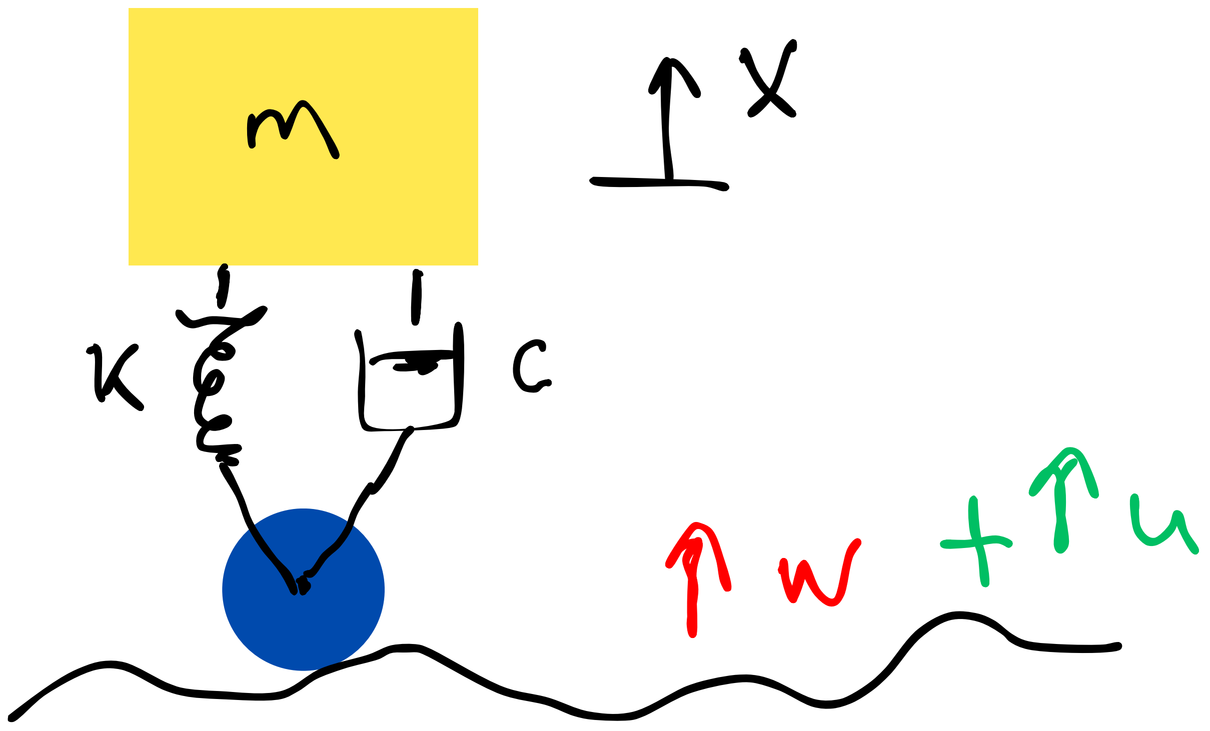
The suspension system in cars, trucks, and other vehicles uses springs and dampers to absorb shocks from the road and provide a smooth ride.

$$m\ddot{x} = F_{total} = -kx + c\,\dot{x} + F$$

Distrubance W

This disturbance represents real-world conditions that a car suspension system might encounter, such as road irregularities, bumps, or potholes

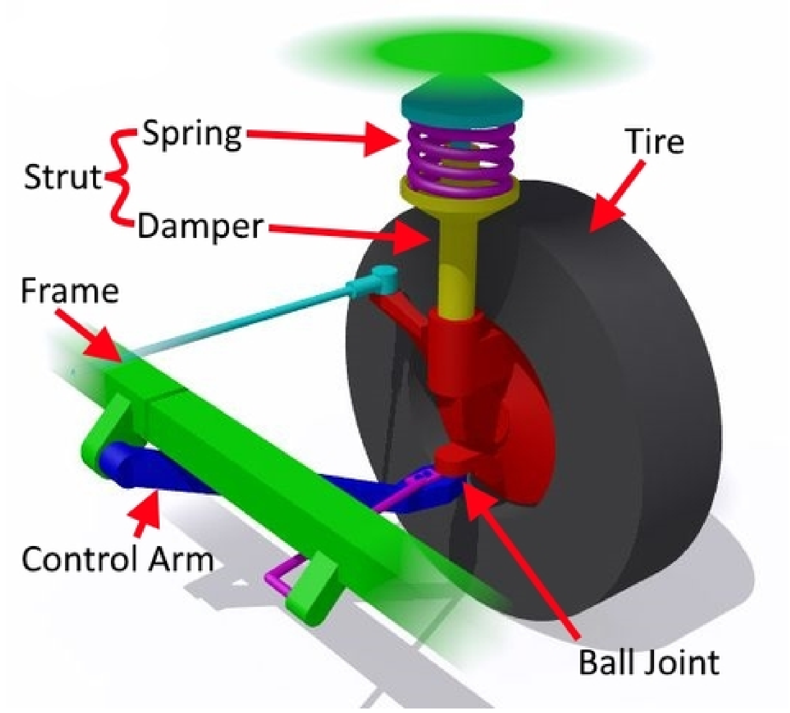
Ex.1: . Vehicle Suspension System



The suspension system in cars, trucks, and other vehicles uses springs and dampers to absorb shocks from the road and provide a smooth ride.

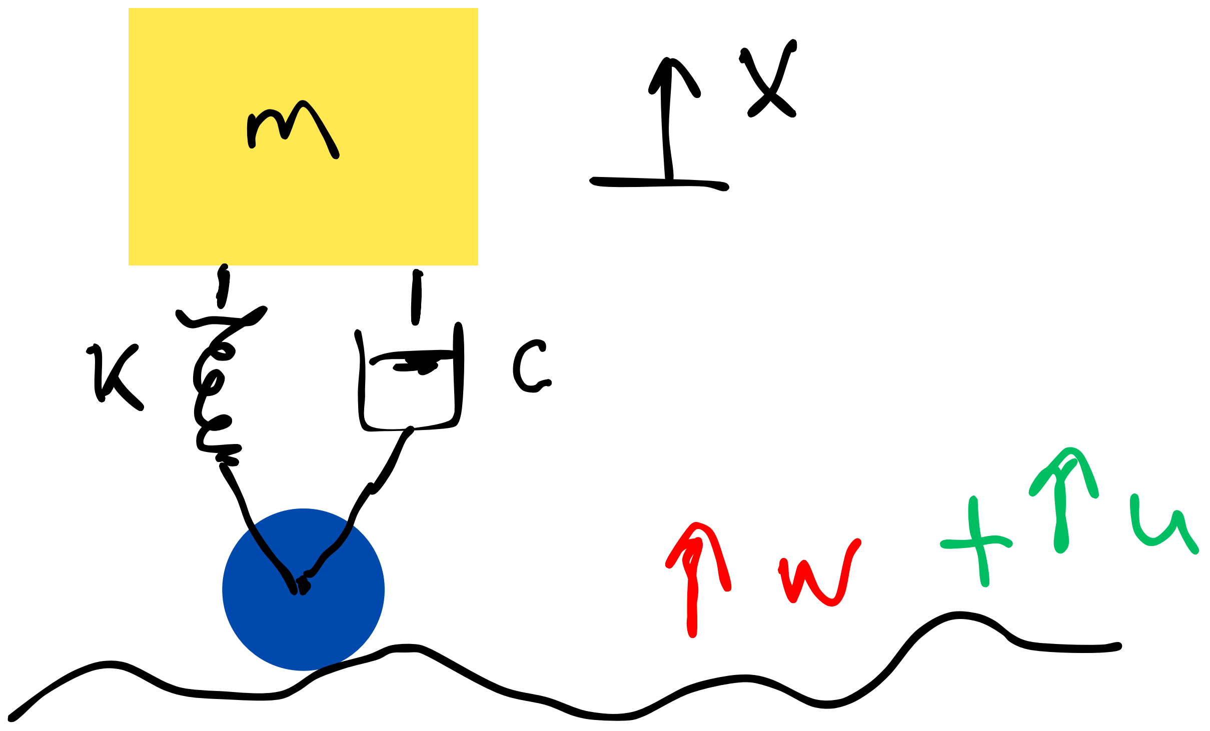
$$m\ddot{x} = F_{total} = -kx + c\,\dot{x} + F$$

Distrubance W + Control U



Design a controller to minimize vibrations and improve ride comfort while maintaining stability.

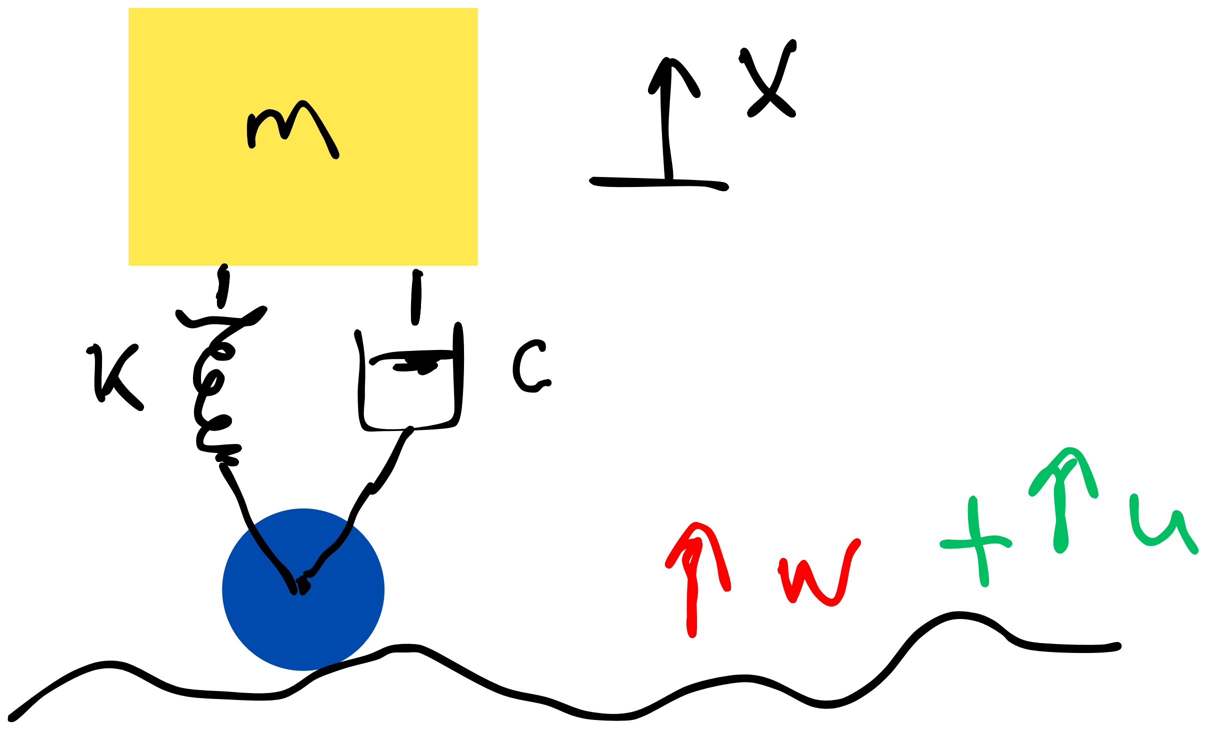
Ex.1: . Vehicle Suspension System



System model

$$\dot{x} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} W}_{\text{process}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u}_{\text{single input}}$$

Ex.1: . Vehicle Suspension System



System model

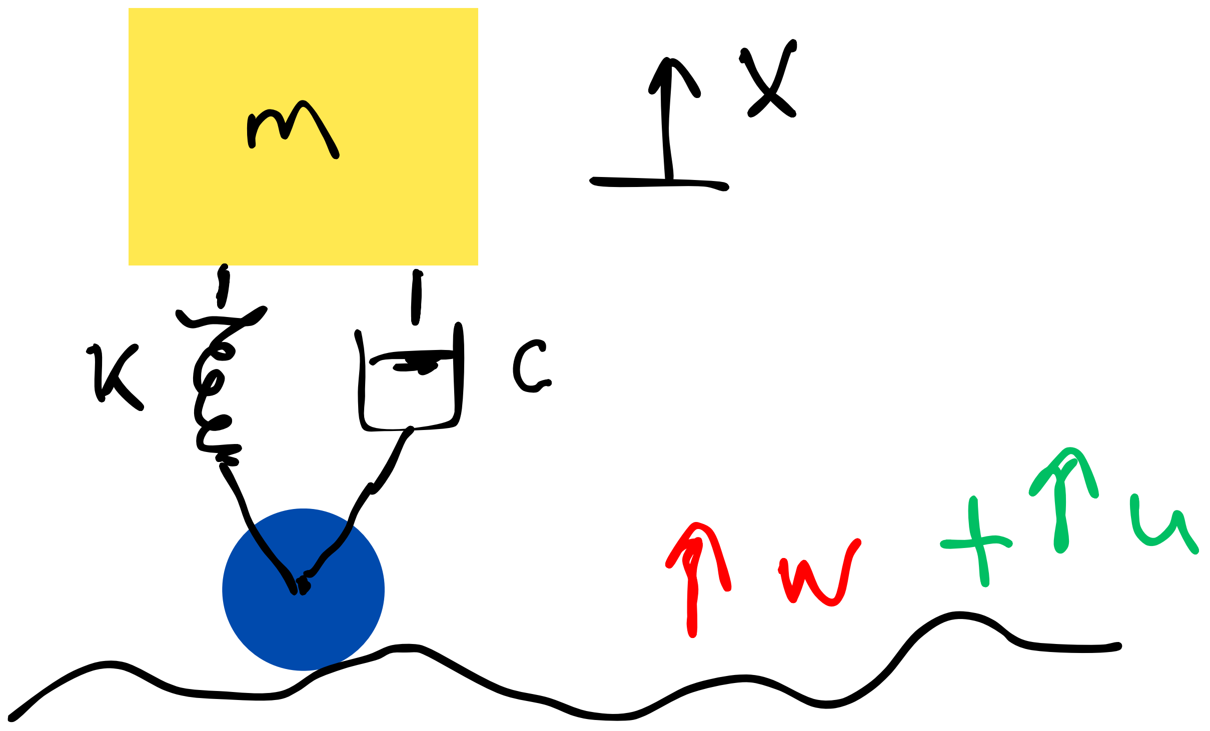
$$\dot{x} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} w}_{\text{process}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u}_{\text{single input}}$$

let us only measure the position of the car, but not it's speed

$$\underbrace{y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}}_{\text{single output}}$$

$$y = x$$

Ex.1: . Vehicle Suspension System



System model

$$\dot{x} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} W}_{\text{process}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix}}_{\text{single input}} u$$

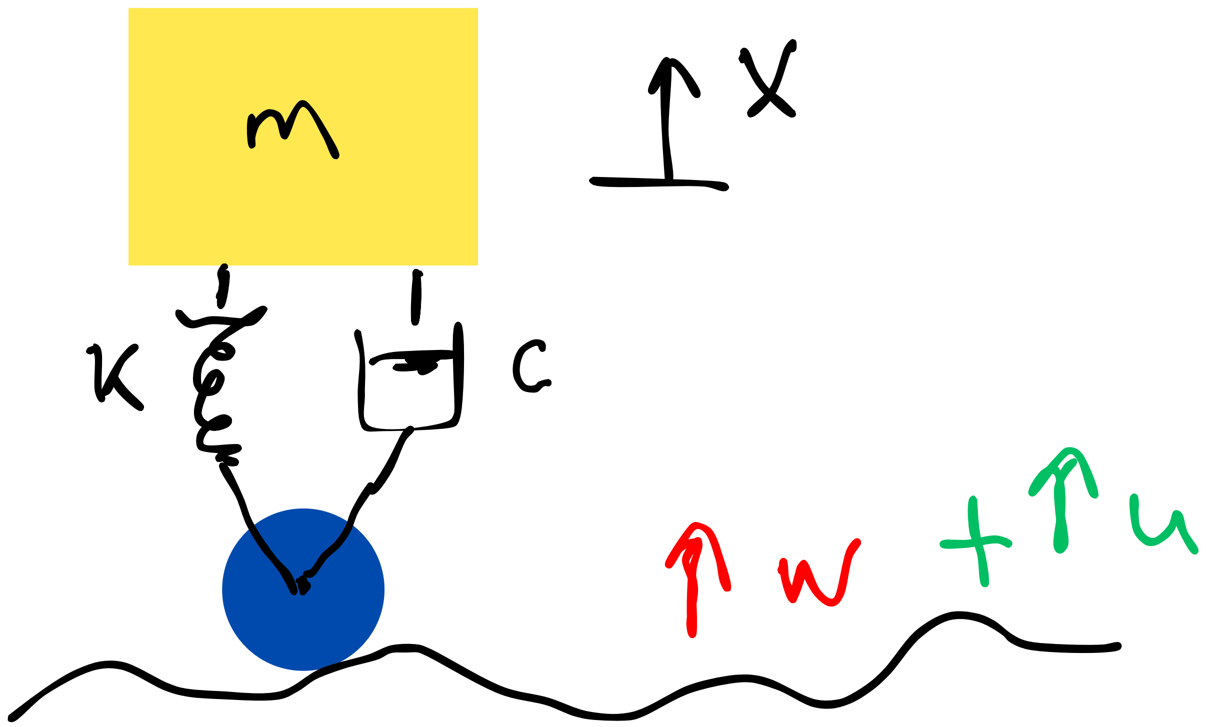
let us only measure the position of the car, but not it's speed

$$\underbrace{y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}}_{\text{single output}}$$

$y = x$

Design a controller to minimize vibrations and improve ride comfort while maintaining stability.

Ex.1: . Vehicle Suspension System



System model

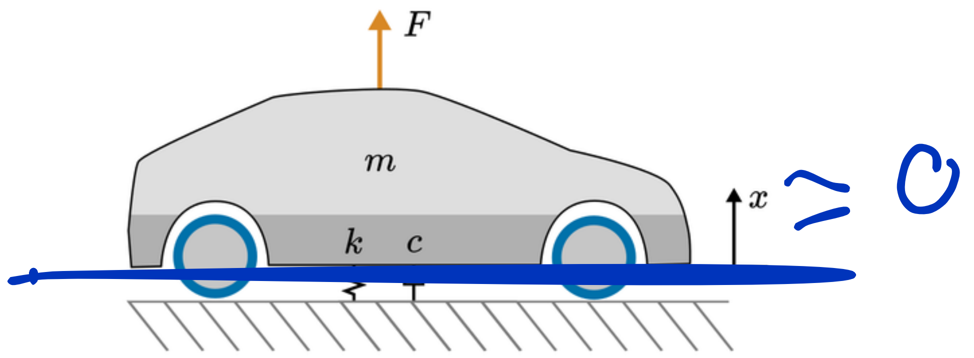
$$\underbrace{\dot{x} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} w}_{\text{process}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u}_{\text{single input}}$$

let us only measure the position of the car, but not it's speed

$$\underline{y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}}$$

single output

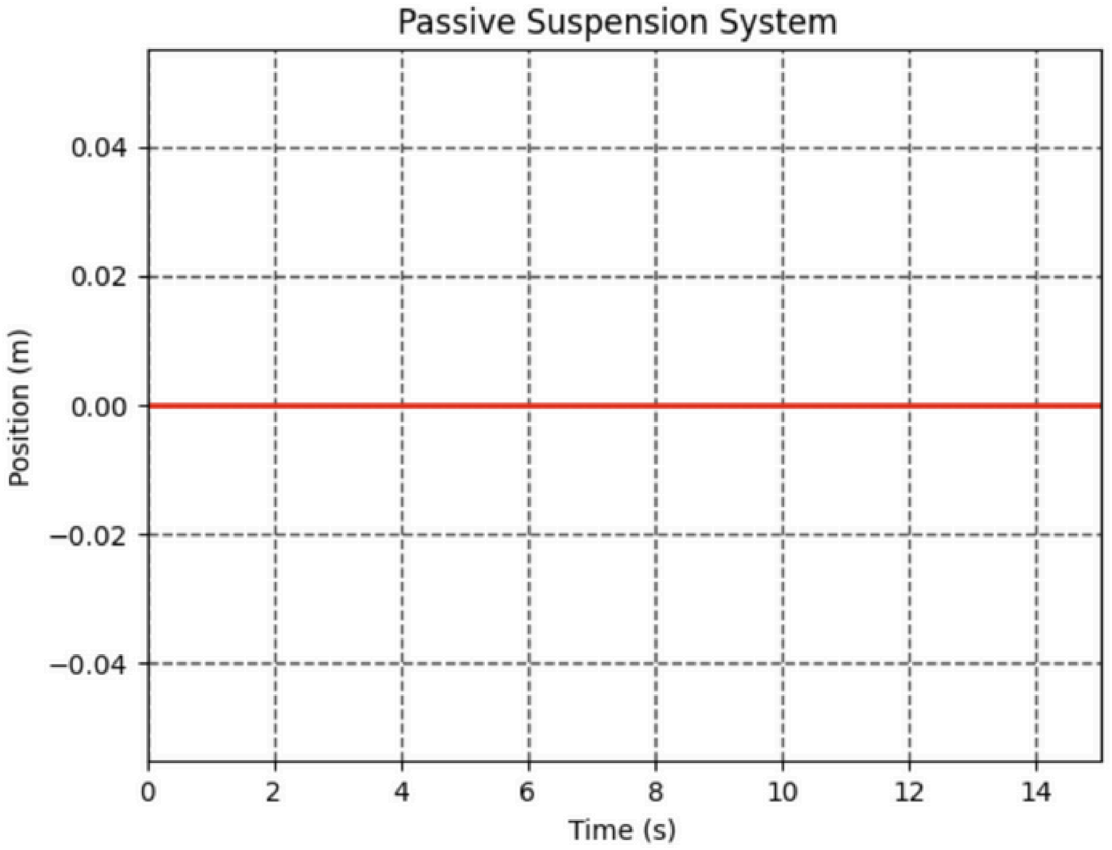
$y = x$

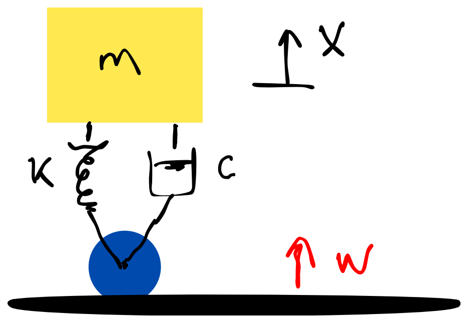


Design a controller that stabilizes the position of the car at the zero level, x = 0.

Why do we need active control?

$w(t) = 0$

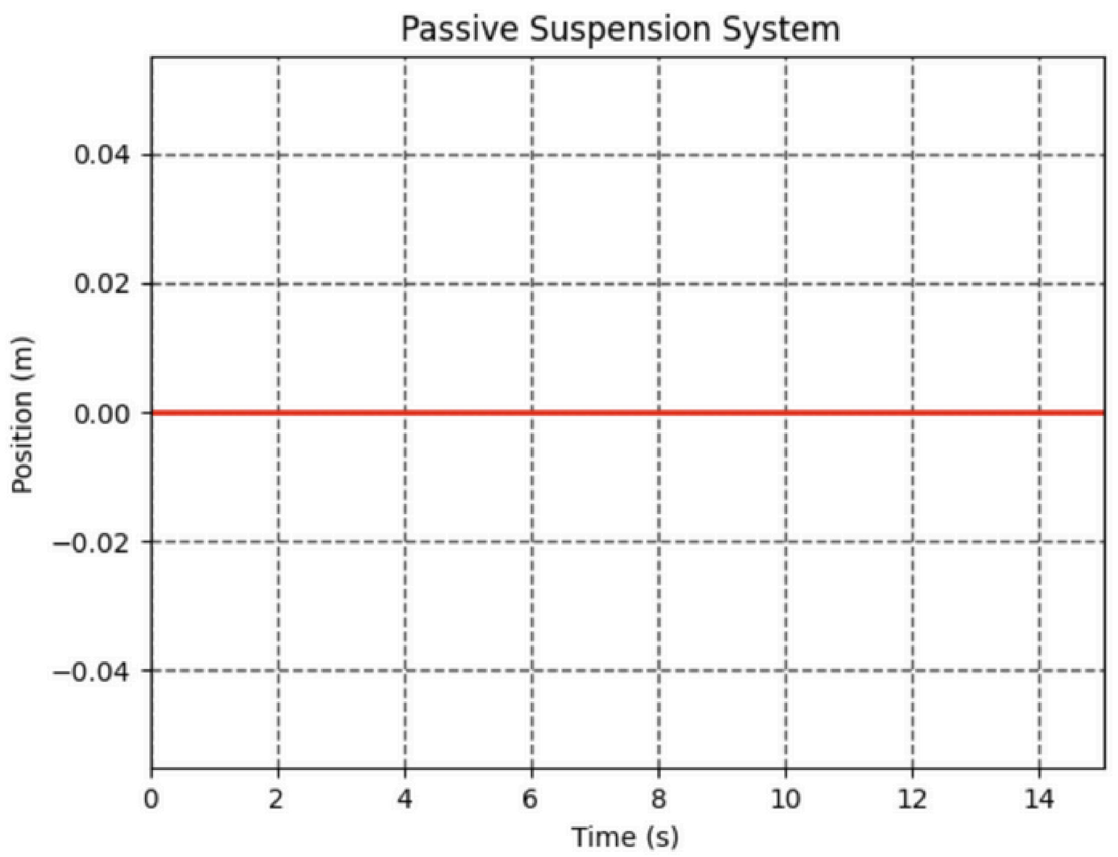


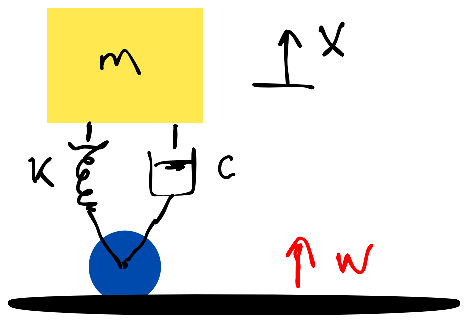


A flat road

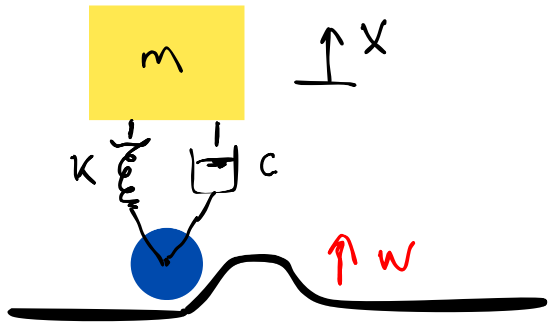
Why do we need active control?

