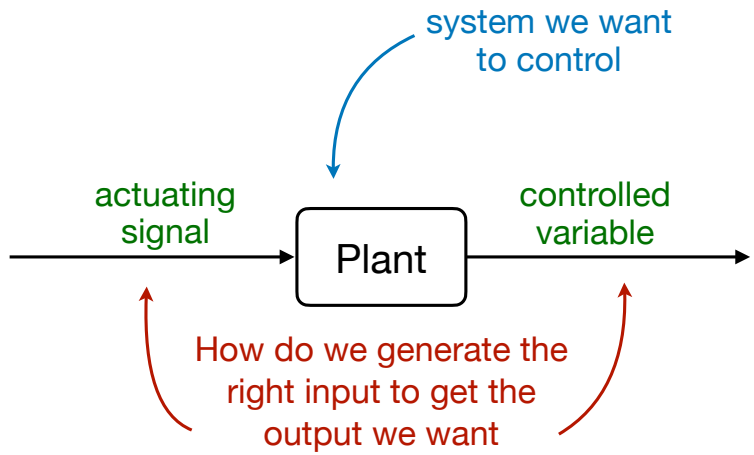
PID control of the Minseg robot

Learning goals

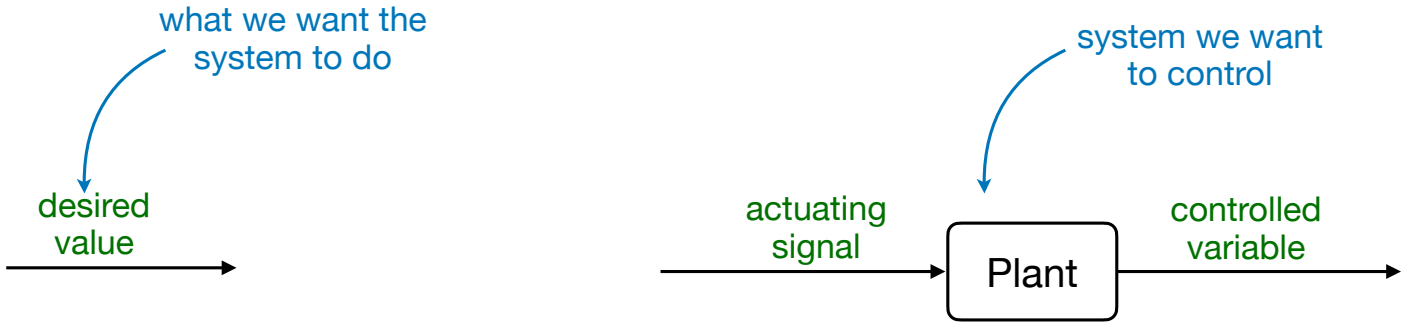
- Further understand the main principles of feedback control
- Understand what is a PID control and what each component does

Feedback Control



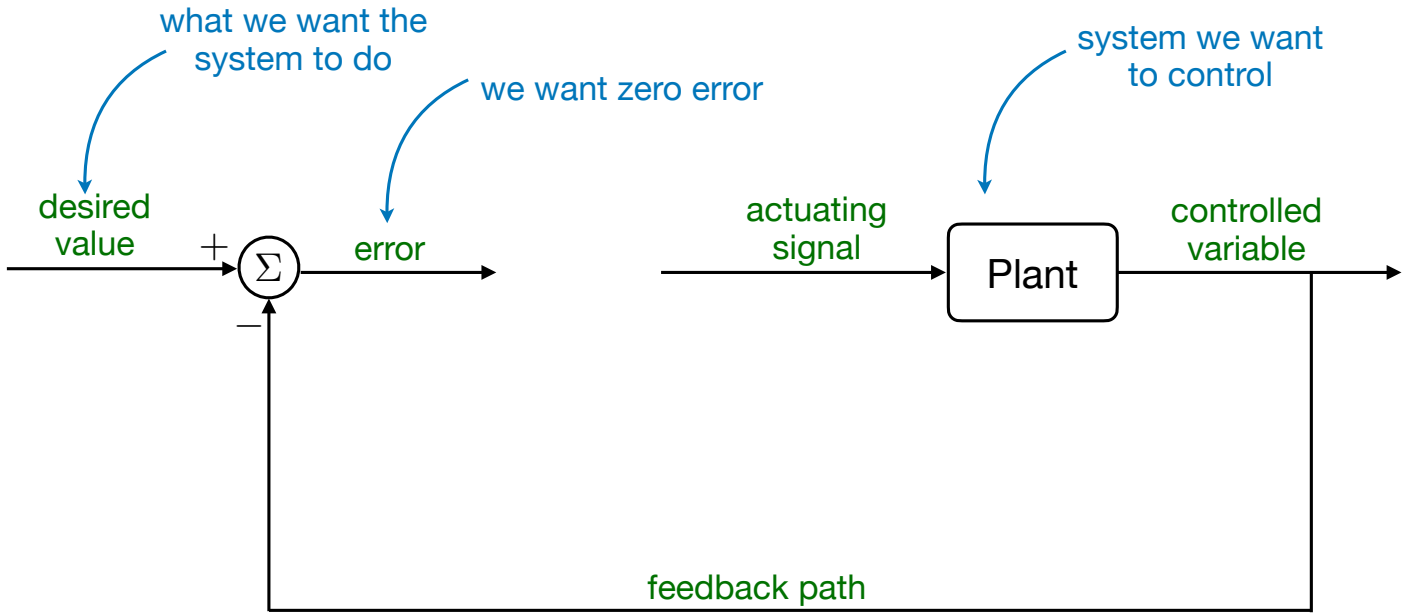
- **Job of the control engineer:** How do we produce the appropriate actuating signal, so that the plant produces a controlled variable that is equal to the desired output?

