Introduction to Control

Problem: Move your robot to a desired state

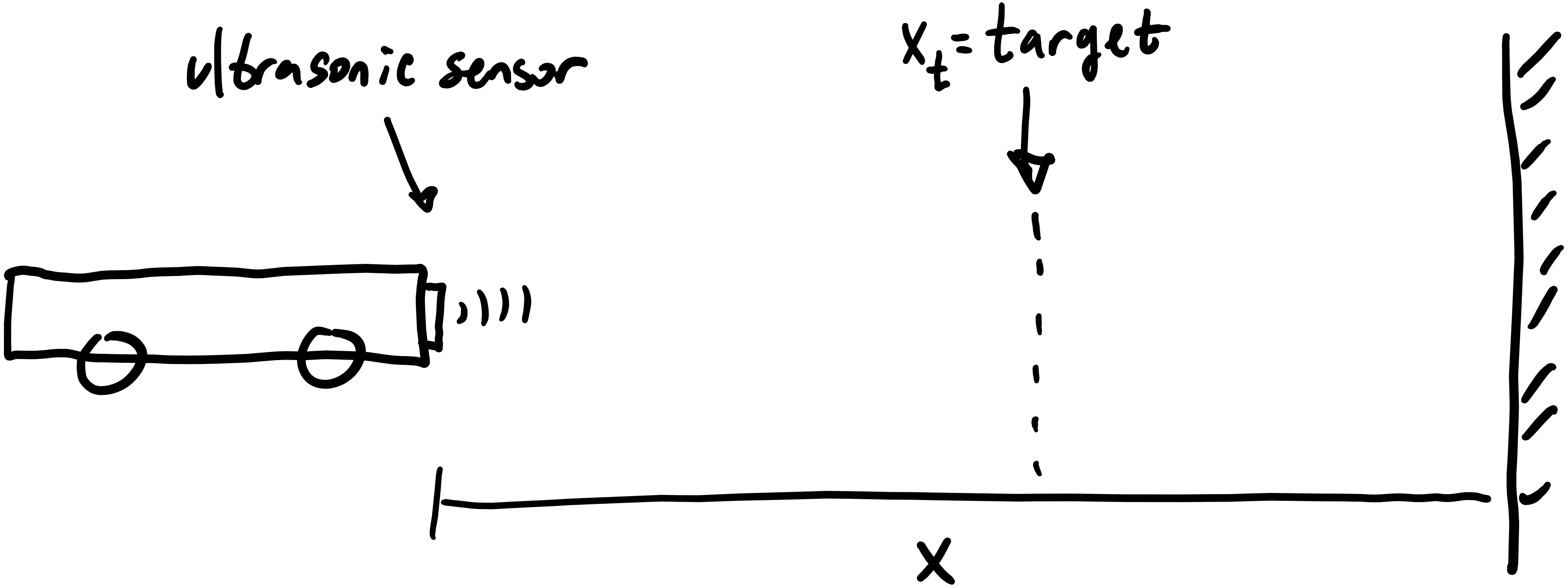
- Move an arm to a target position
- Follow a wall
- Maintain a given speed
- Maintain a set distance from another robot

Solution: Let’s implement “control”!

- *High-level control* typically refers to goal-seeking at the system level:
  - Decide which way to drive
  - Decide if it’s time to get recharged
  - Grasp an object

- *Low-level control* focuses on components:
  - Control the speed of a motor
  - Control the temperature in a vessel
  - Control the position of an arm joint

