Introduction to Cognitive Robotics

Module 3: Mobile Robots
Lecture 4: Closed-loop control and PID control;
the go-to-position problem; divide-and-conquer controller

www.cognitiverobotics.net

Terminology

Goal

- Get some process or plant to a desired state
- Maintain that state

Example states:

Water level in a tank
Temperature of water in a tank
Flow rate of a pipeline
Speed of a mobile robot
Position of a mobile robot
Orientation of a mobile robot

These are referred to as "process variables"

"plant" is a term used to refer to the system being controlled
e.g. water tank, pipeline, mobile robot

Terminology

- Plant: the process or plant to be controlled
- Process variable (PV): the actual state of the process or plant
- Set point (SP): the desired state of the process or plant
- Error: the difference between PV and SP

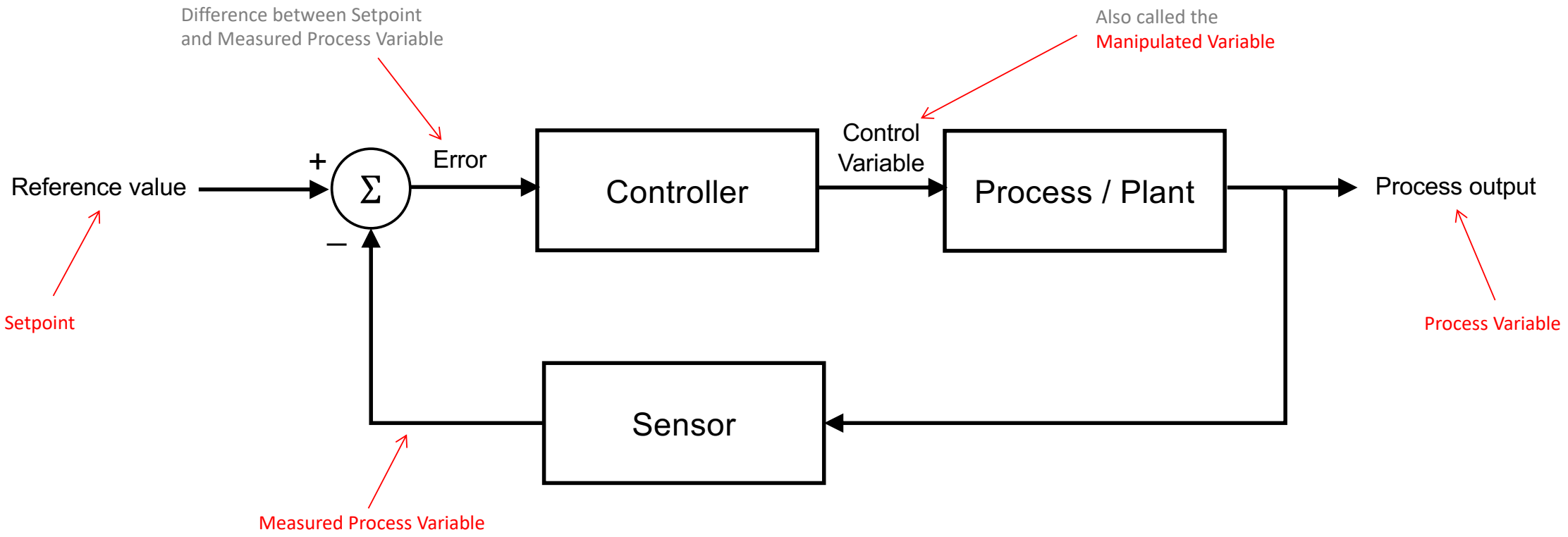
Terminology

- Effector: a mechanism that changes the state of the process or plant (i.e. control action)
- Sensor: a mechanism that measures the state of the process or plant
- Control variable: the input to the effector
- Controller:

Can be a physical device (e.g. a mechanical governor)
or software implementing a control algorithm

