

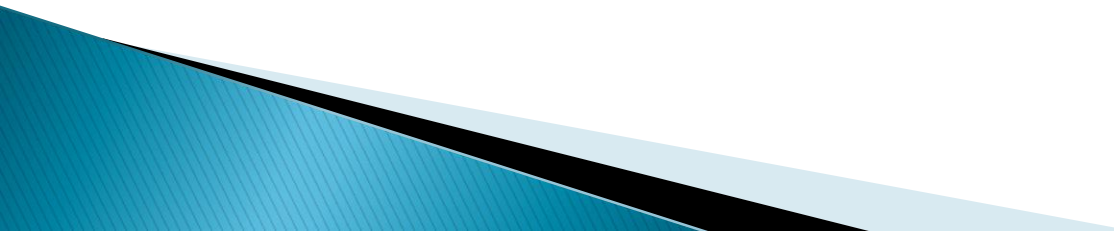
# Feedback Control

Benjamin N. Passow

[benpassow@dmu.ac.uk](mailto:benpassow@dmu.ac.uk)

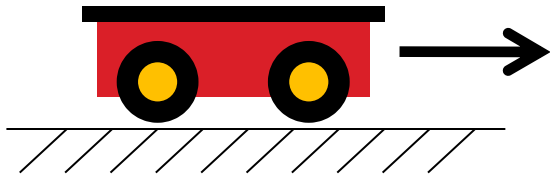
De Montfort University, UK

# Overview

- ▶ Why feedback control
  - ▶ Block diagrams
  - ▶ Binary control
  - ▶ Hysteresis
  - ▶ Proportional control
  - ▶ PID control
  - ▶ PID tuning
  - ▶ Fuzzy logic control
- 

# Why Feedback Control?

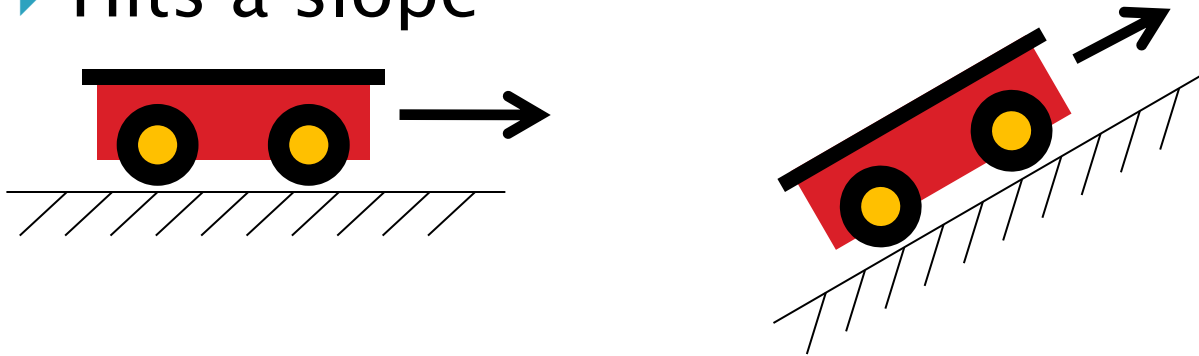
- ▶ Imagine a wheel robot
- ▶ You programmed it to run at 0.5m/s



- ▶ Motor power setting found empirically
- ▶ Open loop control

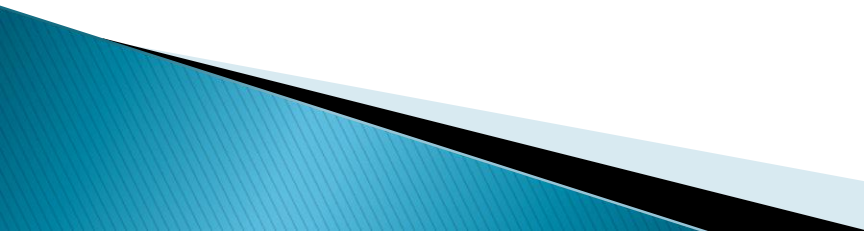
# Why Feedback Control?

- ▶ Imagine a wheel robot
- ▶ You programmed it to run at 0.5m/s
- ▶ Hits a slope



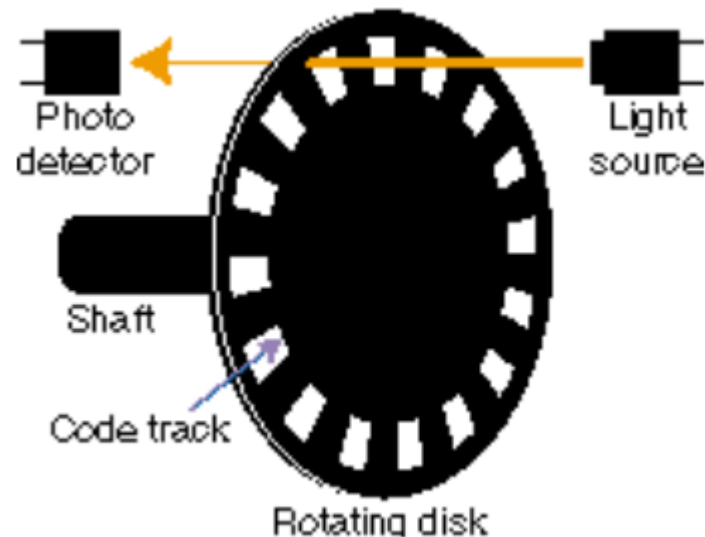
- ▶ Same controller doesn't provide enough power to run at 0.5m/s up a slope

# Why Feedback Control?

- ▶ Solution:
  - ▶ Measure how fast the robot is travelling
  - ▶ Apply power accordingly using a transfer function:
    - Binary
    - Proportional
    - Proportional, Integral, Derivative
    - Even fuzzy
- 

# Encoder (optical)

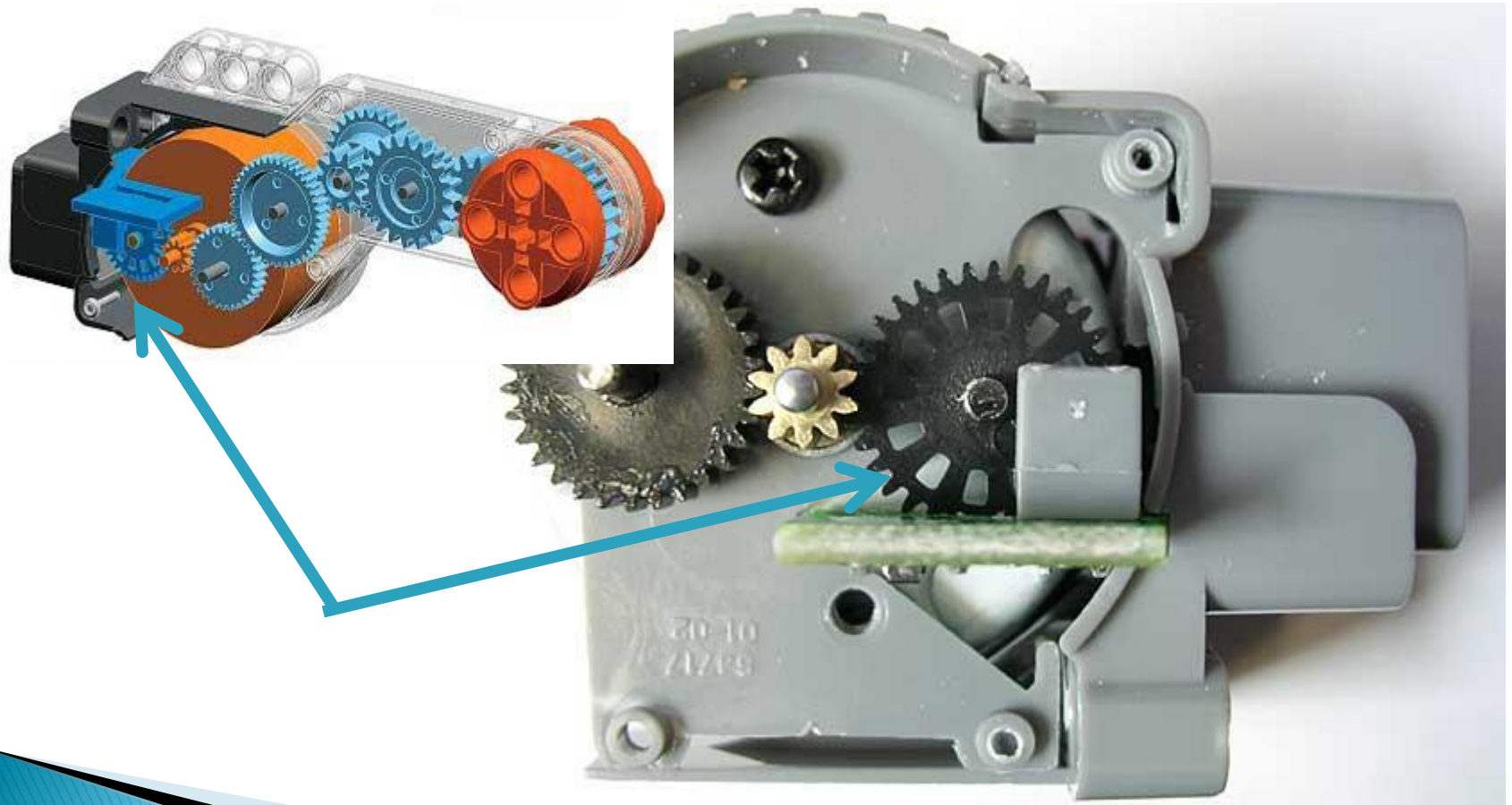
- ▶ To measure the wheel speed we need an encoder:



