# G54ARS
# Autonomous Robotic Systems

PID Control



## Last week…

- Background
  - The behaviour based approach revisited
  - Behaviour arbitration
- Subsumption architecture
  - Brooks' assumptions about mobile robot design
  - levels of competence
  - layers of control
  - structure of layers
  - extensions; finite state machines
- Sensors
  - Sensor Characteristics
  - Sensor Types and categories
- Actuators
  - Effectors and actuators
  - DC motors and Servo Motors
  - DC motors – how they work
  - Degrees of Freedom



Kahoot!

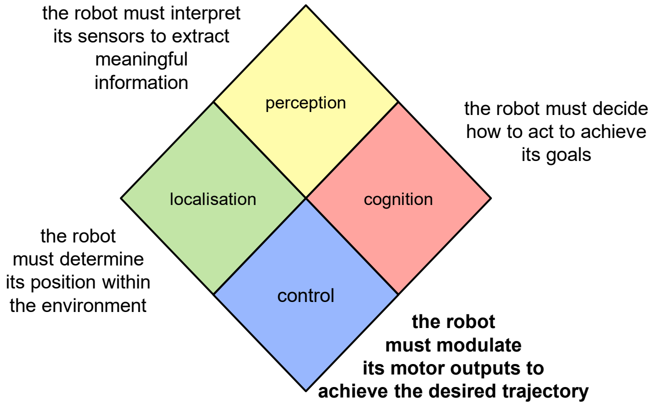
https://kahoot.it/



## This week…

- Control
  - The problem
  - Open-Loop Control
- PID Control
  - PID principles
  - PID parameter effects
  - PID tuning
  - Live example – PID DC motor control
  - PID implementation
- Video - The Grand Challenge

## Autonomous Mobile Robots - Navigation



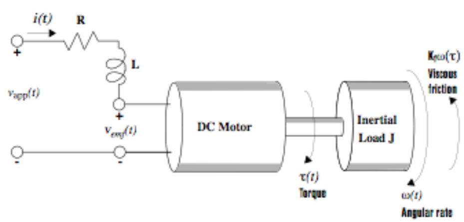



## The Problem



## DC Motor

Input: voltage $v_{app}(t)$

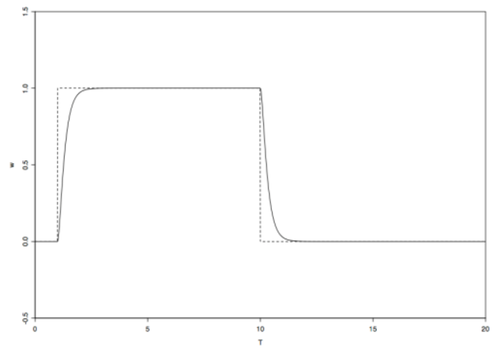



Maintain desired Output: angular velocity $\omega(t)$



## Open-Loop Control

$v_{app}(t) = 26.681\text{V}$



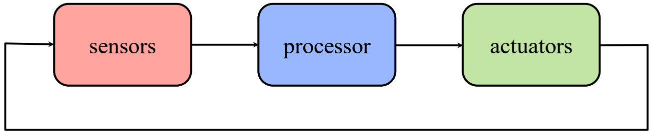
$\omega(t)$ reaches approx. 1 rotation per second

**P**roportional
**I**ntegral
**D**erivative

**Control**



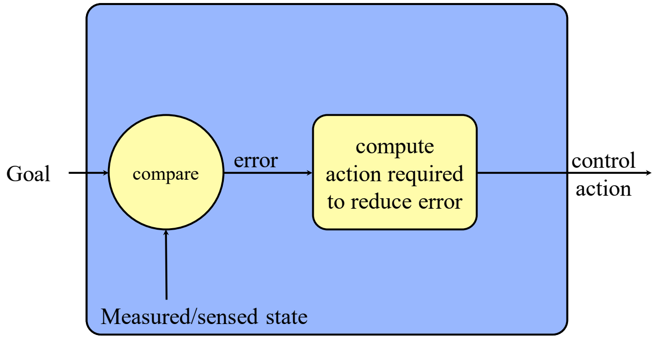
## The Sense-Think-Act Cycle




- Repeat
  - sense the current state
  - reduce difference between current state and goal state
- Until
  - current state = goal state