Feedback Control



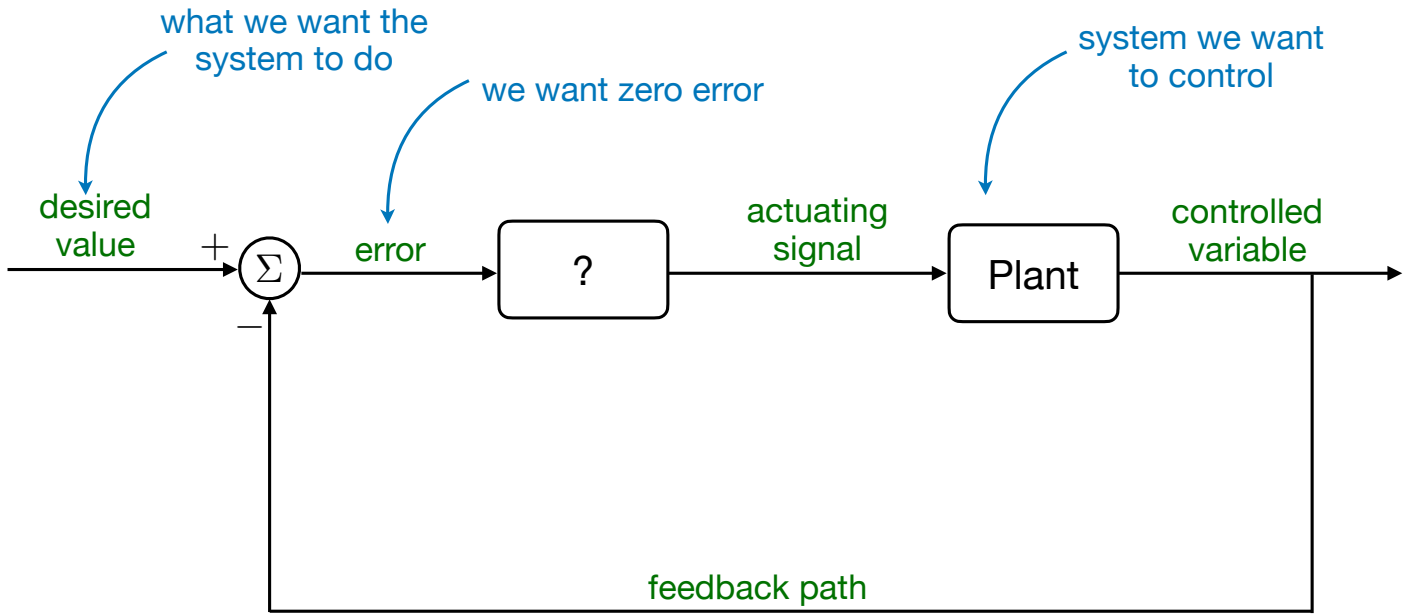
- **Job of the control engineer:** How do we produce the appropriate actuating signal, so that the plant produces a controlled variable that is equal to the desired output?
- The desired output is designated by a desired value/reference.

Feedback Control



- **Job of the control engineer:** How do we produce the appropriate actuating signal, so that the plant produces a controlled variable that is equal to the desired output?
- The desired output is designated by a desired value/reference.
- In feedback control, the controlled variable is fed back (and hence the name) and compared to the desired value. The difference between the two is the error term.

Feedback Control



- **Job of the control engineer:** How do we produce the appropriate actuating signal, so that the plant produces a controlled variable that is equal to the desired output?
- The desired output is designated by a desired value/reference.
- In feedback control, the controlled variable is fed back (and hence the name) and compared to the desired value. The difference between the two is the error term.
- The question is: How do we take this error and transform it to suitable actuating signals so that the error goes to zero?

Proportional Controller

- Obvious approach: multiply the error by a gain k_p
- So, a bigger error leads to a larger actuating signal end hence to a bigger effect onto the system.