- ▶ Each change of light level represents a set rotation of the shaft

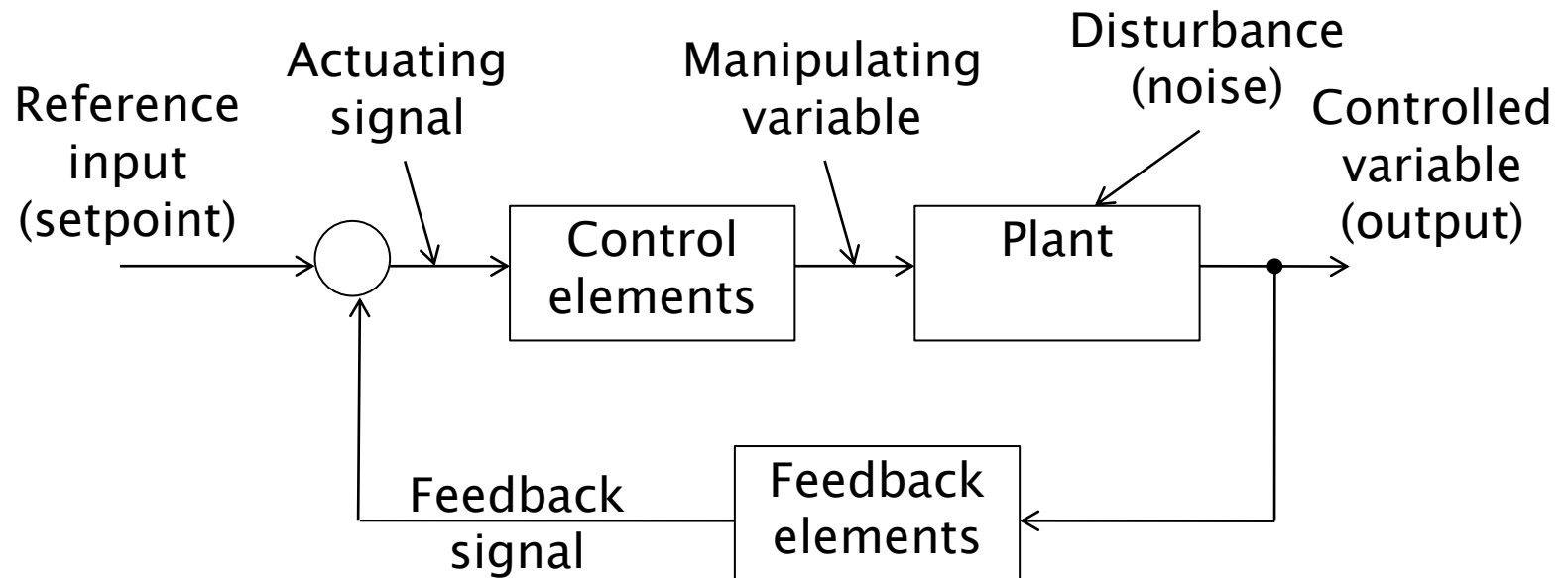
# Why Feedback Control?