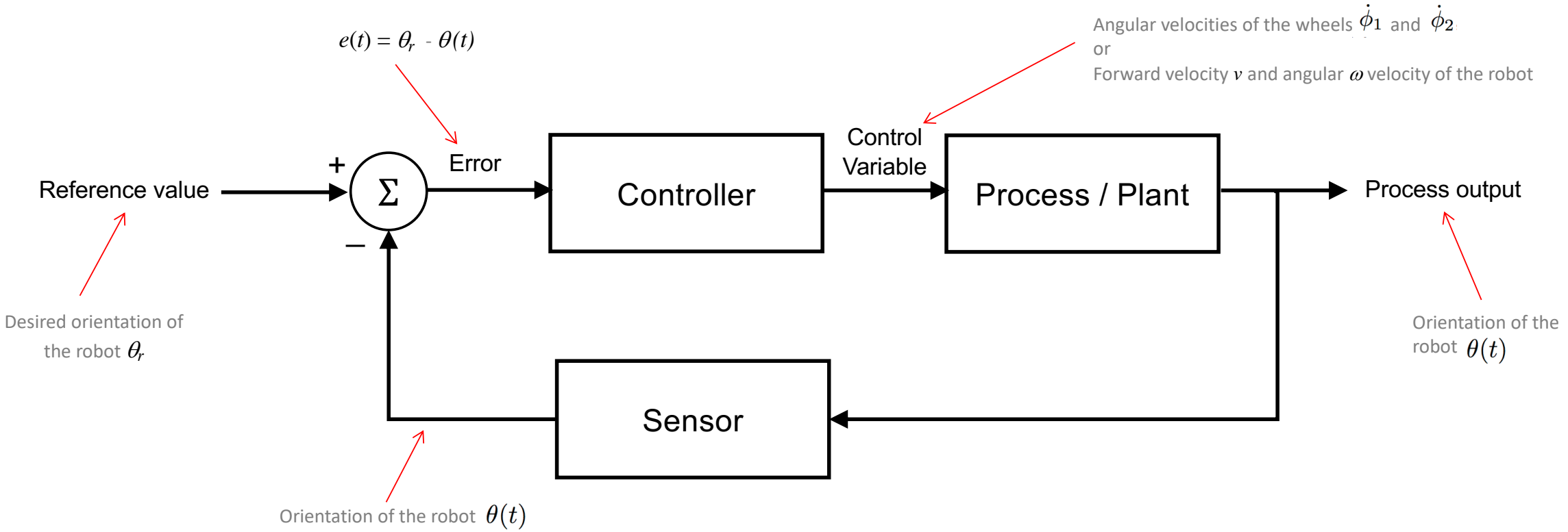
A mechanism to identify the value of the control signal that
reduces the error to zero as quickly as possible, without overshoot,
in a stable manner

Closed-loop Feedback Control



Closed-loop Feedback Control

For example, controlling the orientation of a mobile robot



Closed-loop Feedback Control

Control variable is a function of the error: $f(e)$

e = error between desired value (i.e. the setpoint) and the actual value (i.e. the measured process value)

PID Controller

Which function?

f = "proportional to e"

f = "proportional to the accumulation of e"

f = "proportional to the rate of change of e"