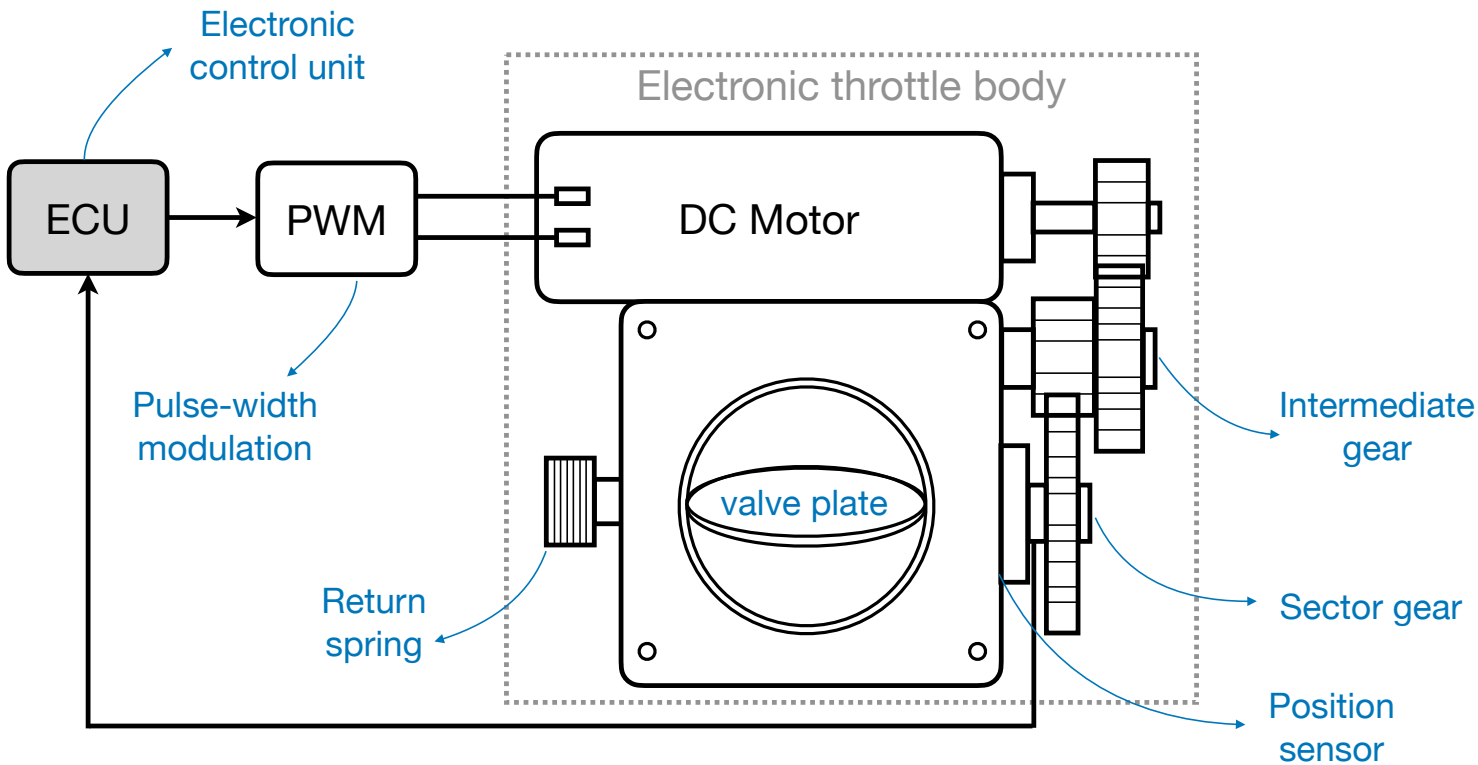
Proportional Controller

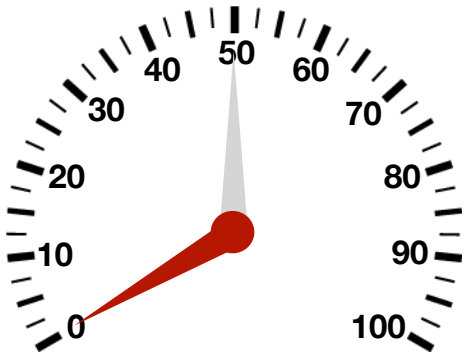
- Obvious approach: multiply the error by a gain k_p
- So, a bigger error leads to a larger actuating signal end hence to a bigger effect onto the system.
- The actuating signal would then be:

$$u(t) = k_p \times e(t)$$

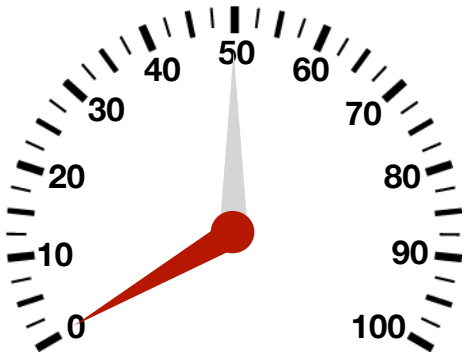
Proportional Controller: example

- Let us consider a simple example: **automatic cruise control (ACC)**
  - The throttle/valve plate rotates within the throttle body, opening the throttle passage to allow more air into the intake manifold.

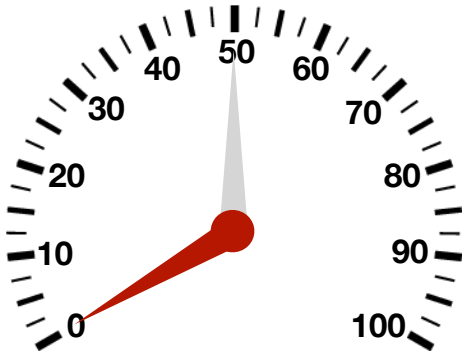




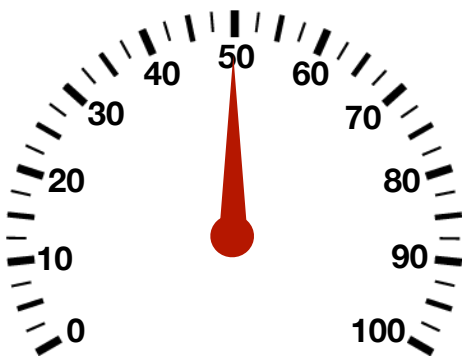
- Initially, the speed is zero (so the error is 50 km/h)



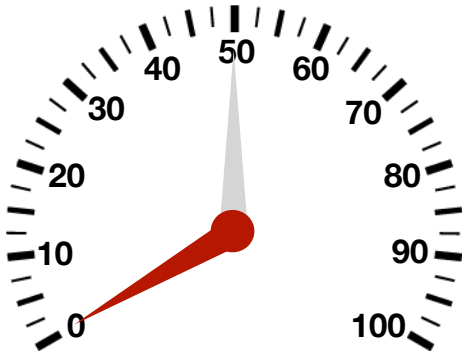
- Initially, the speed is zero (so the error is 50 km/h)
- The throttle plate will open and the car will start moving, thus reducing the error.



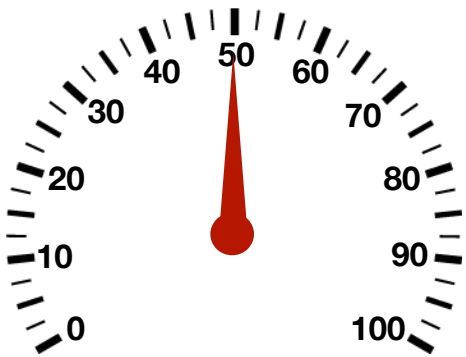
- Initially, the speed is zero (so the error is 50 km/h)
- The throttle plate will open and the car will start moving, thus reducing the error.



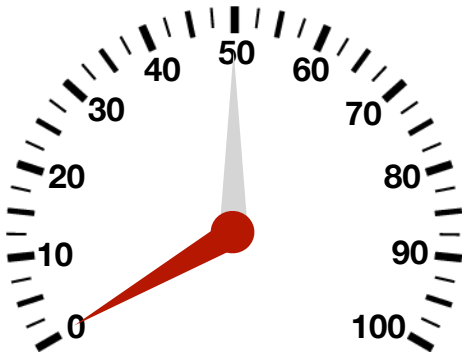
- What if the speed is already at 50 km/h?



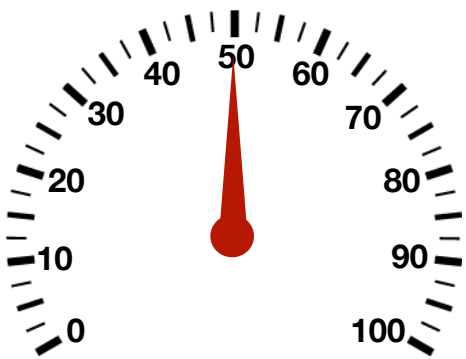
- Initially, the speed is zero (so the error is 50 km/h)
- The throttle plate will open and the car will start moving, thus reducing the error.



- What if the speed is already at 50 km/h?
- The error would become zero, and the throttle valve would close and the car would start reducing speed.
- When that happens, the error increases and the throttle valve opens again.



- Initially, the speed is zero (so the error is 50 km/h)
- The throttle plate will open and the car will start moving, thus reducing the error.