- ▶ NXT's have encoders



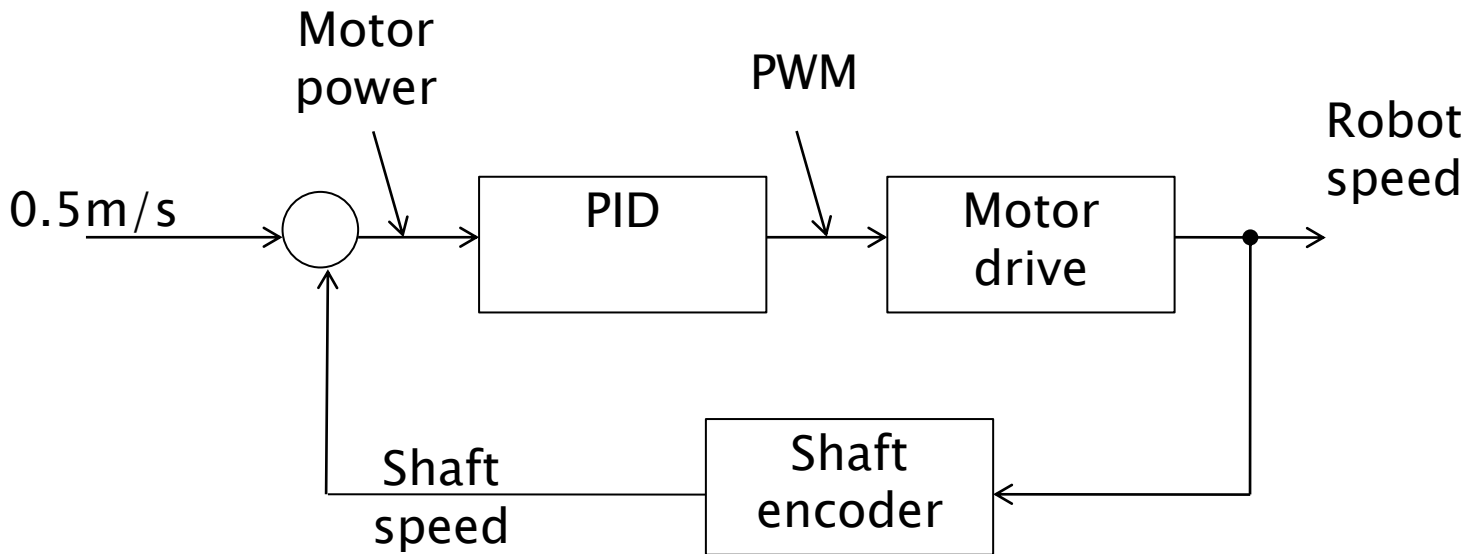
# Block Diagrams

- ▶ Our robot example



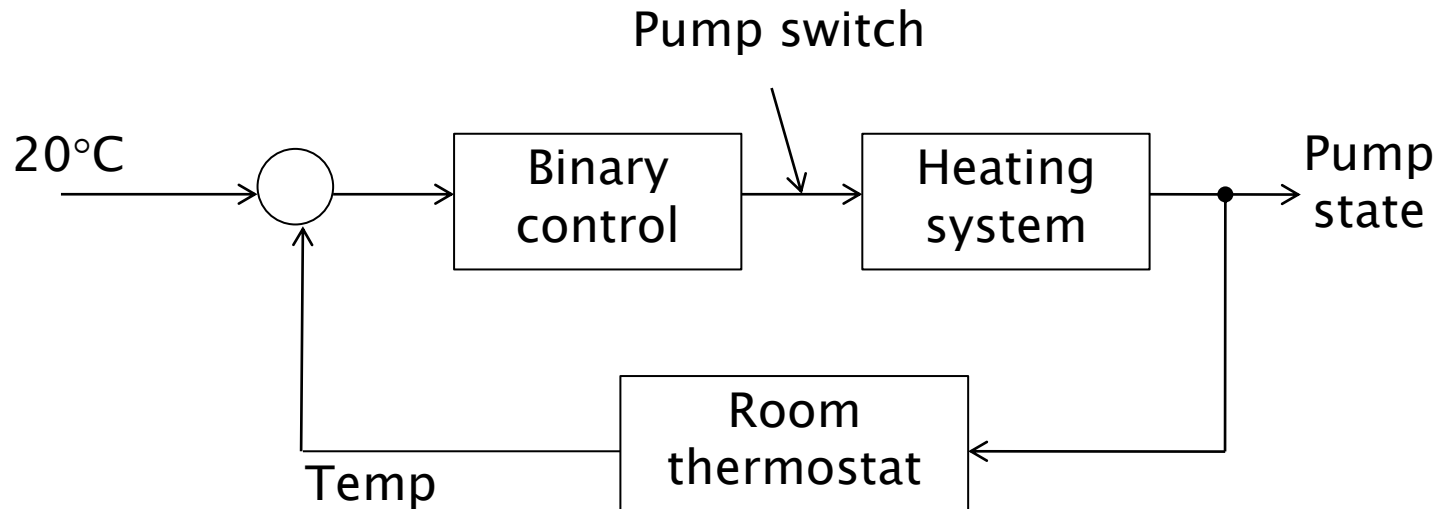
# Block Diagrams

- Describe a system in terms of elements and connections between elements



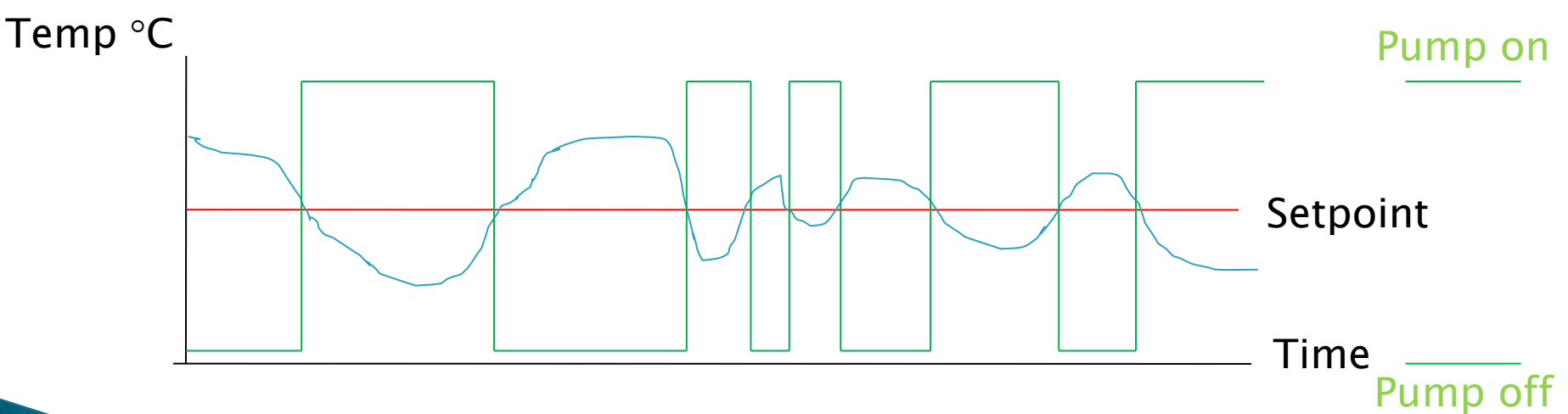
# Binary Control

- ▶ Actuators which can only be on or off
- ▶ For example central heating pump:



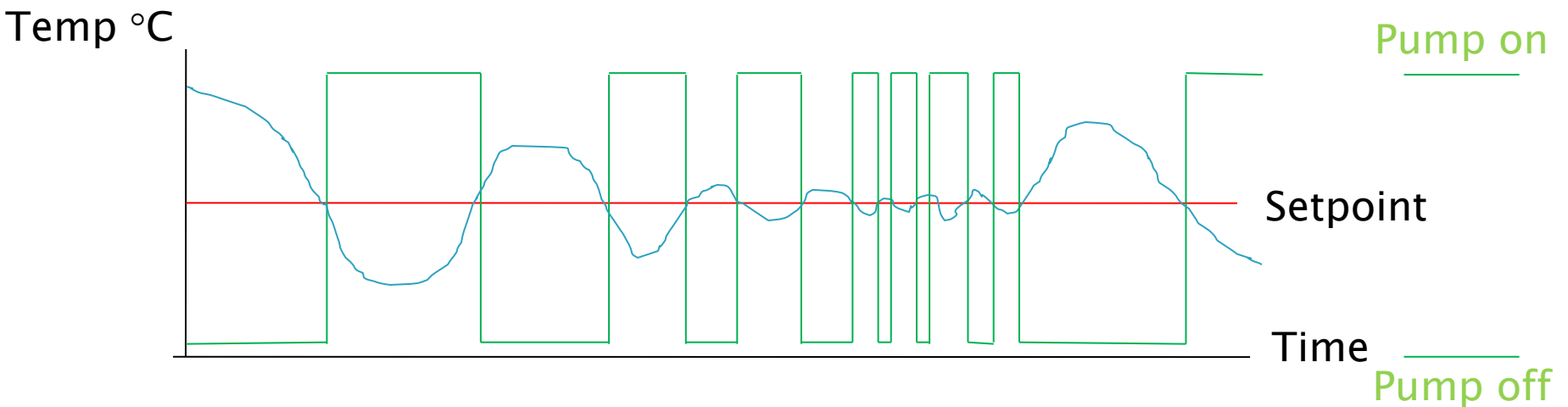
# Binary Control

- ▶ Simple binary rules:
  - If temperature goes below set point turn heating pump on
  - If temperature goes above set point turn heating pump off



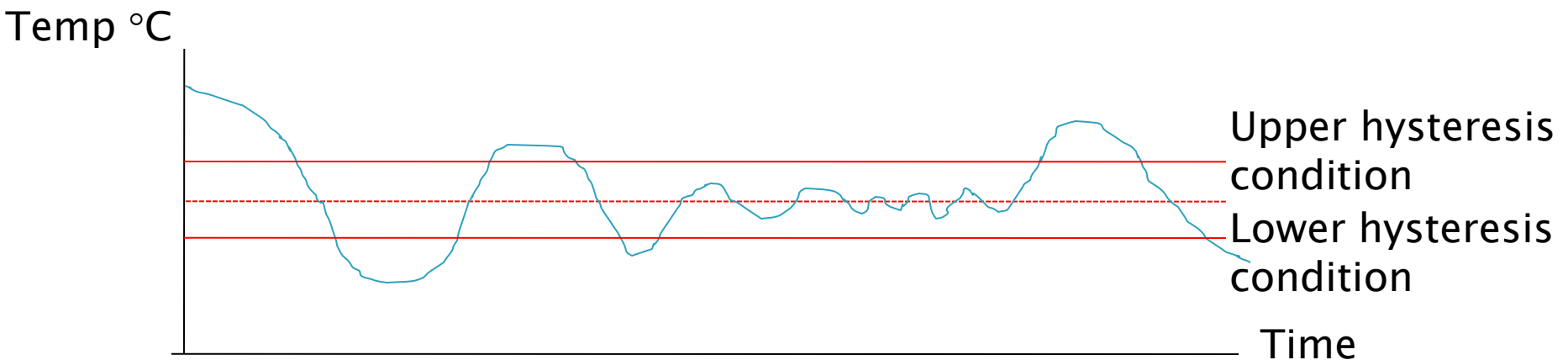
# Binary Control

- ▶ Problems arise when control variable moves frequently about the set point



# Hysteresis

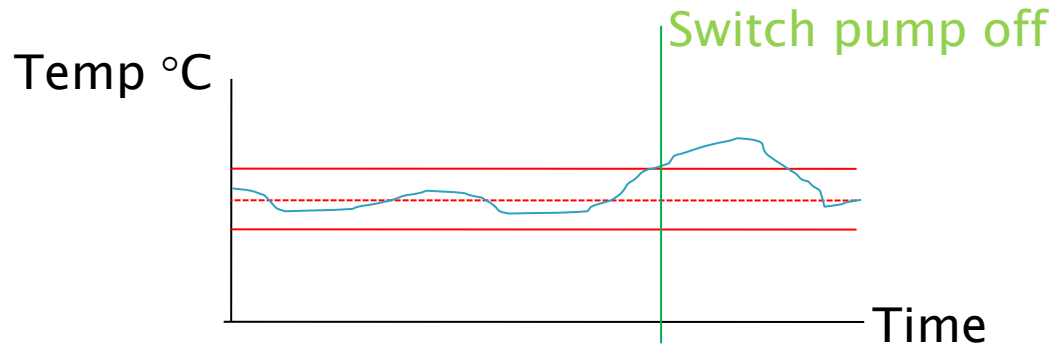
- ▶ Add upper and lower bounds to the setpoint



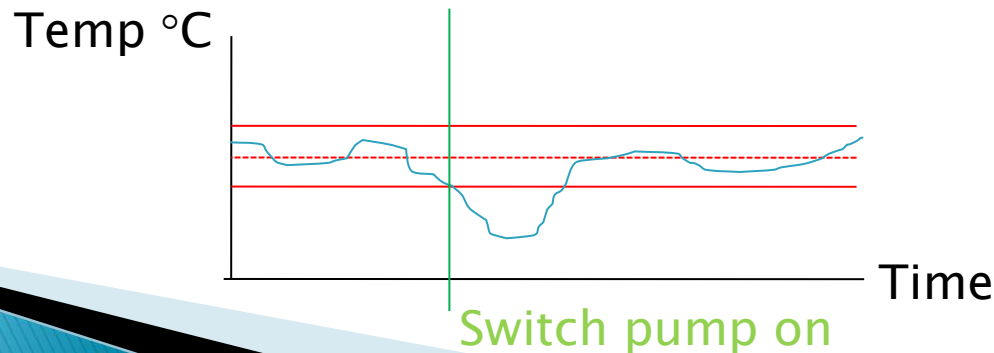
# Hysteresis

- ▶ Two rules:

- Switch off when signal goes from below upper hysteresis condition to above it

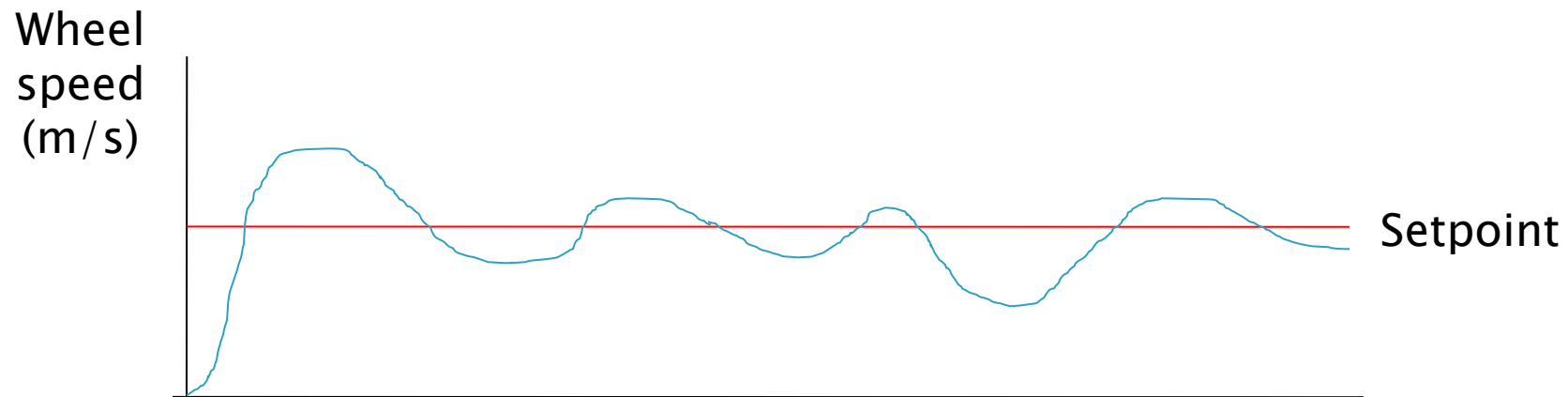


- Switch on when signal goes from above lower hysteresis condition to below it



# Proportional Control

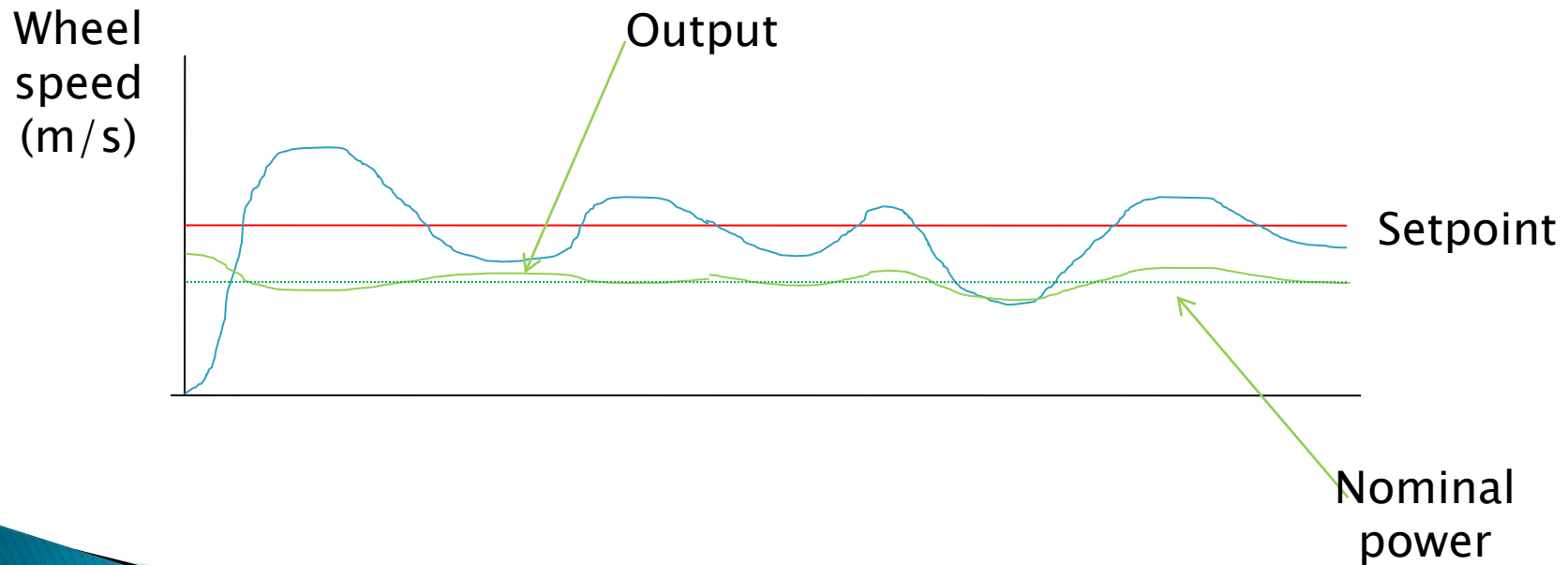
- ▶ Control response is proportional to the error
- ▶ Back to our robot example



What does  
this mean?

# Proportional Control

- ▶ Back to our robot example
- ▶  $K_p = 0.25$



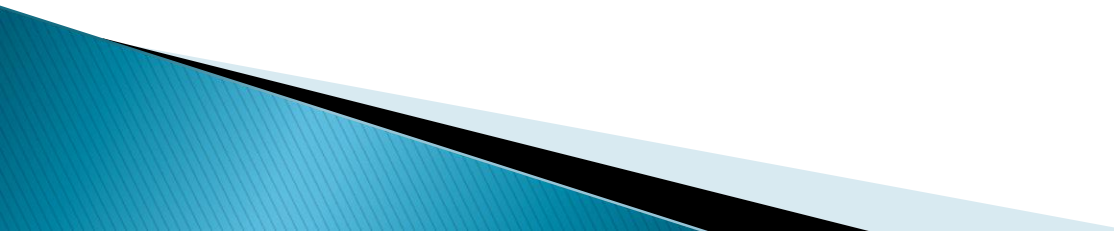
# Recap

- ▶ Feedback control used to adjust the output signal to the conditions
- ▶ Binary control:
  - On or Off systems
  - Hysteresis used to reduce switching around the setpoint
- ▶ Proportional control:
  - Control response is proportional to the error

# Feedback Control

## PID

# Overview

- ▶ Why PID?
  - ▶ PID
  - ▶ Designing PID
  - ▶ Implementing PID
- 

# Why PID?

- ▶ Proportional control does not solve all problems:
  - Changes to system or robot
  - Control speed vs. Oscillation