$w(t) = 0$





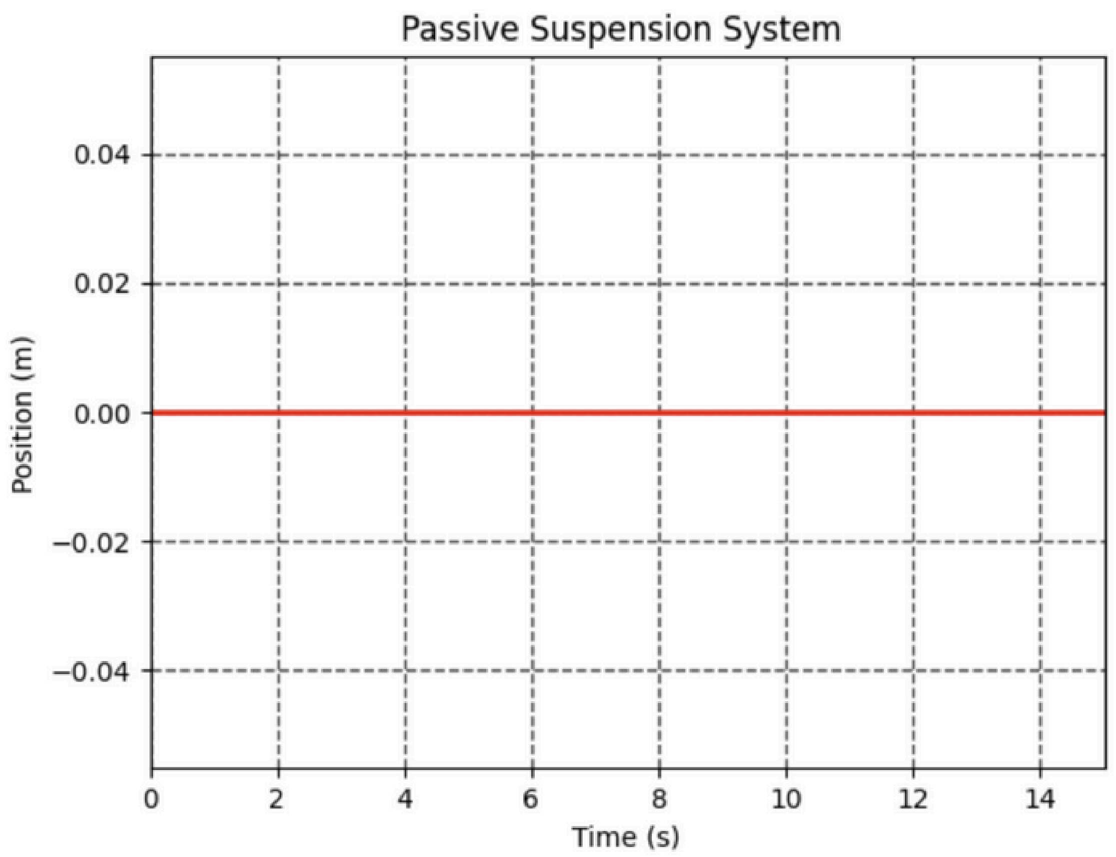
A flat road

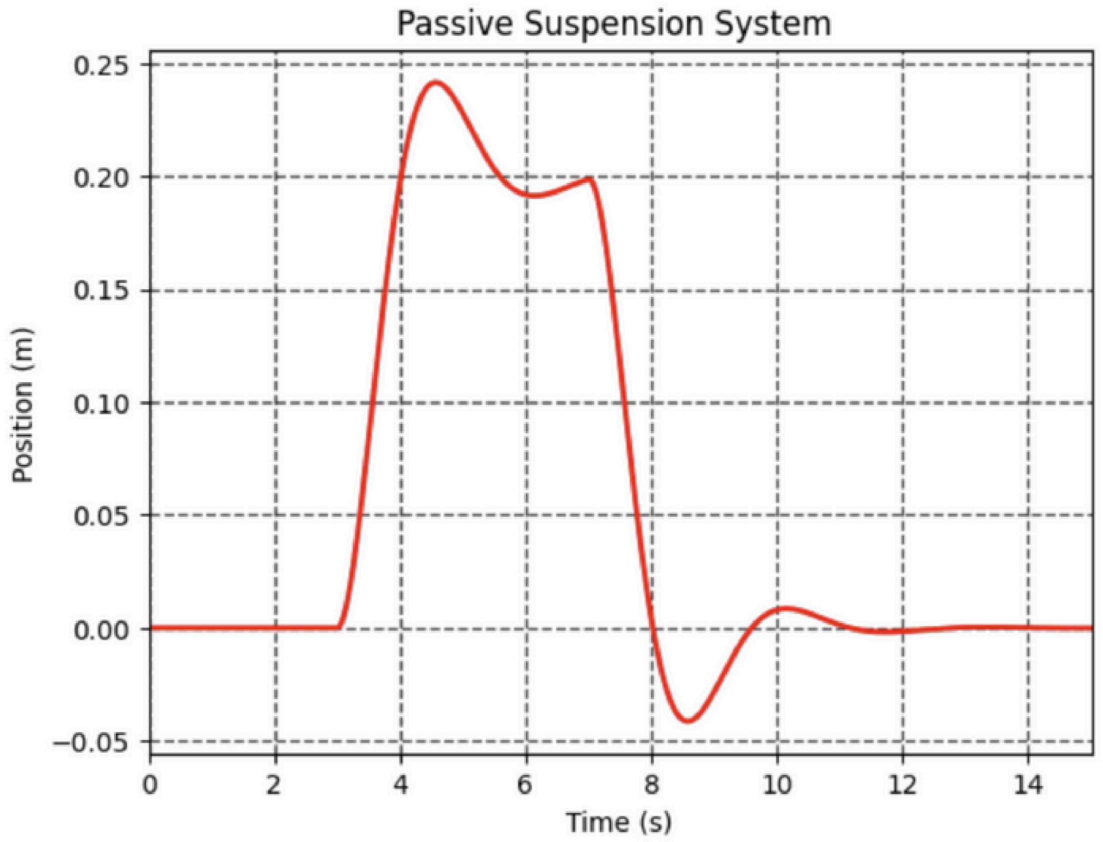


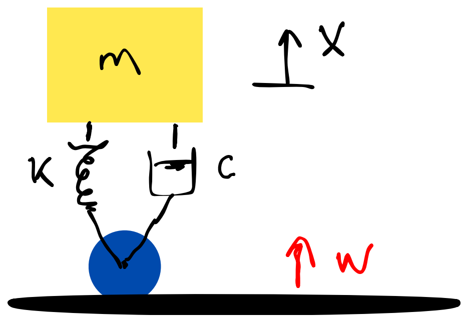
Why do we need active control?

$w(t) = 0$

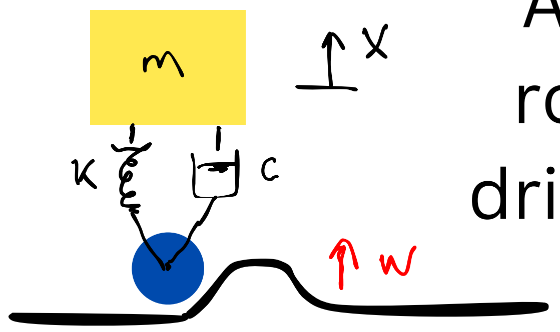
$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$







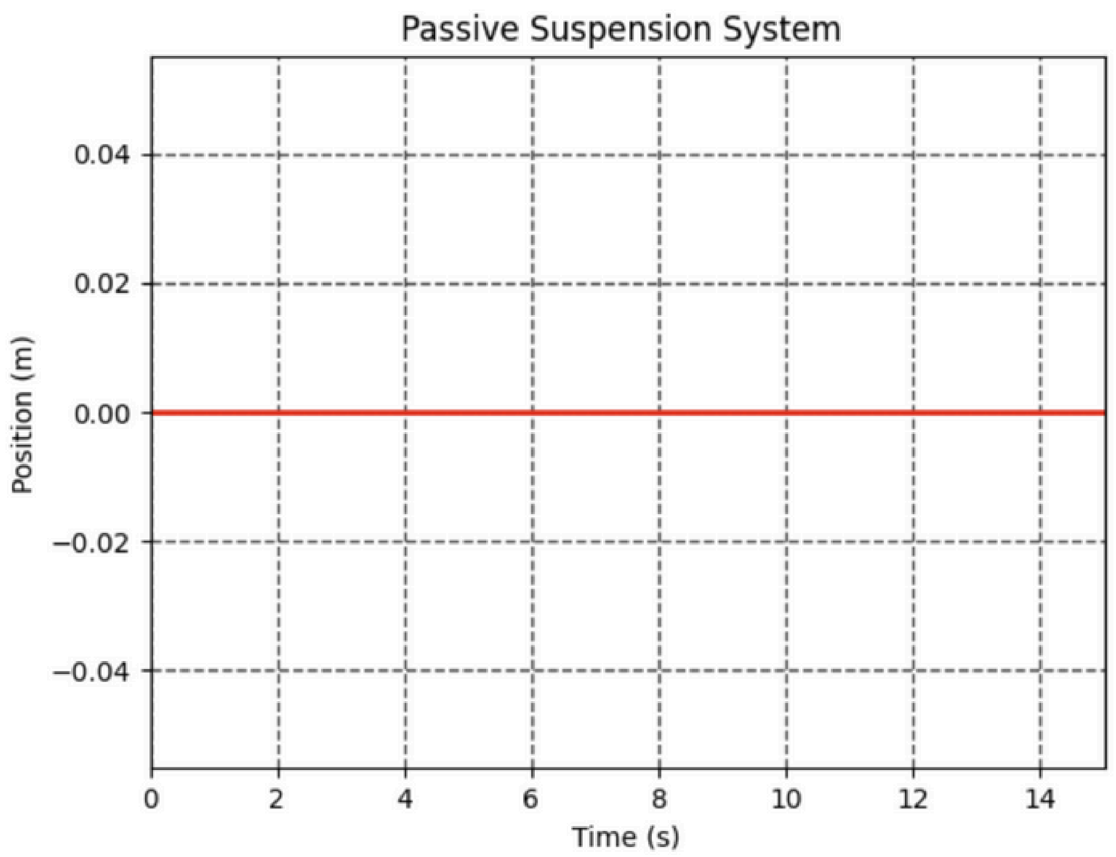
A flat road

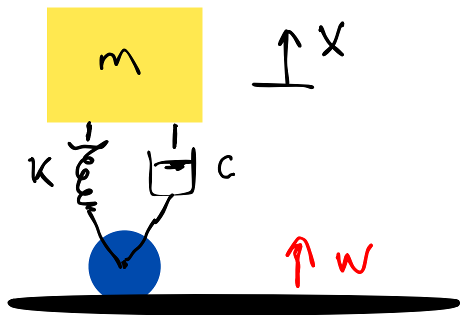


A sudden change in road height, such as driving over a bump or into a pothole.

Why do we need active control?

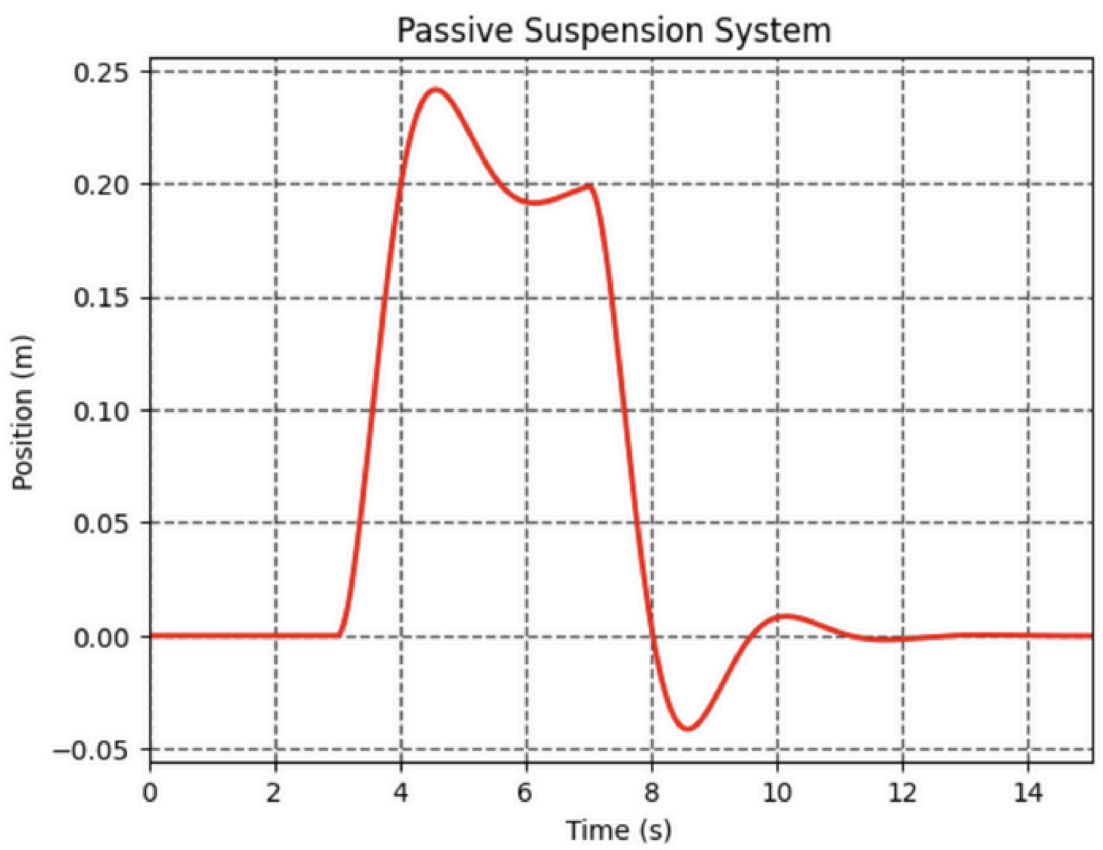
$$w(t) = 0$$

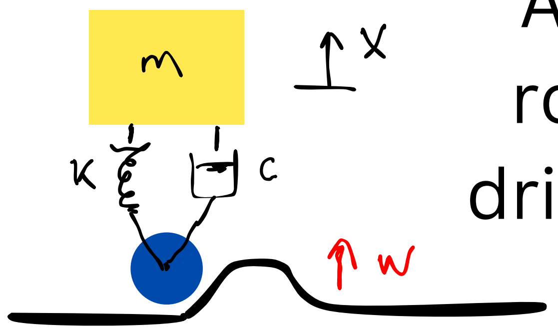




A flat road

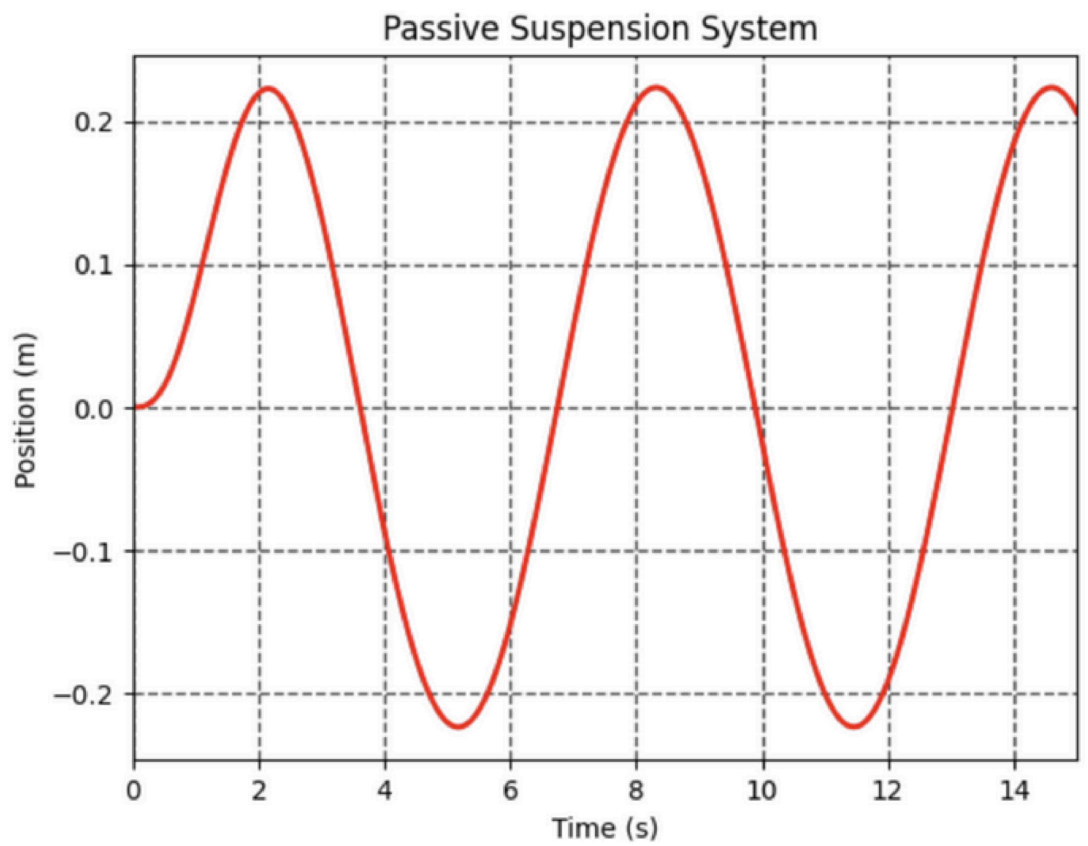
$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

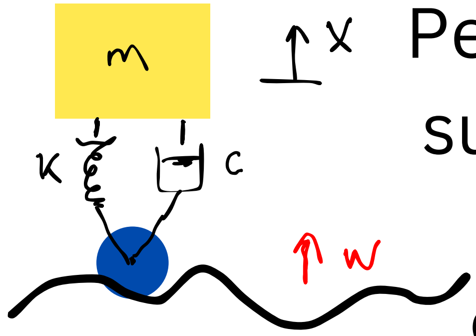




A sudden change in road height, such as driving over a bump or into a pothole.

$$w = sin(t)$$





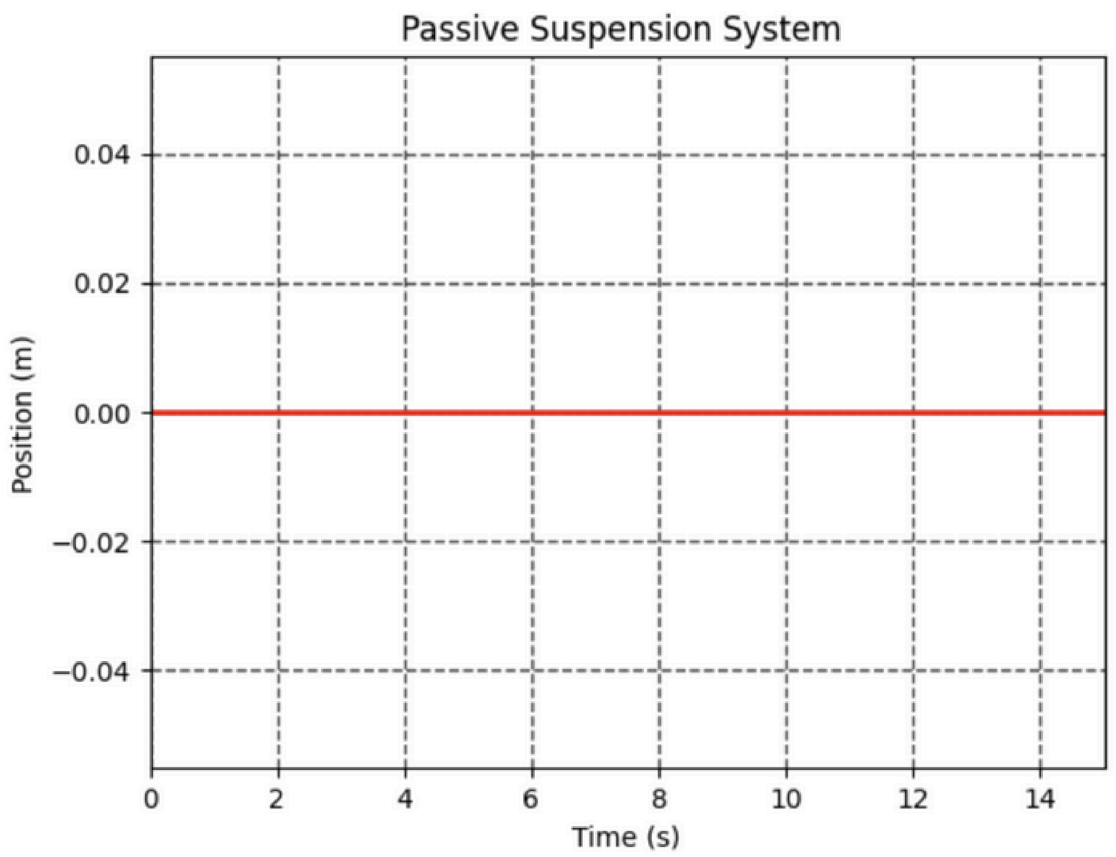
Periodic road profile, such as driving on a washboard or corrugated road.

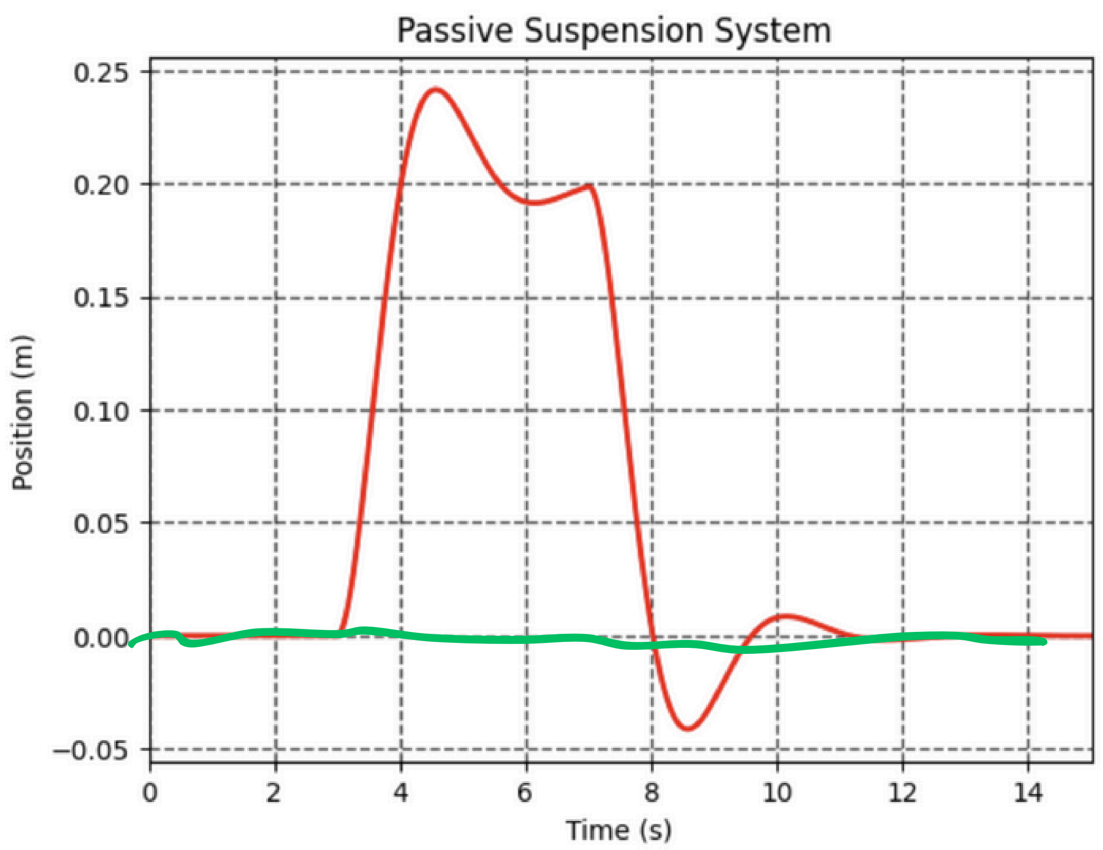
Why do we need active control?

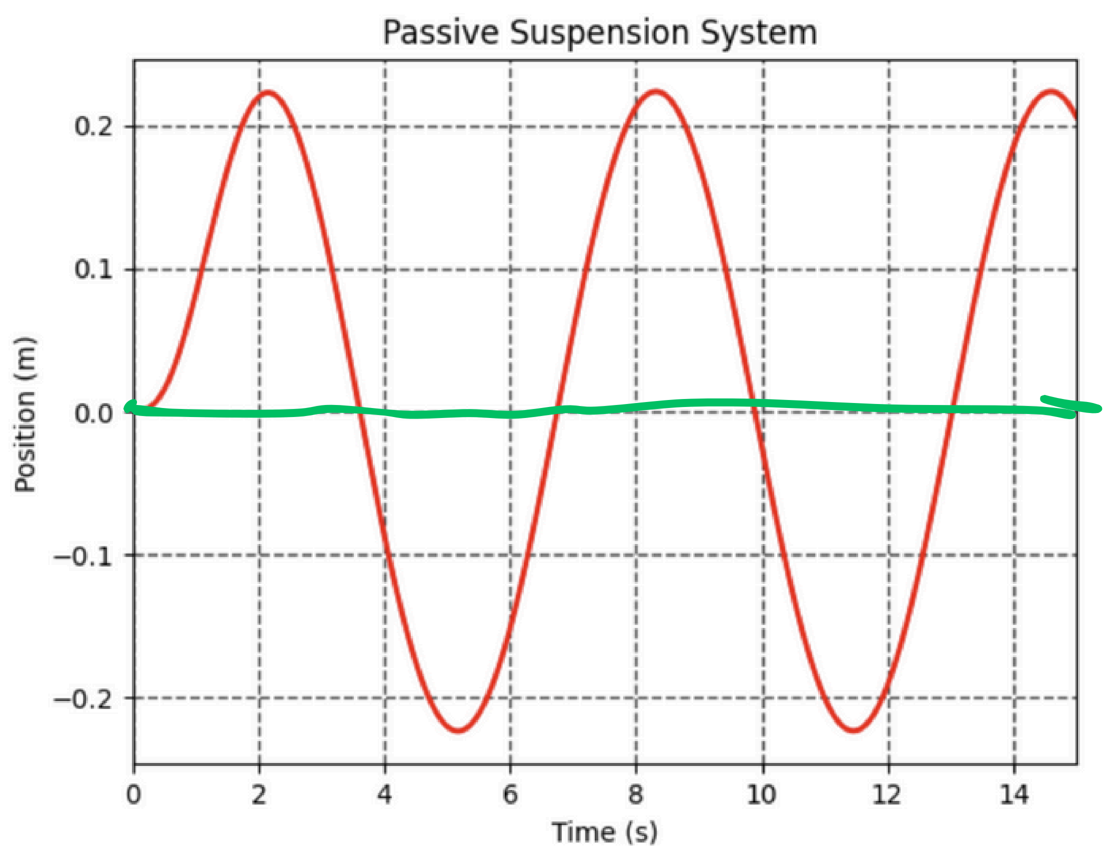
$$W(t) = 0$$

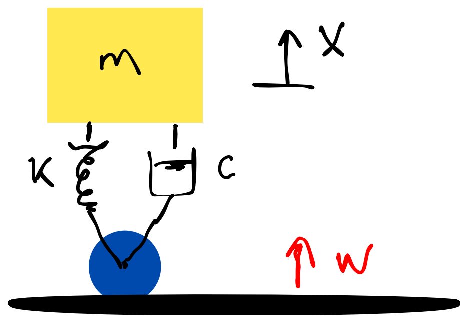
$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

$$w = sin(t)$$









A flat road

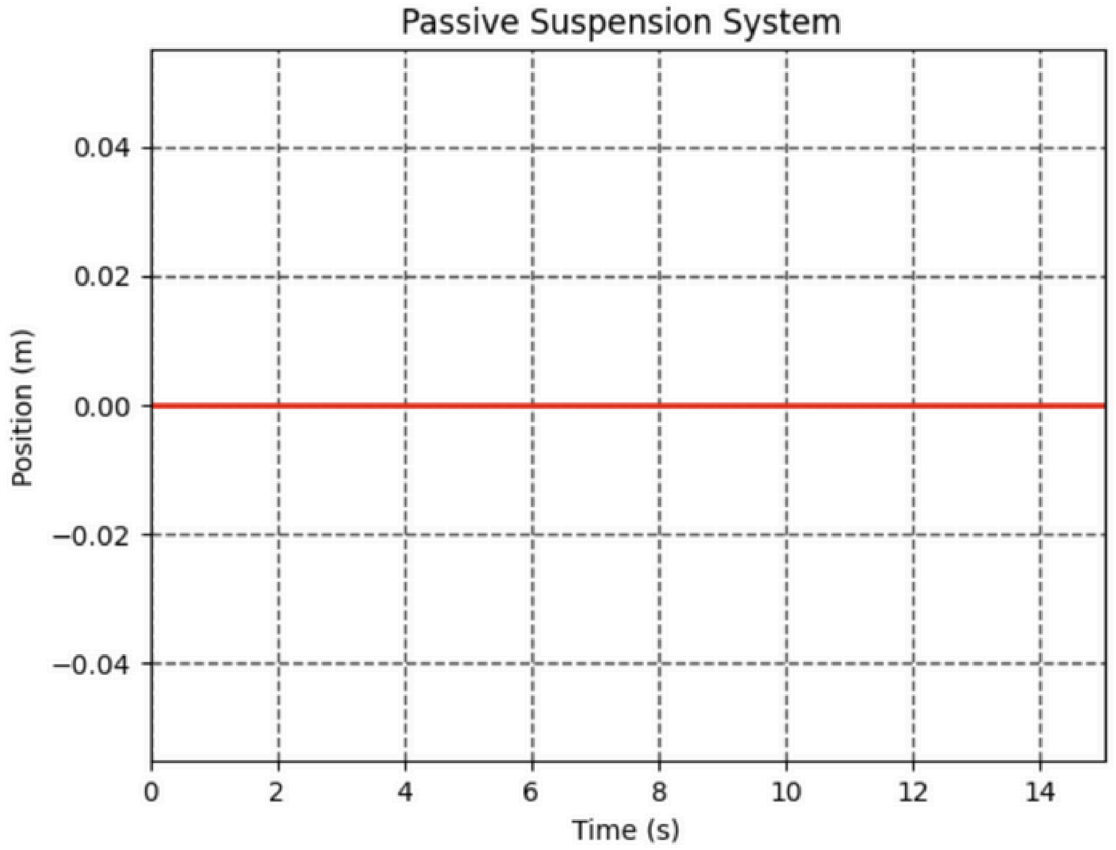
— desired trajectory

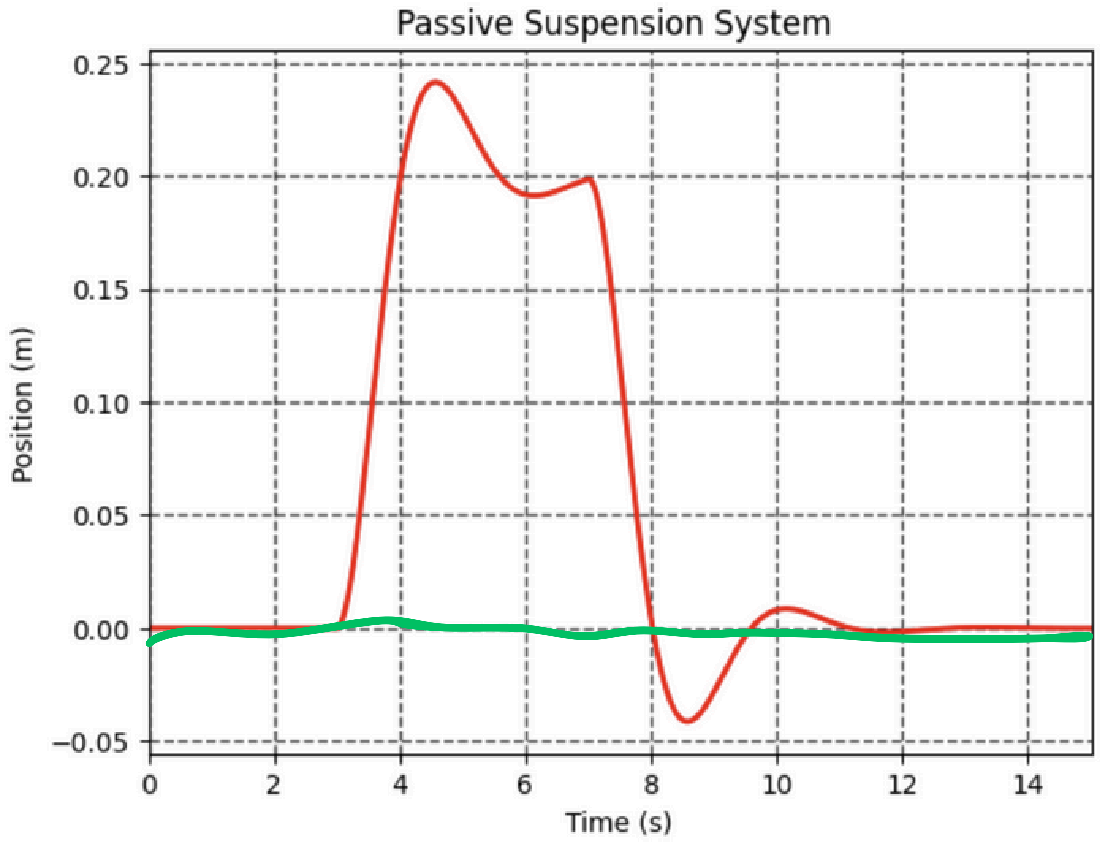
Could we solve the problem using open loop control?