- What if the speed is already at 50 km/h?
- The error would become zero, and the throttle valve would close and the car would start reducing speed.
- When that happens, the error increases and the throttle valve opens again.
- There exists a throttle angle such that the speed remains at 50 km/h.

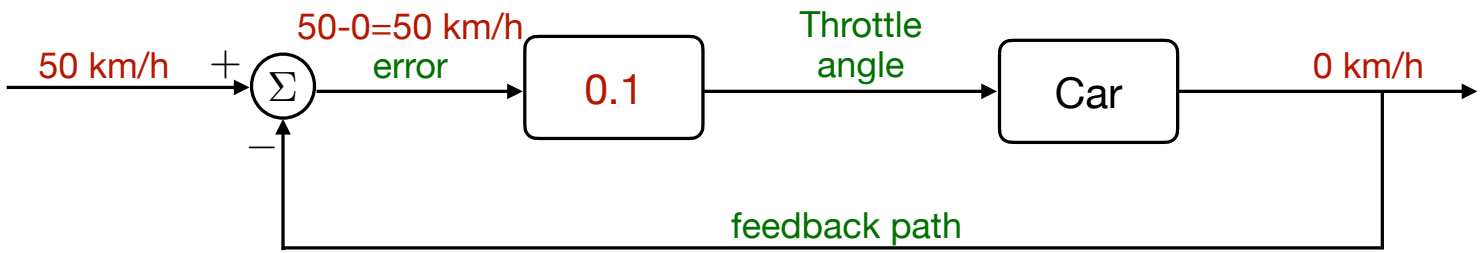
How does the proportional controller make the ACC stay at that speed?

How does the proportional controller make the ACC stay at that speed?

- Let's assume that the car needs a throttle angle of 20° to maintain its speed constant.
- Also, suppose that the throttle angle is given by

$$\text{Throttle angle} = \text{error} \times \text{gain}$$

- If the gain is 0.1



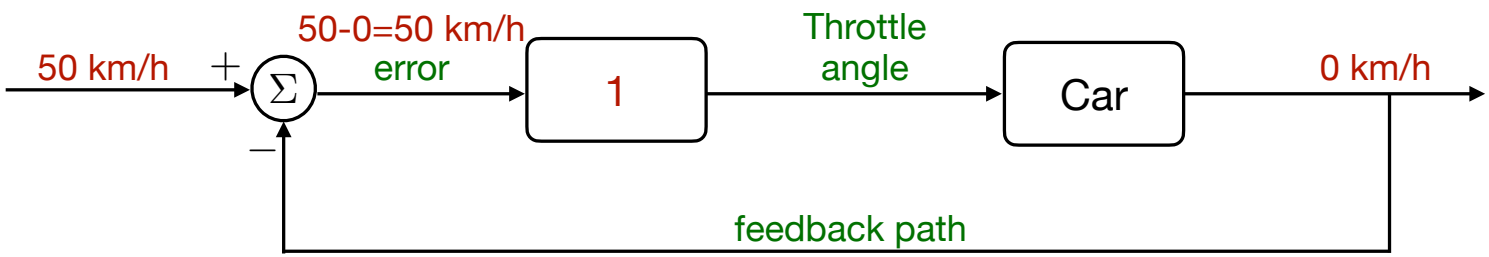
the throttle angle is 5° and while the error is reduced the car will never reach 50 km/h.

How does the proportional controller make the ACC stay at that speed?

- Let's assume that the car needs a throttle angle of 20° to maintain its speed constant.
- Also, suppose that the throttle angle is given by

$$\text{Throttle angle} = \text{error} \times \text{gain}$$

- If the gain is 1



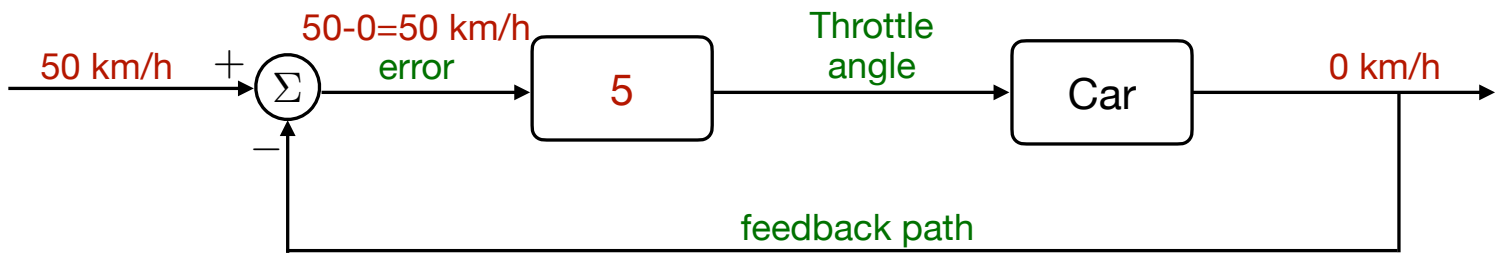
the throttle angle is 50° and while the error is reduced, when the error becomes 20 km/h (i.e.,when the cars has speed 30 km/h), the car will stop accelerating since the throttle angle will be 20°.

How does the proportional controller make the ACC stay at that speed?

- Let's assume that the car needs a throttle angle of 20° to maintain its speed constant.
- Also, suppose that the throttle angle is given by

