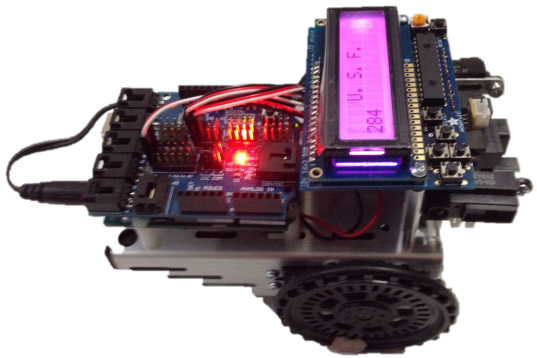
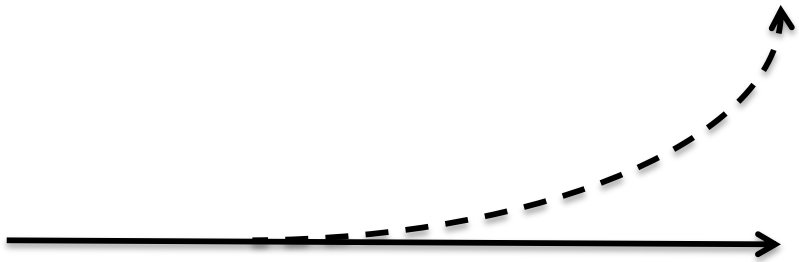
PID Control

Alfredo Weitzenfeld

Problem

- Suppose you have a system that needs to be controlled
- From command to response there is usually a delay
- Your software gives commands, the system responds to it:
  - Turn x degrees to the right
  - Move forward 15 wheel rotations
- Can you always trust your commands will be executed accurately?





PID Control

PID controllers have proven to be robust and extremely beneficial in the control of many important applications.

PID stands for:

P (*Proportional*)

I (*Integral*)

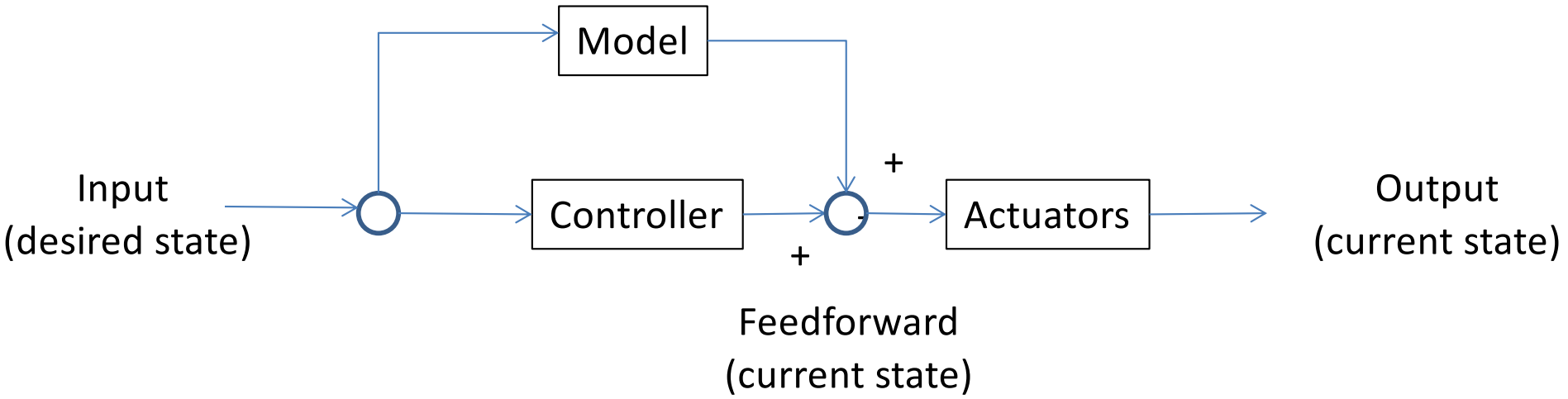
D (*Derivative*)

Definitions

- *Feedback control* is a means of getting a system or robot to achieve and maintain a desired state or *set point* by continuously comparing its current state with its desired state.
- *Desired state* or *goal state* of a system is where the system should be:
  - *Achievement goal* are states the system tries to reach, such as a location.
  - *Maintenance goal* require ongoing active work to keep, such as robot balancing to avoid falling.

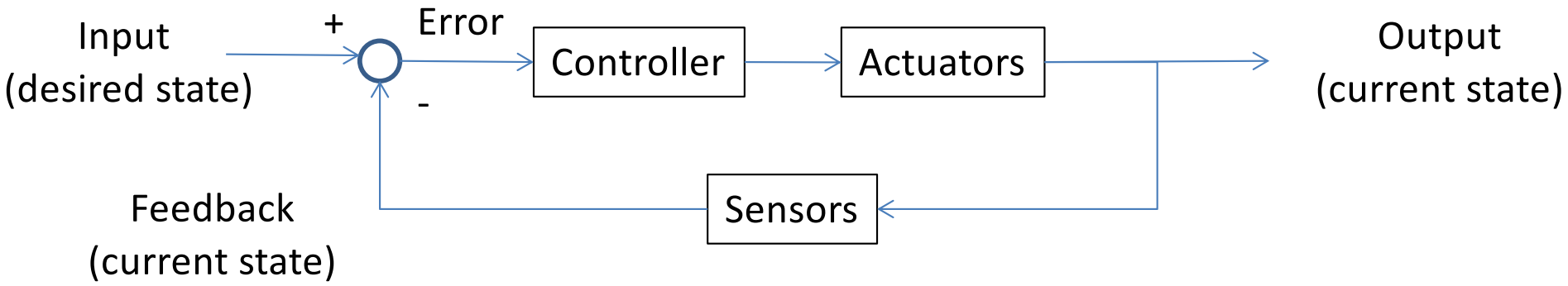
Feedforward or Open Loop Control

- *Feedforward* or *open loop* control does not use sensory feedback and state is not fed back into the system.
- System output is based on predictions in well structured predictable environments.