**PID:**

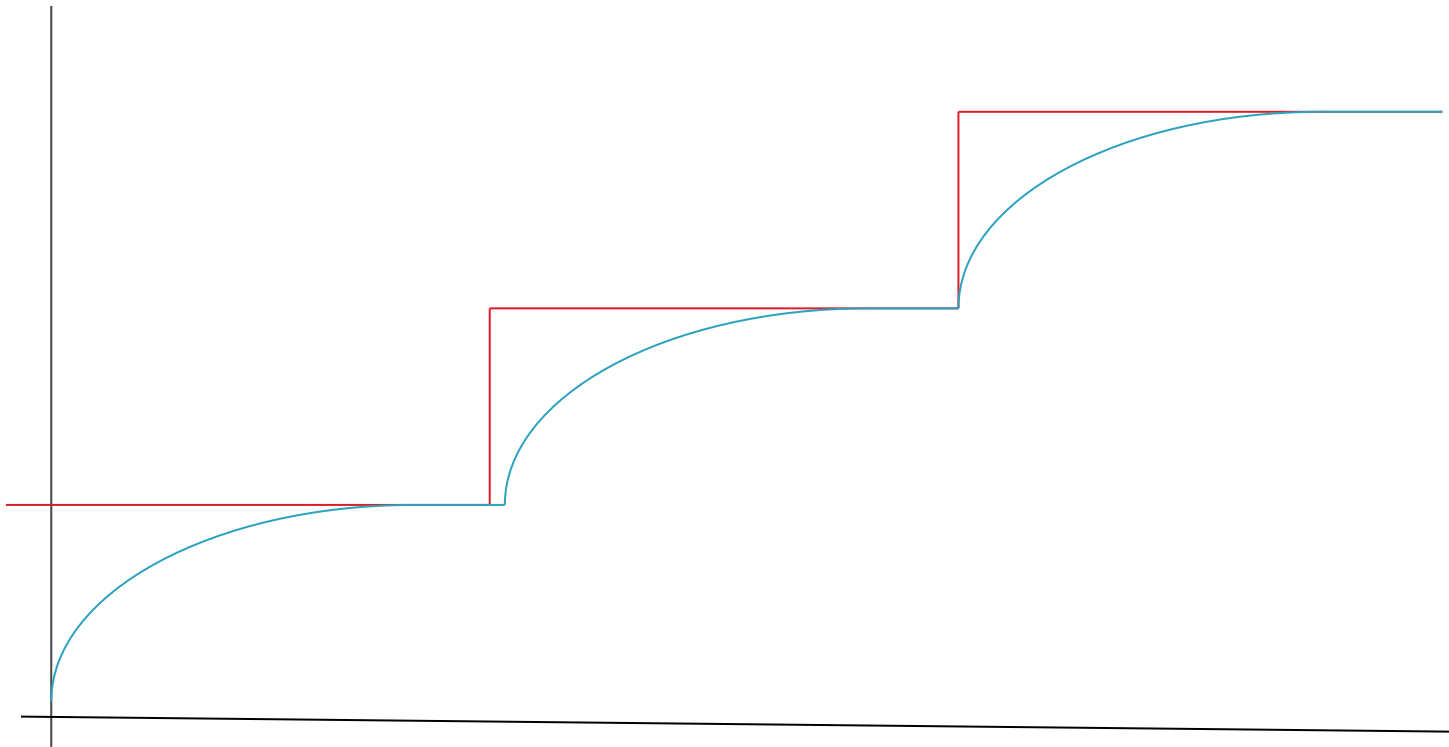
Proportional + Integral + Derivative

3 controllers that handle different aspects



# Why PID?

- ▶ We want



# PID

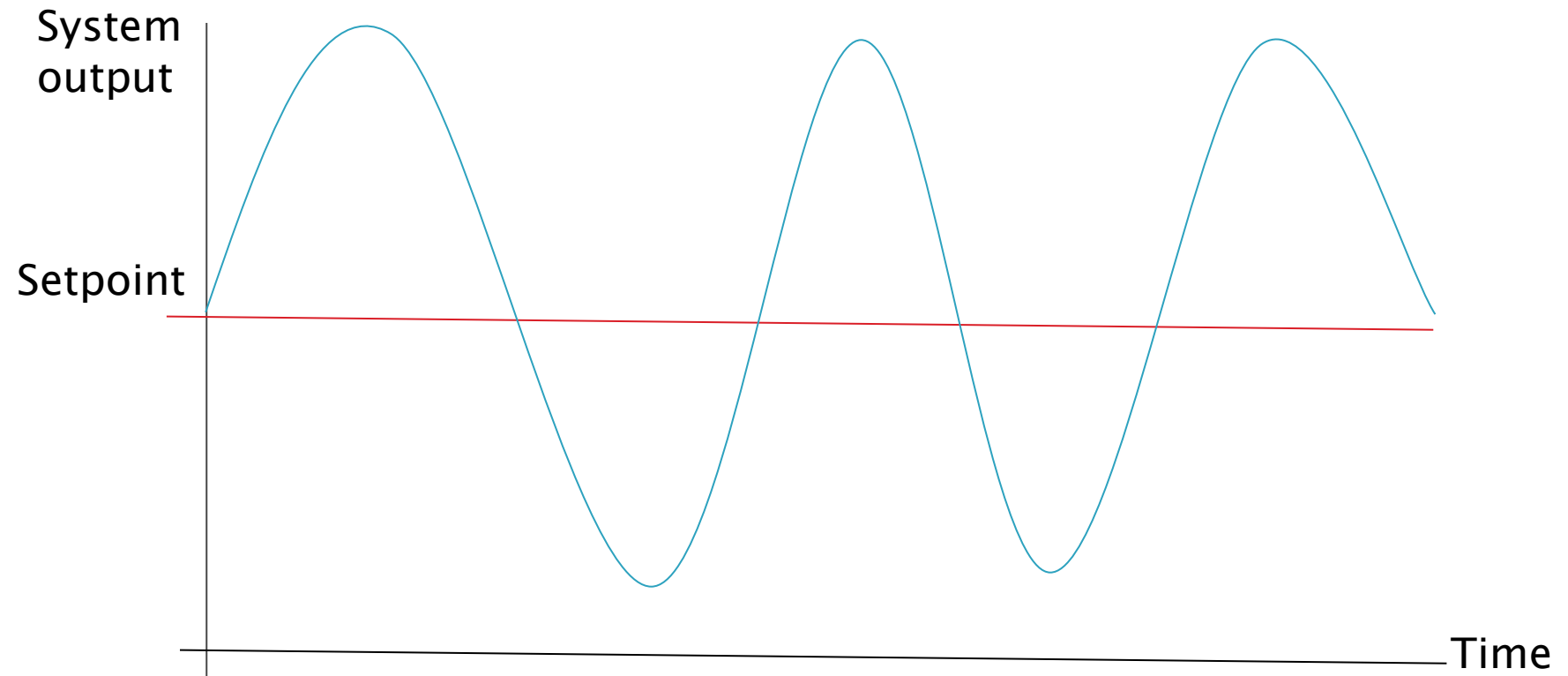
- ▶ Most widely use feedback control technique
  - Well understood
  - Well studied
- ▶ Controller made up of three elements:
  - Proportional
  - Integral
  - Derivative
- ▶ Advantages:
  - Can adapt to different setpoints
  - Faster dampening

# PID

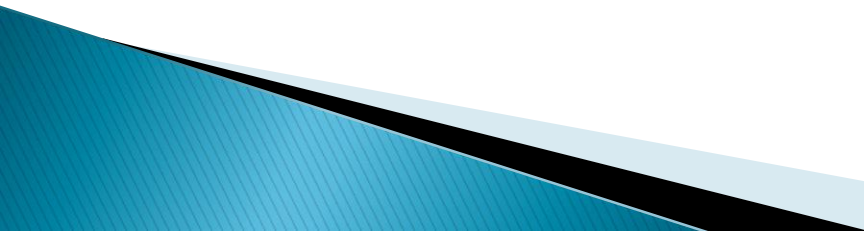
- ▶ **Proportional**
- ▶ Very simple – gets back to the setpoint when there is an error
- ▶  $P = K_p \times e(t)$
- ▶ P is the output of the proportional controller
- ▶  $K_p$  is the gain of the proportional controller
- ▶  $e(t)$  is the error at time t

# PID

## ► Proportional



# PID

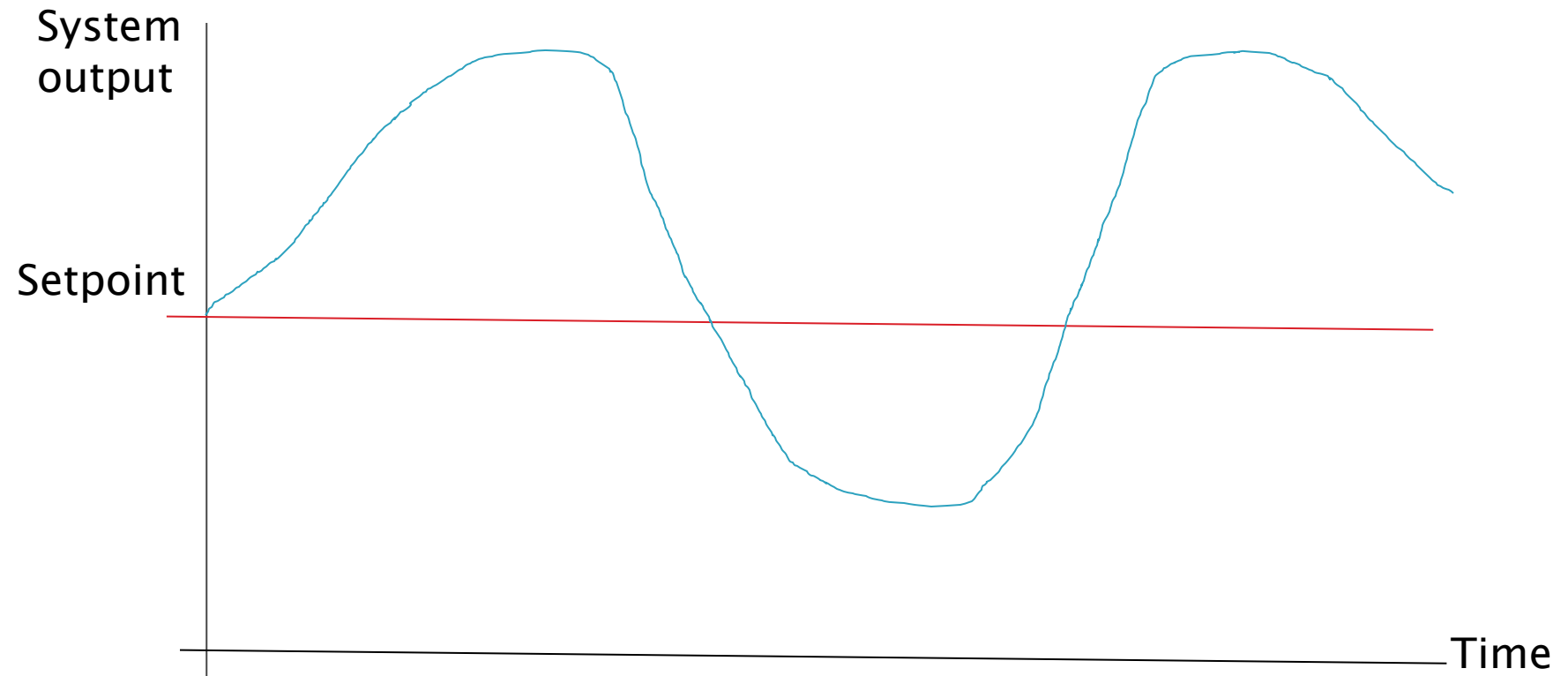
- ▶ **Integral**
  - ▶ Reduces long term errors
  - ▶  $I = K_i \times \int e(\tau) \times d(\tau)$
  - ▶ I is the output of the integral controller
  - ▶  $K_i$  is the gain of the integral controller
  - ▶ e is the error
  - ▶  $\tau$  is the integration variable
- 

# PID

- ▶ **Integral**
- ▶ **Coding it up:**
  - Store error for the last n control cycles
  - Sum errors
  - Threshold – make it's a sensible value
  - Multiply by gain
- ▶ **OR**
  - Add new errors to variable
  - Limit variable to a MAX and MIN boundary