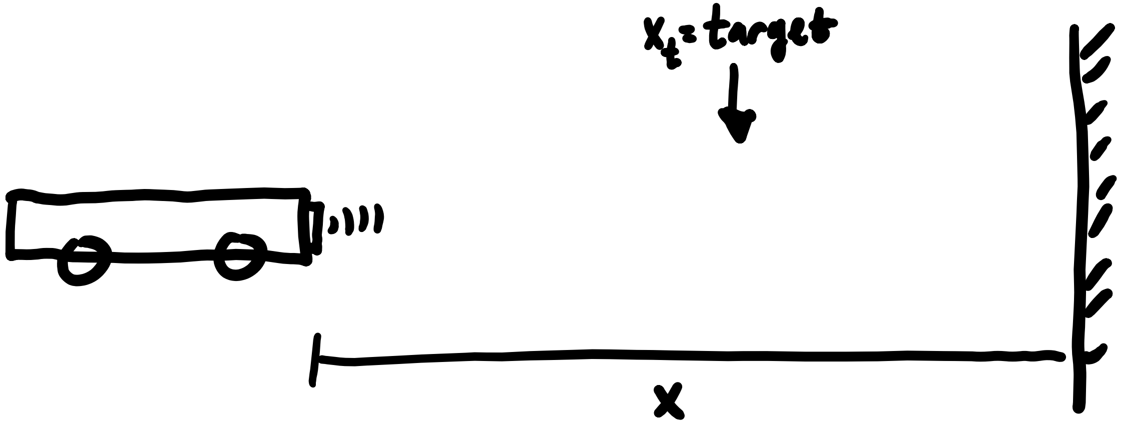
Controlling the distance from a wall



Let's set the speed of the BaseBot to be *proportional* to the error

$$\text{error} = e = x_t - x$$

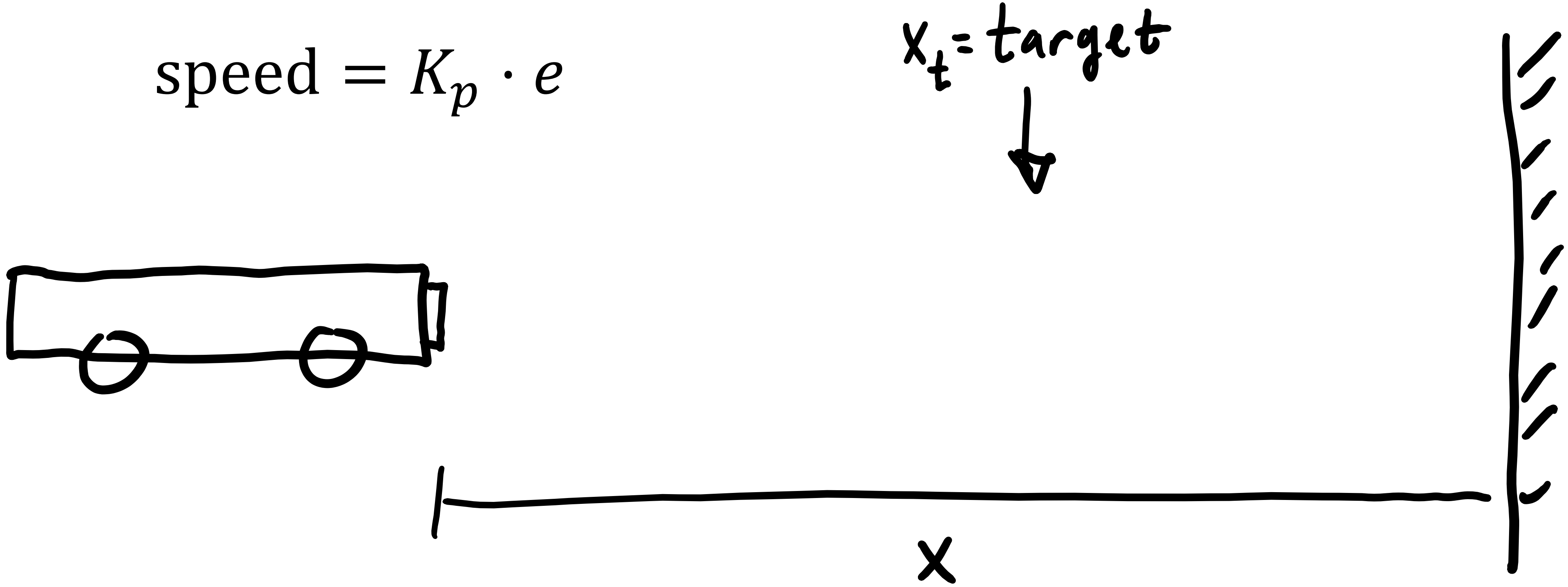
$$\text{speed} = K_p \cdot e$$



What will happen?

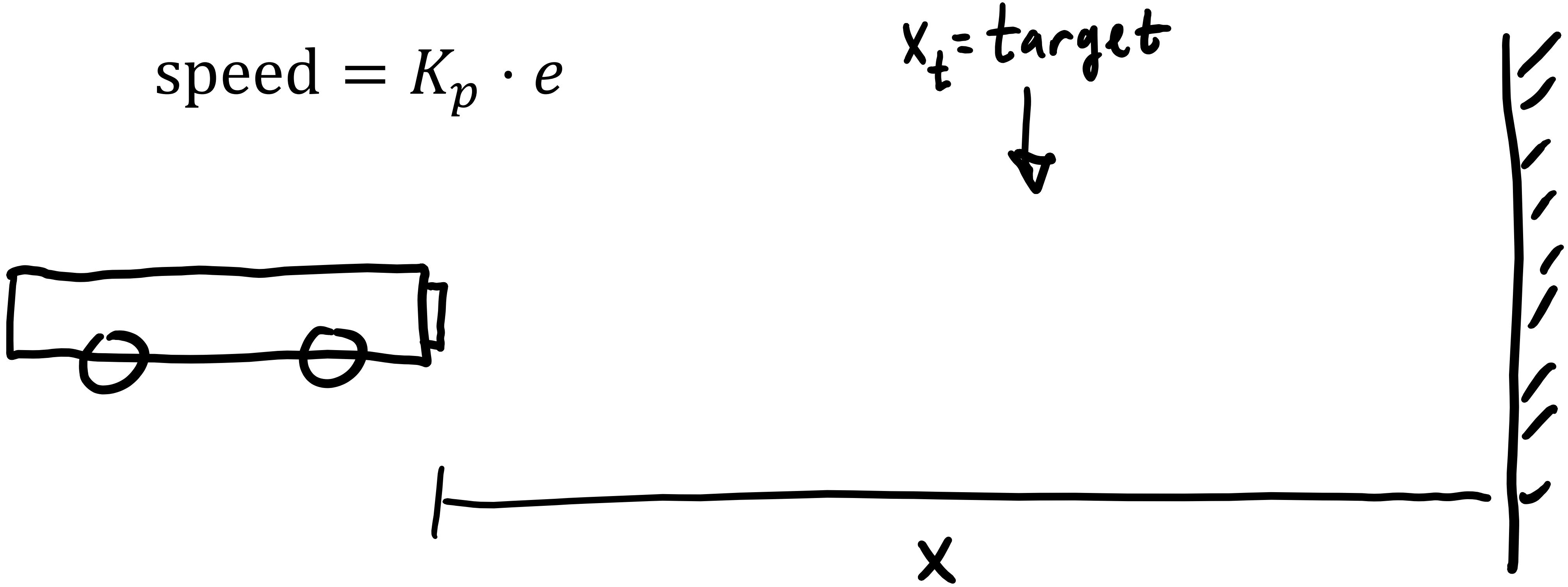
What will happen if K_p is small?