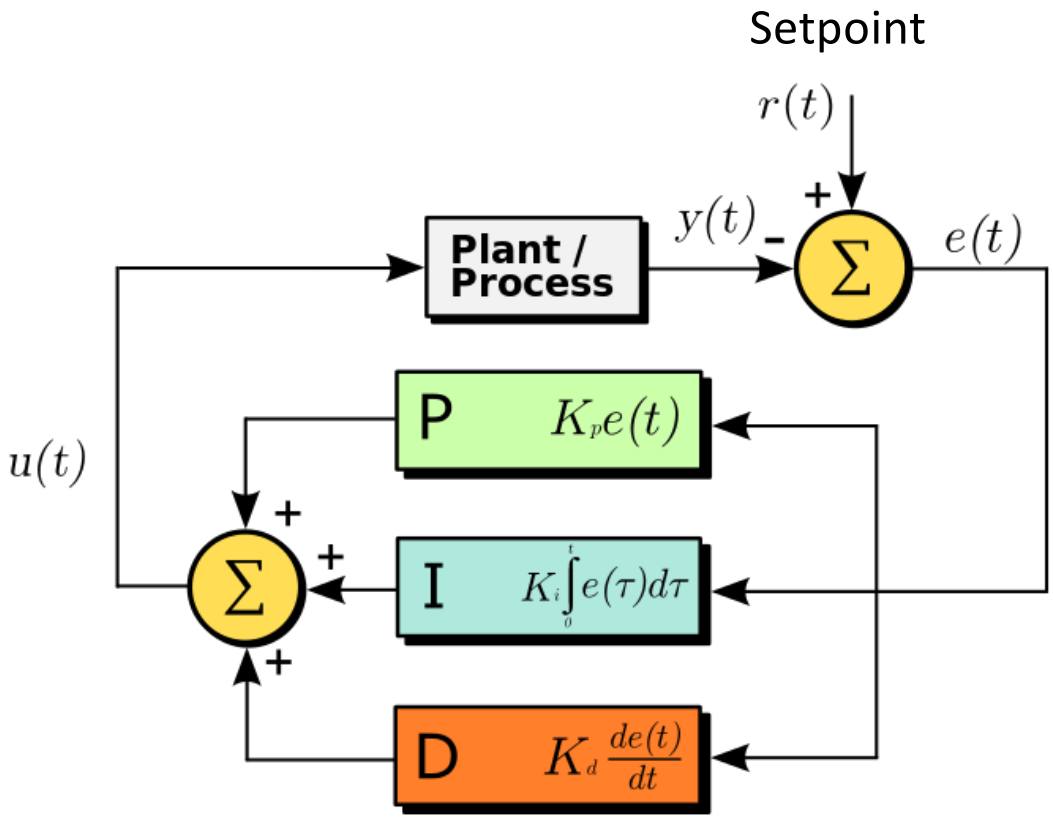


Feedback or Closed Loop Control

- *Feedback* control or *closed loop* control closes the loop between the input and the output providing the system with a feedback error.
- *Error* is the difference between current and desired state of a system. The goal is to minimize this error.
- *Magnitude* of the error describes how big the error with respect to some *direction* (positive or negative error)