## The Sense-Think-Act Cycle






## Compute the Control Action

- Present
  - the current error
    - *proportional*
- Past
  - the sum of errors up to present time
    - *integral*
- Future
  - the rate of change of the error
    - *derivative*

# PID Control

- A proportional-integral-derivative controller (PID controller) simply combines these three terms in a weighted sum to obtain the control action
- The origins of PID control go back to research on automating and optimizing ship steering by N. Minorsky in the early 1900s. If you are interested, see: “Nicolas Minorsky and the Automatic Steering of Ships” by S. Bennett (http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1104827)

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

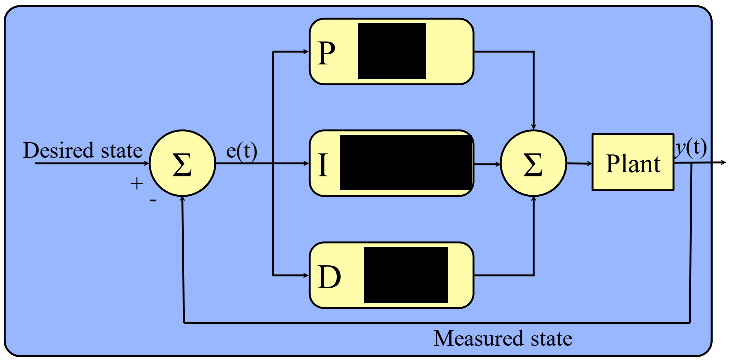


# Appropriate Parameters

- Some applications may not require the use of all three terms
  – appropriate parameters can be set to zero in order to remove unused terms
- PI, PD, P or I controllers
  – PI particularly common since
    - the integral term is required in order to remove steady-state error
    - the derivative term is very sensitive to measurement noise (example?)



# PID Block Diagram






# PID Parameters and their Effects

# Proportional Term

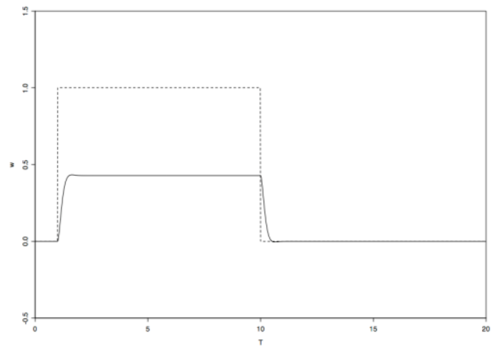
$$P_{\text{out}}(t) = K_p e(t)$$

- A high proportional gain results in a large change in the output for a given error
  - if the proportional gain is too high, the system can become unstable
  - if the proportional gain is too low, the system will be very slow responding to changes
- Proportional term is often the dominant one
  - but pure proportional control will not settle at setpoint