$$\text{speed} = K_p \cdot e$$



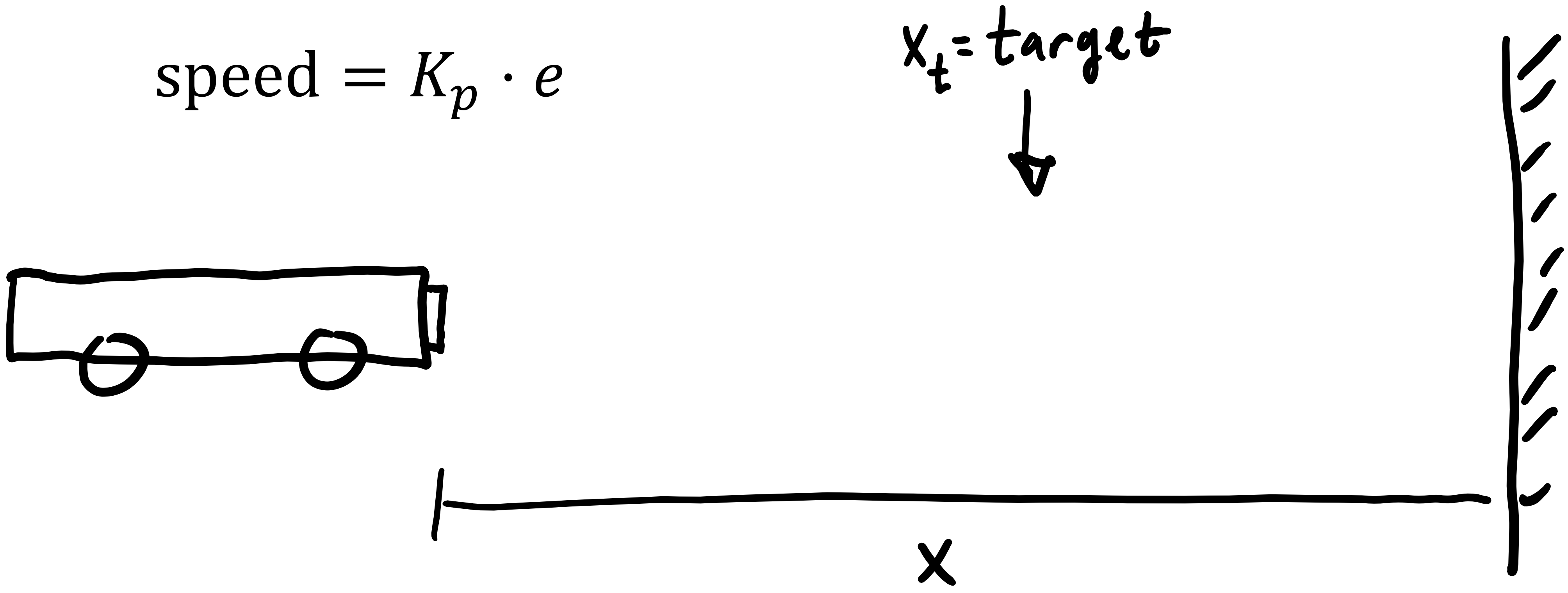
What will happen if K_p is large?

$$\text{speed} = K_p \cdot e$$



What will happen if K_p is “juuuuuust right”?

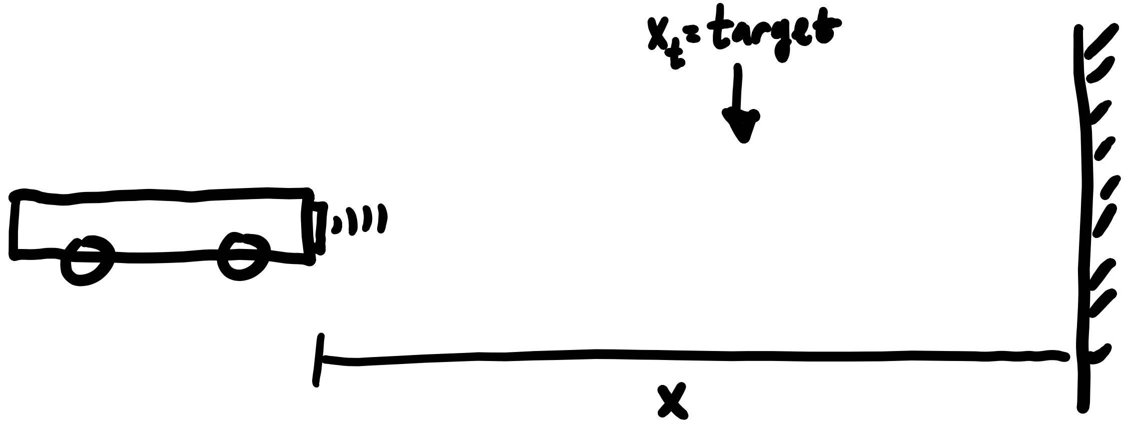
$$\text{speed} = K_p \cdot e$$



Let's look at the math

$$e = x_t - x$$

$$\text{speed} = \frac{dx}{dt} = K_p \cdot e$$



$$\frac{dx}{dt} = \frac{dx}{dt} - \frac{dx_t}{dt} = -\frac{de}{dt} = K_p \cdot e$$