PID Controller



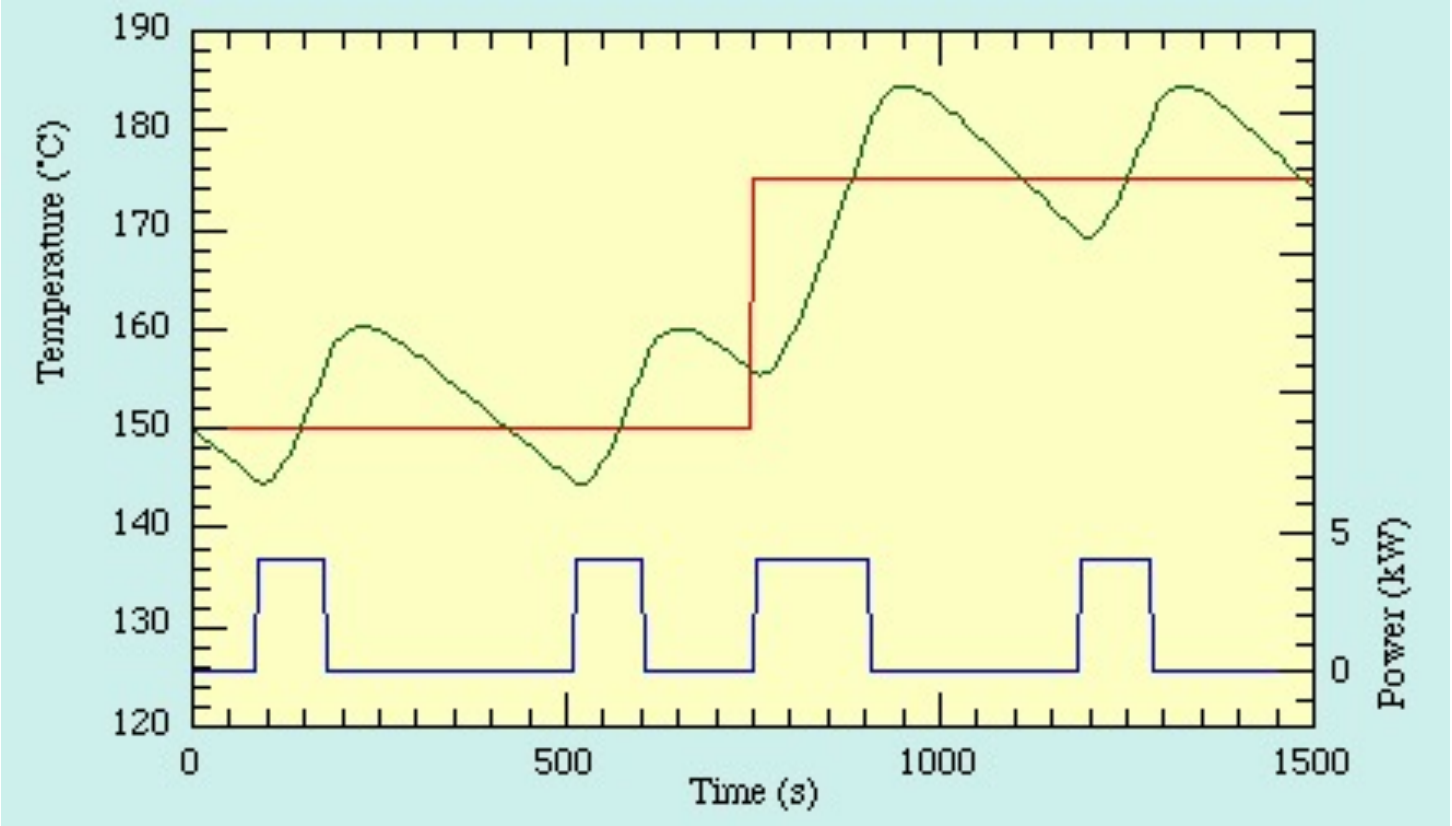
$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t) \cdot dt + K_d \cdot \frac{de(t)}{dt}$$

$$u(t) = K_p \cdot e(t) + K_i \cdot \sum_0^i e(i) + K_d \cdot [e(t) - e(t-1)]$$

$$u(t-1) = K_p \cdot e(t-1) + K_i \cdot \sum_0^{i-1} e(i) + K_d \cdot [e(t-1) - e(t-2)]$$

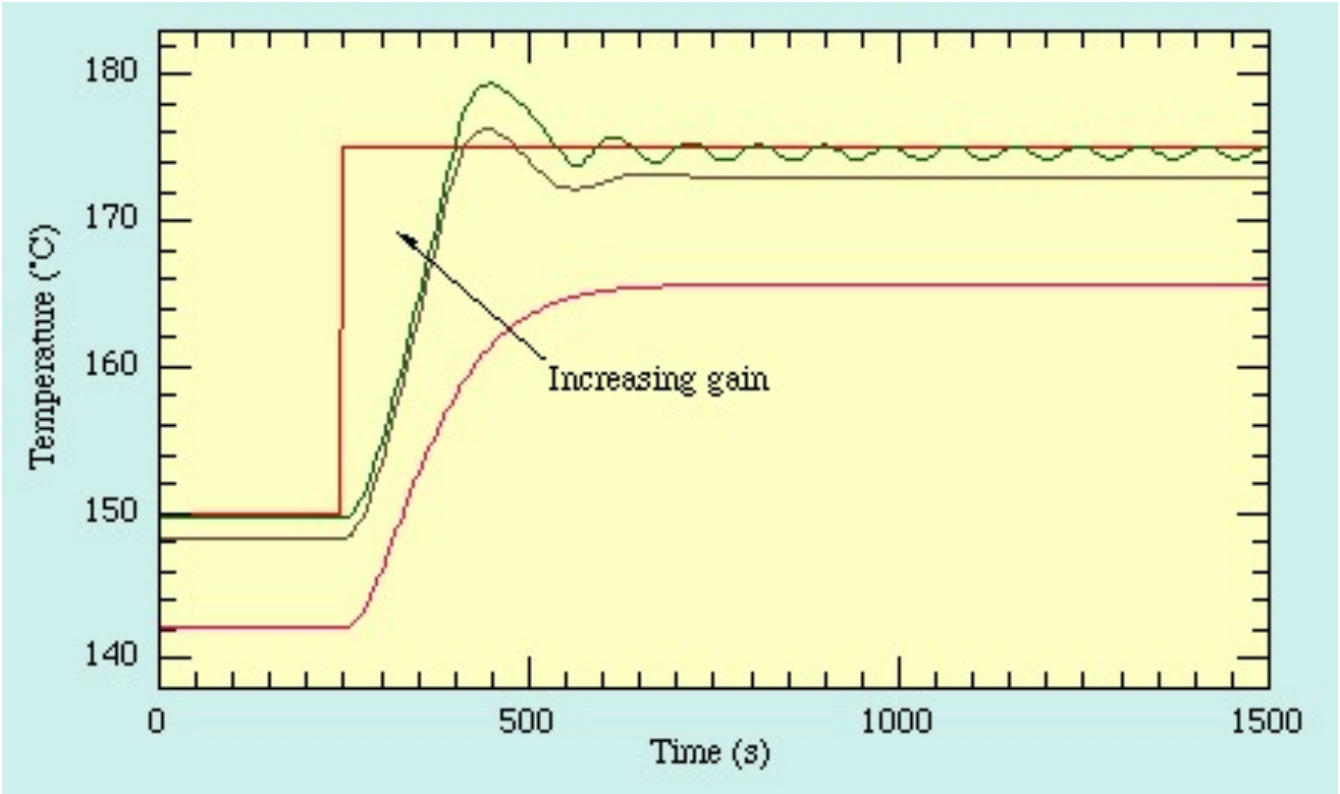
$$u(t) = u(t-1) + K_p \cdot [e(t) - e(t-1)] + K_i \cdot e(t) + K_d \cdot [e(t) - 2e(t-1) + e(t-2)]$$