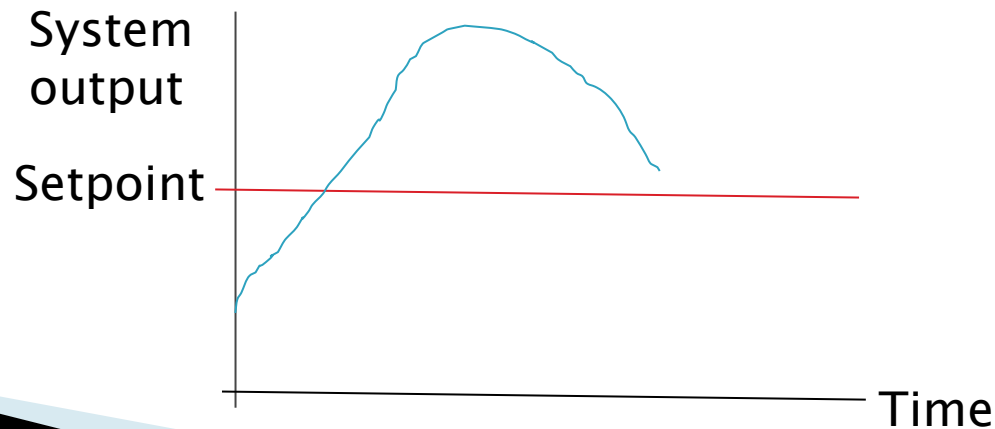
# PID

## ▶ Integral



# PID

- ▶ So far...
- ▶ Proportional control based on error right now
- ▶ Integral control based on previous errors
- ▶ We need a way to control the rate of change



# PID

- ▶ **Derivative**

- ▶ Rate of change

- ▶  $D = K_d \times \frac{\delta e(t)}{\delta t}$

- ▶ D is the output of the derivative controller

- ▶  $K_d$  is the gain of the derivative controller

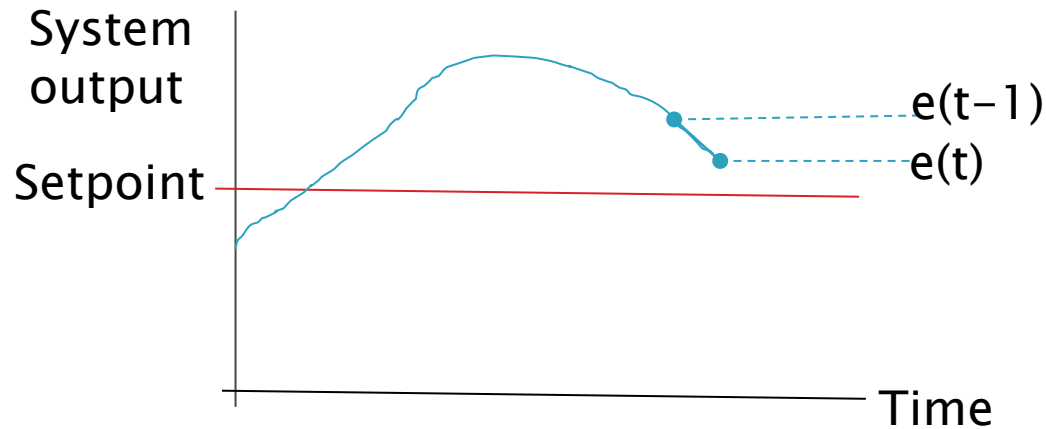
- ▶ e is the error

- ▶ t is time



# PID

- ▶ **Derivative**
- ▶ **Coding it up:**
  - Need to know direction and magnitude of error
  - Result = Gain X (e(t-1) - e(t))



# PID

## Proportional

$$P_{out}(t) = K_p e(t)$$

Pushes towards the set point.

## Integral

$$I_{out}(t) = K_i \int_{t-1}^t e(\tau) d\tau$$

Accelerates movement towards the set point.  
Can cause overshoot.  
Gives accumulation of recent error.

## Derivative

$$U(t) = K_d \frac{d}{dt} e(t)$$

Slows the rate of change of the controller.  
Reduces overshoot.  
Most noticeable close to the set point.

# PID

error = setpoint - measurement

integral error = integral error + error

if integral error > integral max

then integral error = integral max

if integral error < integral min

then integral error = integral min

derivative error = error - previous error

out P = error \* proportional gain

out I = integral error \* integral gain

out D = derivative error \* derivative gain

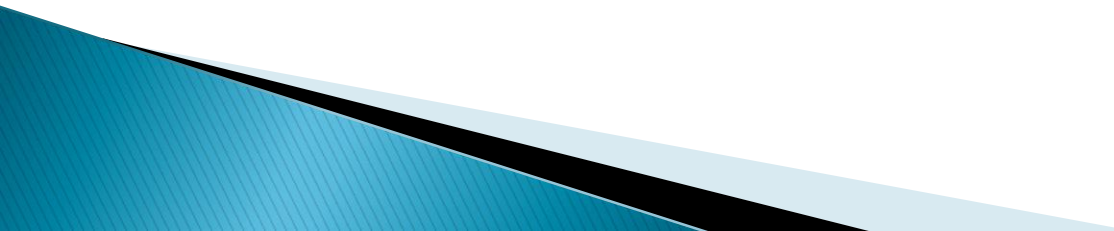
output = out P + out I + out D

previous error = error

# Tuning

- ▶ How do we find good values for the Gains?

# Tuning

- ▶ How do we find good values for the Gains?
  - ▶ Start by setting  $K_p$  to low value say 1
  - ▶ You may see the system oscillate
  - ▶ Small oscillation is okay for now
- 

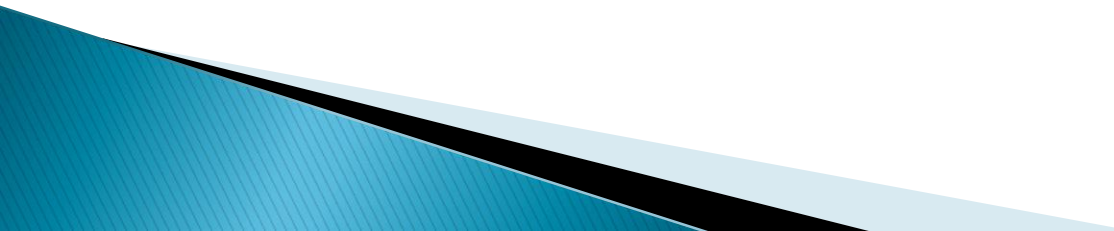
# Tuning

- ▶ How do we find good values for the Gains?
- ▶ Now tune  $K_d$
- ▶ Start with a value much larger than  $K_p$
- ▶ E.g.  $K_d = 10$
- ▶ Raise  $K_d$  until the system shows fast, large oscillation
- ▶ Reduce  $K_d$  by a factor of 2 or 4

# Tuning

- ▶ How do we find good values for the Gains?
- ▶ Now tune  $K_p$
- ▶ If the system is oscillating:
  - Drop  $K_p$  by a factor of 10 until oscillation stops
- ▶ If the system is not oscillating:
  - Increase  $K_p$  by a factor of 10 until oscillation begins
  - Drop  $K_p$  by a factor of 2 or 4
- ▶ Adjust by factor of less than two until it looks good

# Tuning

- ▶ How do we find good values for the Gains?
  - ▶ Now tune  $K_i$
  - ▶ Try a much much smaller gain
  - ▶ Up them until you get oscillation
  - ▶ Drop by a factor of 2
  - ▶ Fine tune
- 

# Recap

- ▶ Proportional, Integral, Derivative
  - P – fix current error
  - I – fix previous errors
  - D – predict and fix future errors
- ▶ Tuning
  - Tune D, then P and finally I
  - Ramp up factors until oscillation
  - Drop by factor 2 or 4
  - Fine tune

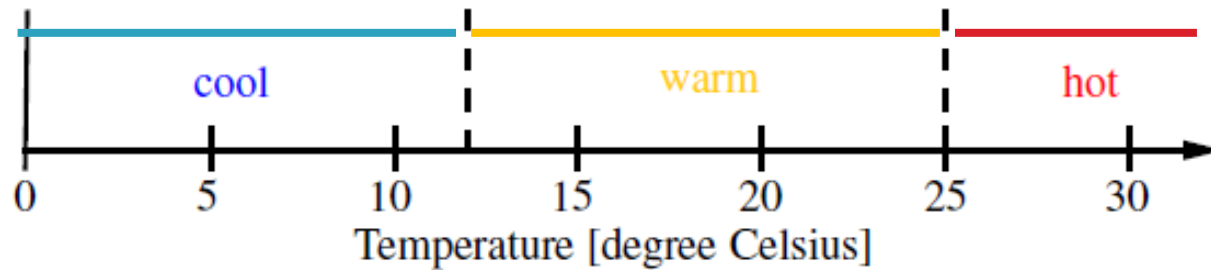
More information: “PID without a PhD” paper