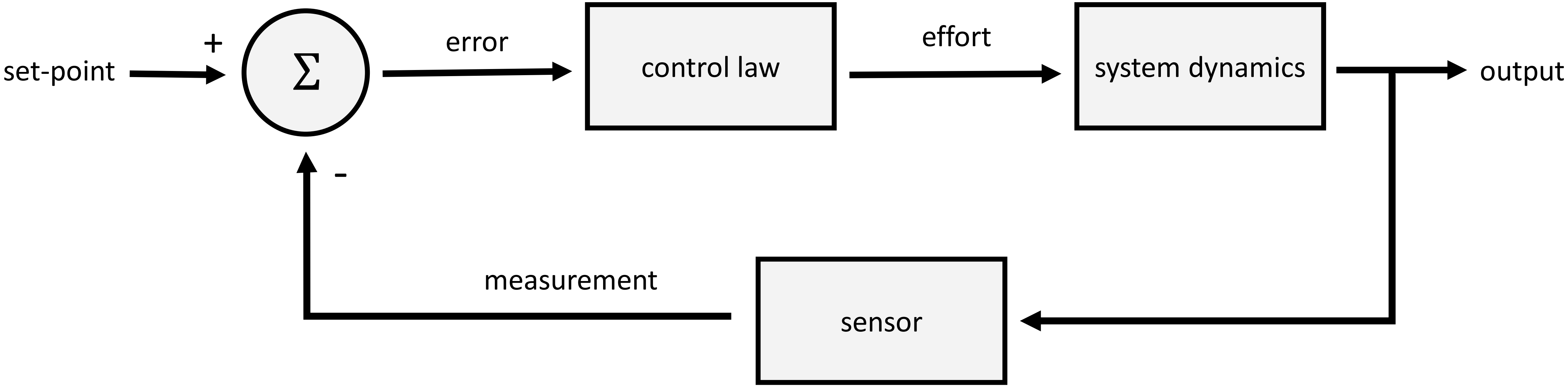
$$\frac{de}{dt} + K_p \cdot e = 0$$

$$\boxed{e(t) = e(0)e^{-K_p t}}$$

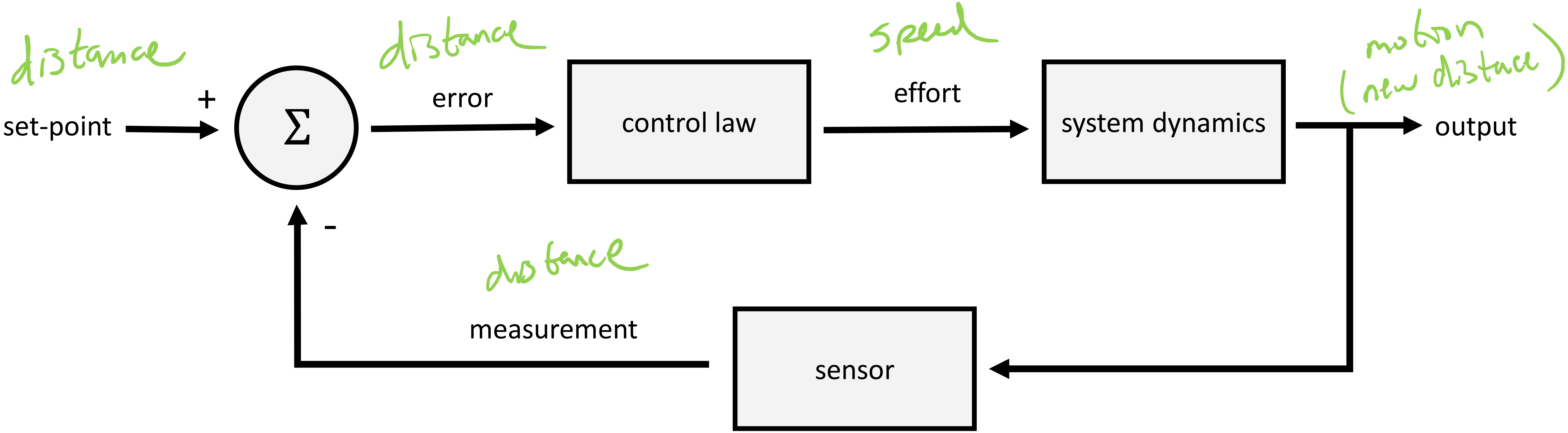
In <u>theory</u>, the error goes to zero as *t* goes to infinity!

(In practice, we ignored a lot of details to get this result.)