On-off control



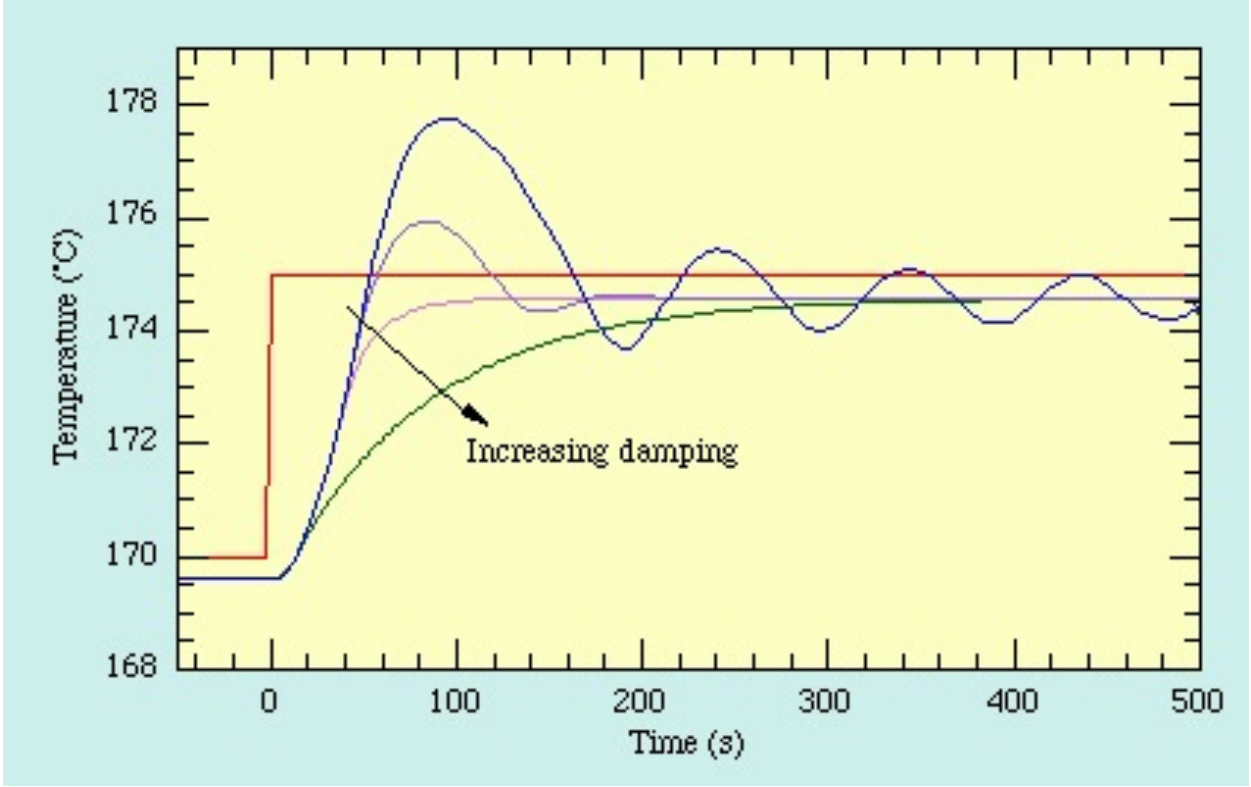
- This is the simplest form of control.
- For some systems, on-off signaling is sufficient
- For example, a thermostat, when the heater is either on or off
- Could cause overshoots and undershoots (ripples)