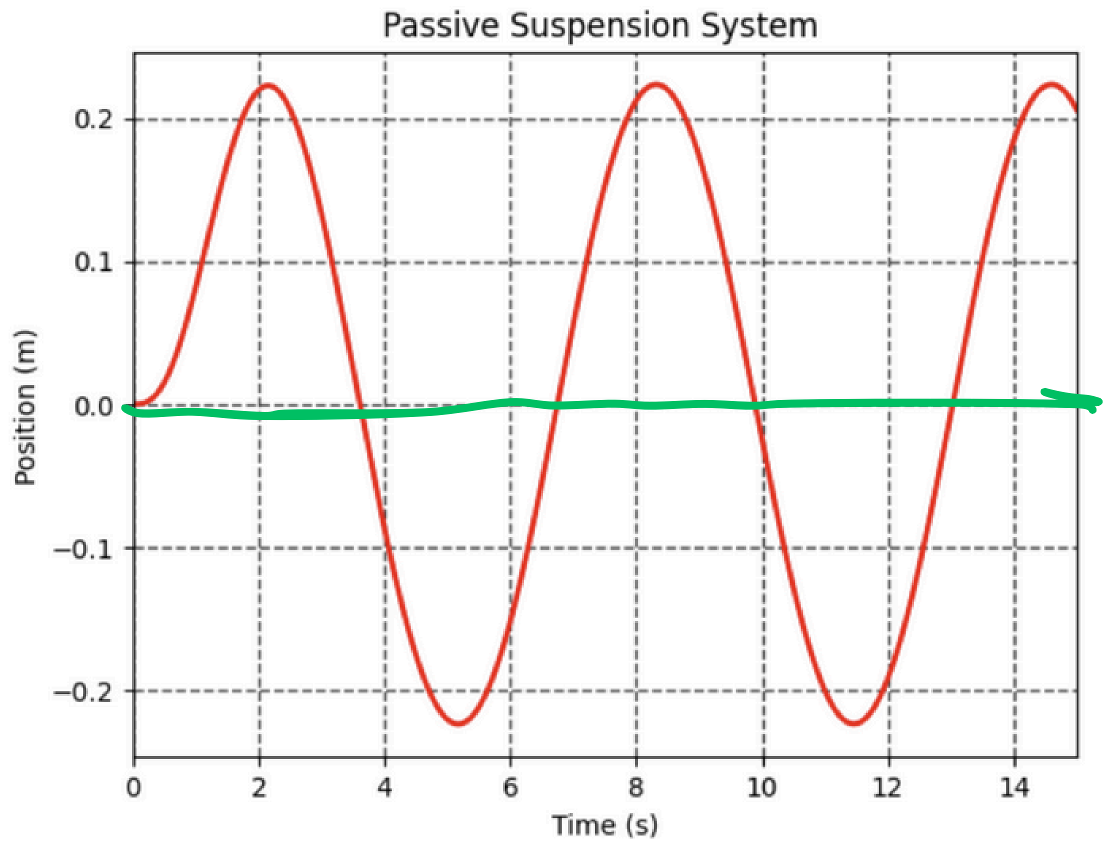
$$W(t) = 0$$

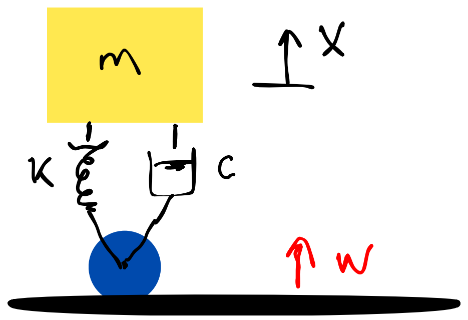
$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

$$w = sin(t)$$



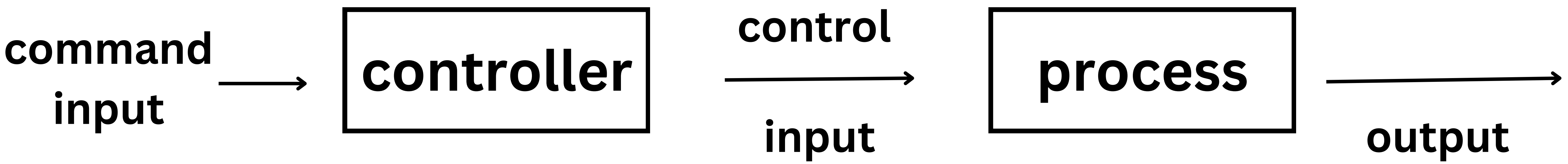






A flat road

Open-loop

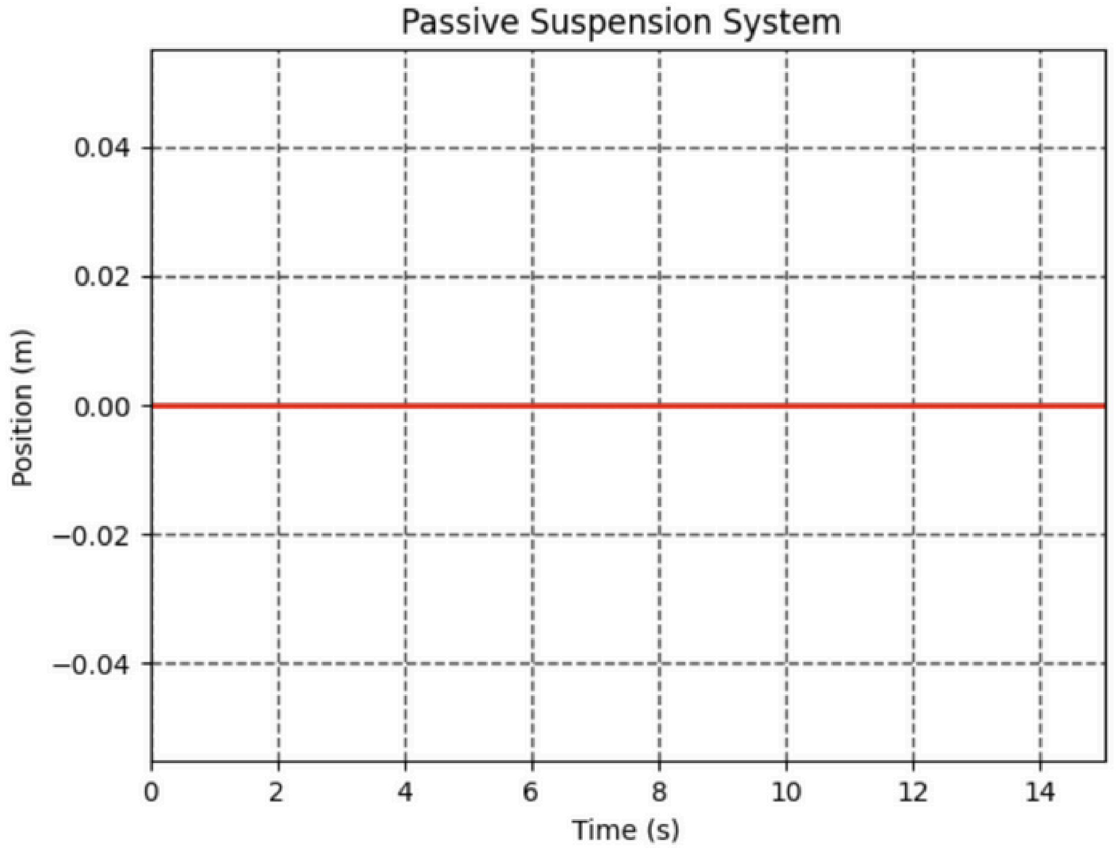


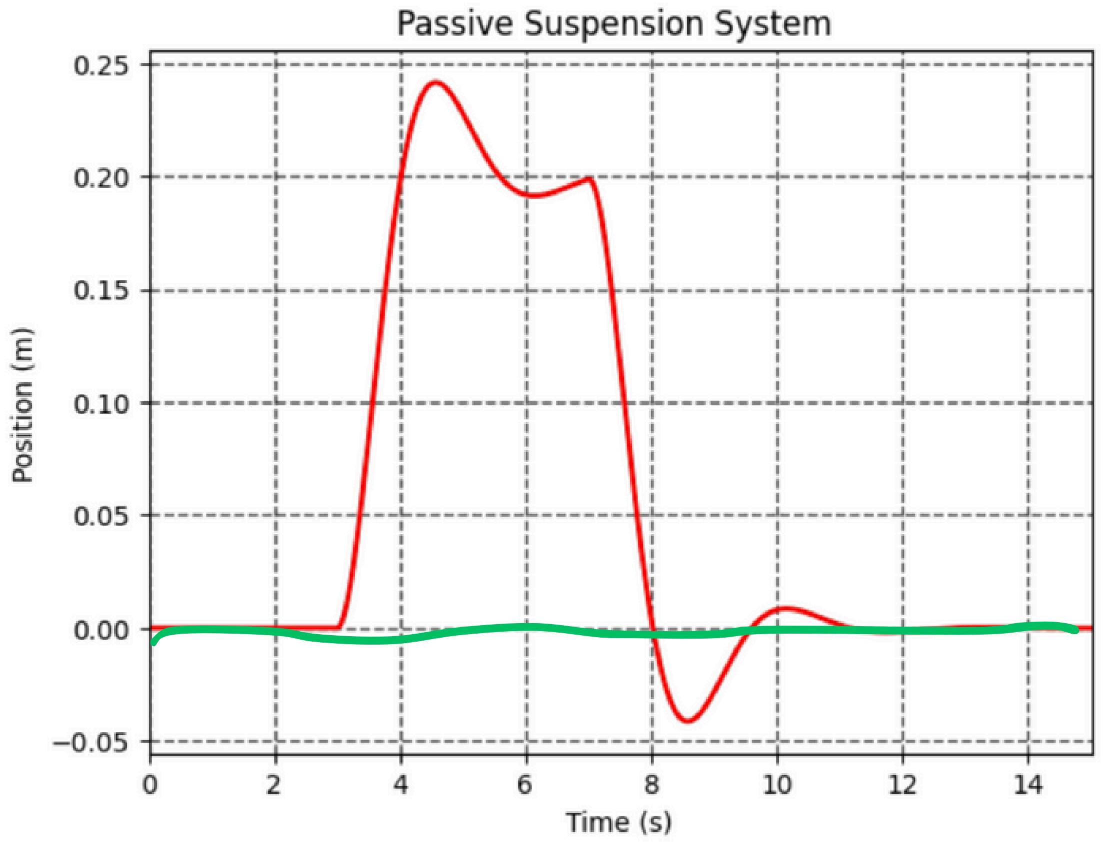
Could we solve the problem using open loop control?

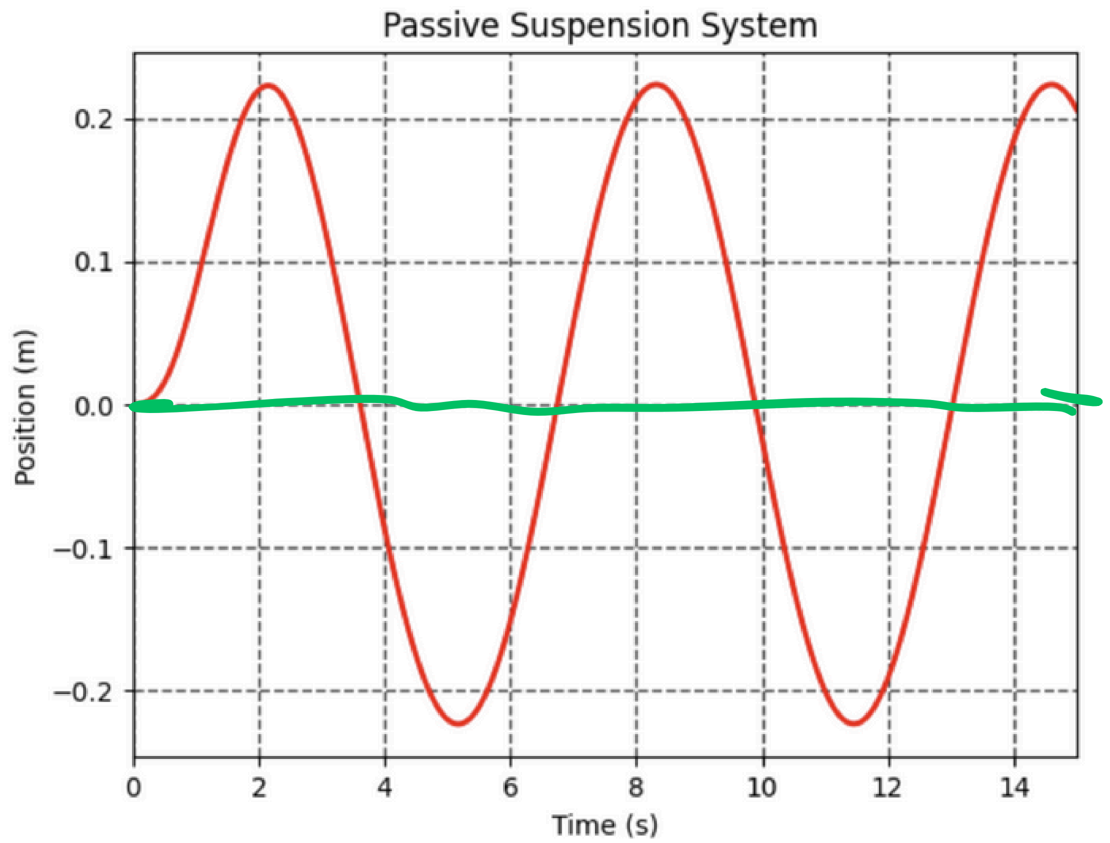
$w(t) = 0$

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

$$w = sin(t)$$







$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} w + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u$$

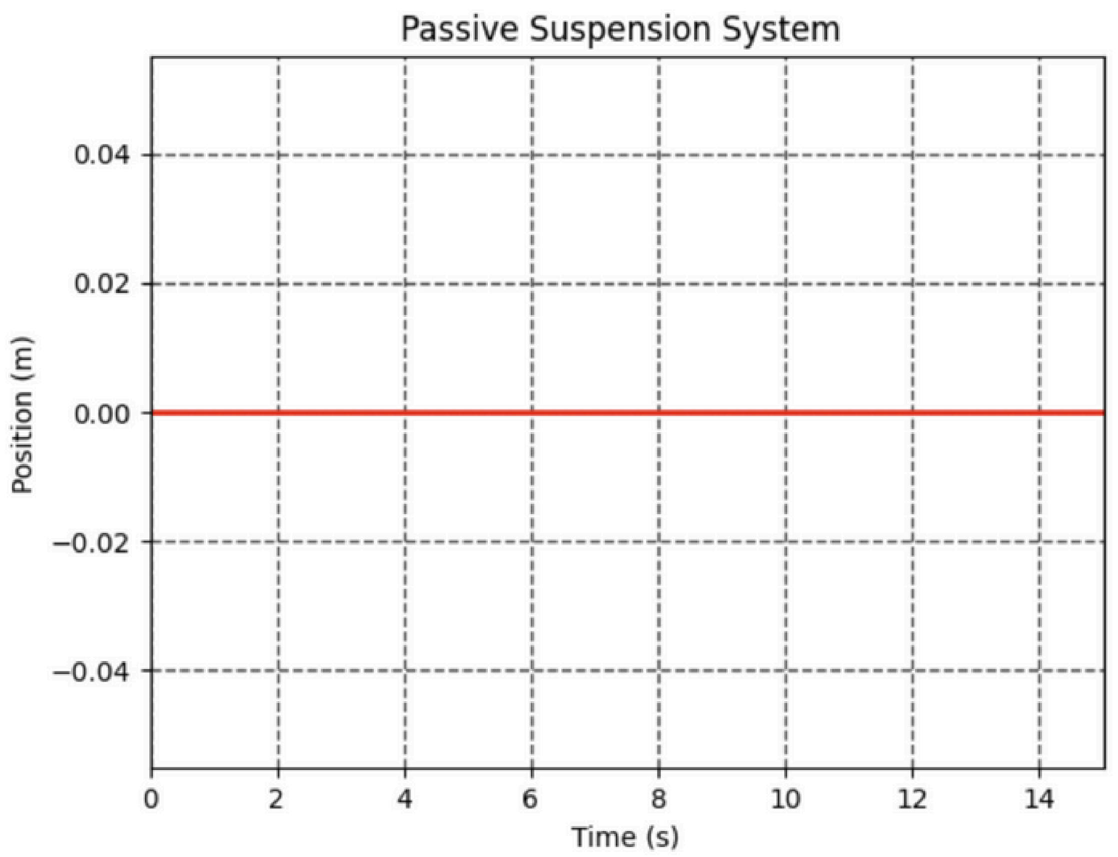
Good candidate to compensate disturbance is u(t) = -w(t)

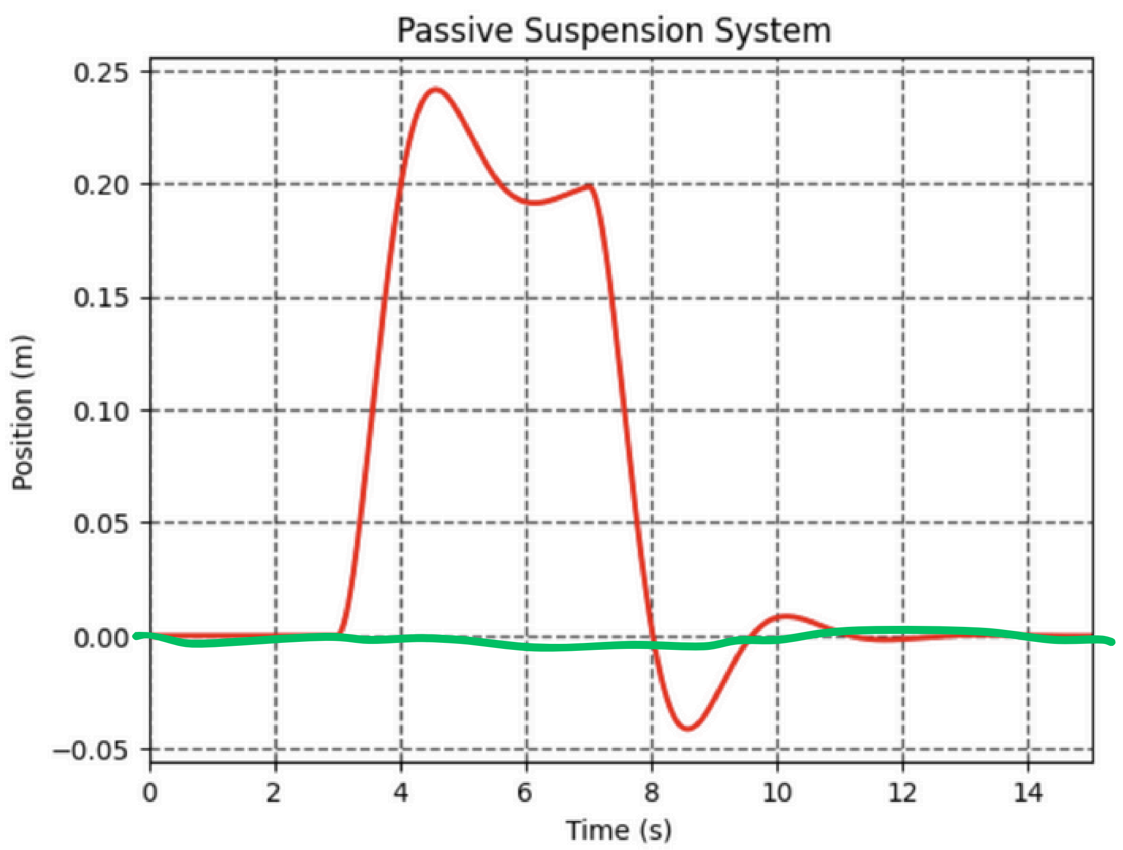
Could we solve the problem using open loop control?

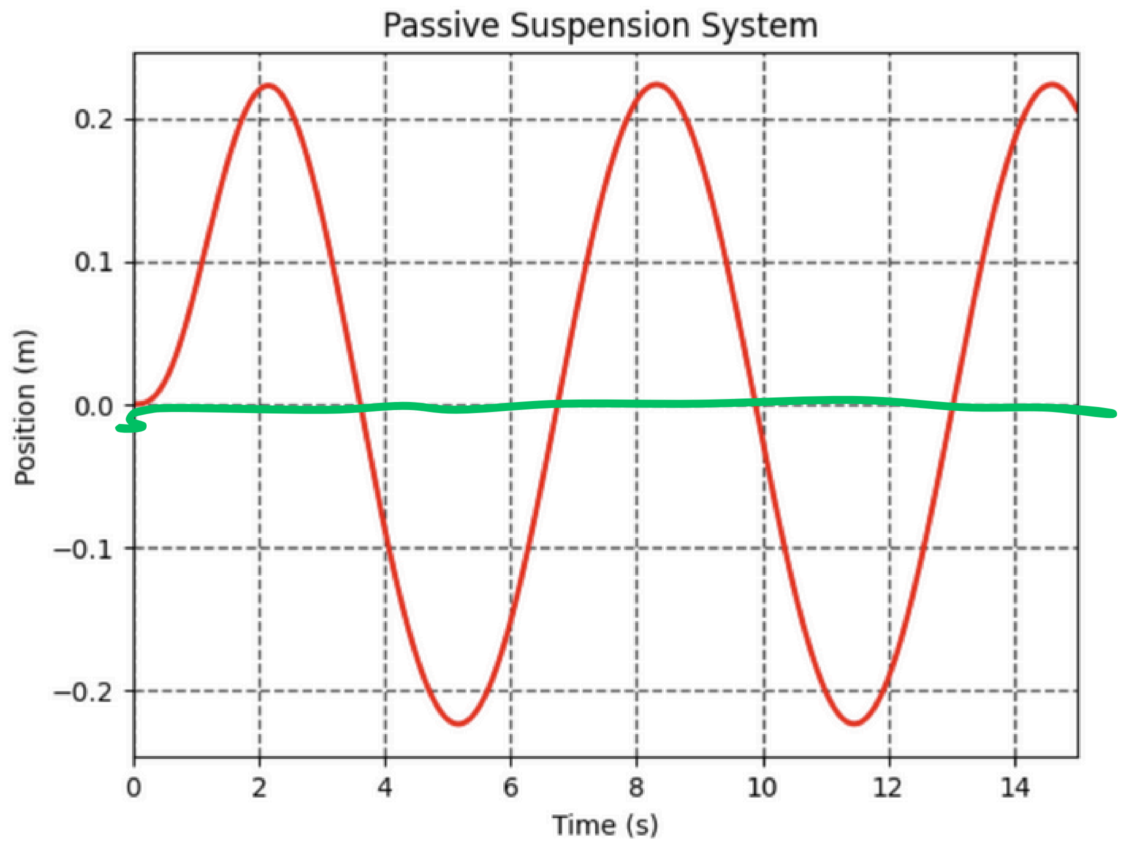
$w(t) = 0$

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

$w = sin(t)$







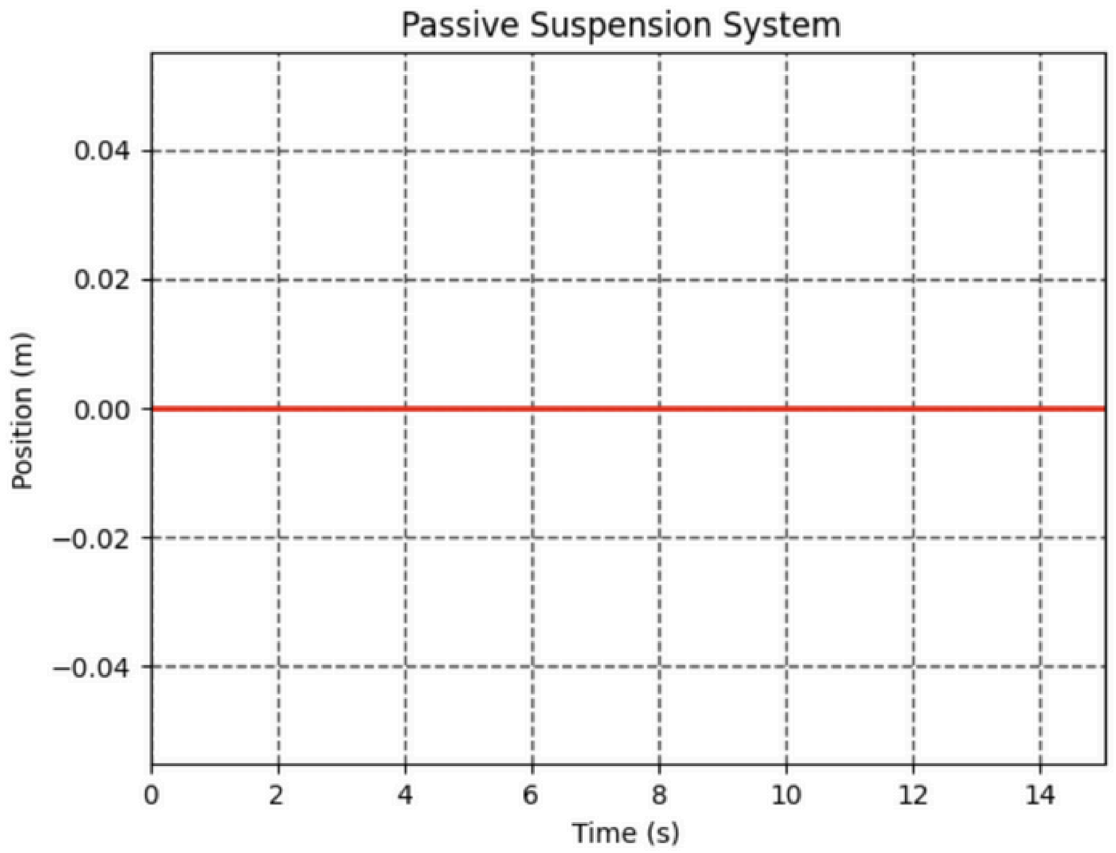
We don't have prior knowledge about the disturbance w(t), and we do not measure it.