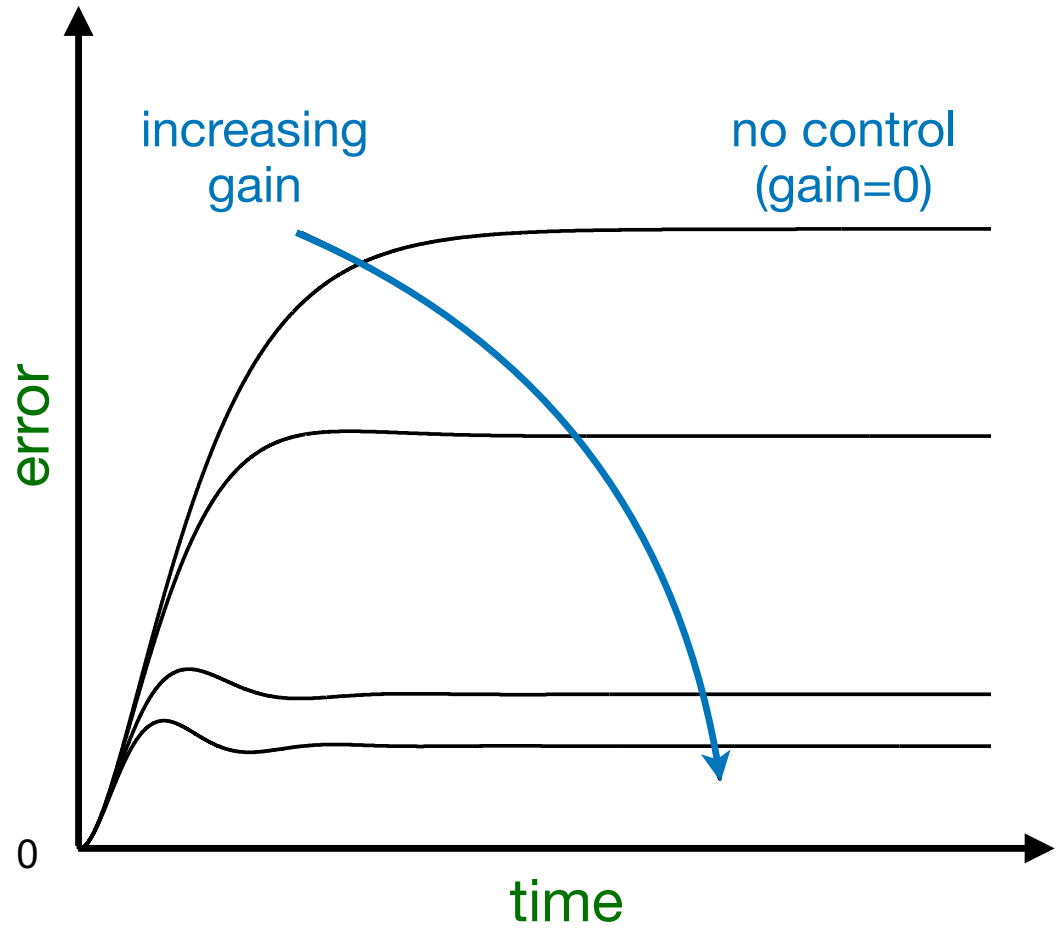
$$\text{Throttle angle} = \text{error} \times \text{gain}$$

- If the gain is 5



when the error becomes 4 km/h (i.e.,when the cars has speed 46 km/h), the car will stop accelerating since the throttle angle will be 20°.

Error does not vanish, it just gets smaller!



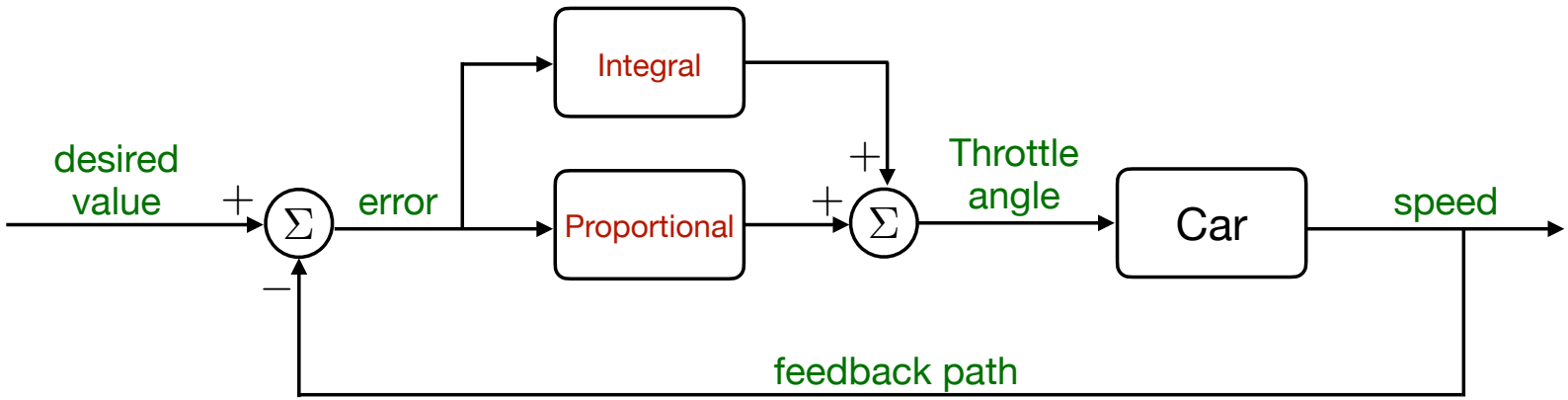
- The error that remains is called steady state error
- How can we get rid of this?

Proportional-Integral (PI) Control

- The idea is to use an integral action (memory) as well.
- Small errors that persist over a long time might indeed be small and hence lead to small values for the proportional controller. But their integral, so the are underneath the curve gets larger overt time:
- Small errors will also have a large (growing) effect over time so that the controller can react to it.
- When the system has error, this is integrated over time, thus increasing the integral term, and as a result, the throttle angle.
- It stops increasing when the error is zero.

Proportional-Integral (PI) Control

- The system then has a controller with two parts:



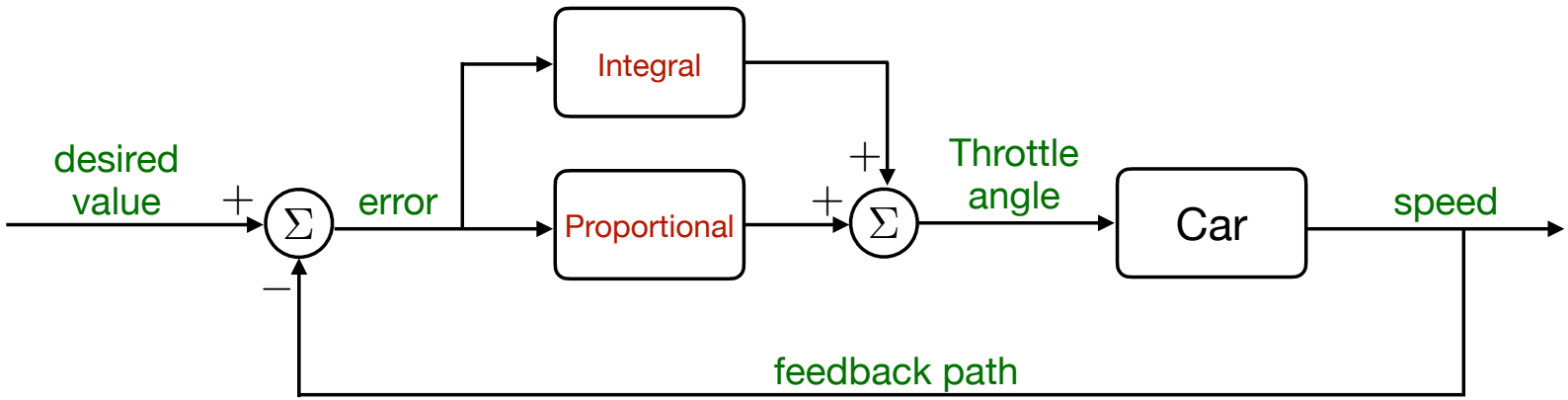
- The actuation signal (here: throttle angle) would then be:

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt$$

- When the system has a steady state error, this error is integrated over time, thus increasing the integral term, and as a result, the throttle angle.