Proportional control



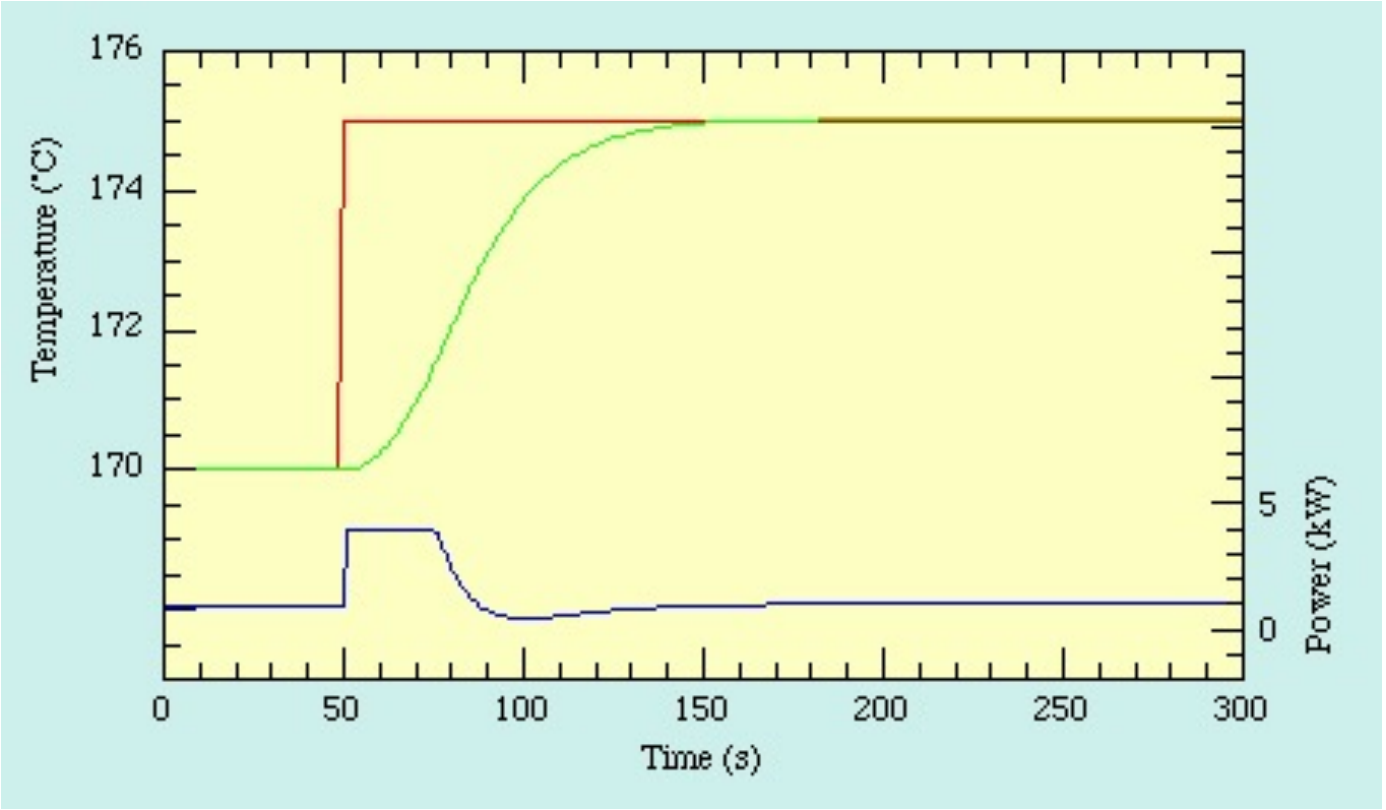
- Signal becomes proportional to the error, the difference between the measured and the setpoint.
- Typically a proportional control decreases response time (quickly gets to the *setpoint*) but increases overshoot.

Proportional, Derivative control



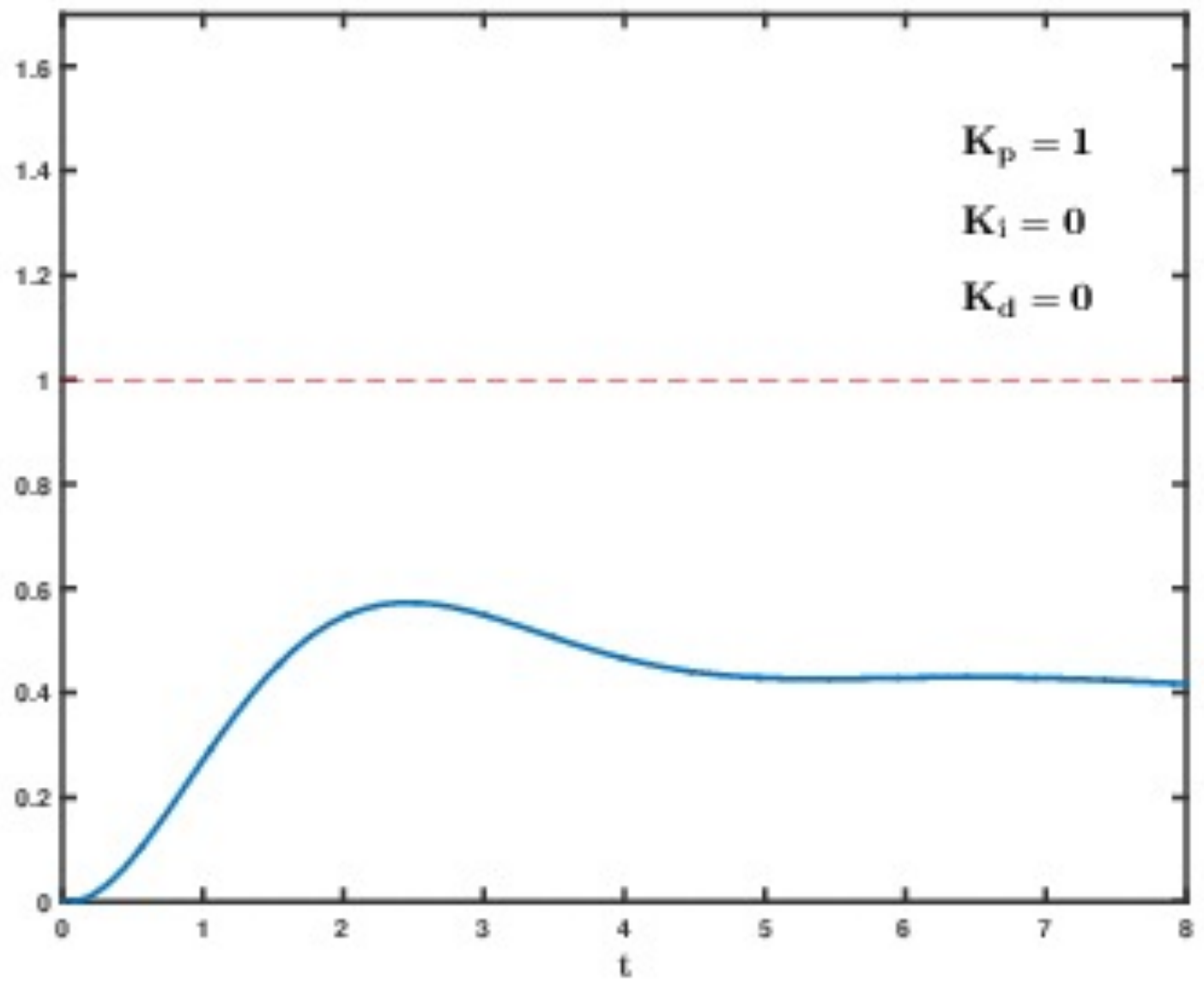
- Derivative control increases stability, reducing the overshoot, and improving the transient response.
- PD controllers are slower than P, but less oscillation, smaller overshoot/ripple.