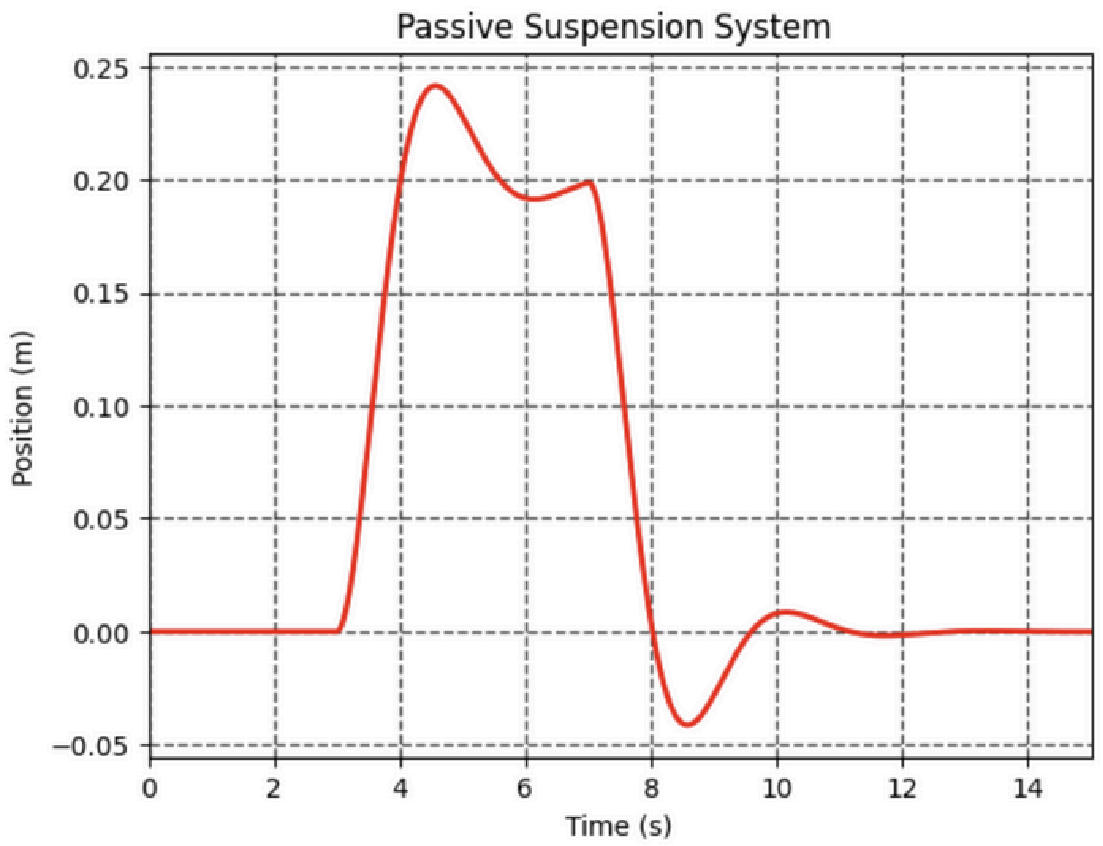
Could we solve the problem using open loop control?

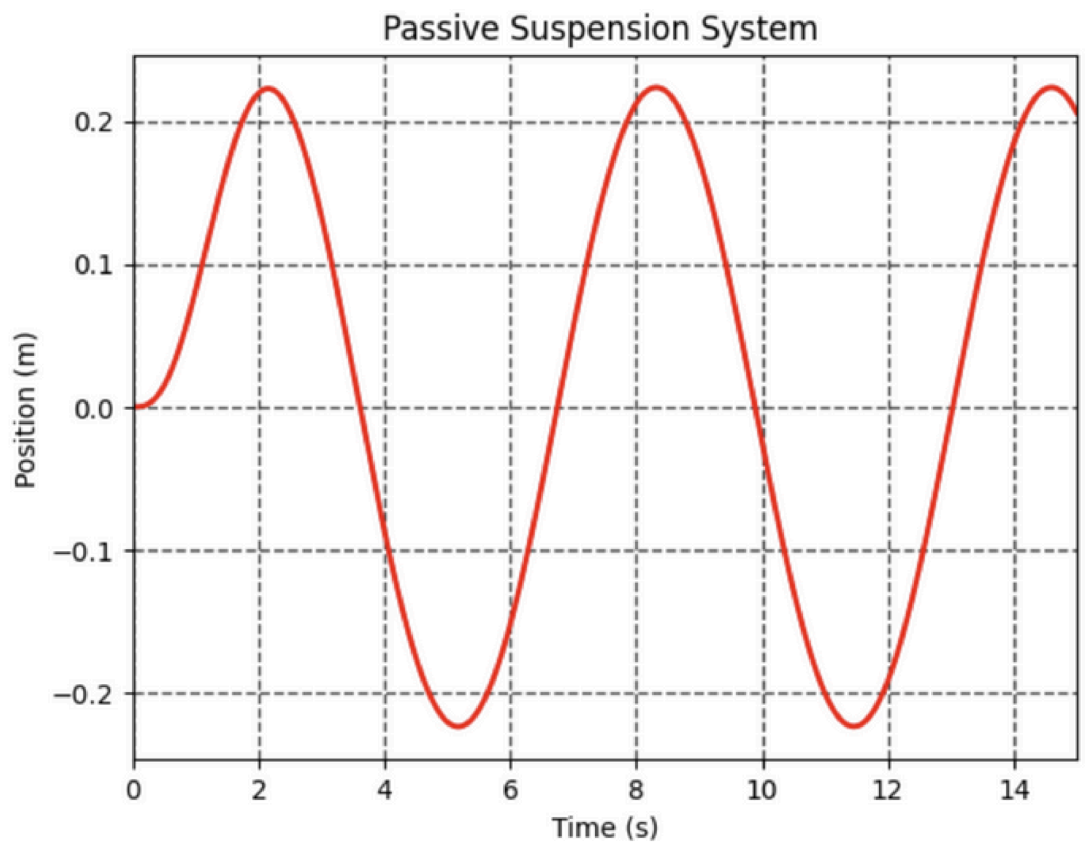
$w(t) = 0$

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

$$w = sin(t)$$



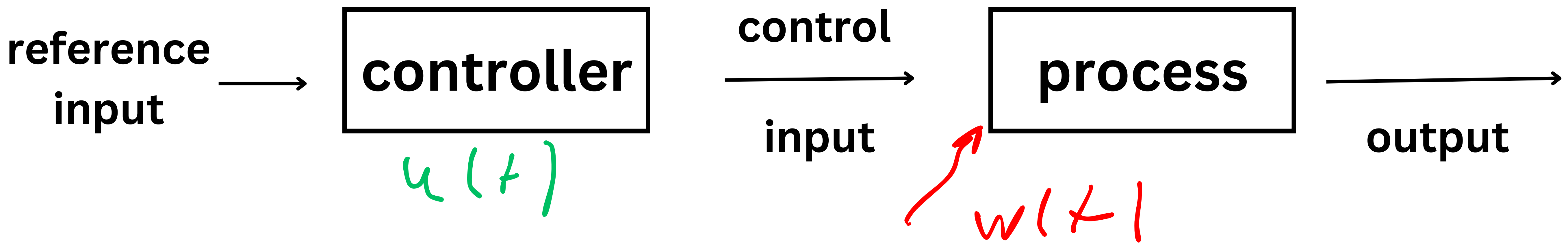




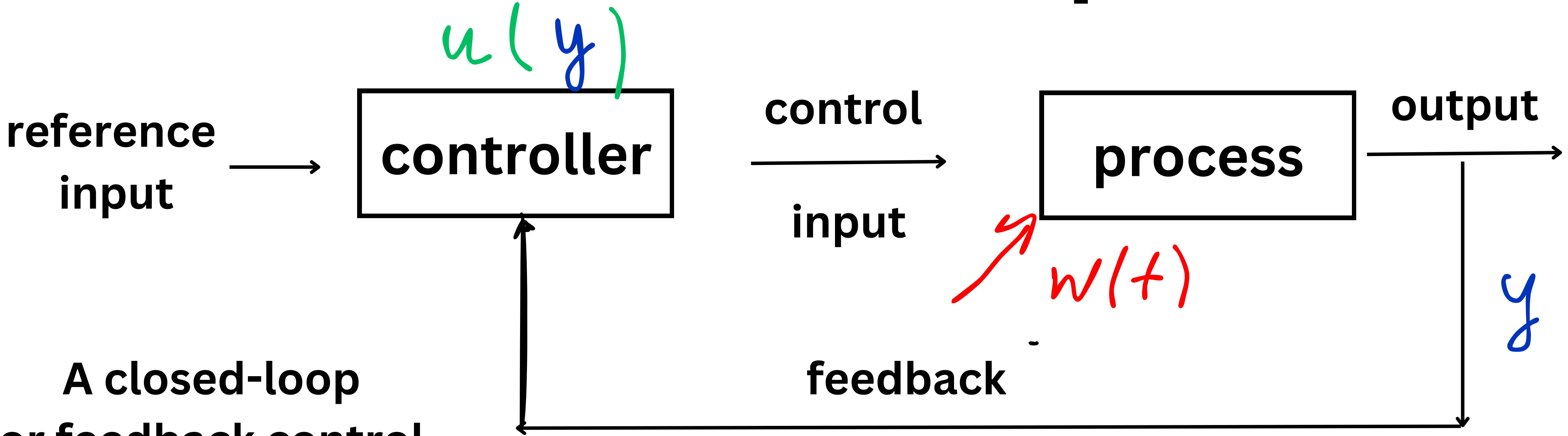
We want to design controller robust to any disturbance w(t)...

Let us use feedback to adjust for changes in the process!!!

Open-loop

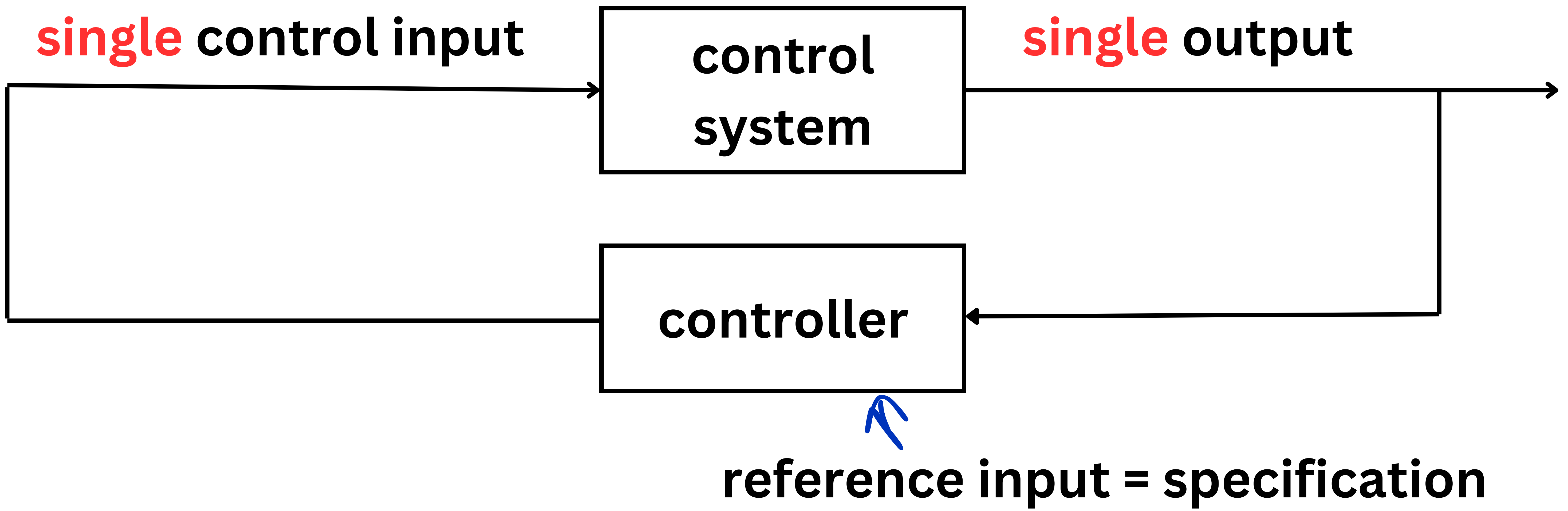


vs Closed-loop



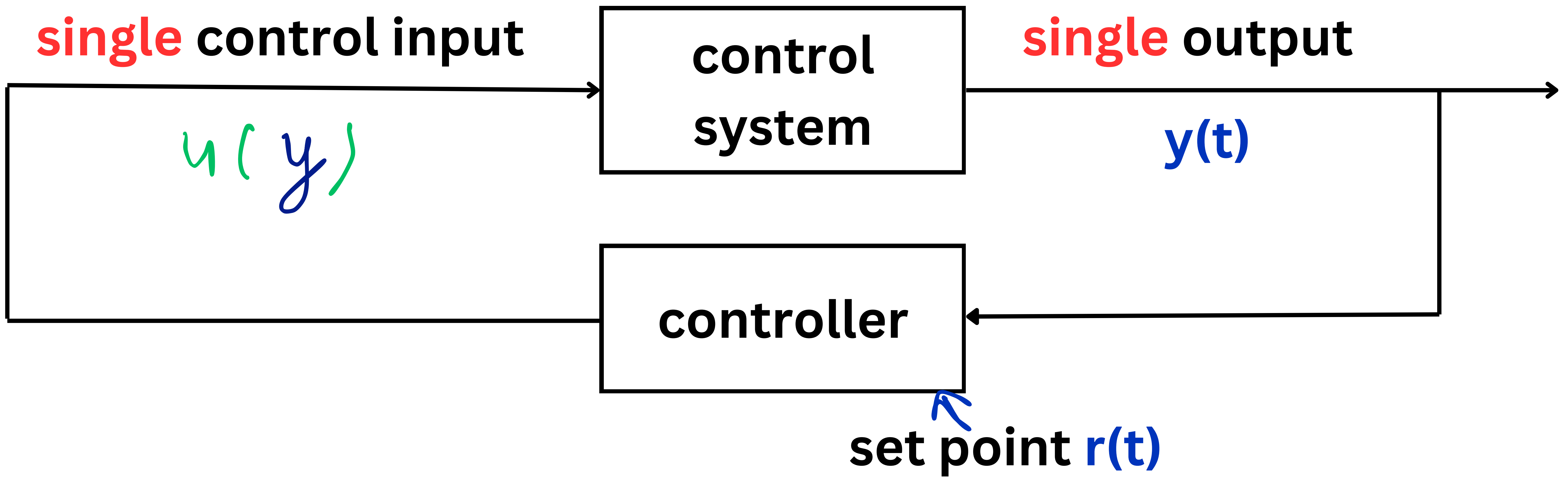
A closed-loop
or feedback control
system

SISO system feedback control



Feedback controller design = how to use the sensor data (output) to generate the correct actuator commands (control input) to ensure that the output of the system satisfies the specification

SISO system feedback control



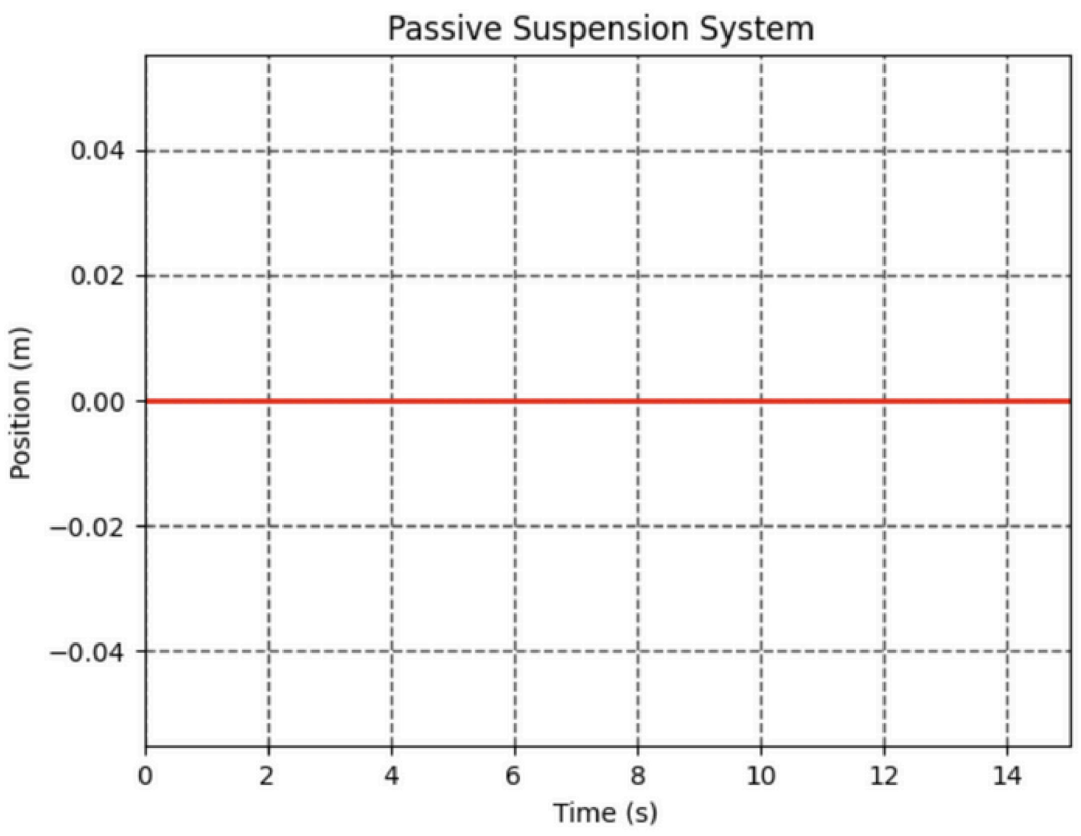
Specification: reference input (set point) $r(t)$

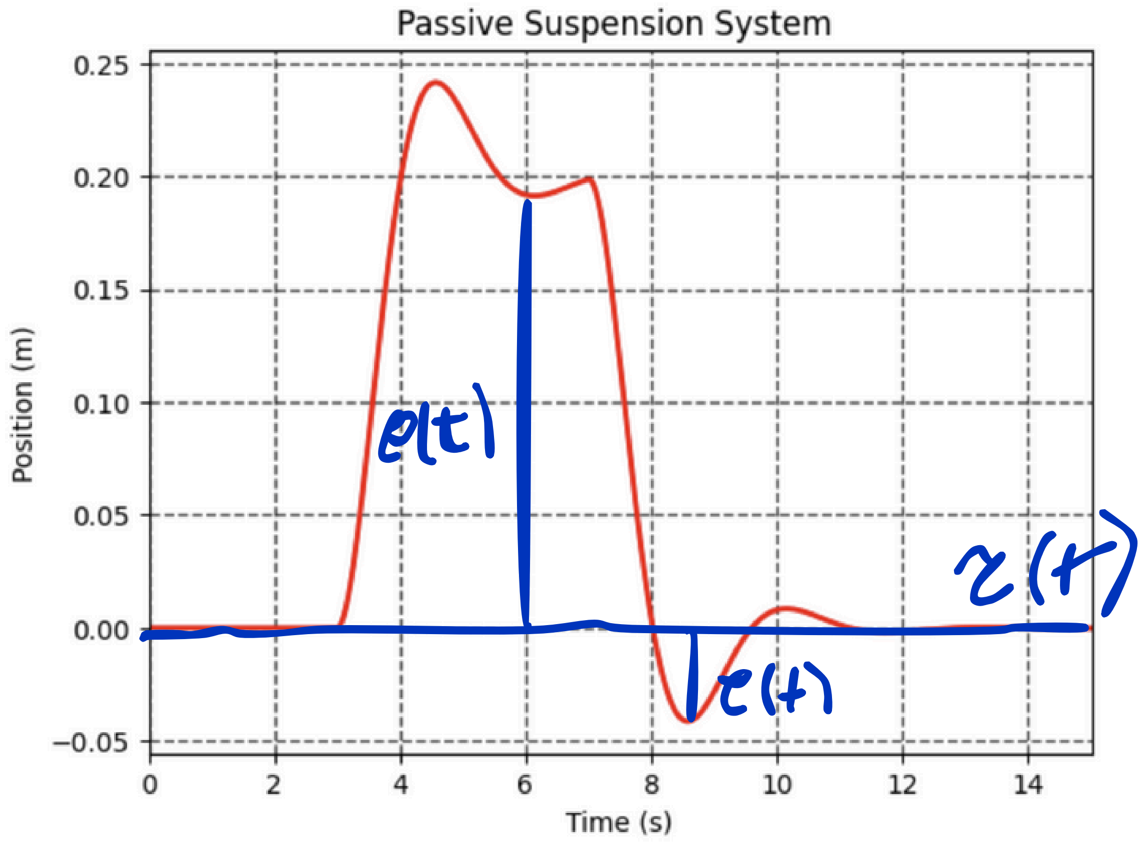
Feedback controller design: generate control input $u(y(t))$ such that the output of the system $y(t)$ coincides with $r(t)$

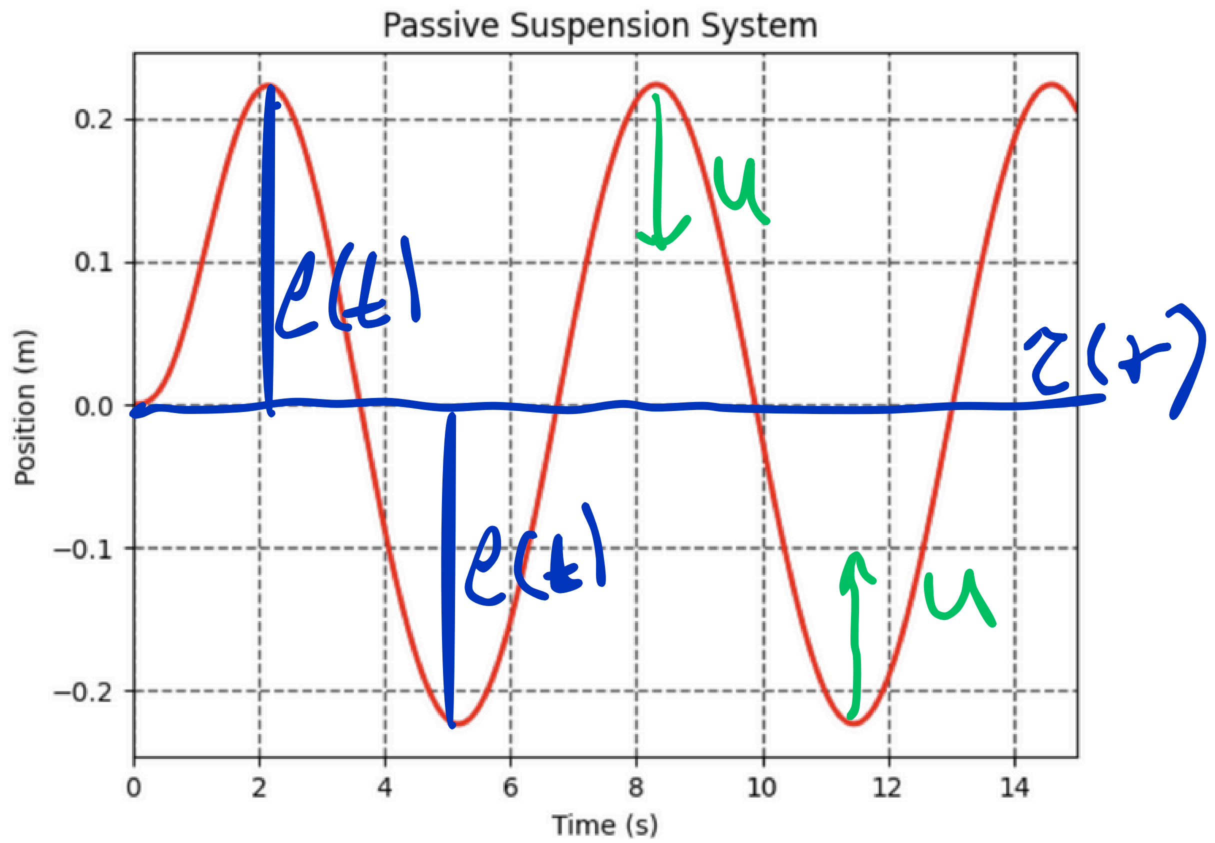
Bang - Bang controller

We want to design controller robust to any disturbance w(t)...

Let us use feedback to adjust for changes in the process!!!







Push back, against the direction of the error

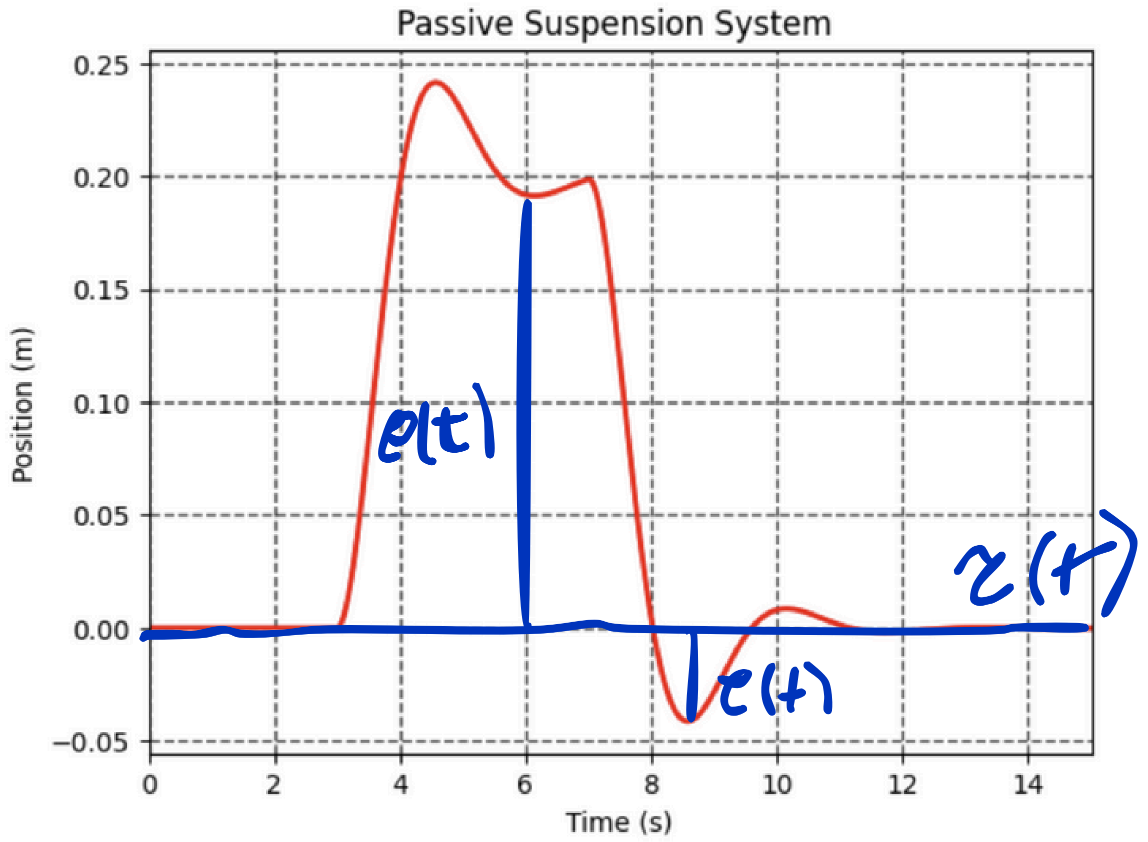
$e(t) = r(t) - y(t)$ with constant action u.

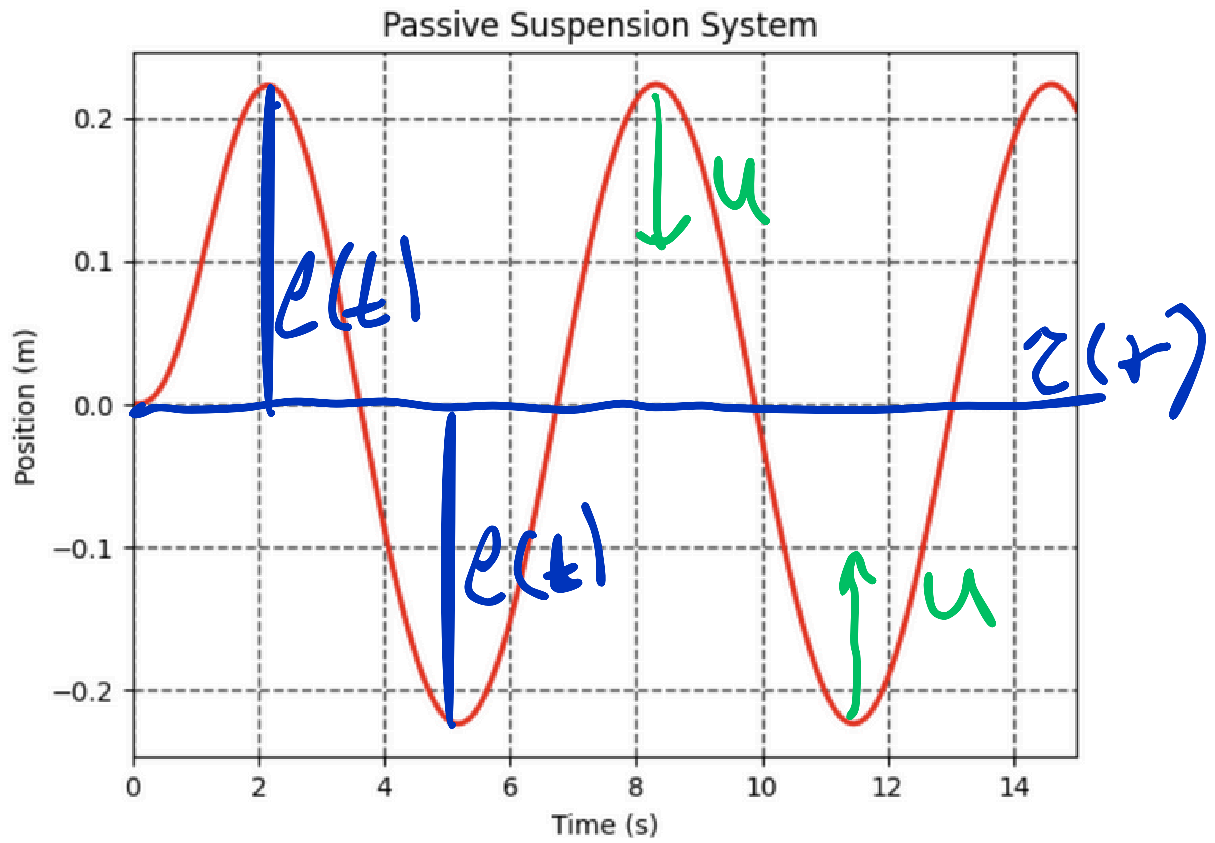
Bang - Bang controller

We want to design controller robust to any disturbance w(t)...