Block diagram for a feedback control process



Example: object stand-off



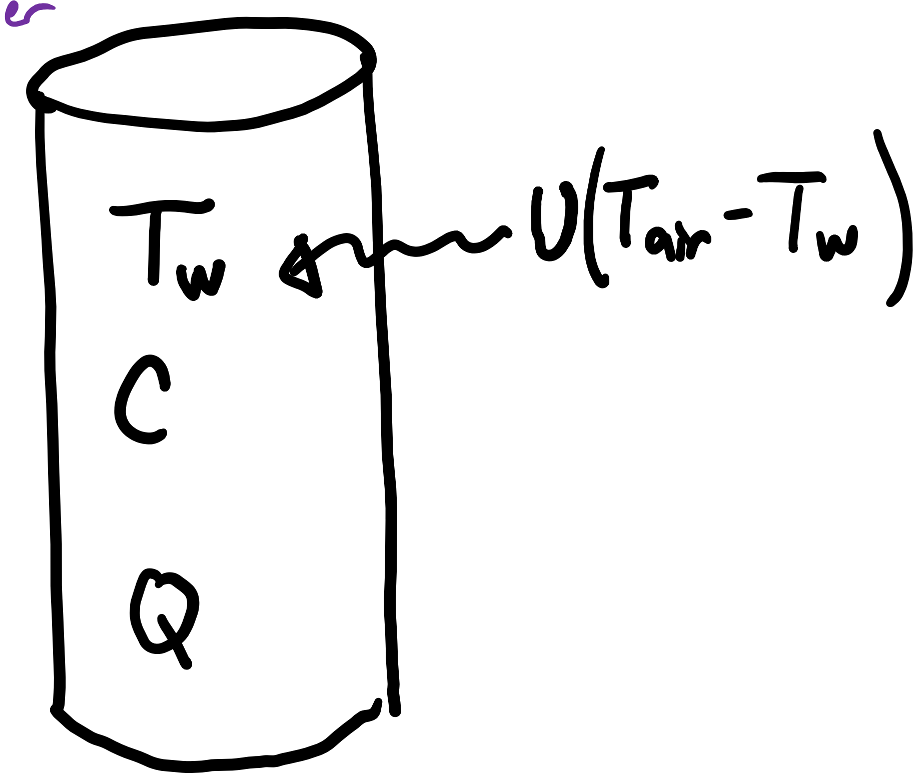
Example: temperature in a water heater

Start with the *governing equation*:

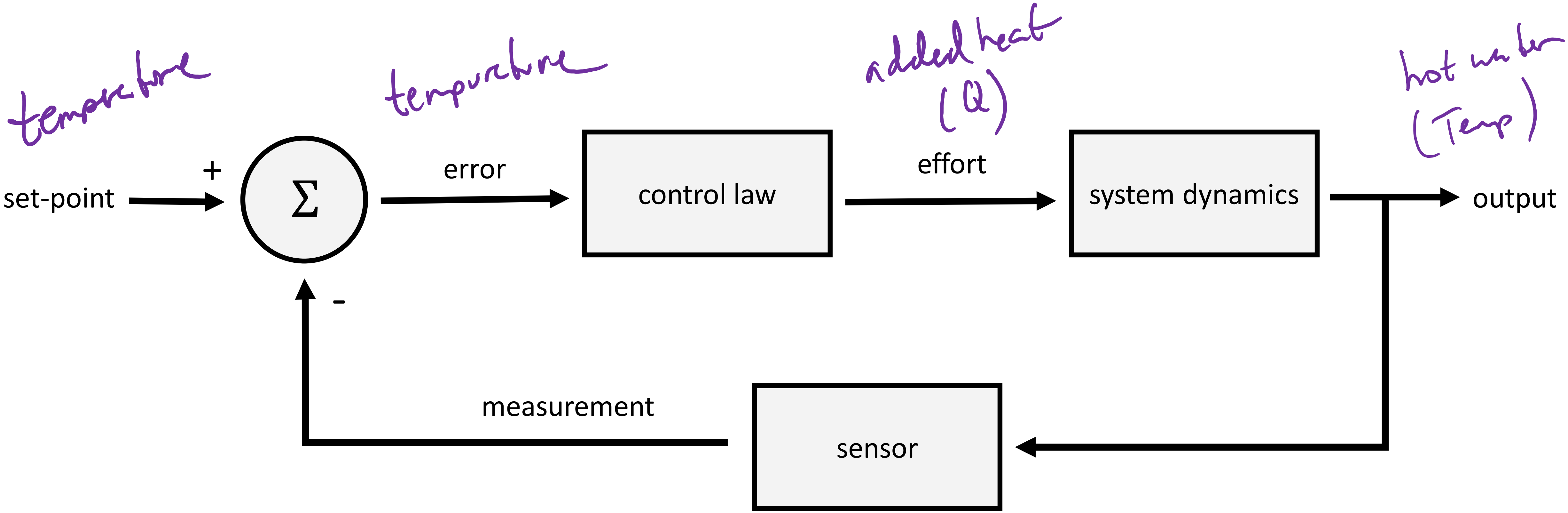
heat from burner

$$\frac{dT_w}{dt} = \frac{U}{C}(T_{air} - T_w) + \frac{1}{C}Q$$

heat +/- from outside



Example: water heater temperature control



Example: temperature in a water heater

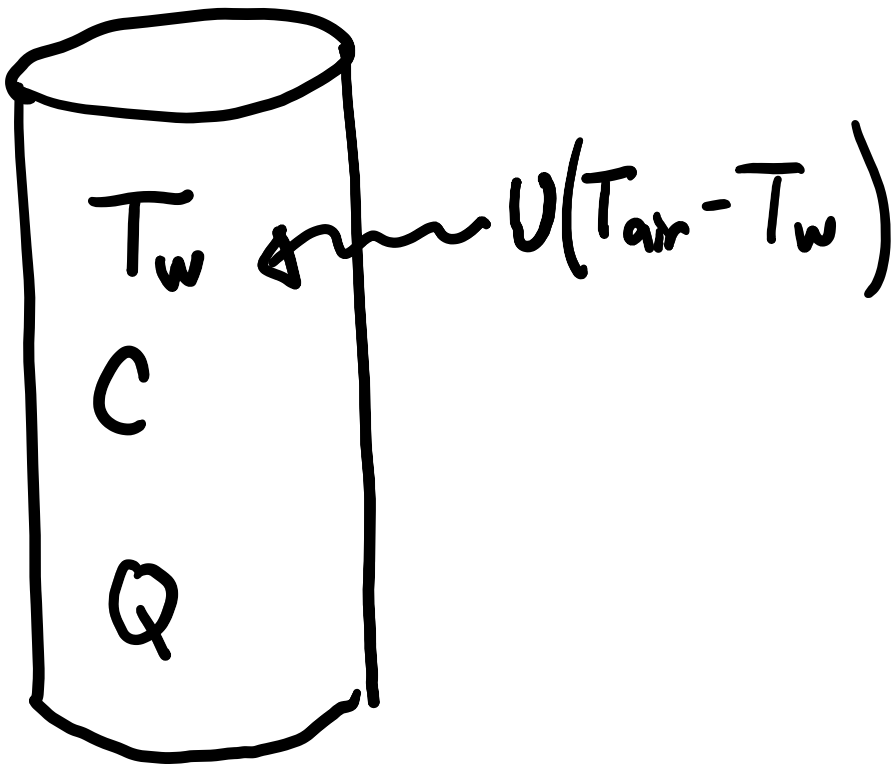
$$\frac{dT_w}{dt} = \frac{U}{C}(T_{air} - T_w) + \frac{1}{C}Q$$