Proportional, Integral, Derivative control



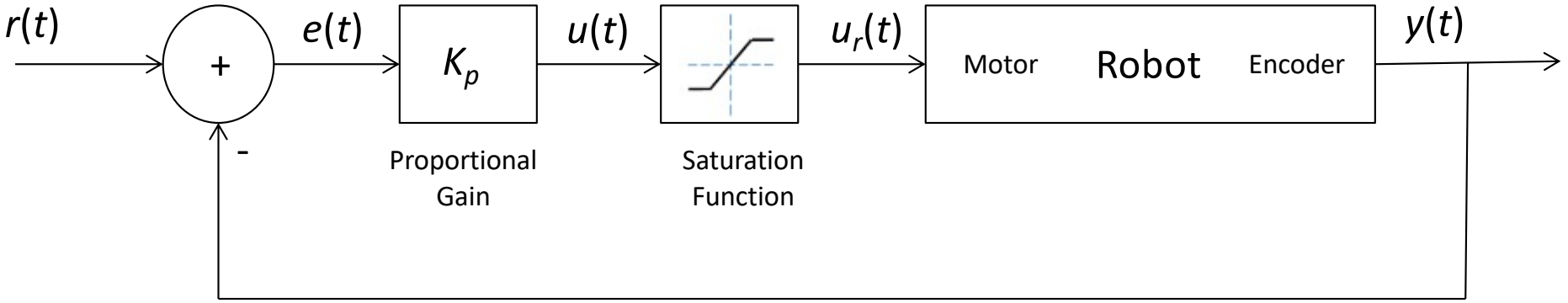
- Integral control will have the effect of eliminating the steady-state error, but it may make the transient response worse.
- A PID adds an integral term to the control function corresponding to the sum of the errors over time.

Controller



Example

Consider the PID system shown in the following figure and equations:



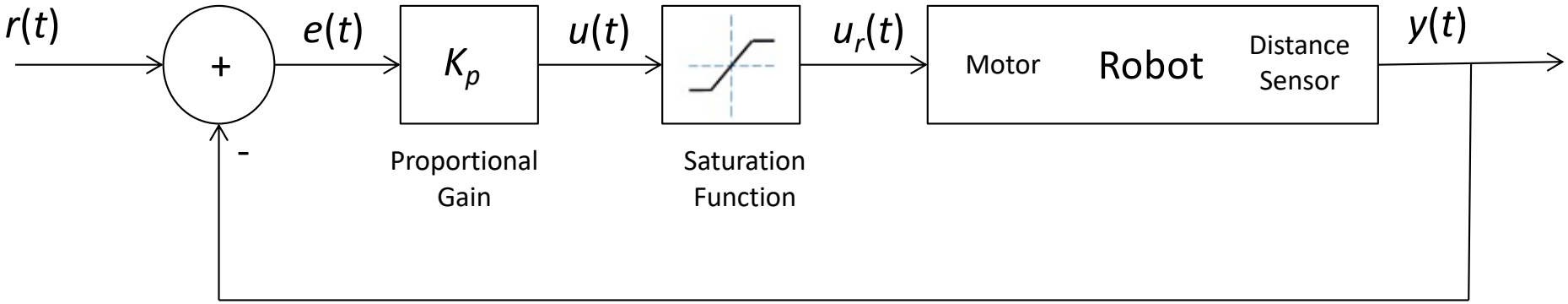
Desired velocity:	$r(t)$
Encoder measured velocity:	$y(t)$
Velocity error function:	$e(t) = r(t) - y(t)$
Motor velocity control function:	$u(t) = K_p * e(t)$
Motor velocity saturated control function:	$u_r(t) = f_{Sat}(u(t))$