Let us use feedback to adjust for changes in the process!!!

if $e(t) > 0: u(t) = u$
if $e(t) < 0: u(t) = -u$
if $e(t) = 0: u(t) = 0$





Push back, against the direction of the error
e(t) = r(t) - y(t) with constant action u.

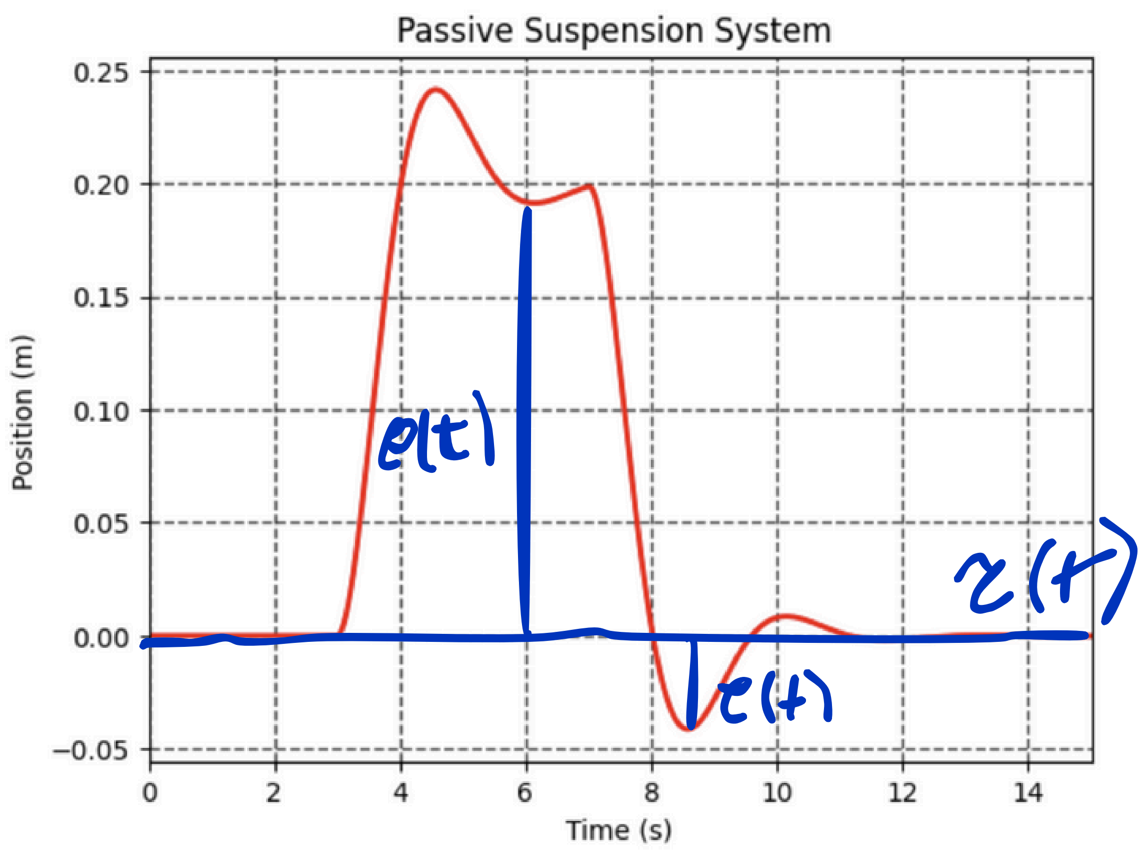
Bang - Bang controller

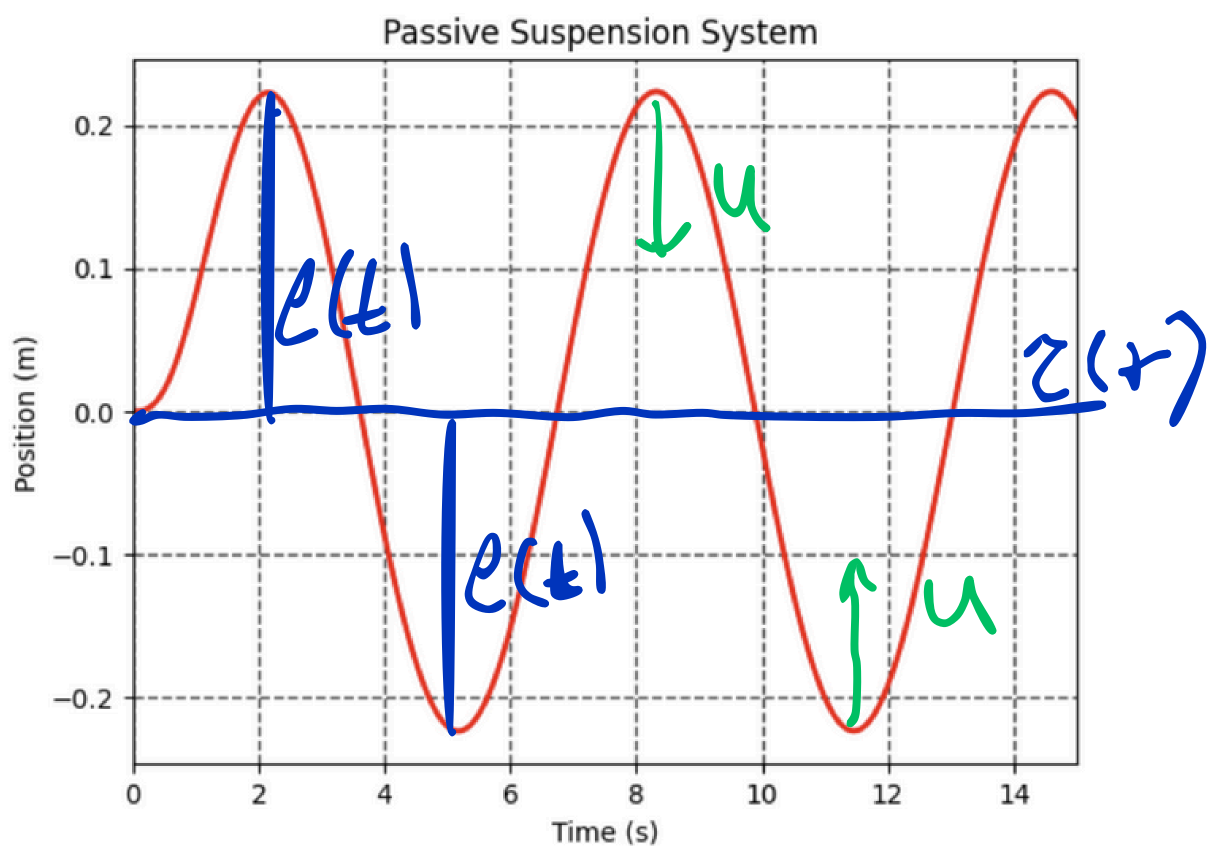
We want to design controller robust to any disturbance w(t)...

Let us use feedback to adjust for changes in the process!!!

Update every t_delay seconds

if $e(t) > 0$: $u(t) = u$
if $e(t) < 0$: $u(t) = -u$
if $e(t) = 0$: $u(t) = 0$





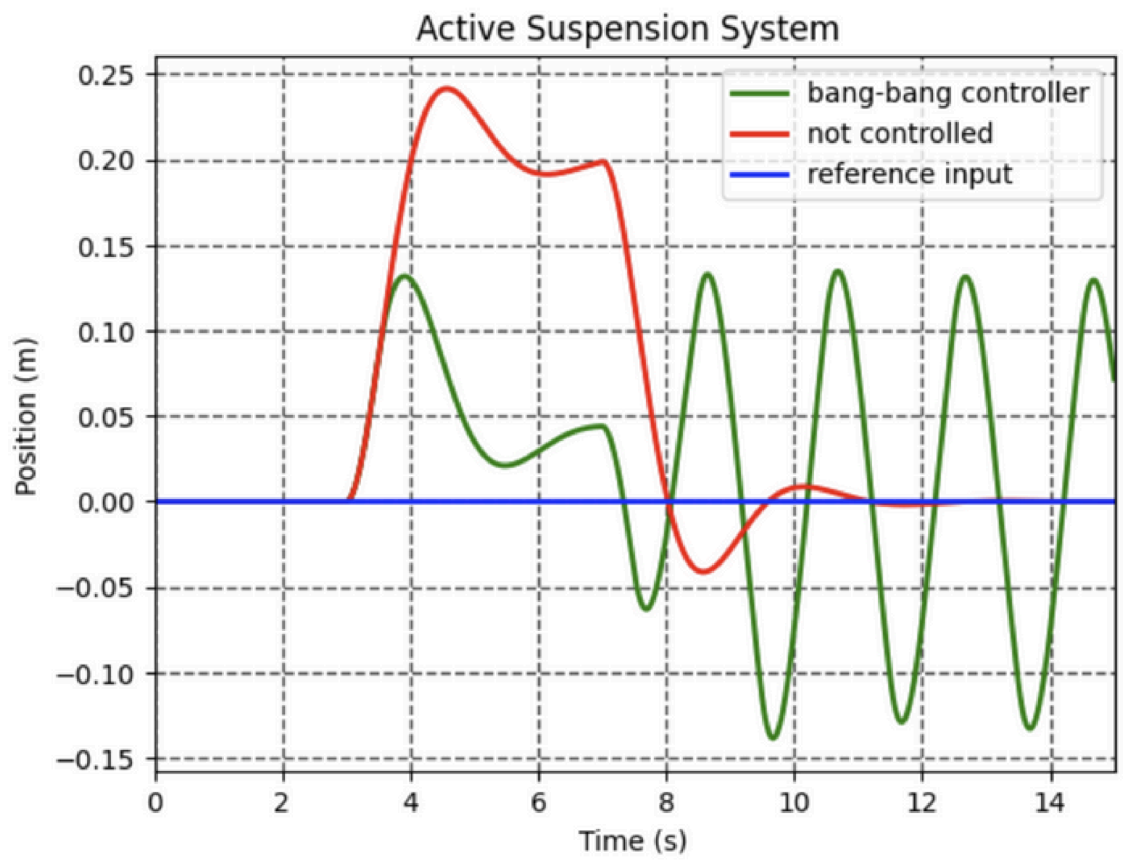
Push back, against the direction of the error
$e(t) = r(t) - y(t)$ with constant action u.
Don't switch more often than t_delay seconds

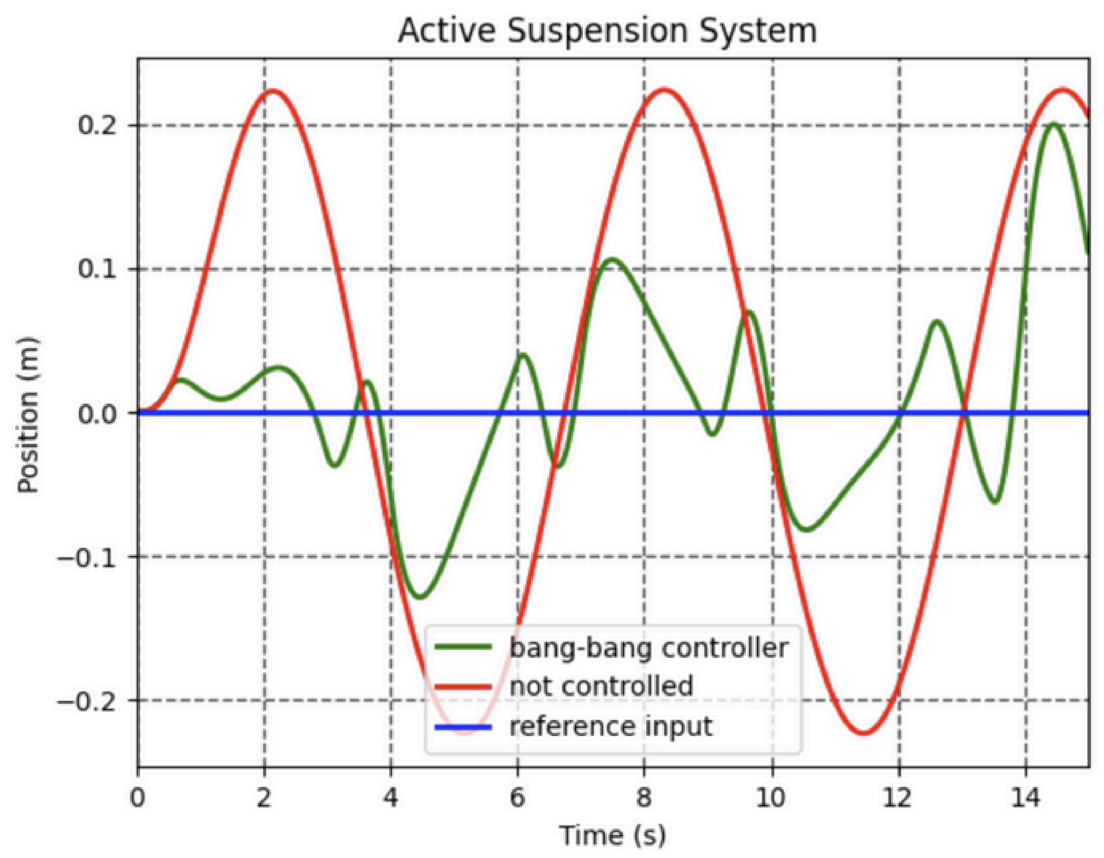
Bang - Bang controller

Since the control is binary, the system cannot achieve fine control over the output. The controller often overreacts, leading to overshoot or undershoot of the desired value.

Update every t_delay seconds

if $e(t) > 0$: $u(t) = u$
if $e(t) < 0$: $u(t) = -u$
if $e(t) = 0$: $u(t) = 0$

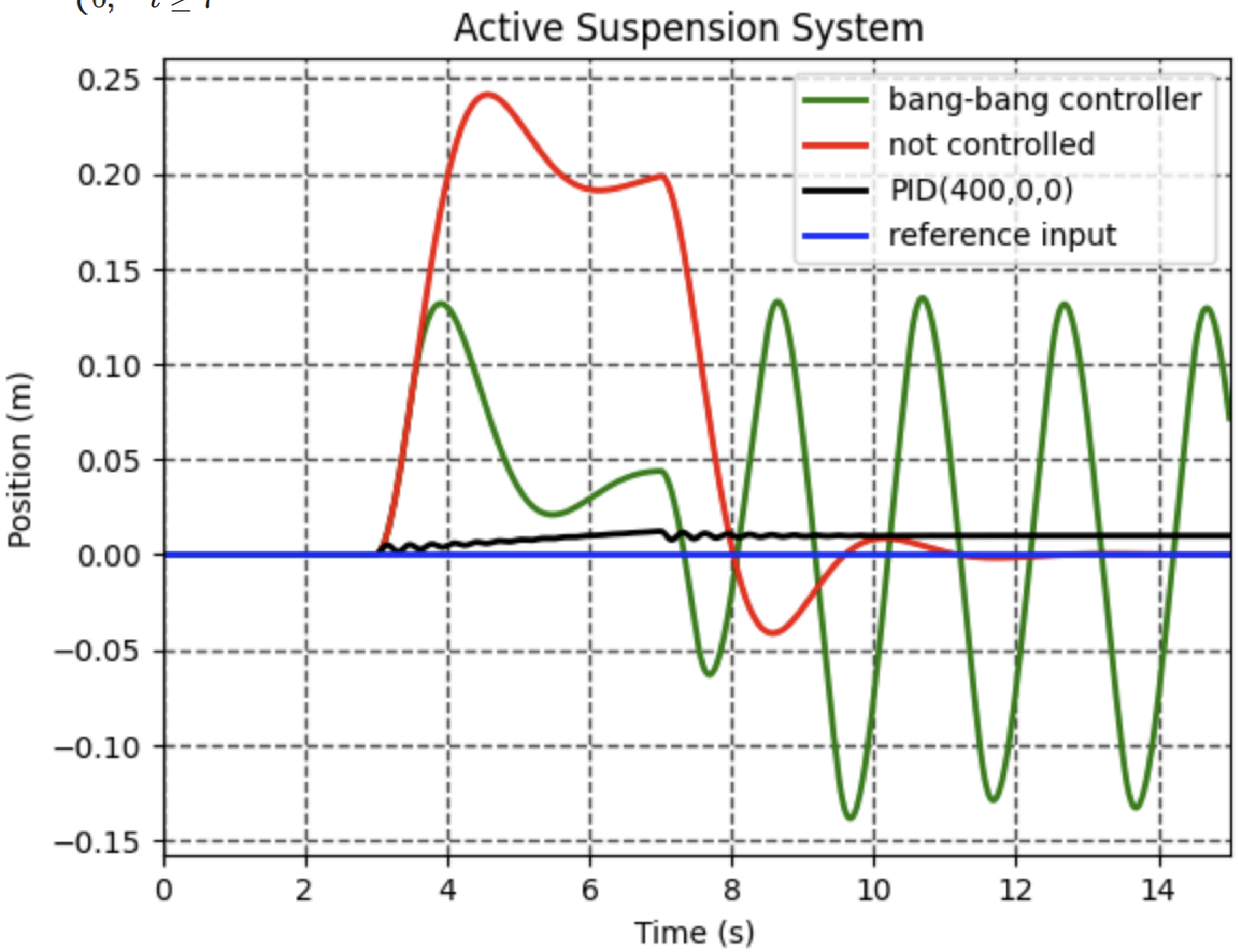




Push back, against the direction of the error $e(t) = r(t) - y(t)$ with constant action u.
Don't switch more often than t_delay seconds

P - Proportional Controller

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

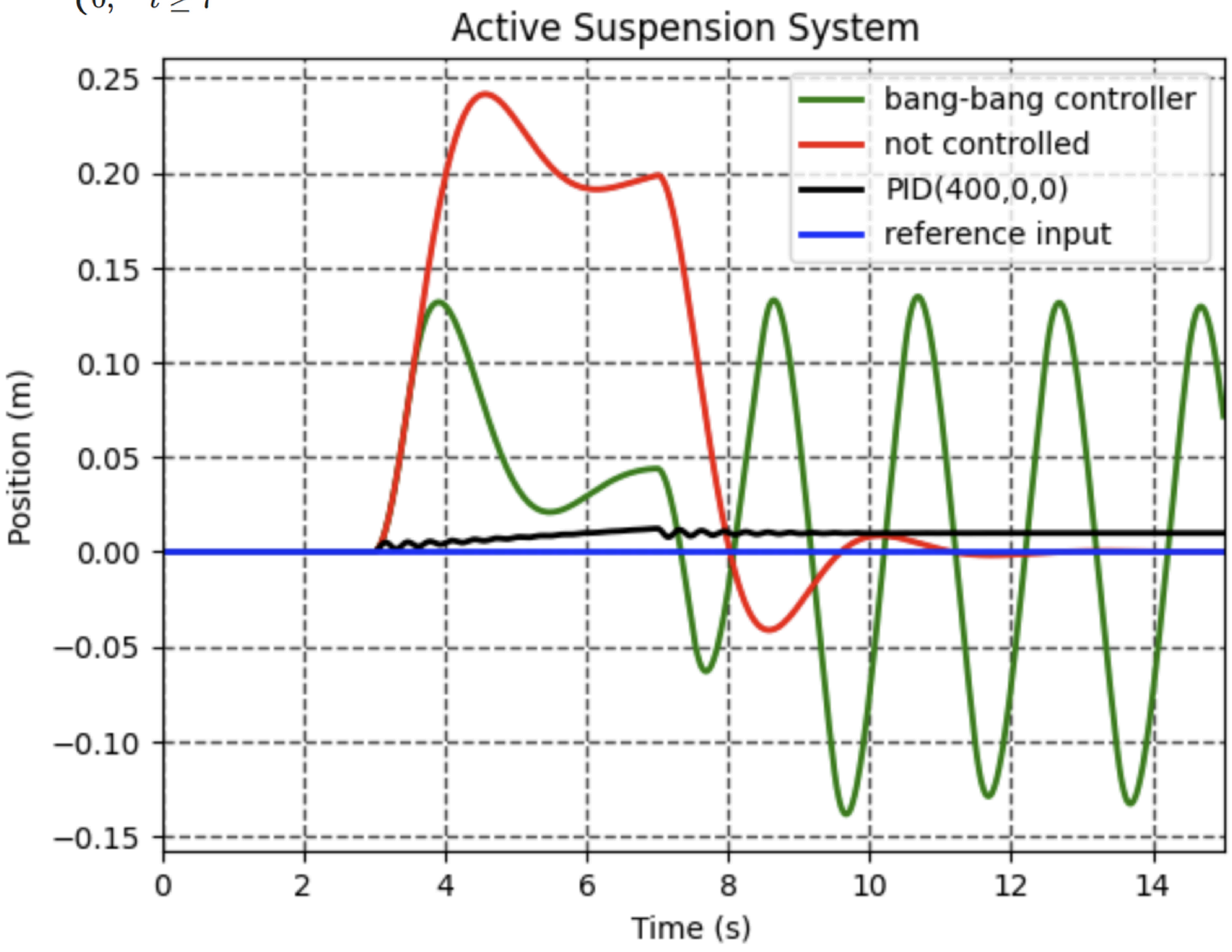


$$u(t) = K_p(r(t) - y(t))$$

The control input tries to move the system in a direction that is opposite to the error, and is proportional to the error in magnitude.

P - Proportional Controller

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$



$$u(t) = K_p(r(t) - y(t))$$

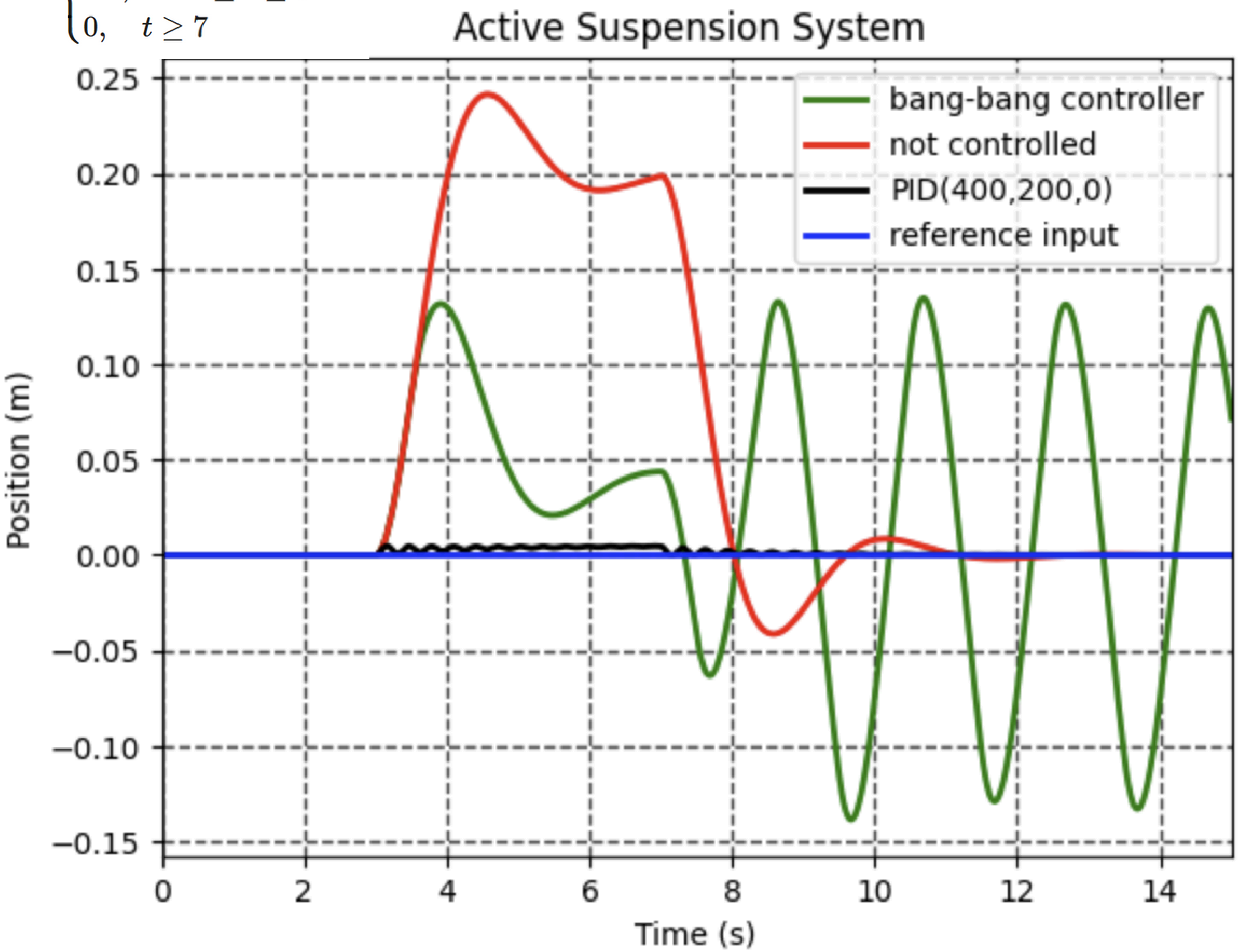
The control input tries to move the system in a direction that is opposite to the error, and is **proportional to the error** in magnitude.