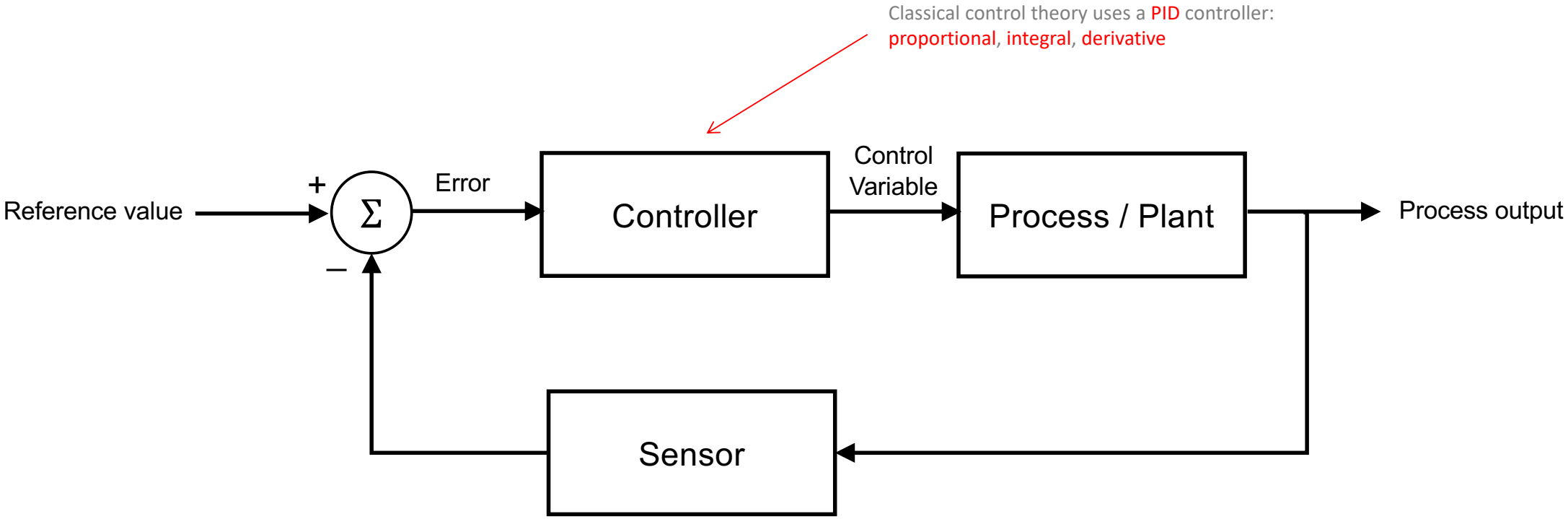
Integral

Derivative

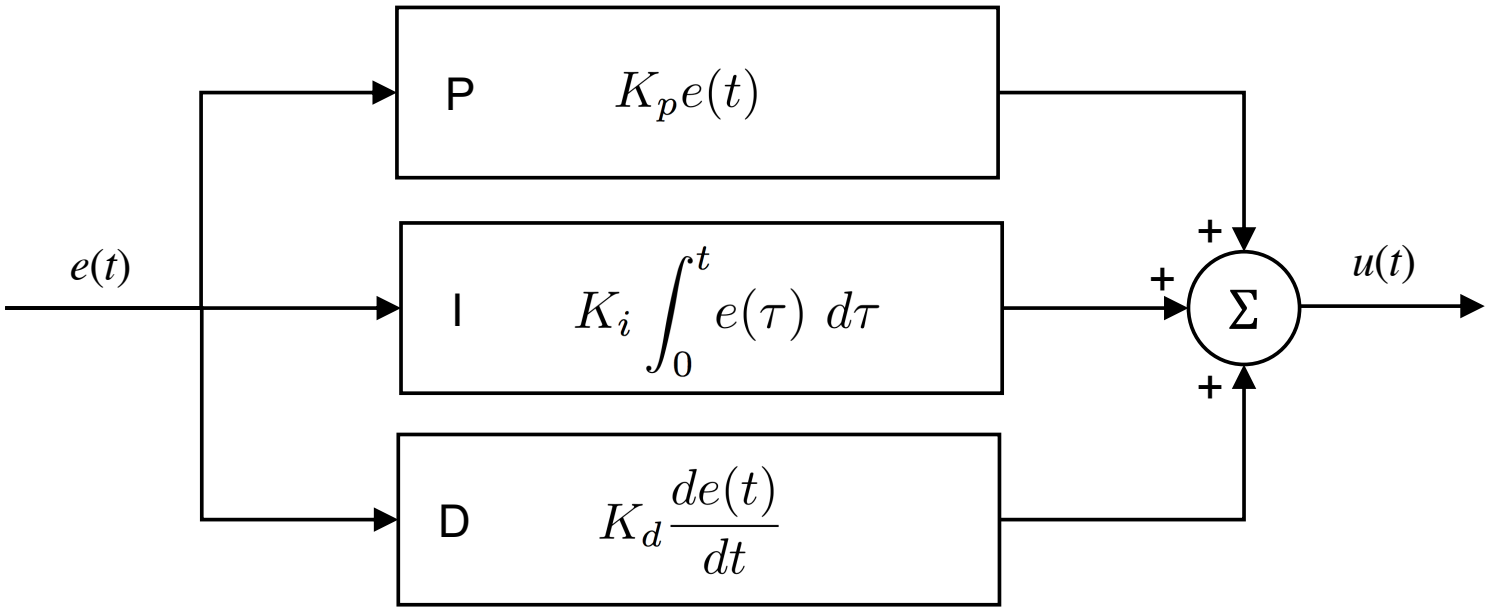
. . . or a combination of these

Each component in is modulated by a respective gain: K_p, K_i, K_d

PID Controller



PID Controller

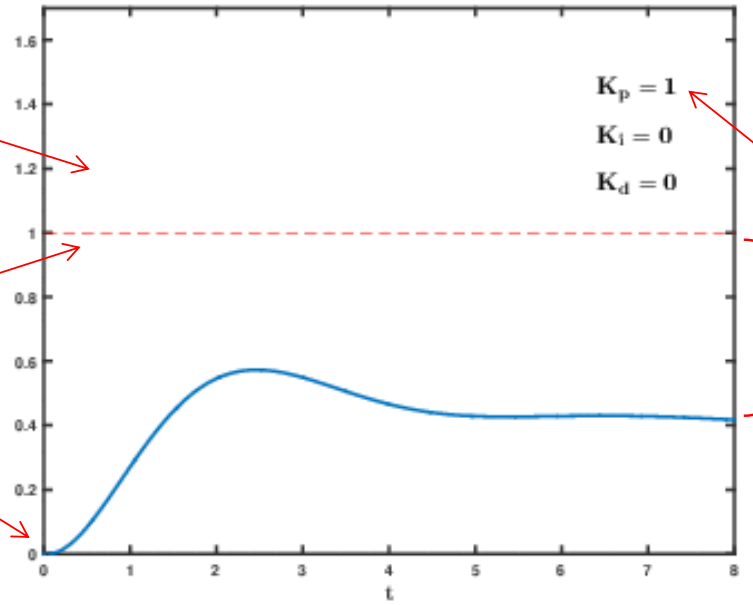


$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

PID Controller

Effect of varying the three gains

The overshoot can be reduced by lowering the K_p but this will increase the time it takes to reduce the error



Setpoint

Initial state of the process variable

Note that there is a steady-state error for pure proportional control,

(i.e. when the gains of the integral and derivative terms are zero).

The integral term eliminates this.