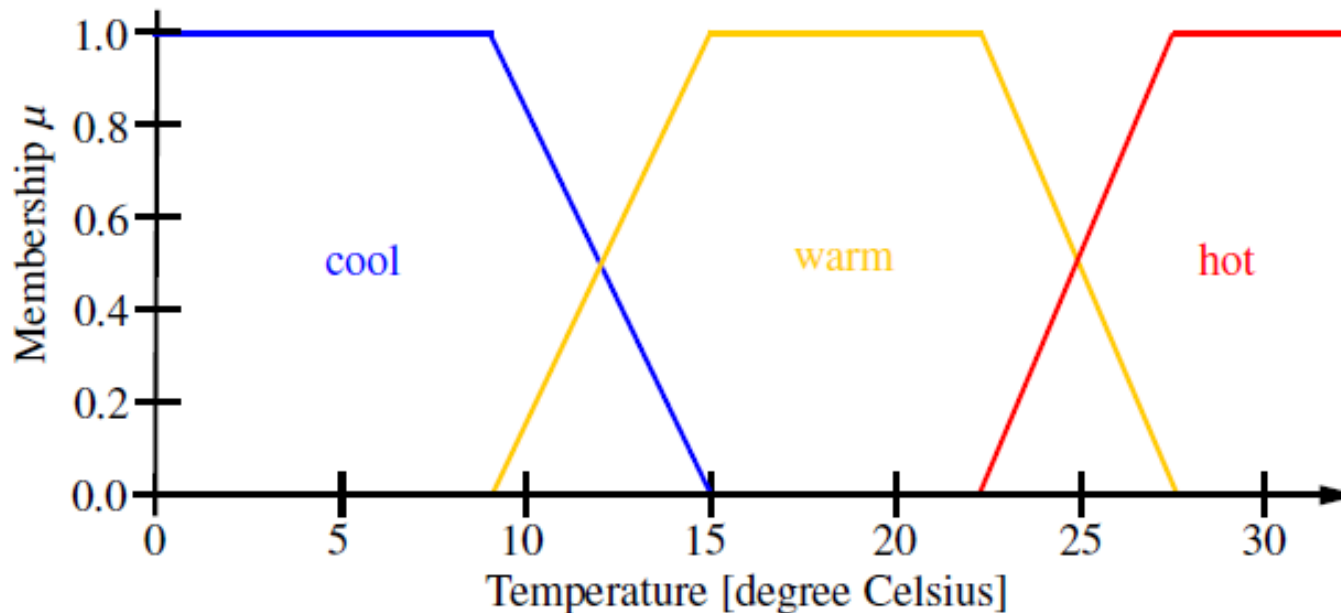
# Fuzzy Control

# Fuzzy Logic

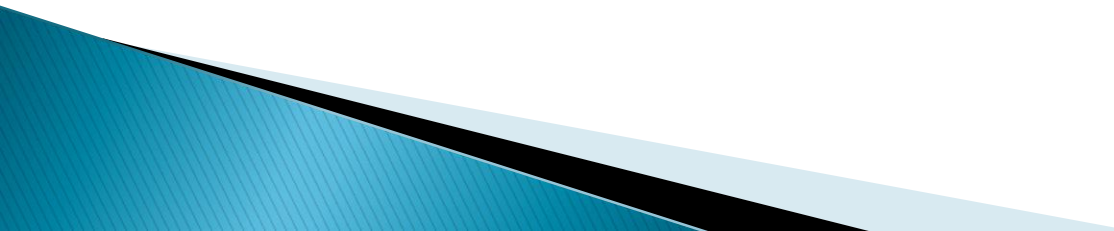
- ▶ Crisp logic:



- ▶ Fuzzy Logic

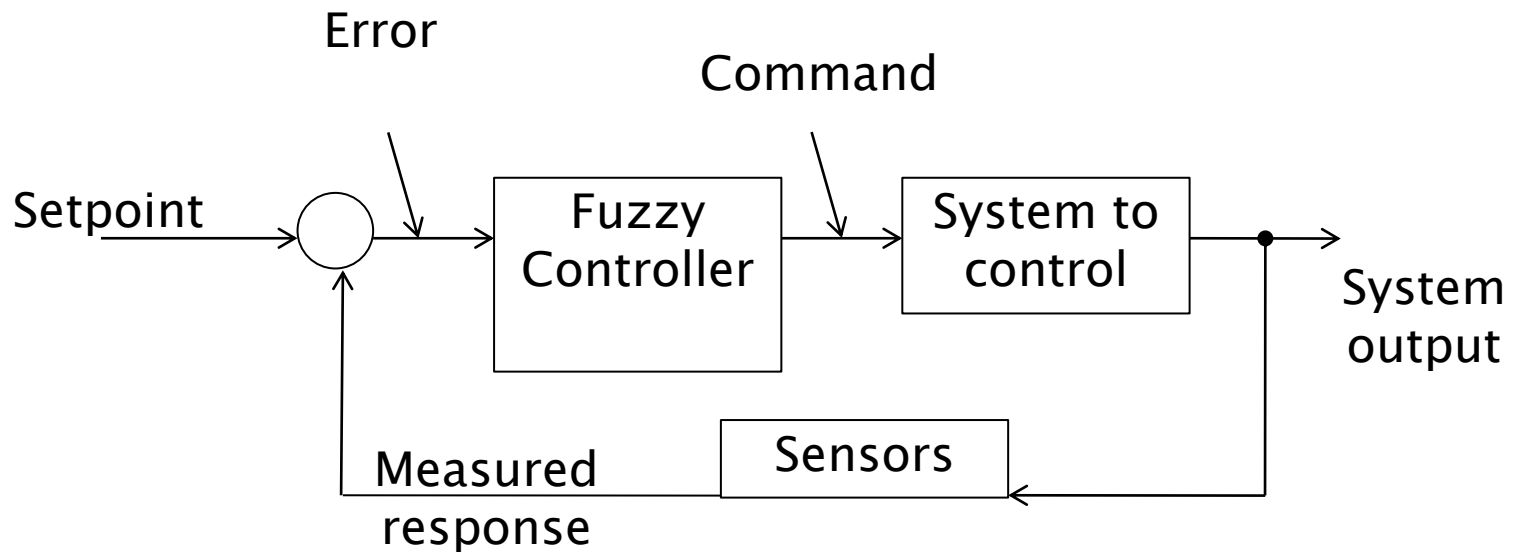


# Fuzzy Logic

- ▶ Intelligent method
    - Rule-based
    - Can model reasoning
  - ▶ Can handle uncertainty
    - Noisy input data
  - ▶ Can be derived from
    - Modelled from expert knowledge
    - Stochastically: e.g. evolutionary algorithm
    - Trained from data: e.g. ANFIS
- 

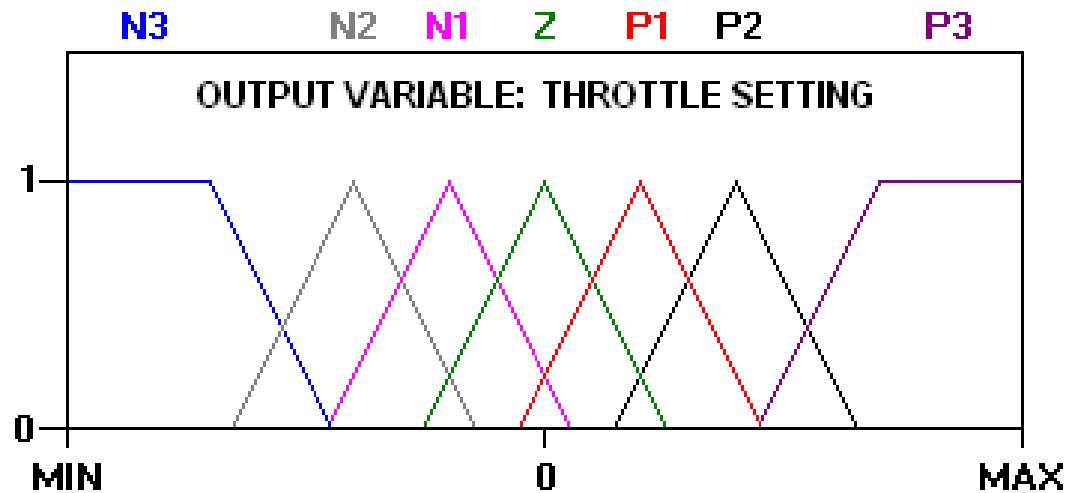
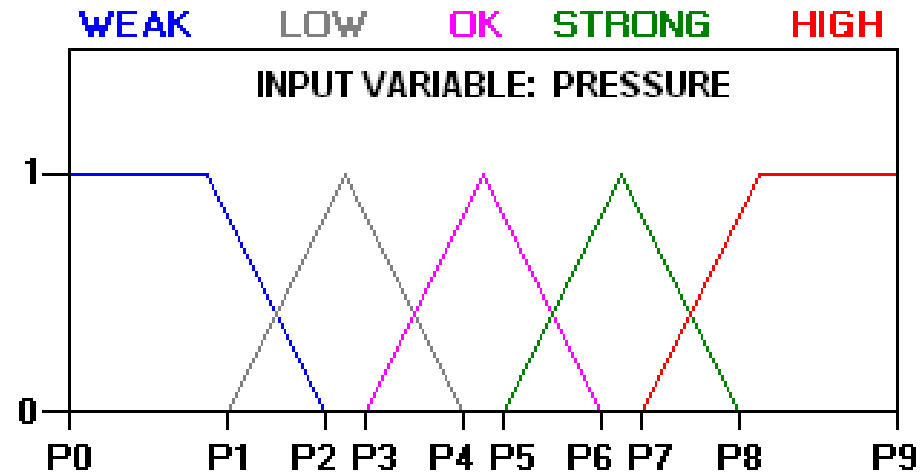
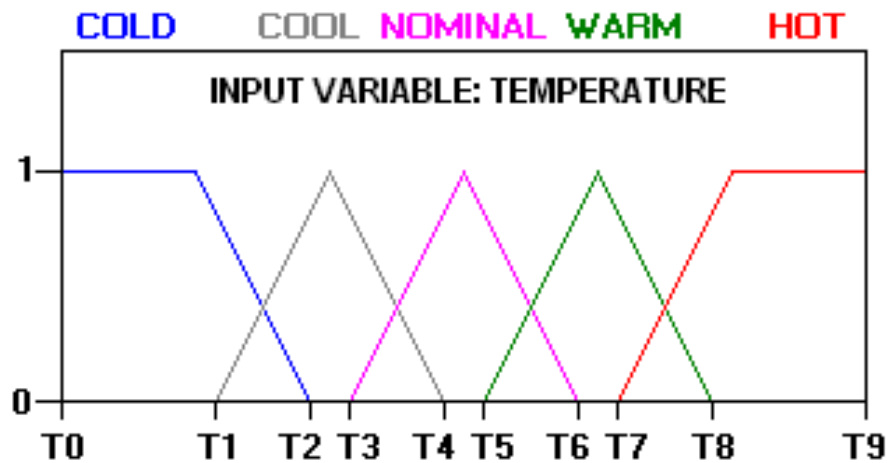
# Block Diagrams