Non zero steady state error

Steady state error is the deviation of the output of control system from desired response during steady state

PI - Proportional Integral Controller

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$



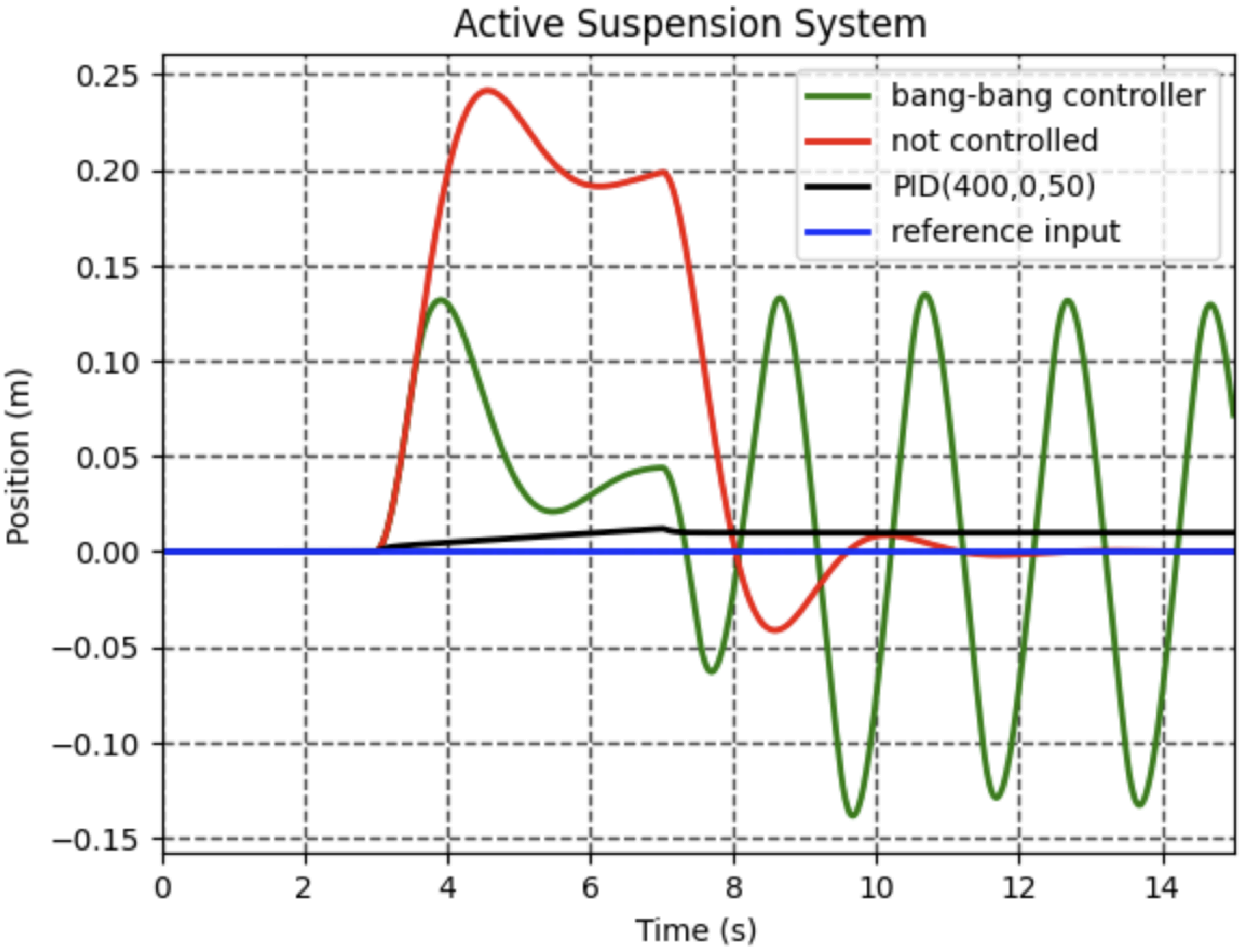
$$u(t) = K_p e(t) + \overbrace{K_i \int_0^t e(\tau)\, d\tau}^{\text{Integral term}}$$

The integral (I) term accumulates past errors over time to eliminate residual error left by proportional (P) control. As error decreases, the integral effect grows, compensating for the diminishing proportional effect. Once the error is zero, the integral term stops increasing.

for linear systems
integral term eliminates steady-state error

PD - Proportional Derivative Controller

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$



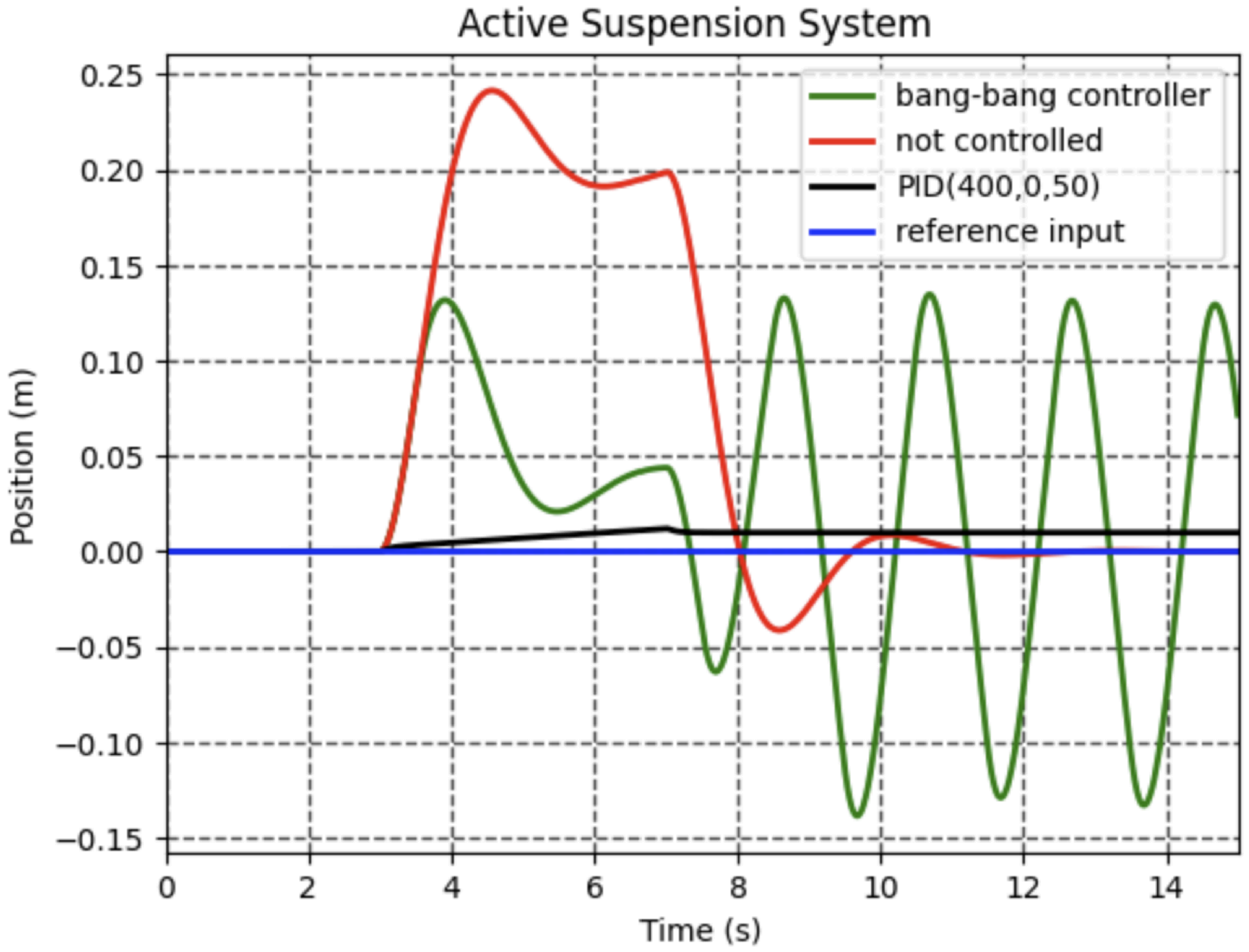
derivative term

$$u(t) = K_p e(t) + \underbrace{K_d \dot{e}(t)}$$

The derivative (D) term predicts future error changes by computing the rate of change of the error.

PD - Proportional Derivative Controller

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$



derivative term

$$u(t) = K_p e(t) + \underbrace{K_d \dot{e}(t)}$$

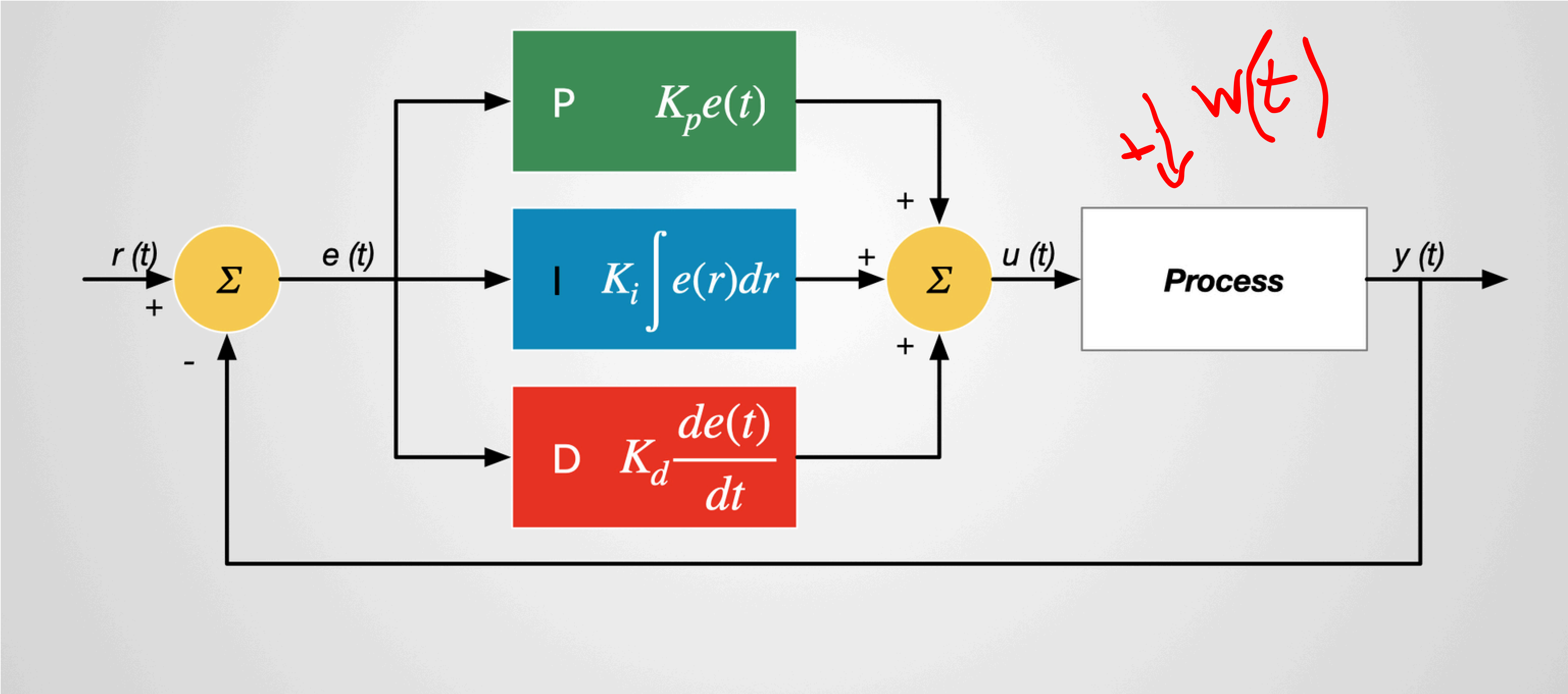
Anticipates error changes: It reacts to how fast the error is changing, rather than just its magnitude.

Improves system stability: It dampens oscillations and reduces overshoot.

Enhances response speed: It helps the system respond faster to disturbances by counteracting rapid error variations.

PID controller

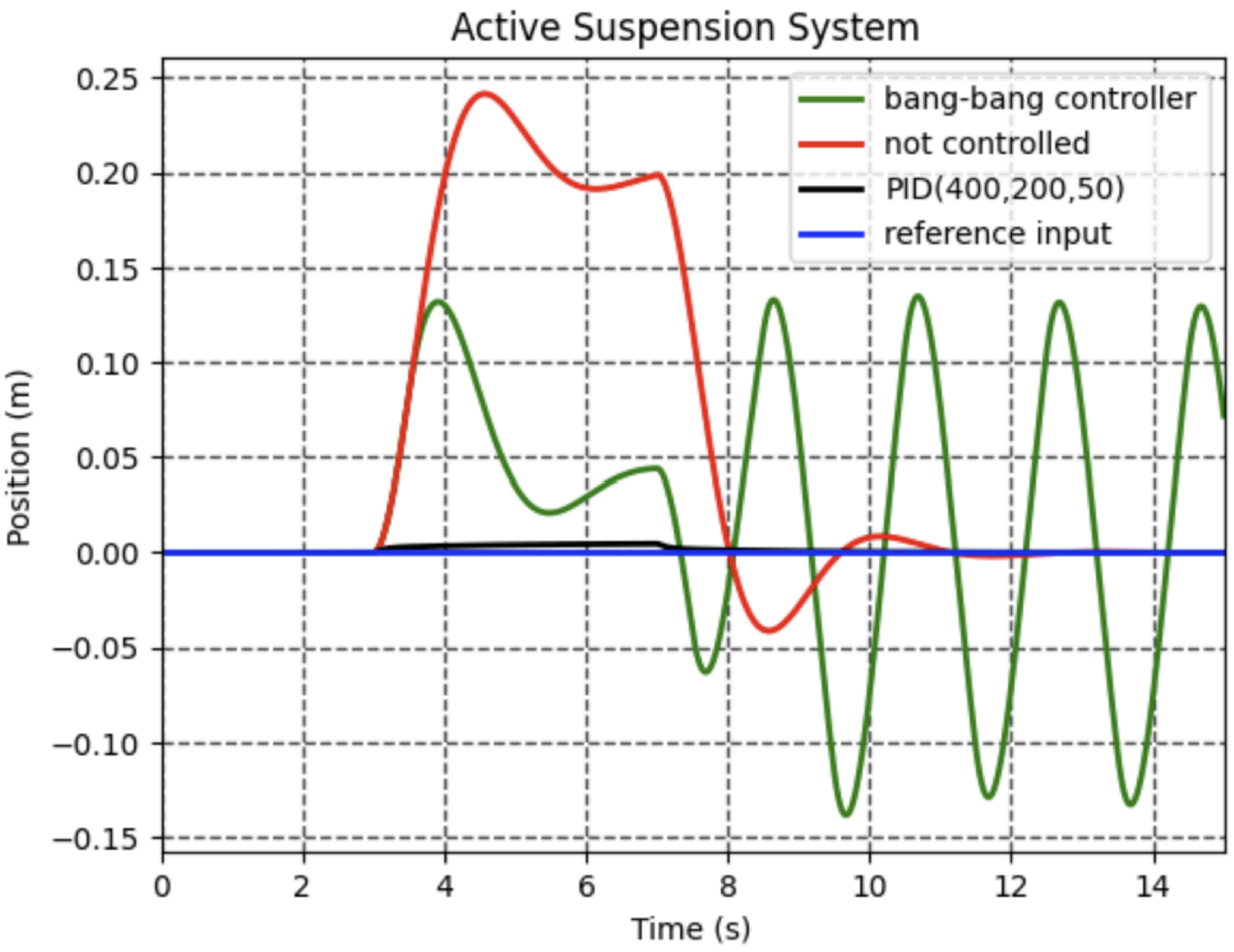
Given a reference input (set point) r(t) generate a controller u(y(t)) such y(t)->r(t), robustly to any disturbance w(t)



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$

where e(t) = r(t) - y(t)

PID controller



Active suspension system

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & \underset{c}{-\frac{\rho}{\cancel{m}}} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} w$$

Specification: r(t) = 0.0

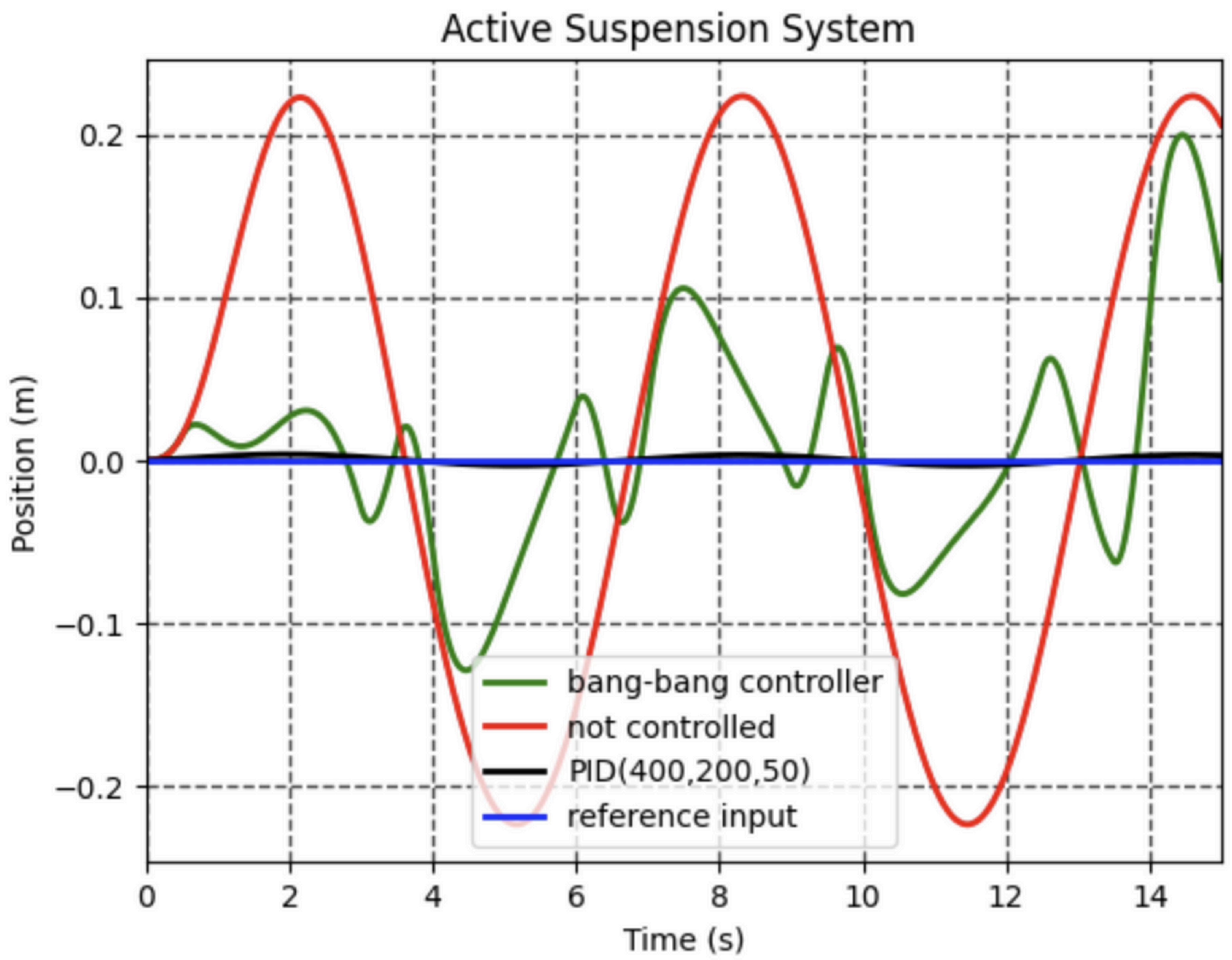
PID(Kp = 400, Ki = 200, Kd = 50)

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$

System response for a step function

$$w = \begin{cases} 0, & t \leq 3.0 \\ 1.0, & 3.0 \leq t \leq 7.0 \\ 0, & t \geq 7 \end{cases}$$

PID controller



Active suspension system

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{\rho}{m} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} w$$

Specification: r(t) = 0.0

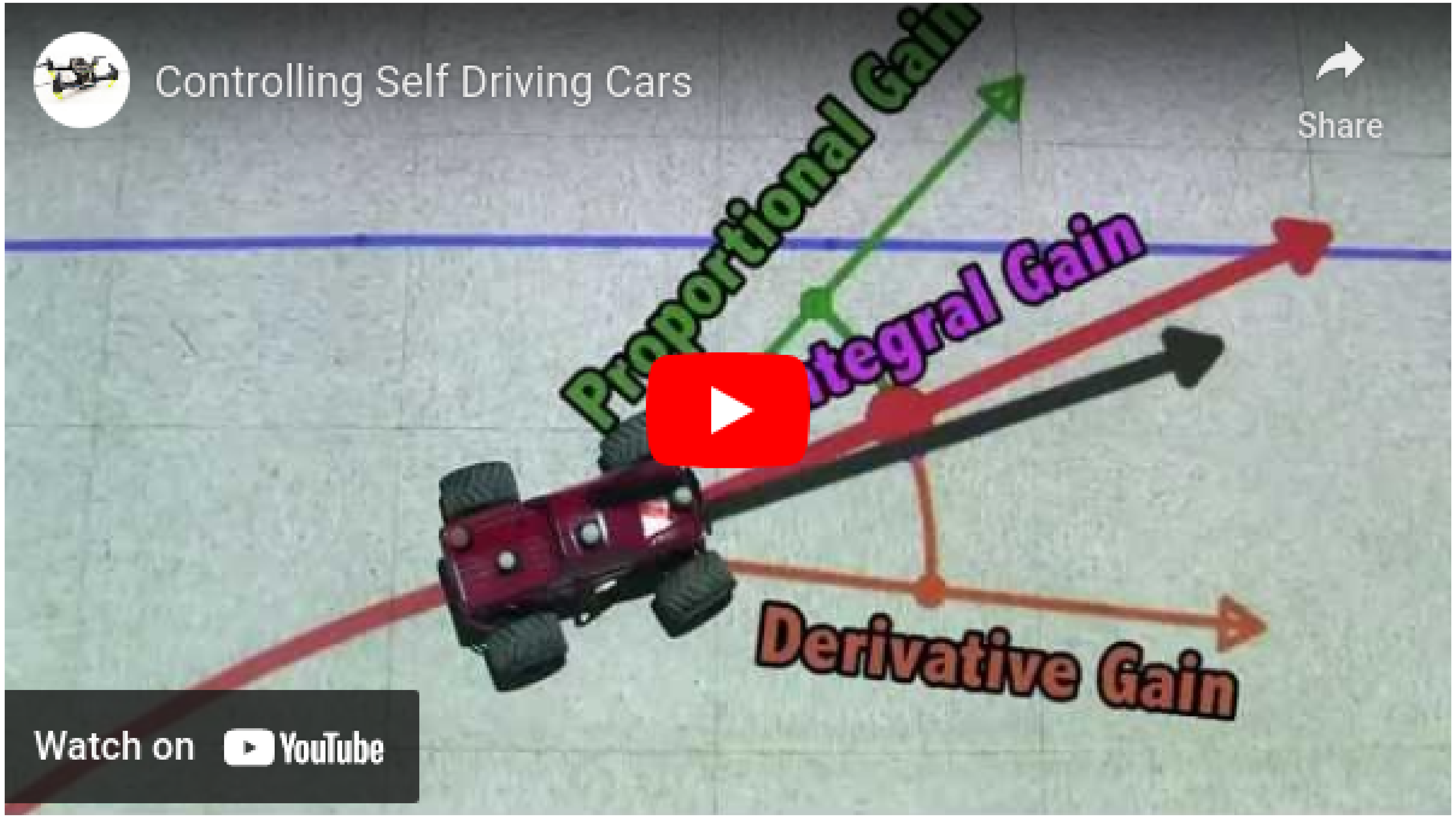
PID(Kp = 400, Ki = 200, Kd = 50)

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$

System response for w(t) = sin(t)

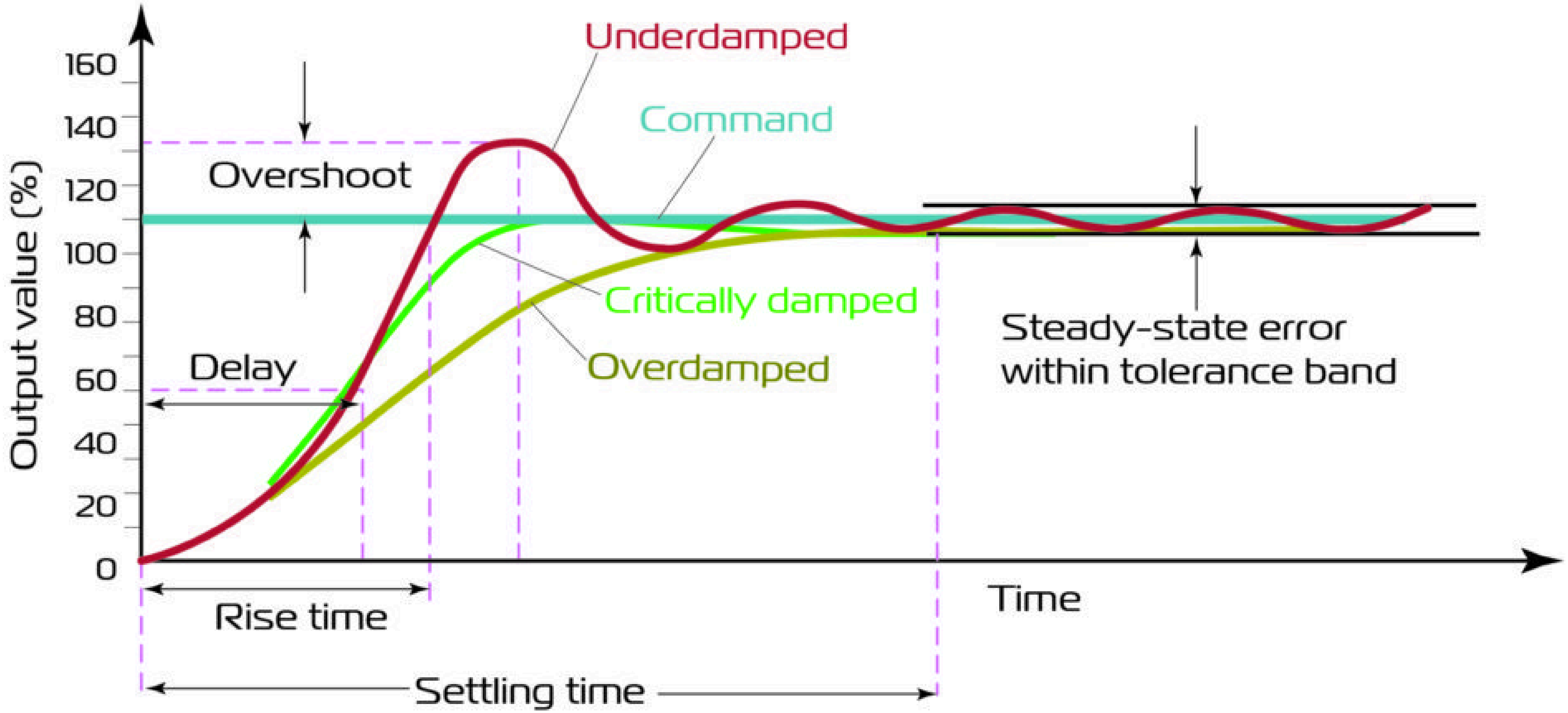
1954: Citroën Traction Avant 15-6H:, self-leveling Citroën hydropneumatic suspension on rear wheels. 1955 - on all four wheels.

PID - for self-driving car



https://www.youtube.com/watch?v=4Y7zG48uHRo

Glossary



Underdamped
Command
Overshoot
Critically damped
Overdamped
Delay
Steady-state error
within tolerance band
Output value (%)
160
140
120
100
80
60
40
20
0
Time
Rise time
Settling time

Glossary

Steady state error is the deviation of the output of control system from desired response during steady state

Settling time is the time required for the response curve to reach and stay within a range of certain percentage (usually 5% or 2%) of the final value.

Overshoot is the occurrence of a signal or function exceeding its target. **Undershoot** is the same phenomenon in the opposite direction.

Delay time is the time required for the response to reach half of its final value from the zero instant.

Rise time is the time required for the response to rise from 0% to 100% of its final value. This is applicable for the under-damped systems.
For the over-damped systems, consider the duration from 10% to 90% of the final value.

Critical damping: the condition in which the damping of an oscillator causes it to return as quickly as possible to its equilibrium position without oscillating back and forth about this position

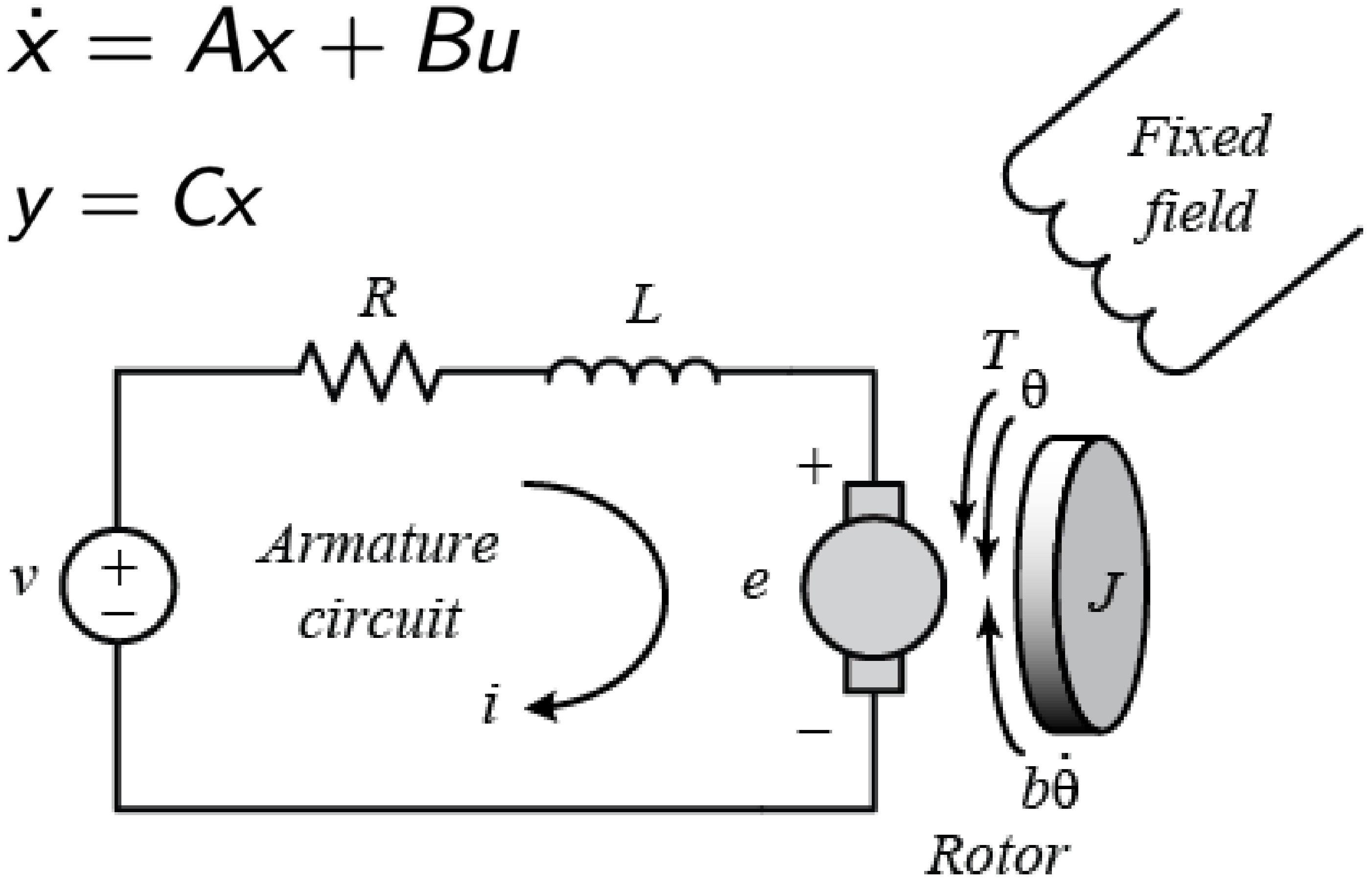
Over damping: the condition in which damping of an oscillator causes it to return to equilibrium without oscillating; oscillator moves more slowly toward equilibrium than in the critically damped system

Under damping: the condition in which damping of an oscillator causes it to return to equilibrium with the amplitude gradually decreasing to zero; system returns to equilibrium faster but overshoots and crosses the equilibrium position one or more times

DC motor control design

$$\dot{x} = Ax + Bu$$

$$y = Cx$$



A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion.