Saturation motor velocity control functions limited by constants c_{max} and c_{min}, corresponding to max and min motor velocities :

$$f_{Sat}\big(u(t)\big) = \begin{cases} if\ u(t) > c_{max} & u_r(t) = c_{max} \\ if\ c_{min} \le u(t) \le c_{max} & u_r(t) = u(t) \\ if\ u(t) < c_{min} & u_r(t) = c_{min} \end{cases}$$

Example

Consider the PID system shown in the following figure and equations:



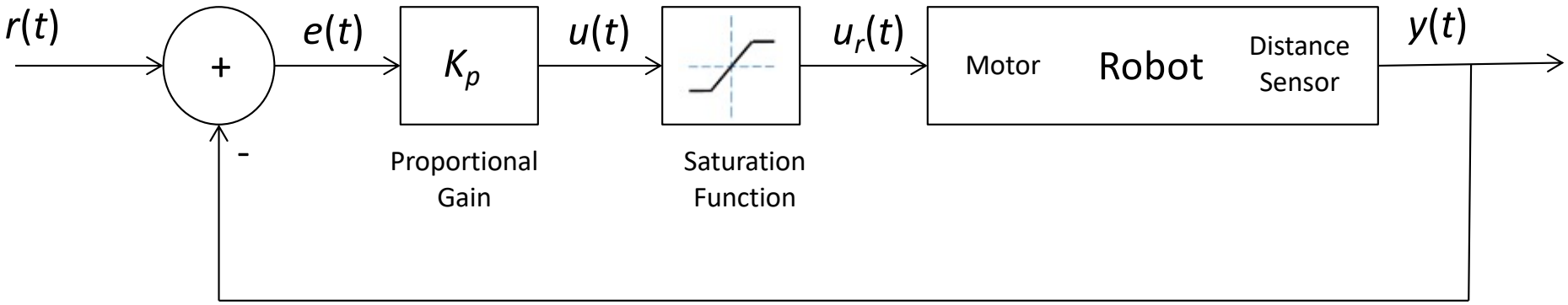
Desired distance:	$r(t)$
Distance measured:	$y(t)$
Distance error function:	$e(t) = r(t) - y(t)$
Motor velocity control function:	$u(t) = K_p * e(t)$
Motor velocity saturated control function:	$u_r(t) = f_{Sat}(u(t))$

Saturation motor velocity control functions limited by constants c_{max} and c_{min}, corresponding to max and min motor velocities :

$$f_{Sat}\big(u(t)\big) = \begin{cases} if\ u(t) > c_{max} & u_r(t) = c_{max} \\ if\ c_{min} \le u(t) \le c_{max} & u_r(t) = u(t) \\ if\ u(t) < c_{min} & u_r(t) = c_{min} \end{cases}$$

Example

Consider the P controller for the system shown in the following figure :



Consider:

- Proportional gain is given by K_p=1.5
- Saturation motor velocity control functions are limited by constants c_{max} = 3 and c_{min} = -3, corresponding to max and min motor velocities of 3 inches per sec and -3 inches per second.
- Desired state is given by r(t) = 5 inch
- Starting robot state is given by y(t) = 10 inch.

Example – Spring 2018

(a) Fill every entry in the table below until the distance error from the 5 inch desired state is less than 25% of 1 inch, i.e. the robot is between 5.25 and 4.75 inch mark. Compute states every second until the above distance error.

t	r(t)	y(t)	e(t)	K_p	u(t)	$u_r(t)$
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

(b) Will the system ever reach a 0.1% inch error from the desired state, i.e. a value between 4.999 or 5.001, and if so after how many steps?

(c) Will the system ever reach a 0% error and if so how after how many steps?

Example – Spring 2018

(a) Fill every entry in the table below until the distance error from the 5 inch desired state is less than 25% of 1 inch, i.e. the robot is between 5.25 and 4.75 inch mark. Compute states every second until the above distance error.

t	r(t)	y(t)	e(t)	K_p	u(t)	$u_r(t)$
0	-5 in	-10 in	5 in	1.5	7.5 in / sec	3 in / sec
1	-5 in	-7 in	2 in	1.5	3 in / sec	3 in / sec
2	-5 in	-4 in	-1 in	1.5	-1.5 in / sec	-1.5 in / sec
3	-5 in	-5.5 in	0.5 in	1.5	0.75 in / sec	0.75 in / sec
4	-5 in	-4.75 in	-0.25 in	1.5	-0.375 in /sec	-0.375 in /sec

(b) Will the system ever reach a 0.1% inch error from the desired state, i.e. a value between 4.999 or 5.001, and if so after how many steps?

Yes. In 12 steps computed at intervals of 1 sec.
The error e(t) is decreasing at a rate of 0.5 (within the "unsaturated" control range). Thus, in the "unsaturated" control portion starting at e(1) = 2, the error can be calculated by $e(t) = 2^{2-t}$ where t is an integer larger or equal than 1.
For $e(t) = 2^{2-t} < 0.001 = 1/1000 < 1/1024 = 1/2^{10} = 1/2^{2-12}$, thus t=12

(c) Will the system ever reach a 0% error and if so how after how many steps?