Proportional-Integral (PI) Control

- The system then has a controller with two parts:



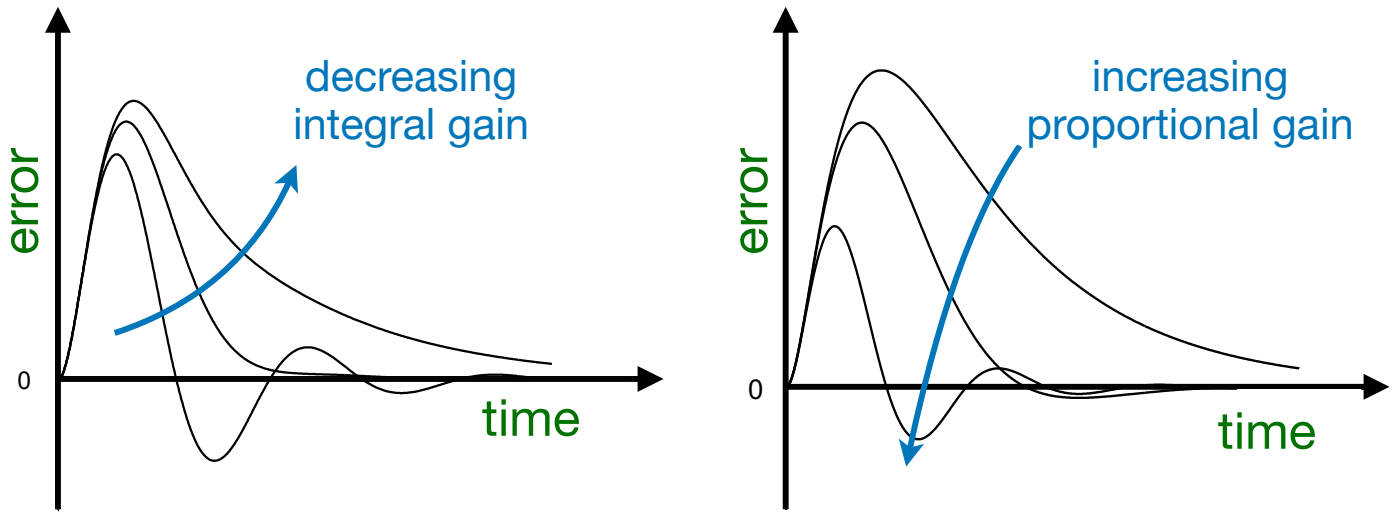
- The actuation signal (here: throttle angle) would then be:

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt$$

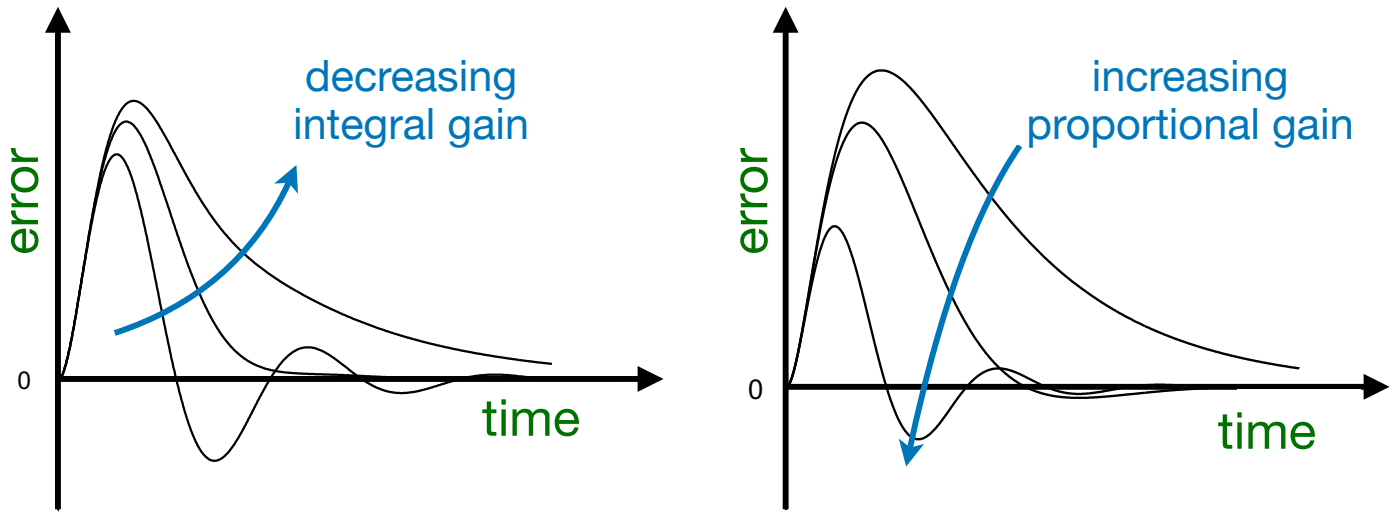
- When the system has a steady state error, this error is integrated over time, thus increasing the integral term, and as a result, the throttle angle.

- An integral term increases action in relation not only to the error, but also the time for which it has persisted $\Longrightarrow$ if applied control action is not enough to bring the error to zero, this control action will be increased as time passes.