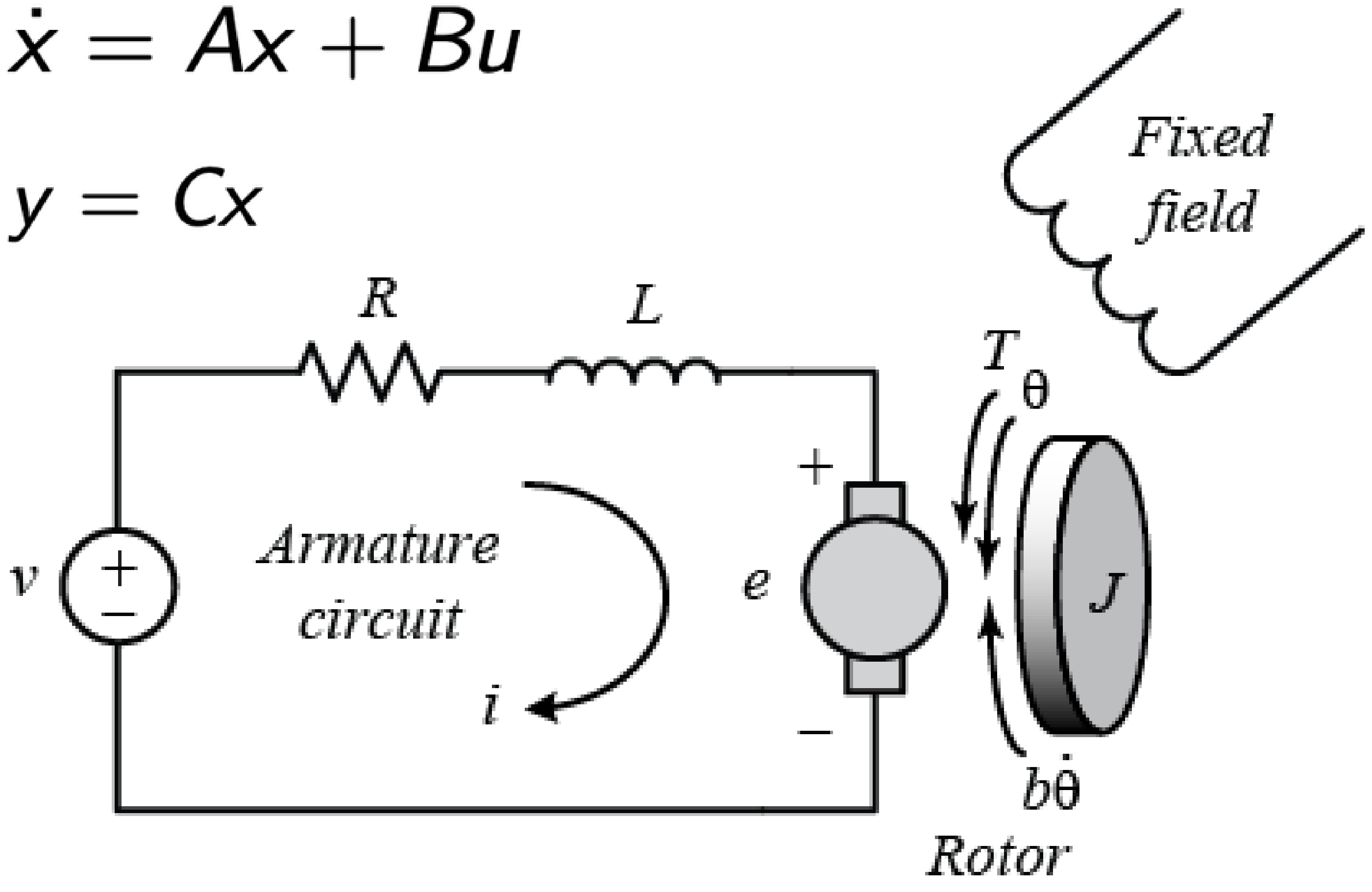
$$A = \begin{pmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{pmatrix}, \ B = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}$$

$$C = (1, 0)\,, x = \begin{pmatrix} \dot{\theta} \\ i \end{pmatrix}, r(t) = 1 \ rad/sec$$

DC motor control design

$$\dot{x} = Ax + Bu$$

$$y = Cx$$



A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion.

$$A = \begin{pmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}$$

$$C = (1, 0), x = \begin{pmatrix} \dot{\theta} \\ i \end{pmatrix}, r(t) = 1 \ rad/sec$$

The design criteria are the following:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$
$$y = Cx$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$ **where e(t) = r(t) - y(t)**

How to rewrite this system in the matrix form?

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$
$$y = Cx$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$ **where e(t) = r(t) - y(t)**

How to rewrite this system in the matrix form?

To implement an integral term, let us introduce a new variable z(t) = $\int_0^t (y(s) - r(s))\, ds$

$z(0) = 0$

then

$\dot{z}(t) = y(t) - r(t)$

$u(t) = K_p e(t) - K_i z(t) + K_d \dot{e}(t)$

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$

$$y = Cx$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$ **where e(t) = r(t) - y(t)**

How to rewrite this system in the matrix form?

To implement an integral term, let us introduce a new variable z(t) = $\int_0^t (y(s) - r(s))\, ds$

$z(0) = 0$

then

$$\dot{z}(t) = y(t) - r(t)$$

$$u(t) = K_p e(t) - K_i z(t) + K_d \dot{e}(t)$$

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$

$$\dot{z} = Cx - r$$

$$u = K_p(r - Cx) - K_i z + K_d(\dot{r} - C\dot{x})$$

$$\Downarrow$$

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$

$$\dot{z} = Cx - r$$

$$u = K_p(r - Cx) - K_i z + K_d(\dot{r} - C\dot{x}), \text{ since } e = r - Cx$$

$$\Downarrow$$

$$\dot{x} = Ax + K_p B r - K_p B C x - K_i B z + K_d B \dot{r} - K_d B C \dot{x} + Dw$$

$$\dot{z} = Cx - r$$

$$\Downarrow$$

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$

$$\dot{z} = Cx - r$$

$$u = K_p(r - Cx) - K_i z + K_d(\dot{r} - C\dot{x}), \text{ since } e = r - Cx$$

$$\Downarrow$$

$$\dot{x} = Ax + K_p B r - K_p B C x - K_i B z + K_d B \dot{r} - K_d B C \dot{x} + Dw$$

$$\dot{z} = Cx - r$$

$$\Downarrow$$

$$\begin{cases} (I + K_d BC)\dot{x} = (A - K_p BC)x - K_i B z + K_p B r + K_d B \dot{r} + Dw \\ \dot{z} = Cx - r \end{cases}$$

PID. Closed-loop system

$$\dot{x} = Ax + Bu + Dw$$

$$\dot{z} = Cx - r$$

$$u = K_p(r - Cx) - K_i z + K_d(\dot{r} - C\dot{x}), \text{ since } e = r - Cx$$

$$\Downarrow$$

$$\dot{x} = Ax + K_p B r - K_p B C x - K_i B z + K_d B \dot{r} - K_d B C \dot{x} + Dw$$

$$\dot{z} = Cx - r$$

$$\Downarrow$$

$$\begin{cases} (I + K_d B C)\dot{x} = (A - K_p B C)x - K_i B z + K_p B r + K_d B \dot{r} + Dw \\ \dot{z} = Cx - r \end{cases}$$

PID. Closed-loop system

$$\begin{cases} (I + K_d BC)\dot{x} = (A - K_p BC)x - K_i B z + K_p B r + K_d B \dot{r} + C w \\ \dot{z} = Cx - r \end{cases}$$

Let us denote $M = I + Kd\, BC$

$x_A = \begin{pmatrix} x \\ z \end{pmatrix} \leftarrow$ **augmented vector space**

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} M^{-1}(A - K_p BC) & -M^{-1} K_i B \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} K_p M^{-1} B \\ -1 \end{bmatrix} r + \begin{bmatrix} Kd\, M^{-1} B \\ 0 \end{bmatrix} \dot{r} + \begin{bmatrix} M^{-1} C \\ 0 \end{bmatrix} w$$

$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

PID. Closed-loop system

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} M^{-1}(A - K_p BC) & \vdots & -M^{-1} K_i B \\ C & \vdots & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} K_p M^{-1} B \\ -1 \end{bmatrix} r + \begin{bmatrix} K_d M^{-1} B \\ 0 \end{bmatrix} \dot{r} + \begin{bmatrix} M^{-1} C \\ 0 \end{bmatrix} w$$

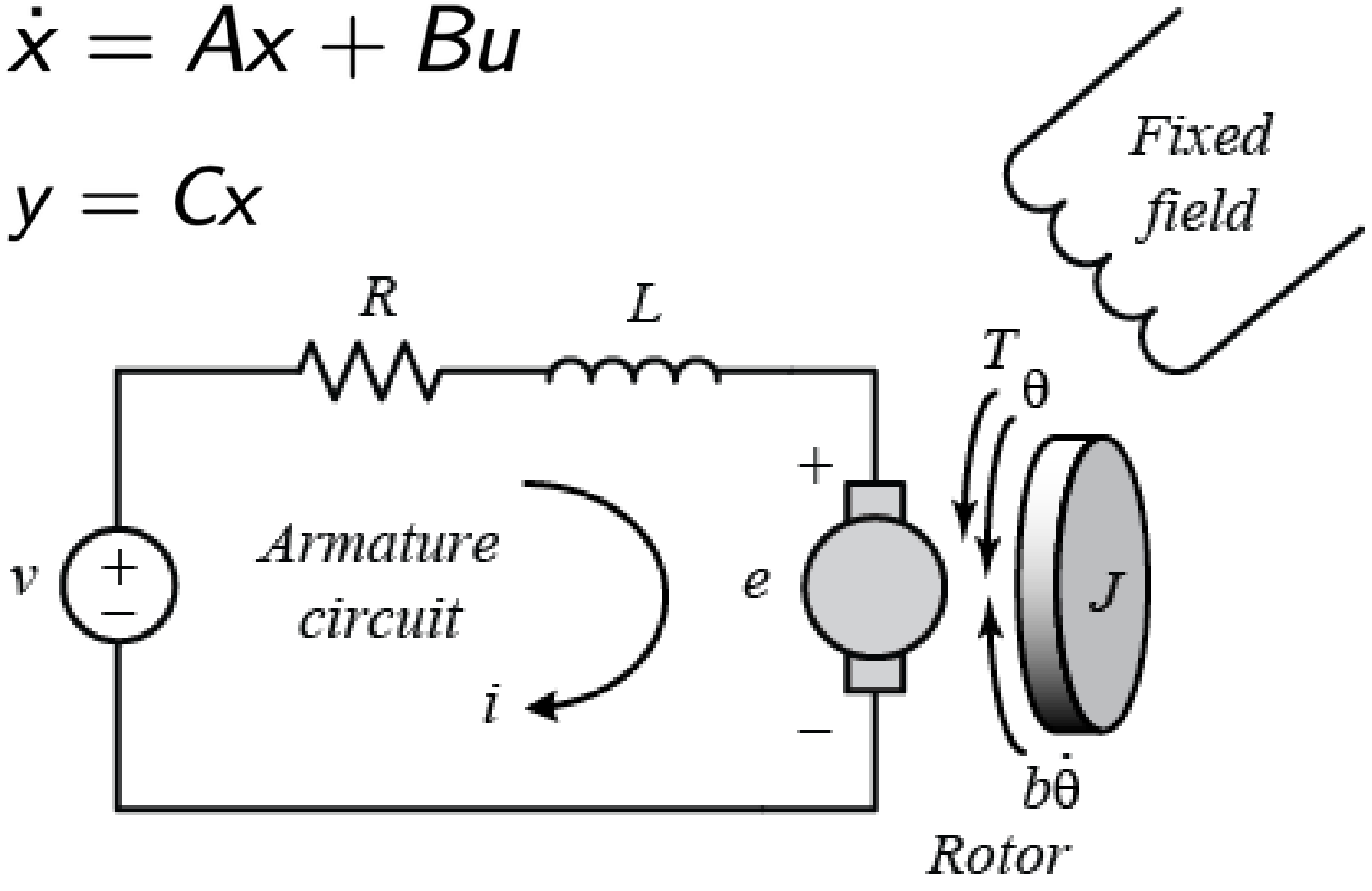
$$y = \begin{bmatrix} C & \vdots & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

$$\begin{cases} \begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = A_a \begin{bmatrix} x \\ z \end{bmatrix} + B_a r + B_{ad} \dot{r} + D_a w \\ y = C_A \begin{bmatrix} x \\ z \end{bmatrix} \end{cases}$$

DC motor control design

$$\dot{x} = Ax + Bu$$

$$y = Cx$$



A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion.

$$A = \begin{pmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \frac{1}{L} \end{pmatrix}$$

$$C = (1, 0), \quad x = \begin{pmatrix} \dot{\theta} \\ i \end{pmatrix}, \quad r(t) = 1 \ rad/sec$$

The design criteria are the following:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%

The design criteria are the following:

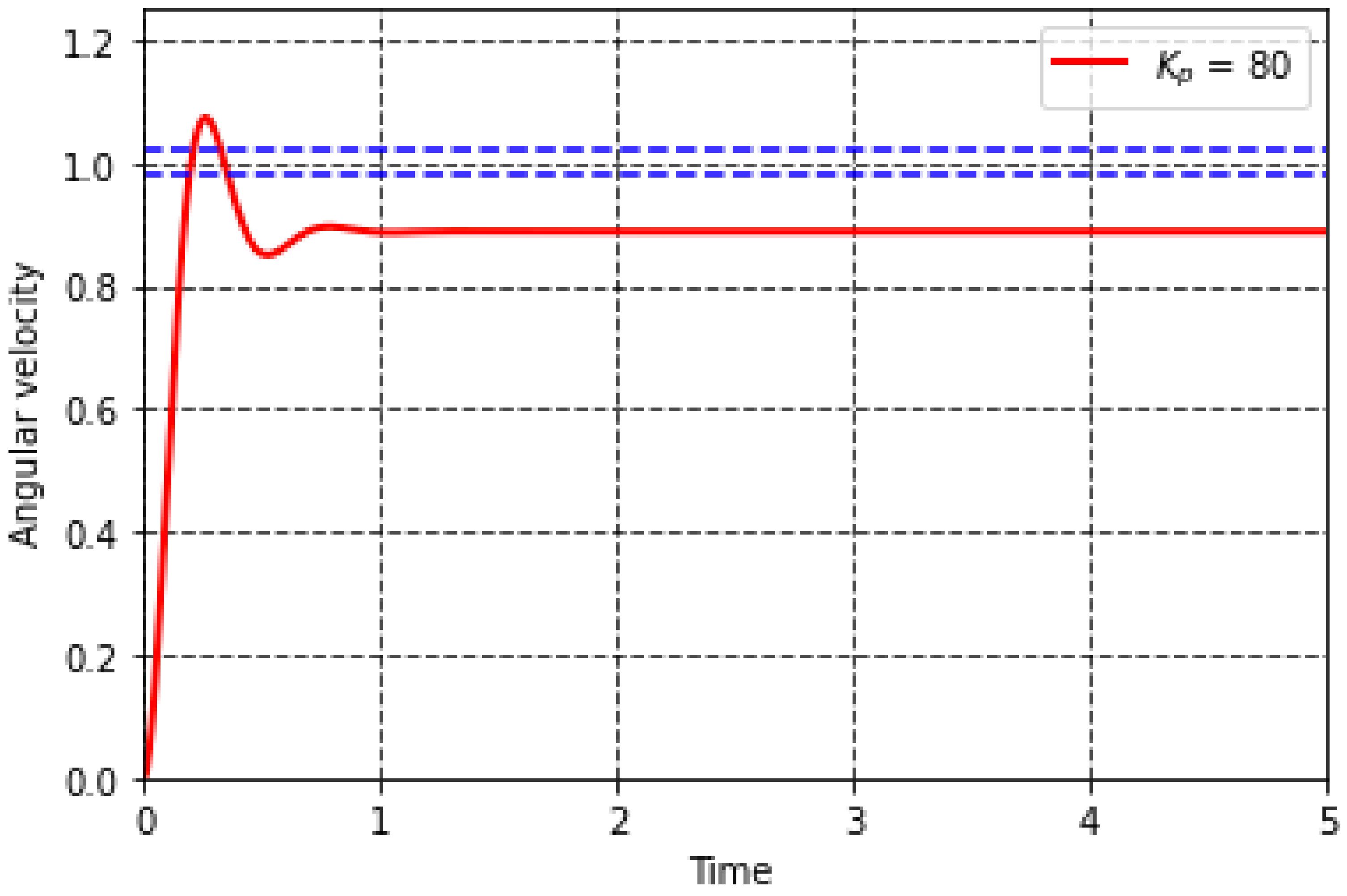
- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%

Proportional term

$$u(t) = K_p(r(t) - y(t))$$

The design criteria are the following:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%



Proportional term

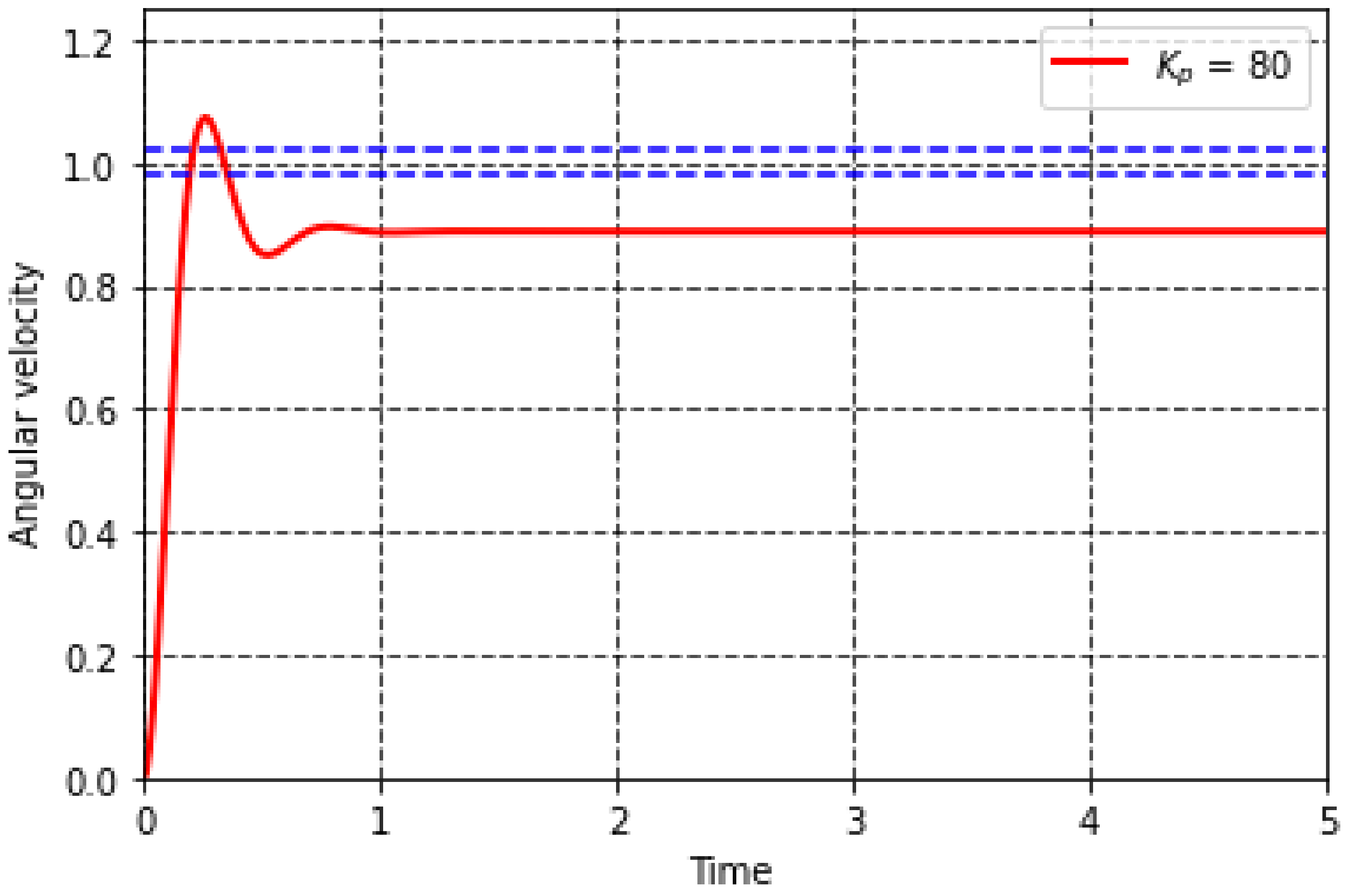
$$u(t) = K_p(r(t) - y(t))$$

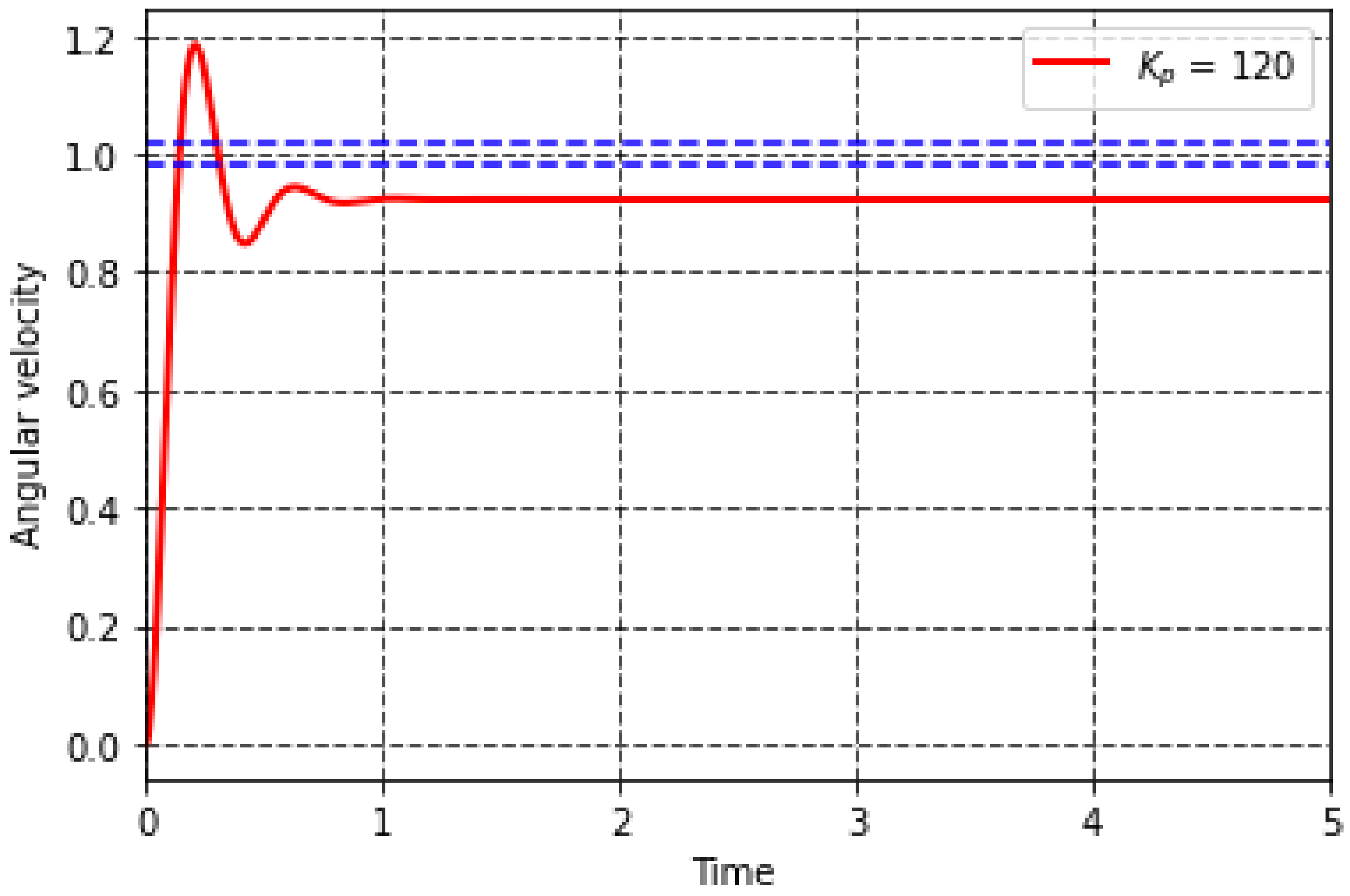
The design criteria are the following:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%

Proportional term

$$u(t) = K_p(r(t) - y(t))$$





The design criteria are the following:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%

Integral term

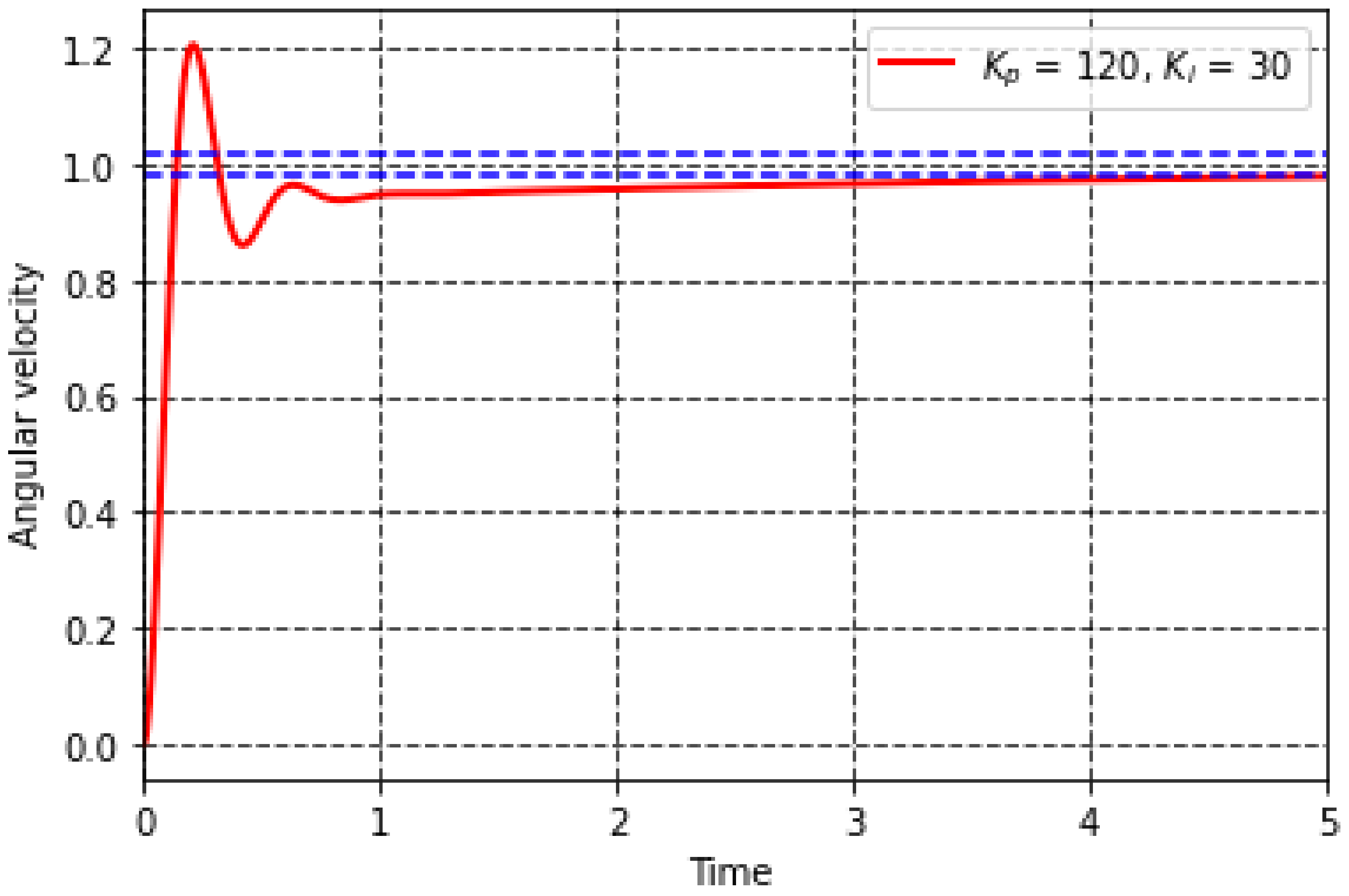
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau$$

The design criteria are the following:

- **Settling time less than 2 seconds**
- **Overshoot less than 5%**
- **Steady-state error less than 2%**