# P Control

P = 20

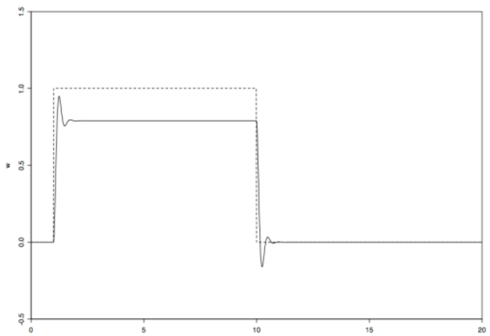


P small: note the steady state error



# P Control

P = 100

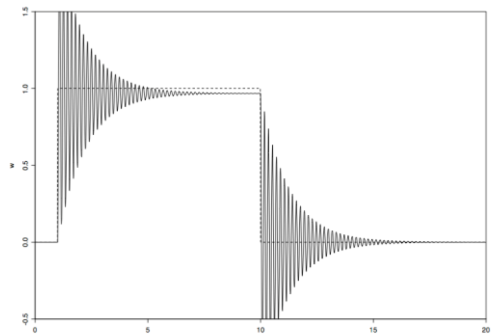


P increases: the steady state error persists, and oscillations appear



# P Control

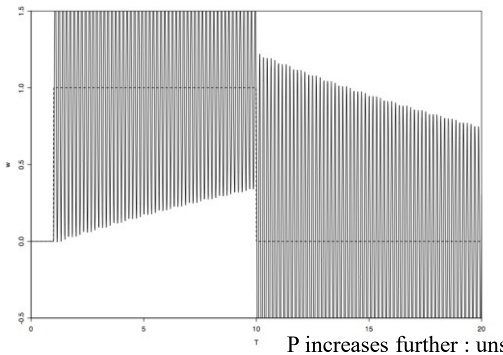
P = 800



P increases: oscillations becoming dominant

## P Control

P = 900



P increases further : unstable behaviour



## Integral Term

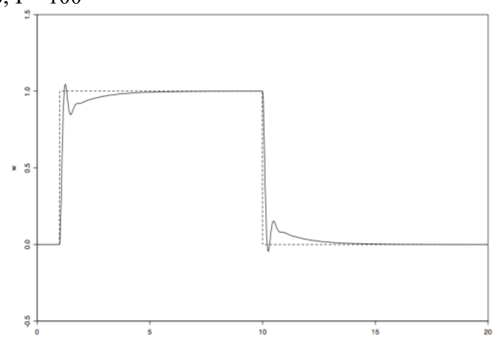
$$I_{\text{out}}(t) = K_i \int_0^t e(\tau)d\tau$$

- Contribution is proportional to both the **magnitude** and the **duration** of the error
  - sum of the instantaneous error over time
- Integral term accelerates the process towards setpoint and eliminates steady-state error
  - but easily causes overshoot (the actual value crosses over the setpoint and creates an error in the opposite direction)



## PI Control

P = 100, I = 100

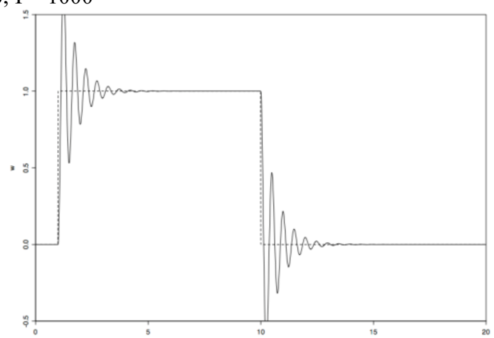


I non-zero: the steady state error is reduced



## PI Control

P = 100, I = 1000



I increases: rise-time improves, but overshoots are significant

## Derivative Term

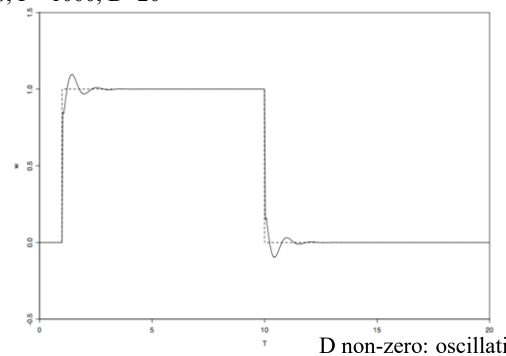
$$D_{\text{out}}(t) = K_d \frac{de(t)}{dt}$$

- Contribution is proportional to the rate of change of the error over time
  - first derivative of the instantaneous error
- Used to slow the rate of change of the error
  - effect is most noticeable near to the setpoint
- Helps reduce overshoot caused by integral term
- Differentiation amplifies signal noise and so derivative can cause instability with noise



## PID Control

P = 100, I = 1000, D=20

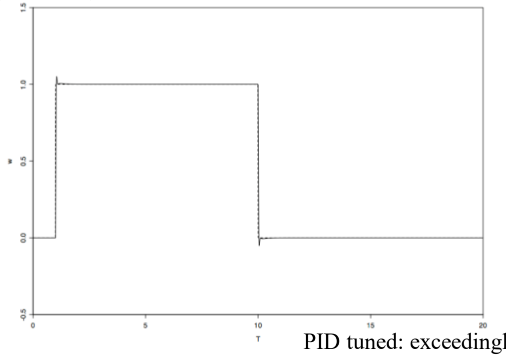


D non-zero: oscillations are damped



## PID Control

P = 300, I = 1000, D=20



PID tuned: exceedingly close control



## PID Summary

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

- Proportional gain: $K_p$

  $\uparrow K_p \Rightarrow$ faster response $\rightarrow$ instability
- Integral gain: $K_i$