- Pro: The integral action can bring the error to zero.
- Con: However, if one chooses a small value of k_i it is slow to react (because the integral grows over time), and if k_i is large, it will create overshoots and oscillations because the integral part only stops changing if the error is zero:

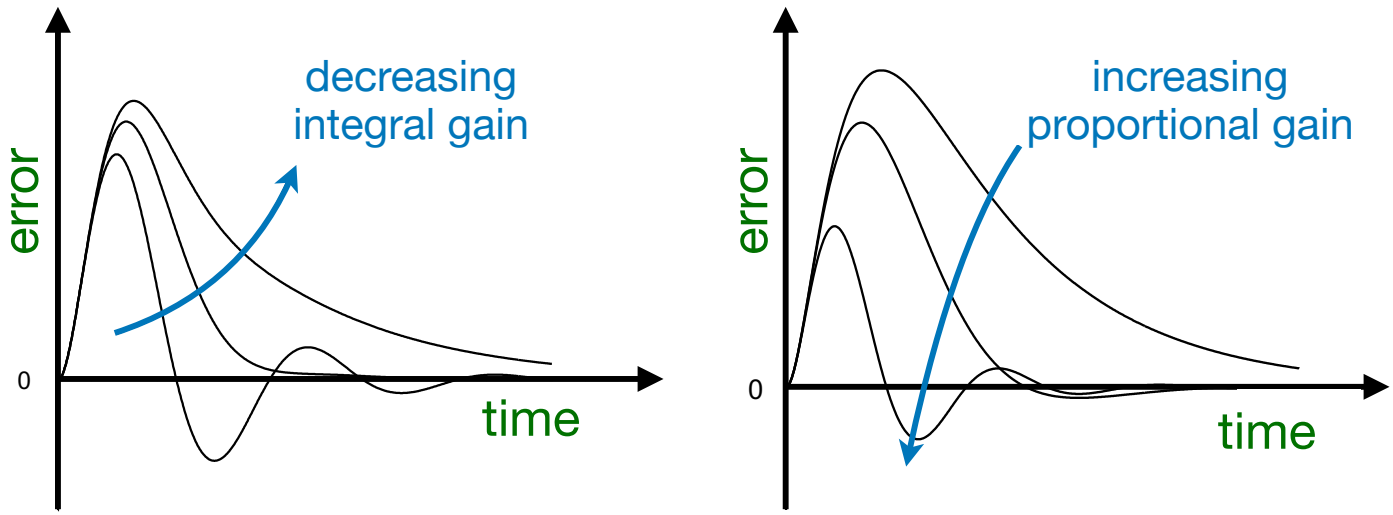


- Pro: The integral action can bring the error to zero.
- Con: However, if one chooses a small value of k_i it is slow to react (because the integral grows over time), and if k_i is large, it will create overshoots and oscillations because the integral part only stops changing if the error is zero:



- These overshoots and oscillations are usually not desired.

- Pro: The integral action can bring the error to zero.
- Con: However, if one chooses a small value of k_i it is slow to react (because the integral grows over time), and if k_i is large, it will create overshoots and oscillations because the integral part only stops changing if the error is zero:



- These overshoots and oscillations are usually not desired.
- Is there any way to compensate for that?

Proportional-Integral-Derivative (PID) Control

- The idea is to predict the future and respond to how fast the error is changing:

- The derivative control term
  - looks at the current rate of change of error (the faster the error is growing, the larger the derivative term becomes),
  - determines how fast the desired value is approached, and
  - prematurely reduces the throttle angle.

Proportional-Integral-Derivative (PID) Control

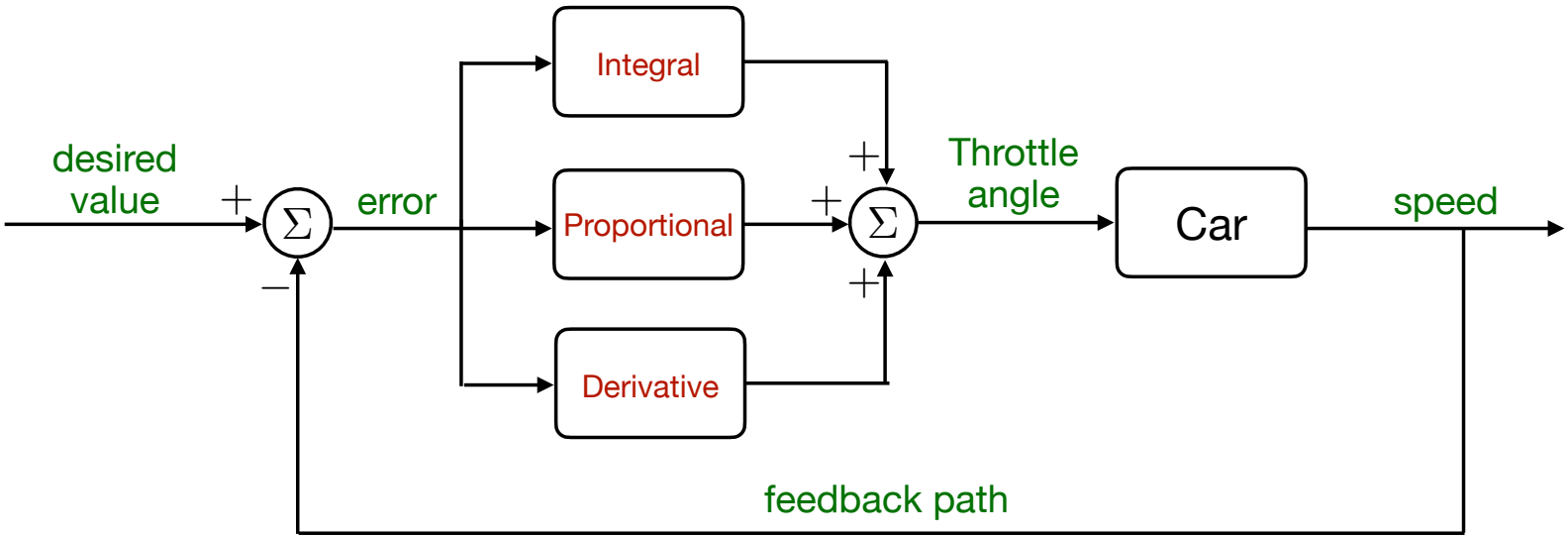
- The idea is to predict the future and respond to how fast the error is changing:

- The derivative control term
  - looks at the current rate of change of error (the faster the error is growing, the larger the derivative term becomes),
  - determines how fast the desired value is approached, and
  - prematurely reduces the throttle angle.

How are these concepts applied to our robot?