Integral term

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau$$

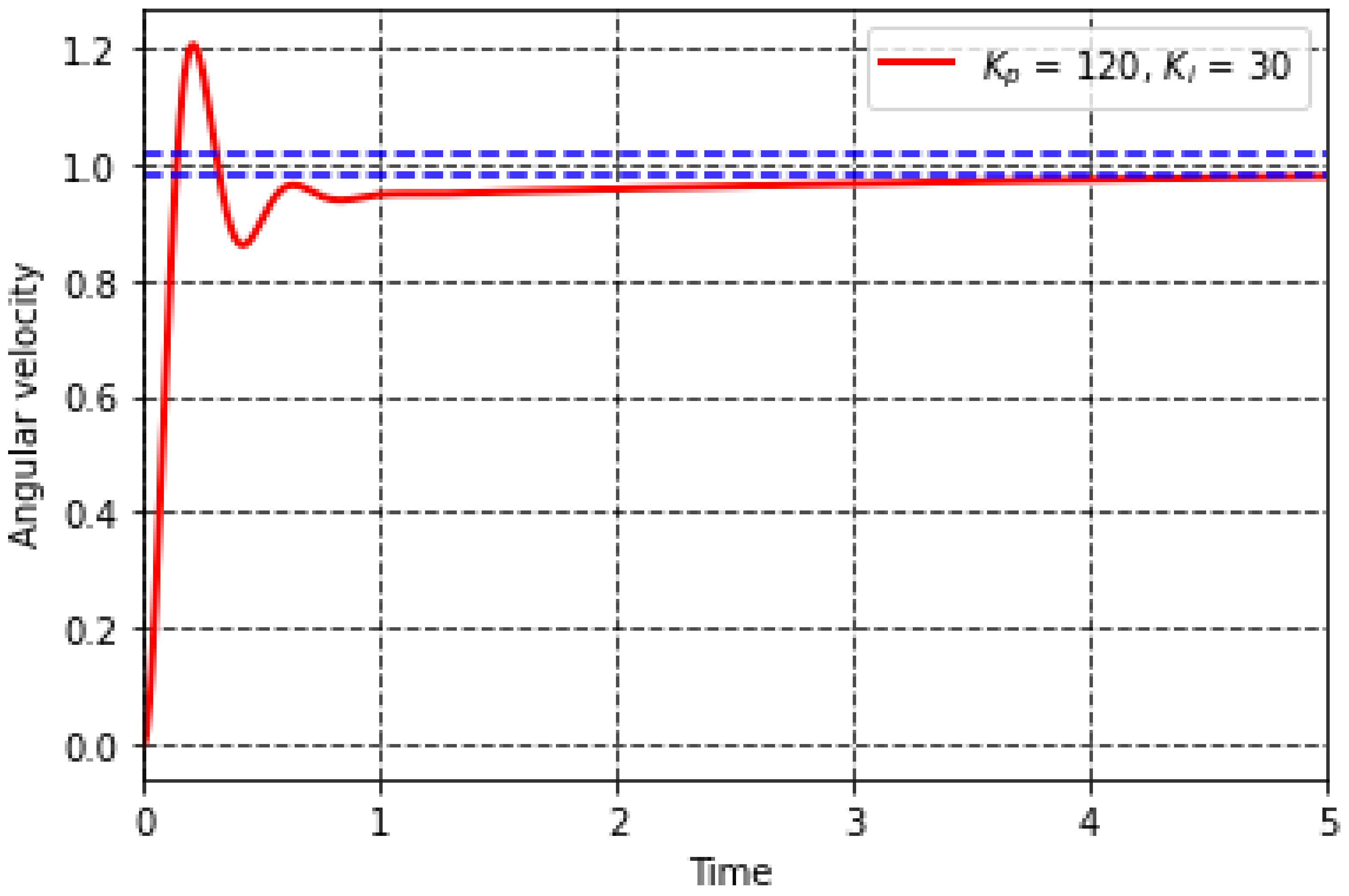


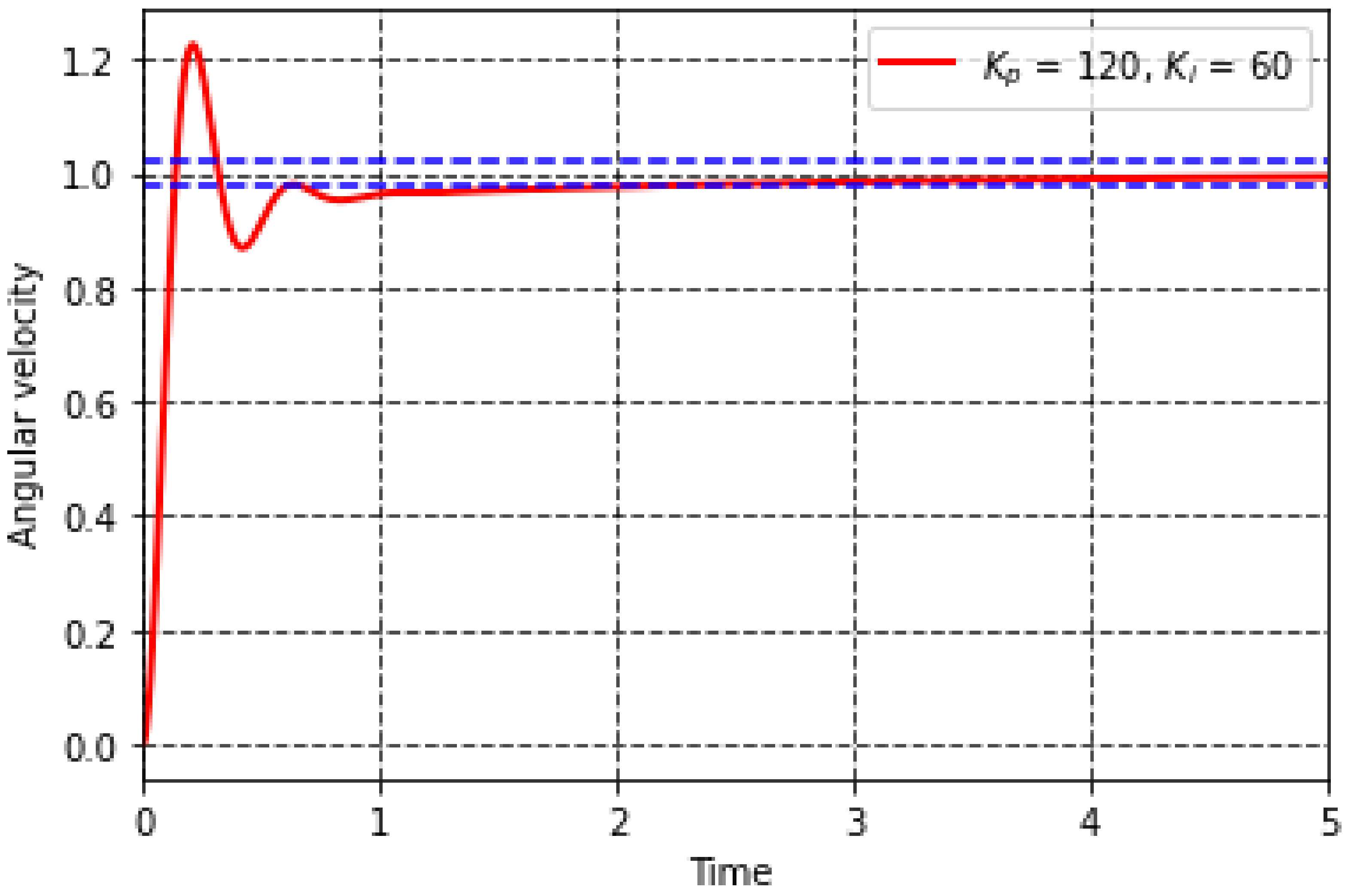
The design criteria are the following:

- **Settling time less than 2 seconds**
- **Overshoot less than 5%**
- **Steady-state error less than 2%**

Integral term

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau$$





The design criteria are the following:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 2%

Derivative term

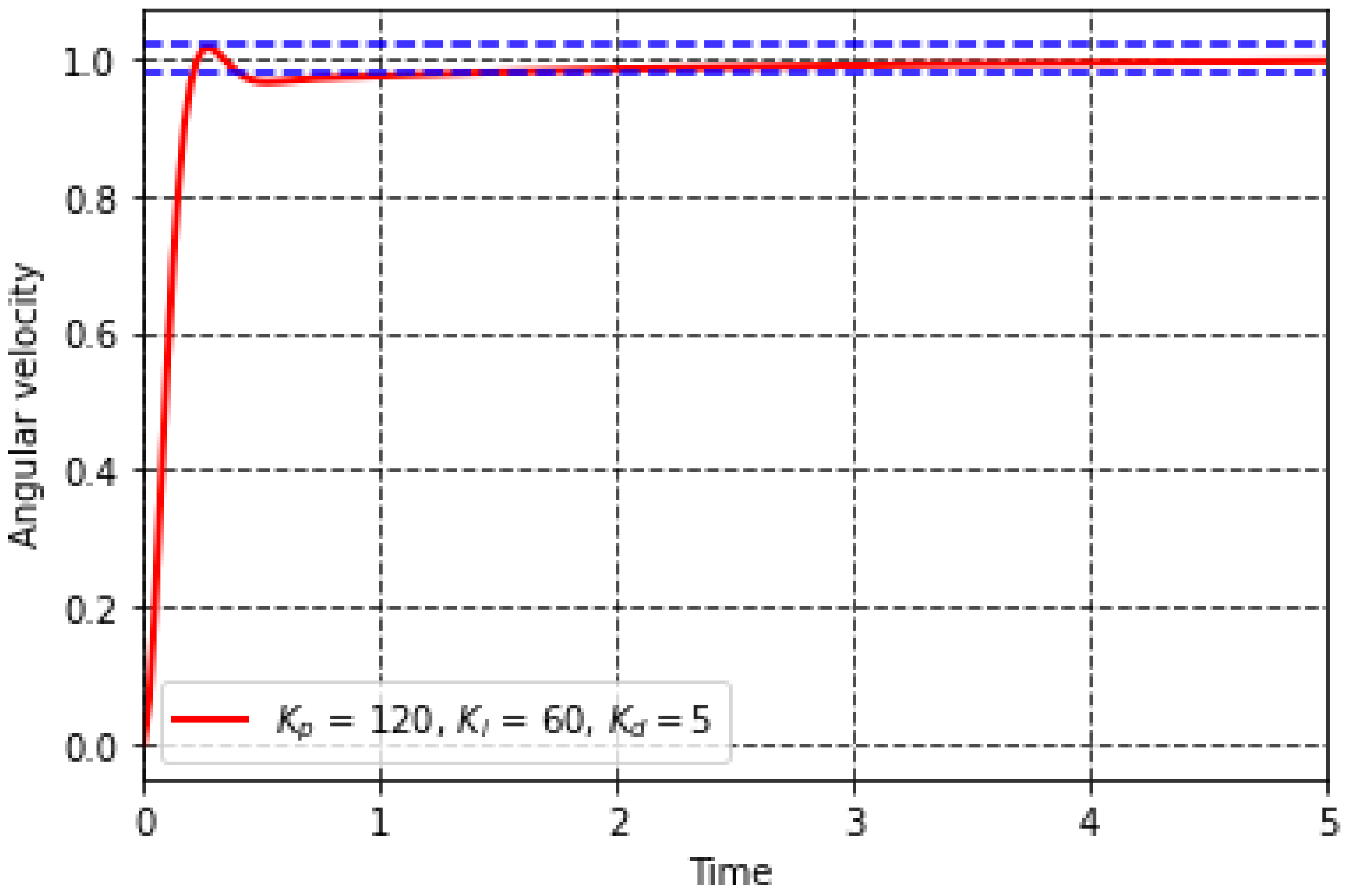
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$

The design criteria are the following:

- **Settling time less than 2 seconds**
- **Overshoot less than 5%**
- **Steady-state error less than 2%**

Derivative term

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$

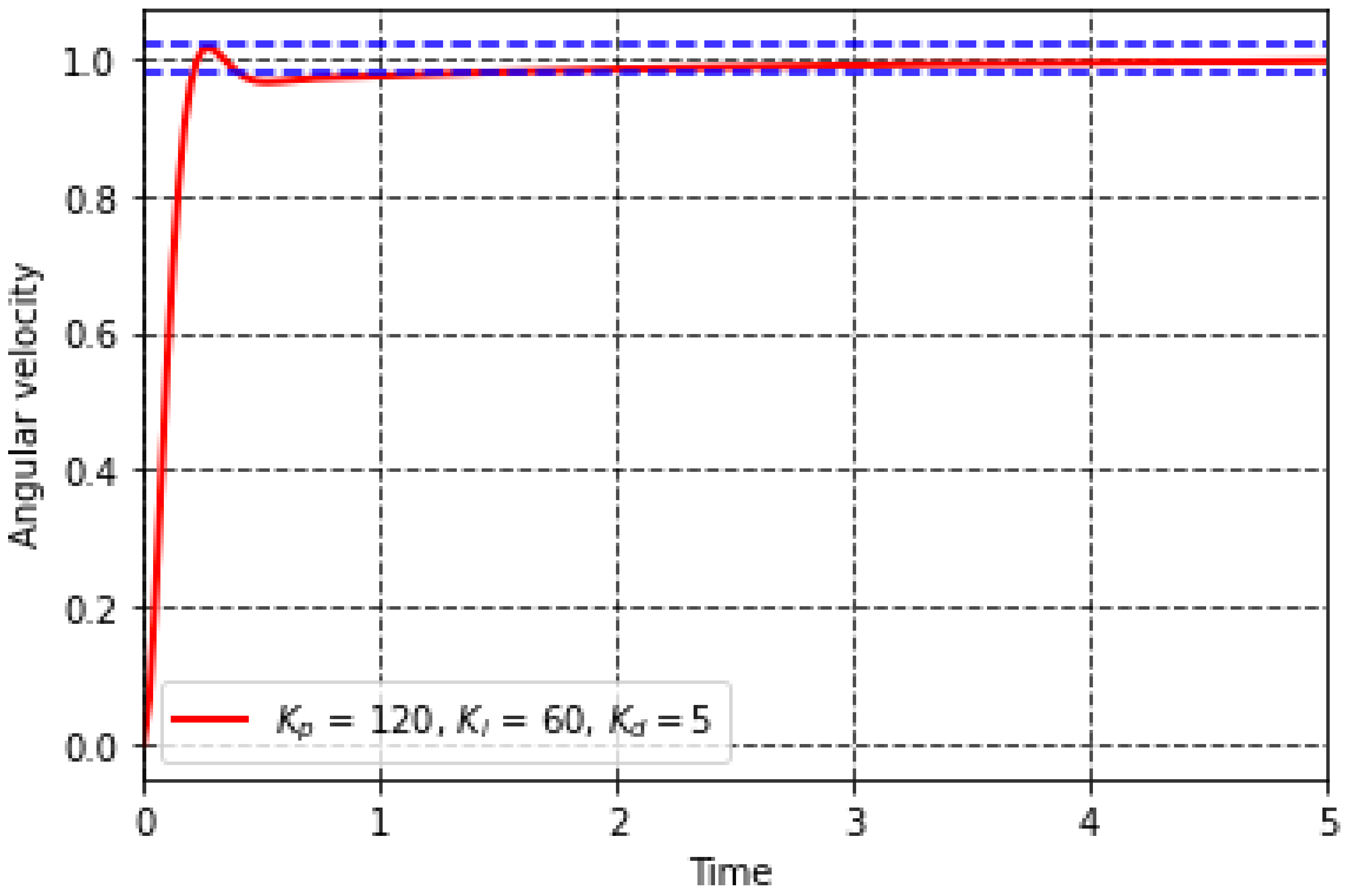


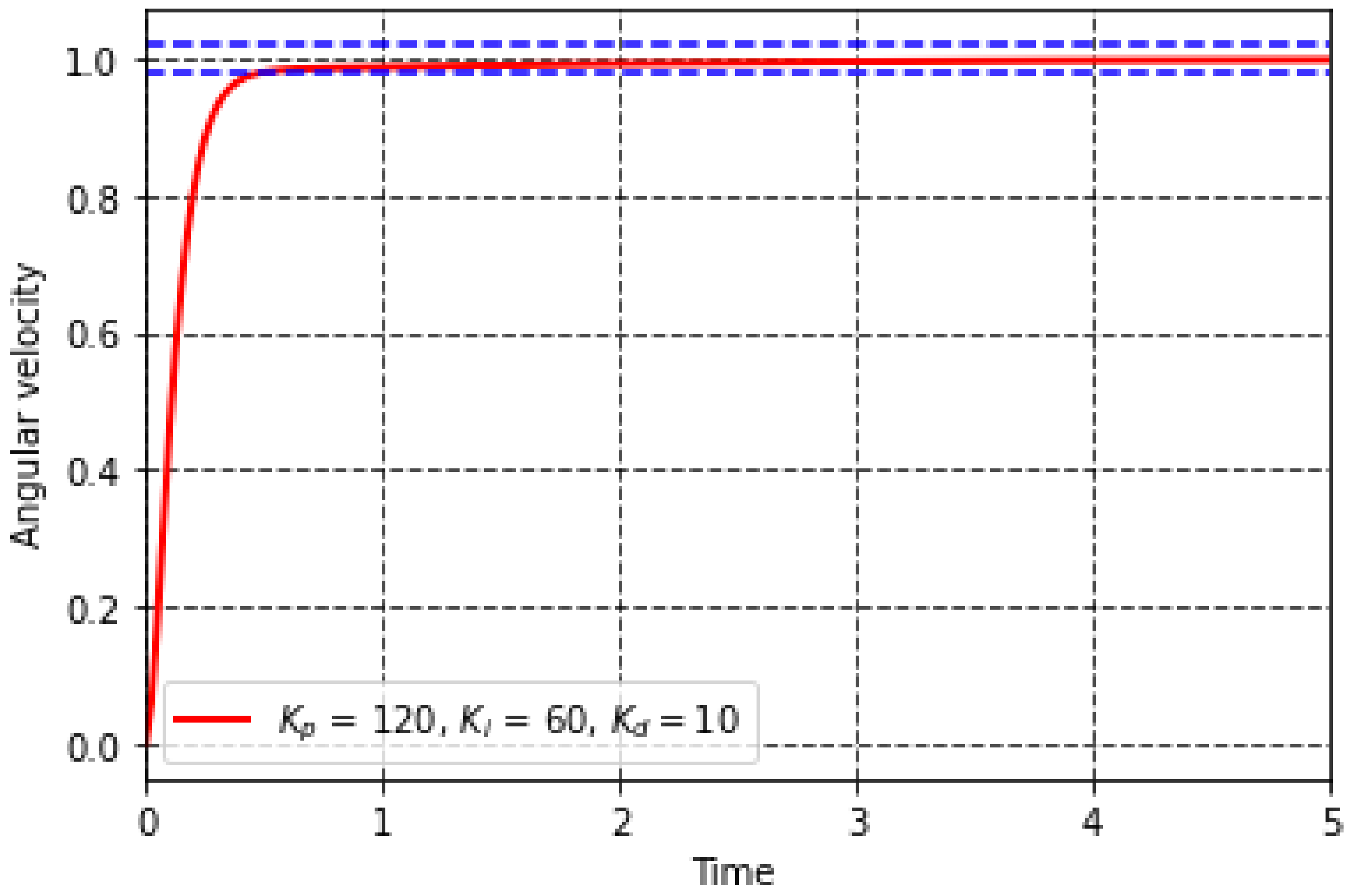
The design criteria are the following:

- **Settling time less than 2 seconds**
- **Overshoot less than 5%**
- **Steady-state error less than 2%**

Derivative term

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$





Manual Tuning PID

- Start with Kp, Ki, and Kd at 0.
- Increase Kp until steady-state error is very low.
- Increase Ki until steady-state error is removed entirely.
- Increase Kd until oscillations are removed.

Tuning PID

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
increase Kp	Decrease	Increase	Small Change	Decrease
increase Ki	Decrease	Increase	Increase	Decrease
increase Kd	Small Change	Decrease	Decrease	No Change

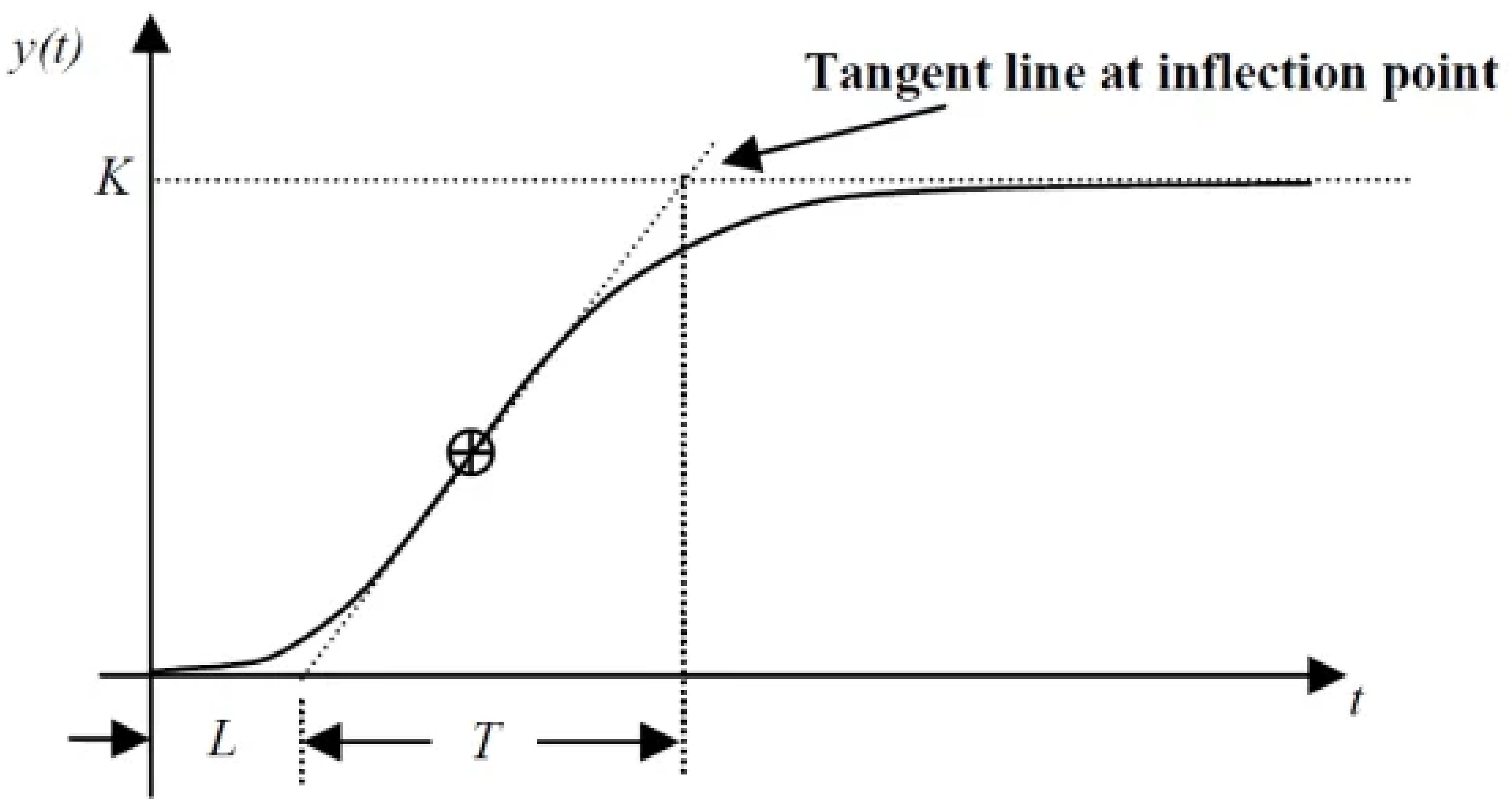
Tuning PID

https://www.youtube.com/watch?v=sFOEsA0Irjs&list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y&index=4

Ziegler-Nichols First Method

Conduct an Open-Loop Step Test:

Apply a step input (sudden change) to the process
open-loop process
Record the process variable's response over time.



If curve has S-shape Identify

Delay Time (L):
Time Constant (T):

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	$1.2\frac{T}{L}$	$2L$	$0.5L$

Based on the measured values of L and T, use the Ziegler-Nichols tuning formulas:

Ziegler-Nichols First Method

Advantages

Provides a simple way to obtain PID parameters.
Effective for systems with slow dynamics.
Good for initial tuning before fine adjustments.

Disadvantages

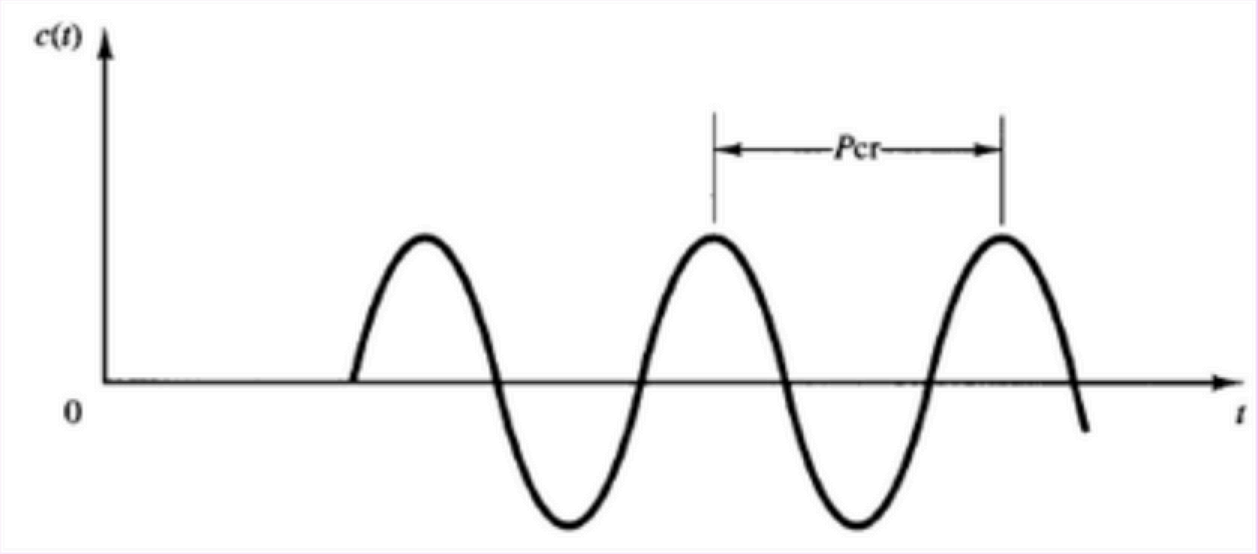
Works best for first-order-like systems (not highly oscillatory or nonlinear).
Can result in aggressive control settings leading to overshoot.
May not be ideal for integrating or unstable systems.

Ziegler-Nichols Second Method

- Apply only proportional control

u(t) = Kp e(t) to the system.

- Increase Kp Until Sustained Oscillations Appear
- The value of Kp at this point is the ultimate gain (Ku).
- Measure the oscillation period (Pct) (the time for one full cycle of oscillation).



Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Ziegler-Nichols Second Method

Advantages

✓ Works for a wide range of processes, including oscillatory and higher-order systems.
✓ Tends to produce fast, responsive control.
✓ Provides a systematic way to tune PID controllers.

Disadvantages

❌ Can lead to aggressive control with high overshoot.
❌ Requires the system to oscillate, which may be risky for unstable processes.
❌ May not work well for nonlinear or integrating systems.

https://www.youtube.com/watch?v=9vZSw_xzC9E&t=293s

PID Tunning

PID tuning depends
on the situation
It's an art!
Follow these rules!
A one size fits all
method doesn't exist

PID controller

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t)$$

where e(t) = r(t) - y(t)

Proportional (P) Control:
Effect: Faster response but steady-state error remains.

Integral (I) Control:
Effect: Improves accuracy but may cause overshoot

Derivative (D) Control:
Effect: Reduces overshoot and improves stability.

PID: Pros

Stability

PID controllers are capable of providing stable and accurate control over systems, ensuring that they reach and maintain the desired setpoint efficiently.

Tuning Flexibility

PID controllers offer flexibility in tuning parameters (Proportional, Integral, and Derivative gains) to achieve optimal performance for different systems and operating conditions.

Simple Implementation

Compared to more complex control algorithms, PID controllers are relatively simple to implement, making them suitable for a wide range of applications and accessible to engineers and technicians with basic control theory knowledge.

Real-Time Control

PID controllers are well-suited for real-time control applications due to their simplicity and efficiency, making them suitable for controlling systems with fast response

PID: Cons

Tuning Complexity:

Tuning PID controllers can be complex, especially for systems with nonlinear dynamics or time-varying parameters. Finding the right balance between stability and performance often requires iterative tuning processes.

Limited Robustness:

PID controllers may lack robustness compared to more advanced control algorithms, particularly in systems with uncertain parameters or external disturbances. Robust PID tuning methods exist but may require additional effort and expertise.

Potential for Oscillations and Instability:

Improper tuning of PID parameters can lead to oscillations or instability in the controlled system, resulting in erratic behavior or even system damage if not addressed promptly.

Integral windup & Derivative term sensitive to measurement errors

PID: Pros

it is not clear how to design a PID controller
when system is not SISO...