Define the *error*:

$$e = T_t - T_w$$

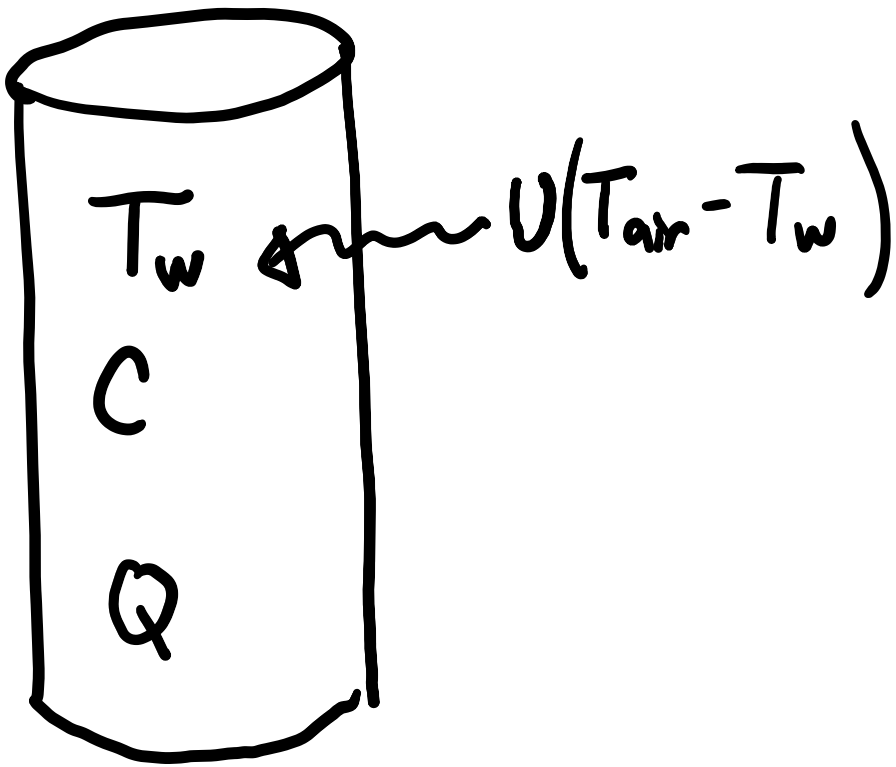
Define the *control law*:

$$Q = K_p \cdot e = K_p(T_t - T_w)$$



Example: temperature in a water heater

$$\frac{dT_w}{dt} = \frac{U}{C}(T_{air} - T_w) + \frac{1}{C}Q$$



Substitute the control law:

$$\frac{dT_w}{dt} = \frac{U}{C}(T_{air} - T_w) + \frac{1}{C}K_p(T_t - T_w)$$

What happens in the steady-state?

The results of our control function:

$$\frac{dT_w}{dt} = \frac{U}{C}(T_{air} - T_w) + \frac{1}{C}K_p(T_t - T_w)$$

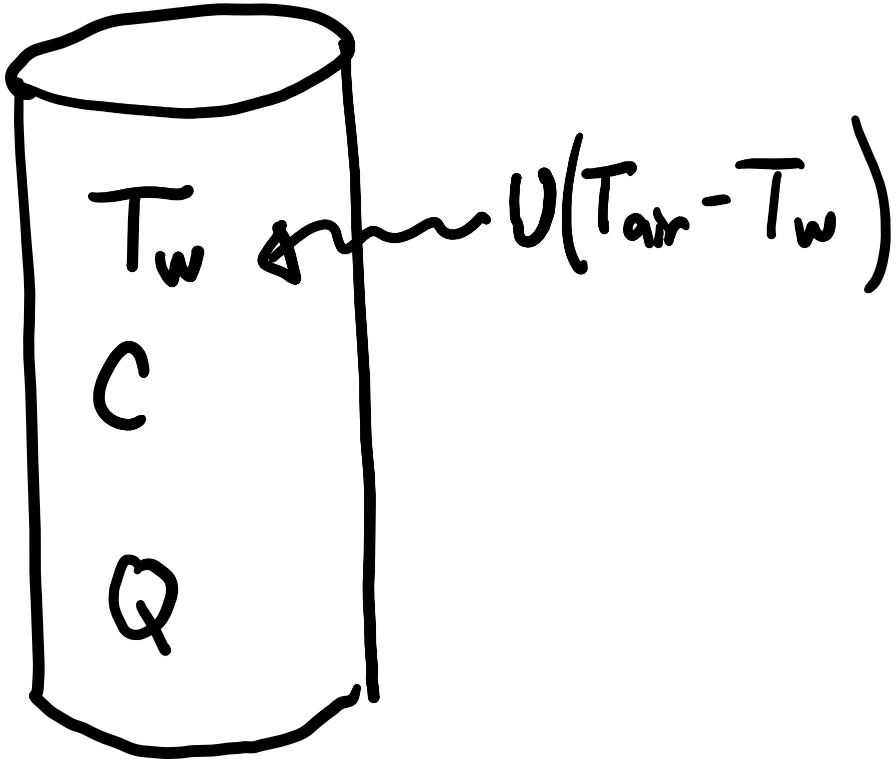
At steady-state, the derivatives go to zero:

