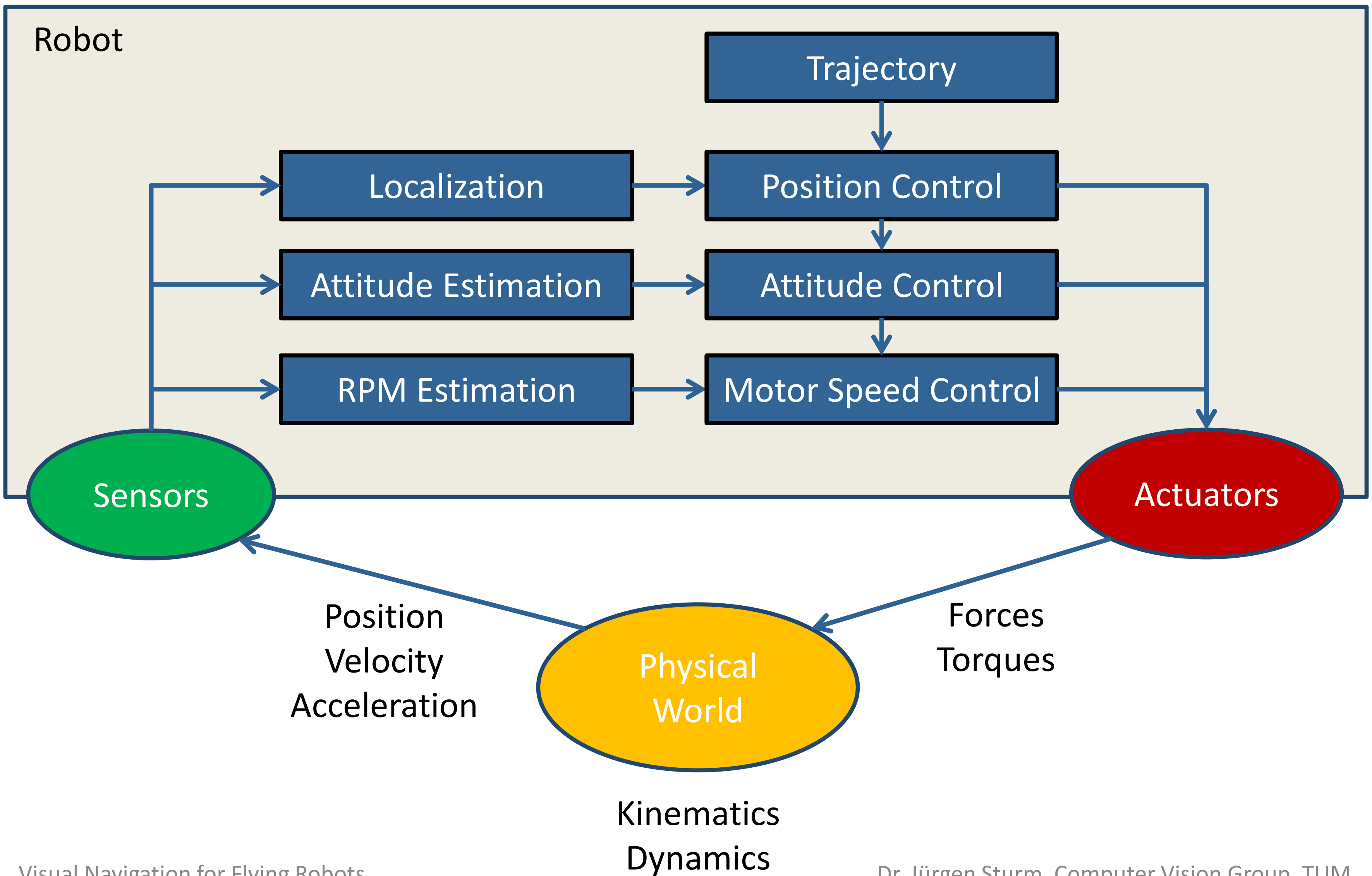
Visual Navigation for Flying Robots

Robot Control

Dr. Jürgen Sturm

Control Architecture

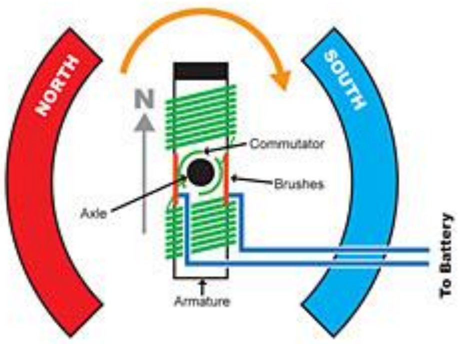


Agenda for Today

- Motors
- Motor Controllers
- Kinematics and Dynamics
- Linear Control

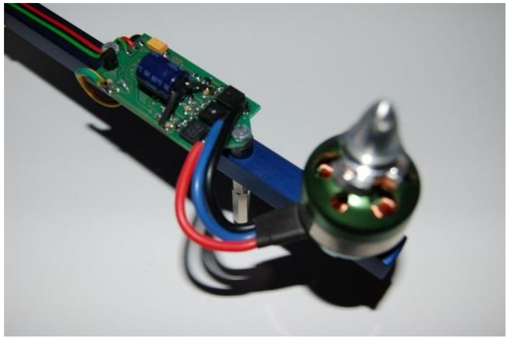
DC Motors

- Maybe you have built one in school
- Stationary permanent magnet
- Electromagnet induces torque
- Split ring switches direction of current