  $\uparrow K_i \Rightarrow$ elimination of steady state error $\rightarrow$ overshoot
- Derivative gain: $K_d$

  $\uparrow K_d \Rightarrow$ decreases overshoot, slows transient response $\rightarrow$ instability (particularly under noise)

# PID Tuning



## Parameter Tuning

- If PID parameters are chosen incorrectly
  - the process may be unstable
  - its output may diverge, with/without oscillation
- The parameters (i.e. weights) must be adjusted to achieve the desired behaviour for a given application
  - PID parameter tuning
- The optimum behaviour is application dependent
  - rise-time must be less than a specified time
  - overshoot may not be allowed (e.g., engines)
  - minimise the energy required to reach setpoint
  - oscillations may not be permitted



## Effects of *Increasing* Parameters

| *Parameter* | *Rise-time* | *Overshoot* | *Settling time* | *Steady-state error* |
|---|---|---|---|---|
| $K_p$ | decrease | increase | small change | decrease |
| $K_i$ | decrease | increase | increase | eliminate |
| $K_d$ | small change | decrease | decrease | none |



## Tuning Methods

- Manual heuristic method
  - set I and D to zero
  - increase P until the output oscillates
  - set P to about half the critical oscillating value
  - increase I until the steady-state error is eliminated in an appropriate amount of time for the application
  - increase D until overshoot is reduced to acceptable level
- Automated (software) tuning
  - repeat
    - automatically induce setpoint changes
    - analyse the process characteristics
    - automatically adjust parameters
  - until acceptable

## Other methods, e.g.: Ziegler-Nichols Method

- Set I and D to zero
  - increase P until it reaches the critical gain, $K_c$, at which output oscillates
  - note the oscillation period, $P_c$

| | | | |
|---|---|---|---|
| | $0.5\ K_c$ | - | - |
| | $0.45\ K_c$ | $1.2\ K_p / P_c$ | - |
| | $0.6\ K_c$ | $2\ K_p / P_c$ | $K_p\ P_c / 8$ |



## Live Example: PID control for a DC Motor

- Example in Matlab / SimuLink
  - The example is available here: http://www.mathworks.co.uk/matlabcentral/fileexchange/26275-pid-controller-design-for-a-dc-motor?s_iid=ovp_custom1_1363833138001-68881_rr
  - The provided model allows you to adjust the PID controller and see the effect of the changes to the parameters in terms of response time and avershoot.
  - Note the drastic increase in rapid voltage variation arising from tuning for rapid response – such variation will decrease the life-time of the DC motor.
  - → PID tuning is often not *only* about the output



# PID Implementation



## PID Control – Implementation Notes

- Continuous form of PID:

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

- In digital implementations, we want a discrete form over the last $k$ iterations, where $\Delta t$ is the sampling time (i.e. time between iterations):

$$C(t) = K_p e(t) + K_i \sum_{i=1}^{k} e(t_i)\Delta t + K_d \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

## PID Pseudo-Code

```
previous_error = setpoint – measured_feedback;
Integral = 0;
while(true)
{
    wait(dt);
    error = setpoint – measured_feedback;
    integral = integral + error * dt;
    derivative = (error – previous_error) / dt;
    previous_error = error;
    output =    Kp * error +
                Ki * integral +
                Kd * derivative;
}
```



## Applications of PID

Examples?



## The Grand Challenge

Source: http://en.wikipedia.org/wiki/File:DesertToCity.jpg

- See here:
  - http://en.wikipedia.org/wiki/DARPA_Grand_Challenge



## Summary

- Summary of this lecture
  - Control
    - The problem
    - Open-Loop Control
  - PID Control
    - PID principles
    - PID parameter effects
    - PID tuning
    - Live example – PID DC motor control
    - PID implementation
- Video: The Grand Challenge