https://en.wikipedia.org/wiki/PID_controller

PID Controller

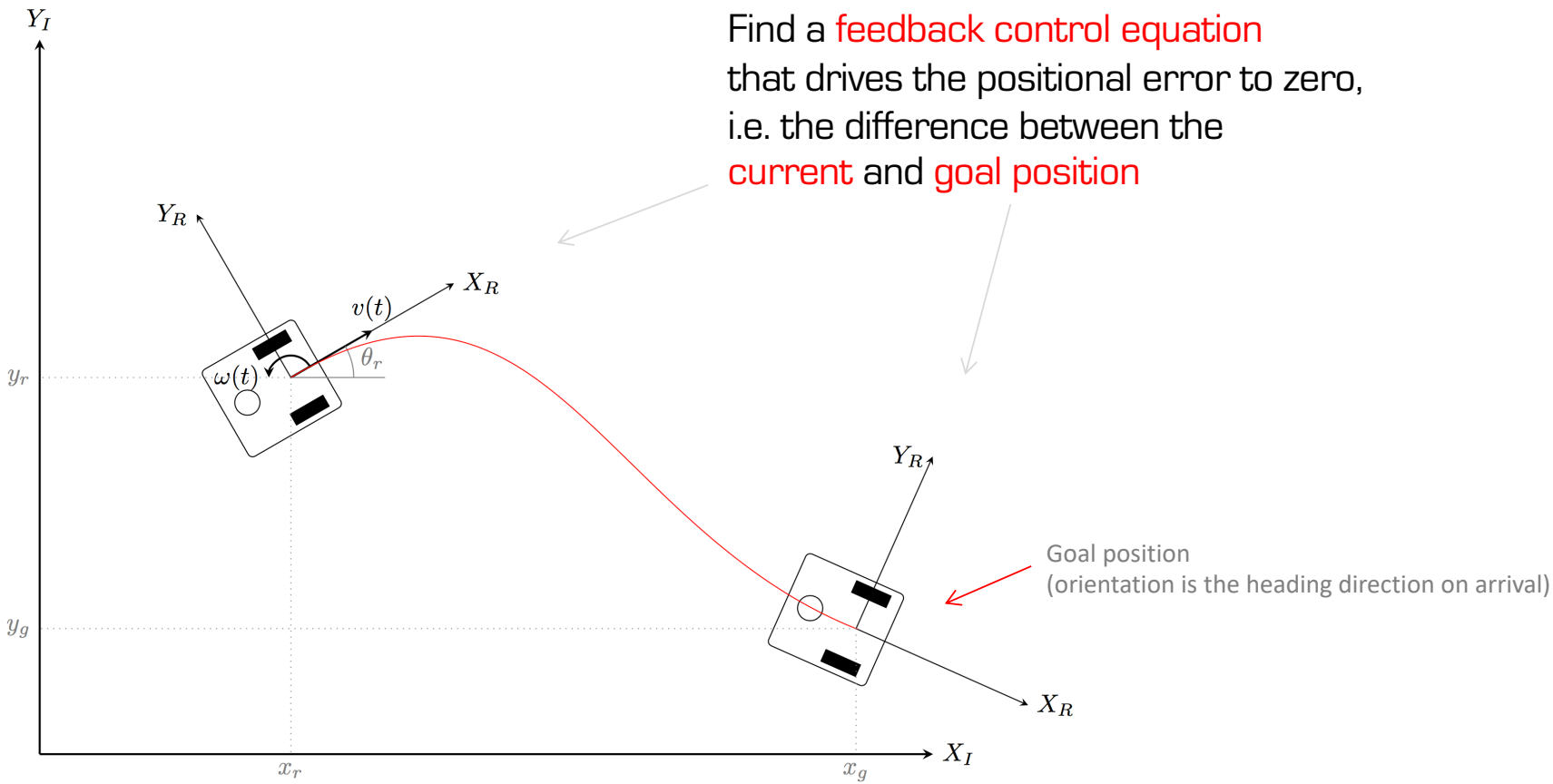
Take-home message

The key to effective PID control is to use the right gain values

but

identifying them is difficult

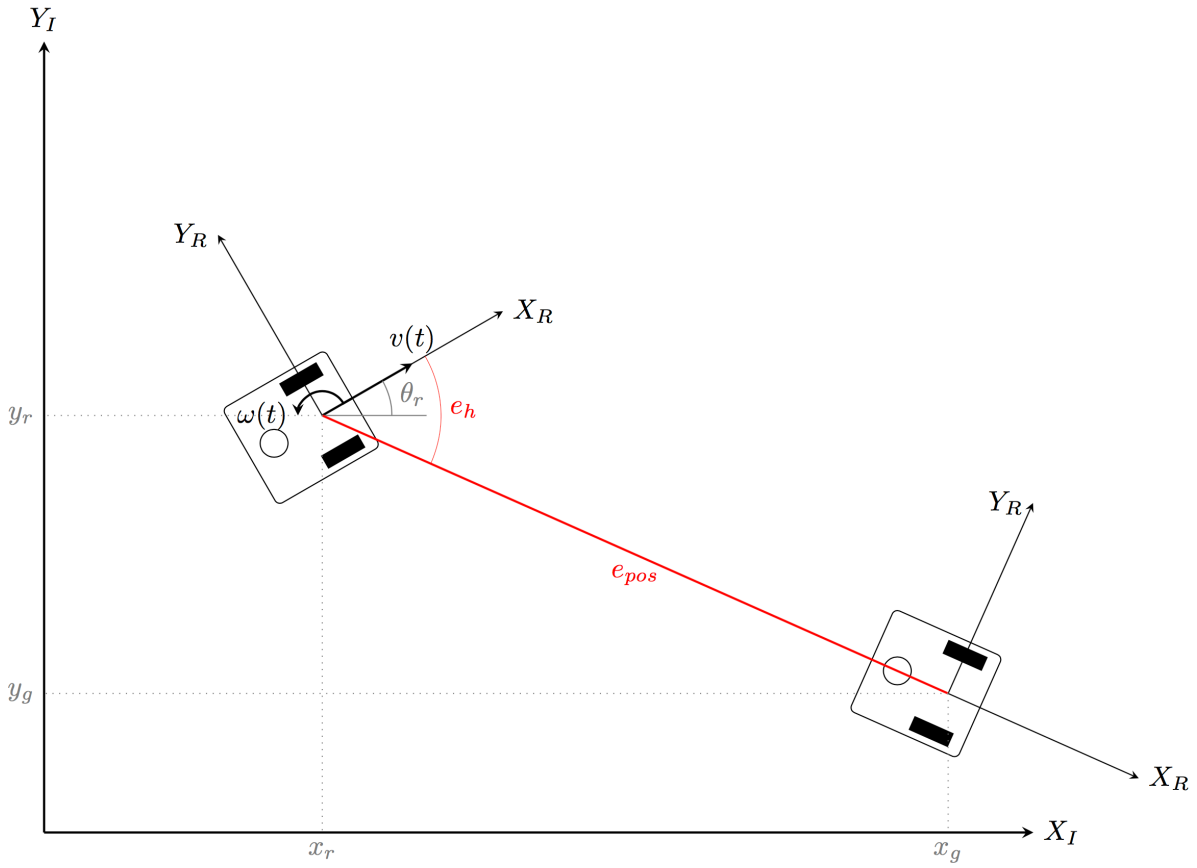
The Go-to-Position Problem



The Go-to-Position Problem

- The robot knows its own global current position
  - (x_r, y_r, θ_r)
- It knows the global position of the goal
  - (x_g, y_g)
- Compute error in position and heading
  - (e_{pos}, e_h)

The Go-to-Position Problem



The Go-to-Position Problem

- The robot knows its own global current position
  - (x_c, y_c, θ_c)
- It knows the global position of the goal
  - (x_g, y_g)
- Compute error in position and heading
  - (e_{pos}, e_h)
- Reduce both errors to zero
  - by generating the appropriate forward and angular velocities (v, ω) or, alternatively,
  - by generating the appropriate angular velocities of the wheels, (v_R, v_L) i.e. $(\dot{\phi}_1, \dot{\phi}_2)$

Go-to-Position as a Control Problem

Solution 1: Divide and Conquer

Decompose 2D problem into two 1D problems

- First, correct the heading:

  rotate to reduce the orientation error to zero

- Second, correct the position:

  translate straight ahead to reduce the position error to zero

Go-to-Position as a Control Problem

Solution 1: Divide and Conquer

Algorithm goto1(x_g, y_g) using proportional control

Global variables: the current robot position and orientation (x_r, y_r, θ_r)

Arguments: - the goal position of the robot (x_g, y_g)

- the proportional gains for controlling position and heading

$$K_p^{pos}$$
$$K_p^{h}$$

- the tolerances on position error Δ_{pos} and orientation error Δ_h

Go-to-Position as a Control Problem

Solution 1: Divide and Conquer

Version A: Control forward and angular velocities of the robot

Do

Compute the current position of the robot (x_r, y_r)