- ▶ A fuzzy feedback controller



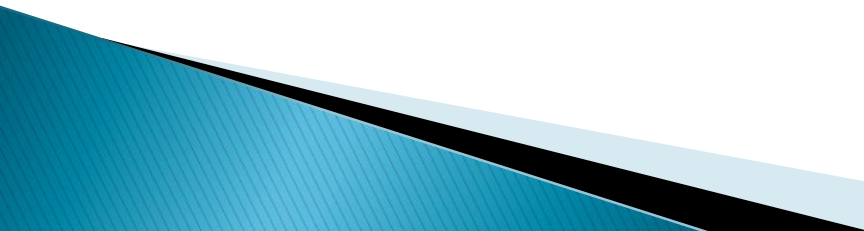
# Fuzzy Control

## - Membership Functions

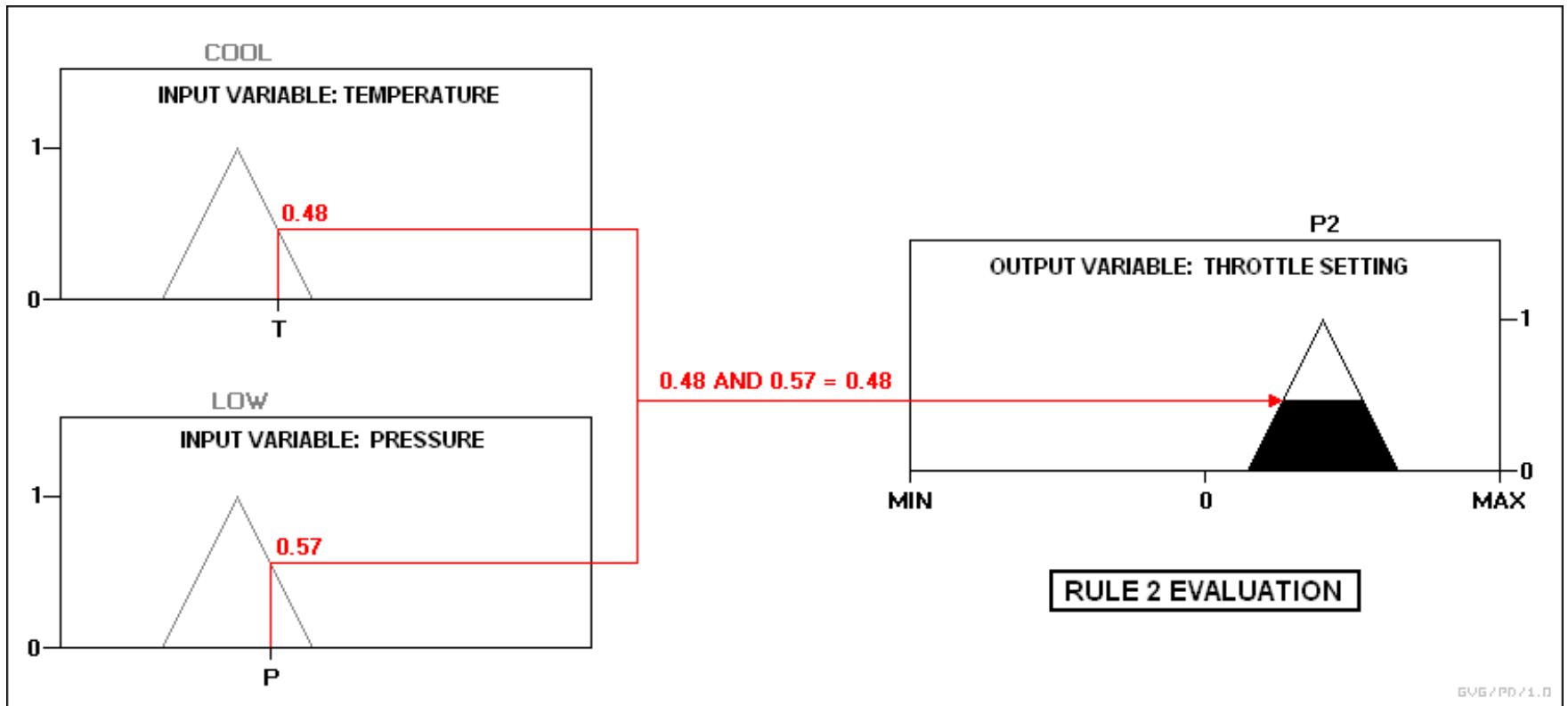


# Fuzzy Control

## – Rule Base (and fuzzy operators)

- ▶ rule 1: IF temperature IS cool AND pressure IS weak, THEN throttle is P3.
  - ▶ rule 2: IF temperature IS cool AND pressure IS low, THEN throttle is P2.
  - ▶ rule 3: IF temperature IS cool AND pressure IS ok, THEN throttle is Z.
  - ▶ rule 4: IF temperature IS cool AND pressure IS strong, THEN throttle is N2.
- 

# Fuzzy Control – Implication

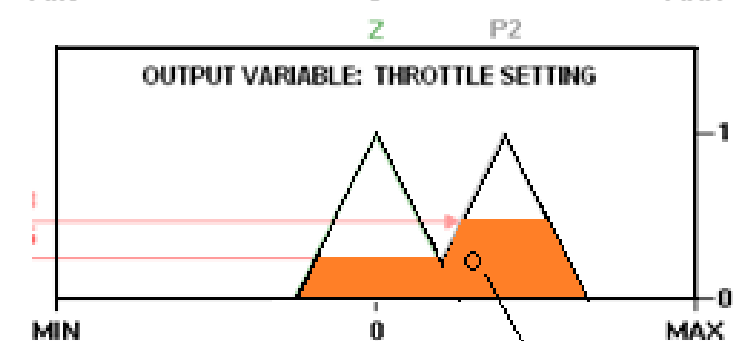
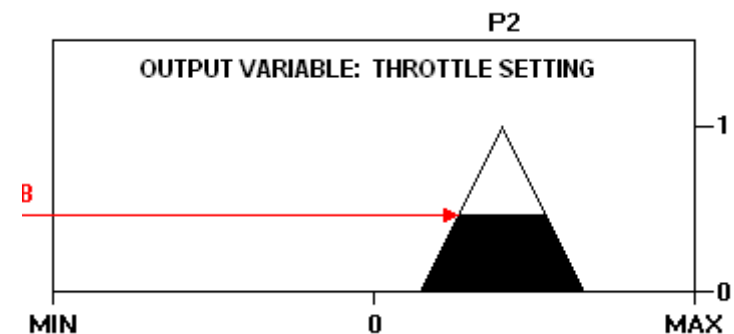
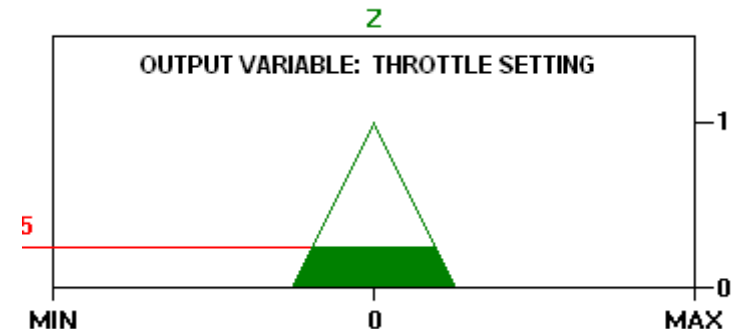


# Fuzzy Control

## – Aggregation & Defuzzification

- ▶ Aggregate all outputs
- ▶ Defuzzify the aggregated output

–> Crisp output to use on robot



Centroid

# Fuzzy Control – Summary

- ▶ Intelligent method
    - Rule-based
    - Can model reasoning
  - ▶ Can handle uncertainty
    - Noisy input data
    - Type-2 Fuzzy logic can handle even more uncertainty
  - ▶ Can be derived from
    - Modelled from expert knowledge
    - Stochastically: e.g. evolutionary algorithm
    - Trained from data: e.g. ANFIS
- 