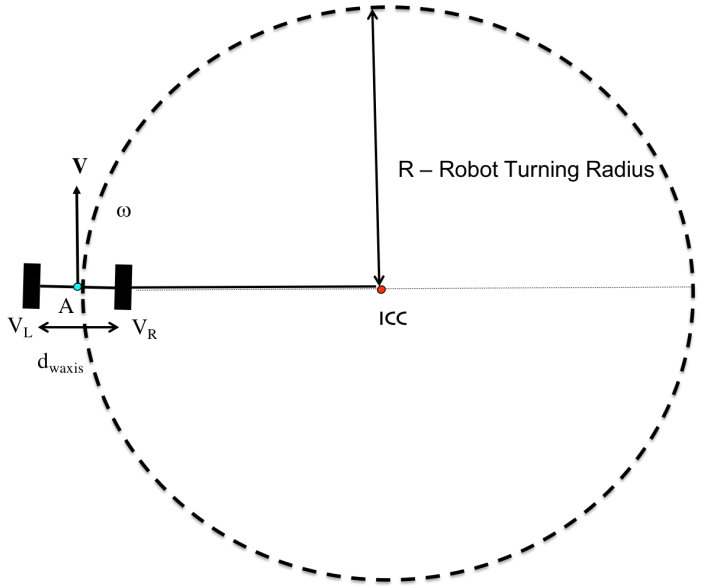
No, *t* will need to reach infinite for this to happen.

Example – Fall 2018

Apply the PID controller shown in the figure below, where the robot needs to complete the full circle around ICC, starting point A.

Consider the data below:

- R = 10 inch ("Robot Turning Radius")
- d_{waxis} = 4 inch (distance between the two robot wheels)
- r = 1 inch (radius of robot wheels)
- c_{max} = 6π inches per sec (max linear robot velocity)
- c_{min} = -6π inches per sec (min linear robot velocity)
- $y(t)$ (measured distance to the goal in inches, for example, at t=0, the measured distance to A from the robot initial location is the full circle, i.e. $y(0) = 2\pi R = 2\pi 10 = 20\pi$ inches)
- $r(t)$ = 0 inch (desired distance to the goal, i.e. distance to A).
- $e(t)$ (distance error), where $-0.5\pi < e(t) < 0.5\pi$,



Example – Fall 2018

Fill every entry in the table, for K_p = 0.5, until reaching the distance error $-0.5\pi < e(t) < 0.5\pi$:

t	$y(t)$	$e(t) = r(t) - y(t)$	$u(t) = K_p * e(t)$	$u_r(t) = f_{Sat}(u(t))$
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Example – Fall 2018

Fill every entry in the table, for K_p = 0.5, until reaching the distance error $-0.5\pi < e(t) < 0.5\pi$:

t	$y(t)$	$e(t) = r(t) - y(t)$	$u(t) = K_p * e(t)$	$u_r(t) = f_{Sat}(u(t))$
0	-20π	20π	10π	6π
1	-14π	14π	7π	6π
2	-8π	8π	4π	4π
3	-4π	4π	2π	2π
4	-2π	2π	1π	1π
5	-1π	1π	0.5π	0.5π
6	-0.5π	0.5π	0.25π	0.25π

Example – Fall 2018

Fill every entry in the table, for K_p = 1.0, until reaching the distance error $-0.5\pi < e(t) < 0.5\pi$:

t	$y(t)$	$e(t) = r(t) - y(t)$	$u(t) = K_p * e(t)$	$u_r(t) = f_{Sat}(u(t))$
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Example – Fall 2018

Fill every entry in the table, for K_p = 1.0, until reaching the distance error $-0.5\pi < e(t) < 0.5\pi$:

t	$y(t)$	$e(t) = r(t) - y(t)$	$u(t) = K_p * e(t)$	$u_r(t) = f_{Sat}(u(t))$
0	-20π	20π	20π	6π
1	-14π	14π	14π	6π
2	-8π	8π	8π	6π
3	-2π	2π	2π	2π
4	0	0	0	0

Example – Fall 2018

Fill every entry in the table, for K_p = 1.5, until reaching the distance error $-0.5\pi < e(t) < 0.5\pi$:

t	$y(t)$	$e(t) = r(t) - y(t)$	$u(t) = K_p * e(t)$	$u_r(t) = f_{Sat}(u(t))$
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Example – Fall 2018

Fill every entry in the table, for K_p = 1.5, until reaching the distance error $-0.5\pi < e(t) < 0.5\pi$:

t	$y(t)$	$e(t) = r(t) - y(t)$	$u(t) = K_p * e(t)$	$u_r(t) = f_{Sat}(u(t))$
0	-20π	20π	30π	6π
1	-14π	14π	21π	6π
2	-8π	8π	12π	6π
3	-2π	2π	3π	3π
4	1π	-1π	-1.5π	-1.5π
5	-0.5π	0.5π	0.75π	0.75π
6	0.25π	-0.25π	-0.125π	-0.125π

Example – Fall 2018

> In theory, for any of the previous K_p, will the robot ever stop exactly at the desired location?

> In practice, will the robot ever stop exactly at the desired location?

Example – Fall 2018

> In theory, for any of the previous K_p, will the robot ever stop exactly at the desired location?

In theory "yes", for $K_p = 1.0$.

> In practice, will the robot ever stop exactly at the desired location?

In practice "no", there are errors and delays affecting control $u(t)$ and measurements $y(t)$. If there were no errors and no delays, then there would be no need for a PID controller!