Compute the distances from the robot position to the target position (d_x, dy)

$d_x = x_g - x_r$

$d_y = y_g - y_r$

Compute the position and heading errors(e_{pos}, e_h)

$e_{pos} = \text{sqrt}\ (d_x^2 + d_y^2)$

$e_h = \text{atan2}\ (d_y, d_x) - \theta_r$

The difference between the desired heading and the current robot orientation

If not oriented correctly, correct heading

if $|e_h| > \Delta_h$

set forward velocity: $v = 0$

set angular velocity: $\omega = K_p^h\ e_h$

Otherwise, correct position

else

set forward velocity: $v = K_p^{pos}\ e_{pos}$ or some maximum velocity v_{max}

set angular velocity: $\omega = 0$

Send velocities (v, ω) to the robot

Pause some time

while $|e_{pos}| > \Delta_{pos}$

Send velocities $(0, 0)$ to the robot

Go-to-Position as a Control Problem

Solution 1: Divide and Conquer

Version B: Control angular velocities of the wheels

Do

Compute the current position of the robot (x_r, y_r)

Compute the distances from the robot position to the target position (d_x, dy)

$$d_x = x_g - x_r$$

$$d_y = y_g - y_r$$

Compute the position and heading errors(e_{pos}, e_θ)

$$e_{pos} = \text{sqrt}\ (d_x^2 + d_y^2)$$

$$e_h = \text{atan2}\ (d_y, d_x) - \theta_r$$

If not oriented correctly, correct orientation

if $|e_h| > \Delta_h$

set right wheel angular velocity: $\dot{\phi}_1 = -K_p^h\, e_h$

set left wheel angular velocity: $\dot{\phi}_2 = -\dot{\phi}_1$

Rotate: left and right wheels go in opposite direction

Otherwise, correct position

else

or some maximum velocity

set right wheel angular velocity: $\dot{\phi}_1 = K_p^{pos}\, e_{pos}$

set left wheel angular velocity: $\dot{\phi}_2 = \dot{\phi}_1$

Translate: left and right wheels go in same direction

Send velocities $(\dot{\phi}_1, \dot{\phi}_2)$ to the robot

Pause some time

while $|e_{pos}| > \Delta_{pos}$

Send velocities $(0, 0)$ to the robot