Proportional-Integral-Derivative (PID) Control

- The system then has a controller with three parts:

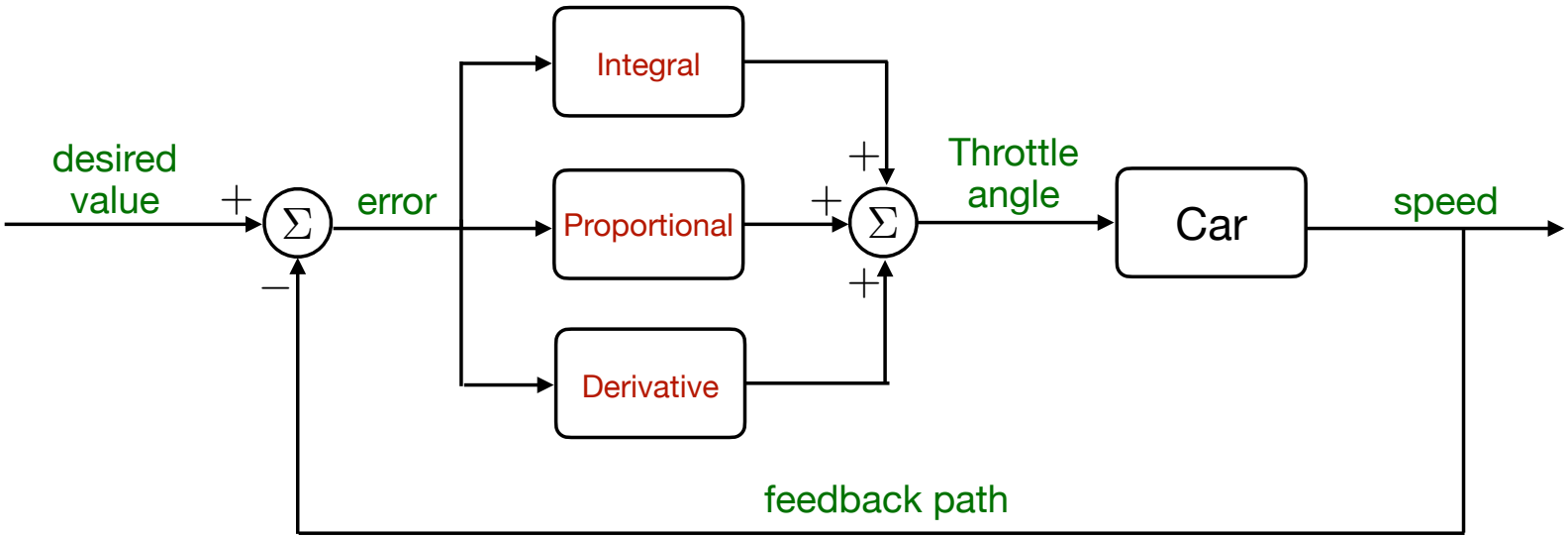


- The actuation signal (here: throttle angle) would then be:

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt + k_d \times \frac{de(t)}{dt}$$

Proportional-Integral-Derivative (PID) Control

- The system then has a controller with three parts:



- The actuation signal (here: throttle angle) would then be:

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt + k_d \times \frac{de(t)}{dt}$$

How are these concepts applied to our robot?

PID Control at the Minseg Robot

See the controller implementation in the library “SEG_CONTROL”!
How is the equation for the actuating signal implemented?

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt + k_d \times \frac{de(t)}{dt}$$

PID Control at the Minseg Robot

See the controller implementation in the library “SEG_CONTROL”!
How is the equation for the actuating signal implemented?

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt + k_d \times \frac{de(t)}{dt}$$

Proportional part:

```
Vout1 = Kp * error + Kd * dTerm + integralTerm;
```

PID Control at the Minseg Robot

See the controller implementation in the library "SEG_CONTROL"!
How is the equation for the actuating signal implemented?

$$u(t) = k_p \times e(t) + k_i \times \int_0^t e(t)dt + k_d \times \frac{de(t)}{dt}$$

Proportional part:

```
Vout1 = Kp * error + Kd * dTerm + integralTerm;
```

How are the integral and derivative parts being calculated?

Integral (I) Control at the Minseg Robot

- We must calculate the integral numerically.
- We do not know the error for all the times but only for some moments where we measured it.

Integral (I) Control at the Minseg Robot

- We must calculate the integral numerically.
- We do not know the error for all the times but only for some moments where we measured it.
- This leads to an approximation of the integral as areas of rectangles

$$\int_0^t e(t)dt \approx \sum_{t_i=t_1}^{t_i \leq t} e(t_i) \times (t_{i-1} - t_i)$$

The rectangles have height $e(t_i)$ and width $t_{i-1} - t_i$.

Integral (I) Control at the Minseg Robot

- We must calculate the integral numerically.
- We do not know the error for all the times but only for some moments where we measured it.
- This leads to an approximation of the integral as areas of rectangles

$$\int_0^t e(t)dt \approx \sum_{t_i=t_1}^{t_i \leq t} e(t_i) \times (t_{i-1} - t_i)$$

  The rectangles have height $e(t_i)$ and width $t_{i-1} - t_i$.

- In the code it reads (including the factor k_i):
  `integralTerm += error * actualDt * Ki;`

Derivative (D) Control at the Minseg Robot

- We must calculate the derivative numerically.
- We do not know the error for all the times but only for some moments where we measured it.

Derivative (D) Control at the Minseg Robot

- We must calculate the derivative numerically.
- We do not know the error for all the times but only for some moments where we measured it.
- This leads to an approximation of the derivative as a slope:

$$\frac{de(t)}{dt} \approx \frac{e(t_i) - e(t_{i-1})}{t_i - t_{i-1}}$$

The slope follows from the height of the triangle $e(t_i) - e(t_{i-1})$ and its width $t_i - t_{i-1}$.

Derivative (D) Control at the Minseg Robot

- We must calculate the derivative numerically.
- We do not know the error for all the times but only for some moments where we measured it.
- This leads to an approximation of the derivative as a slope:

$$\frac{de(t)}{dt} \approx \frac{e(t_i) - e(t_{i-1})}{t_i - t_{i-1}}$$

  The slope follows from the height of the triangle $e(t_i) - e(t_{i-1})$ and its width $t_i - t_{i-1}$.

- In the code it reads:
  `dTerm = (error - lastErr) / actualDt;`