$$\cancelto{0}{\frac{dT_w}{dt}} = \frac{U}{C}(T_{air} - T_w) + \frac{1}{C}K_p(T_t - T_w)$$

A little clever rearrangement:

$$0 = U(T_{air} - T_t + T_t - T_w) + K_p(T_t - T_w)$$

$$0 = U(T_{air} - T_t + e) + K_p e = U(T_{air} - T_t) + (U + K_p)e$$

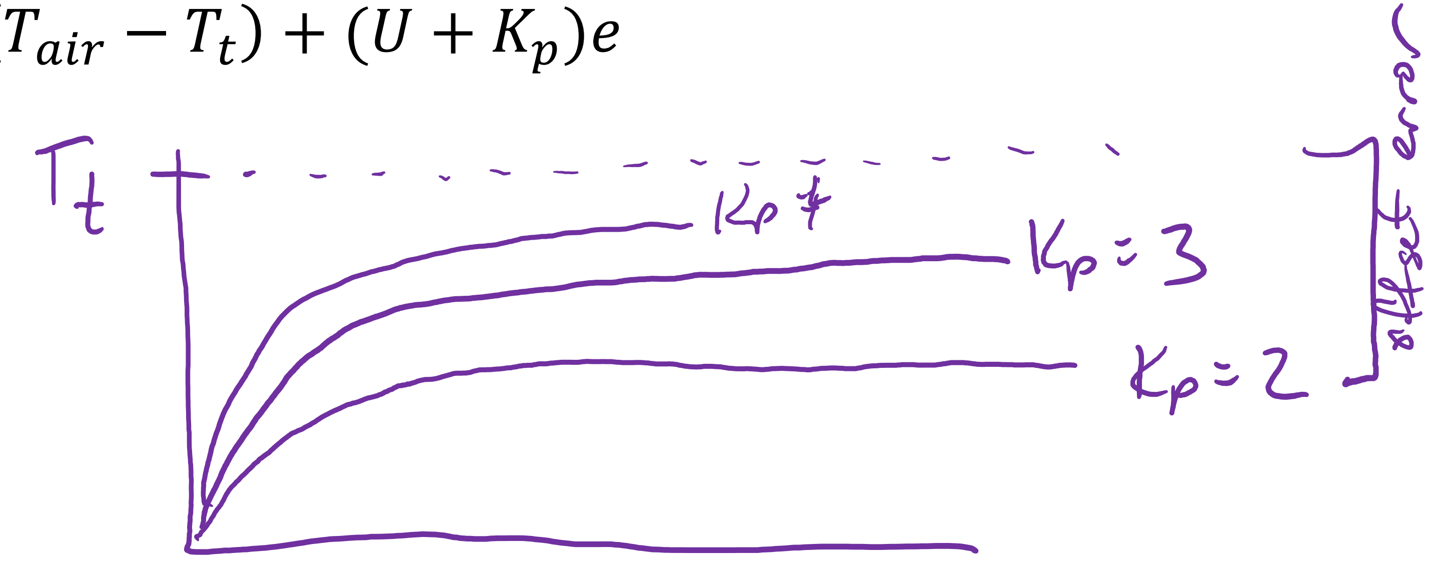


What happens in the steady-state?

$$0 = U(T_{air} - T_t + e) + K_p e = U(T_{air} - T_t) + (U + K_p)e$$

And finally, solve for the steady-state error,

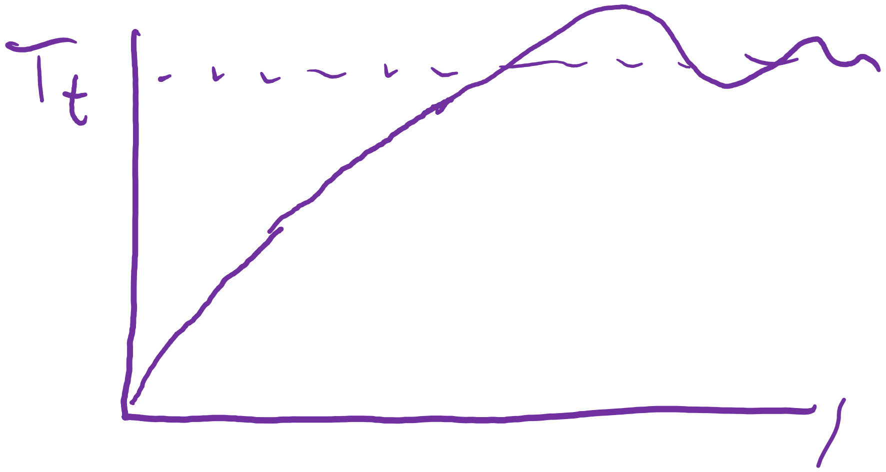
$$e\Big|_{t\to\infty} = \frac{U(T_t - T_{air})}{U + K_p}$$



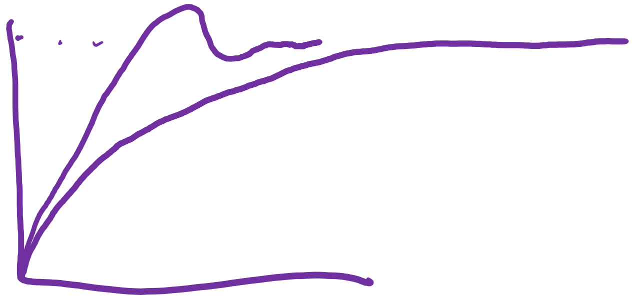
For many control problems, proportional control will leave a steady-state, or *offset*, error.

Integral control is typically used to remove the offset

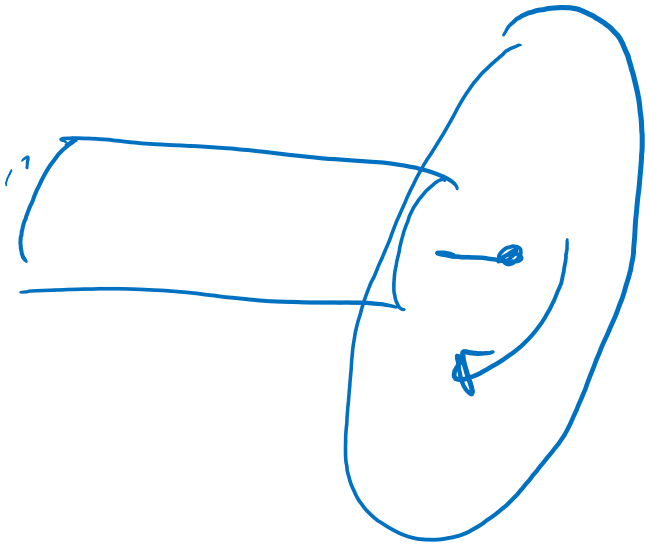
$$Q = \underbrace{K_p \cdot e}_{\text{proportion}} + \overbrace{K_i \int_0^t e(\tau)d\tau}^{\text{integral term}}$$

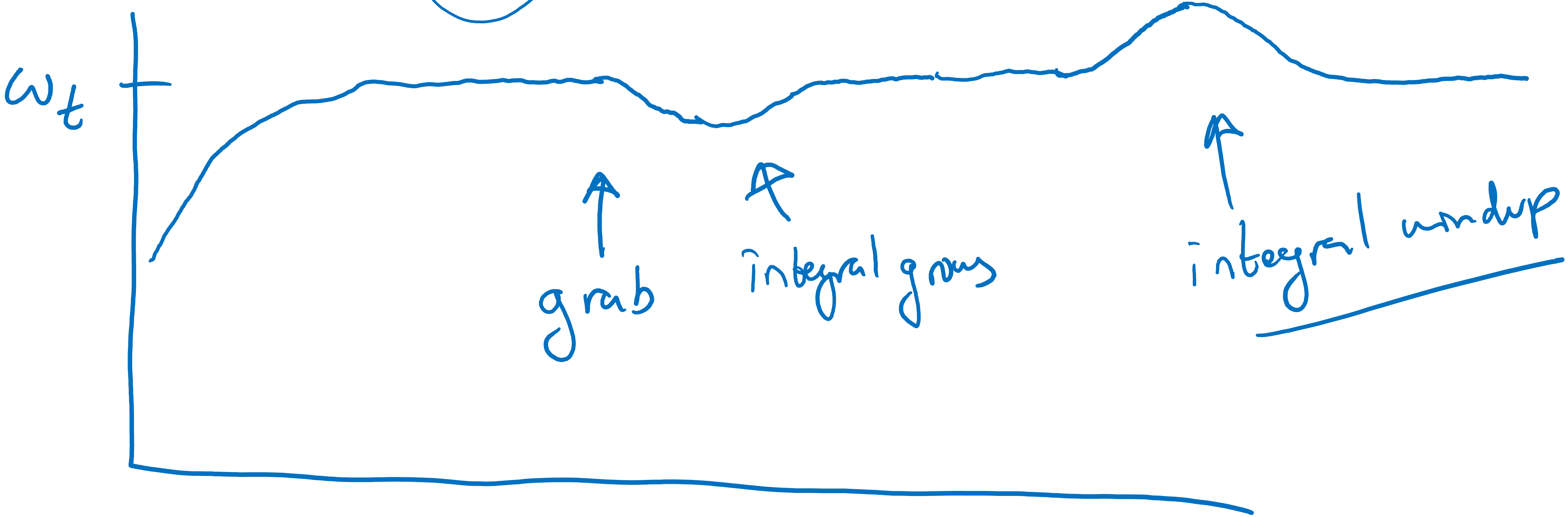


Without showing the proof, this control law will remove the offset for the water heater problem



Controlling motor speed with PI





In code

P & I control

```
const float Kp = <some value>;
const float Ki = <some other value>;

float error_sum = 0;
float target = <some value>;

float CalcEffort(void)
{
  float observed = ReadSensor();
  float error = target - observed;

  error_sum += error;

  float effort = Kp * error + Ki * error_sum;

  return effort;
}
```

Practical details

```
const float Kp = <some value>;
const float Ki = <some other value>;

float error_sum = 0;
float target = <some value>;

float CalcEffort(void)
{
  float observed = ReadSensor();
  float error = target - observed;

  error_sum += error;    <-- Cap the sum of the error

  float effort = Kp * error + Ki * error_sum;

  return effort;
}
```