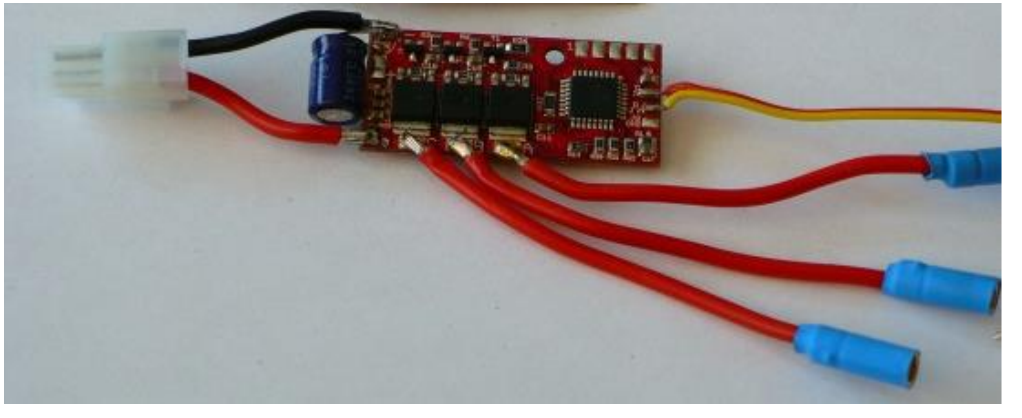



Brushless Motors

- Used in most quadrocopters
- Permanent magnets on the axis
- Electromagnets on the outside

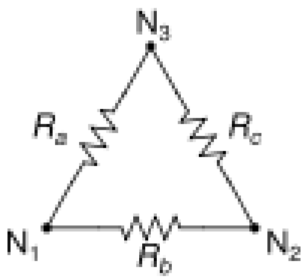


→ Does not require brushes (less maintenance)

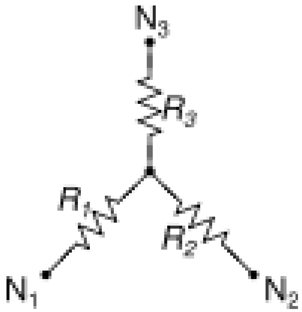


Brushless Motors

- Winding styles

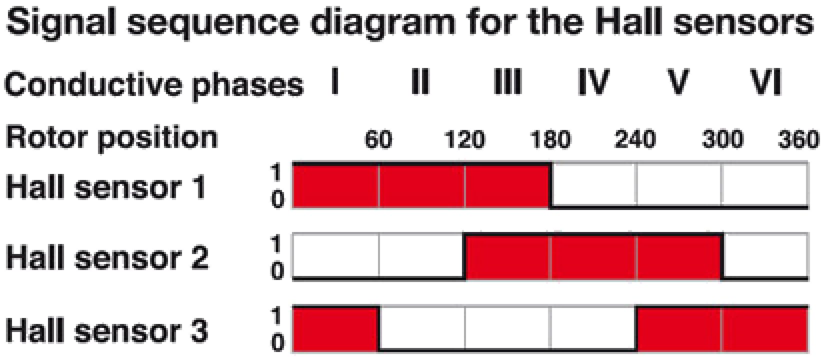


good at
low speeds



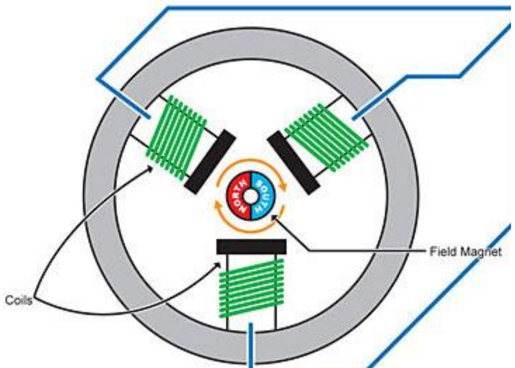
good at
high speeds

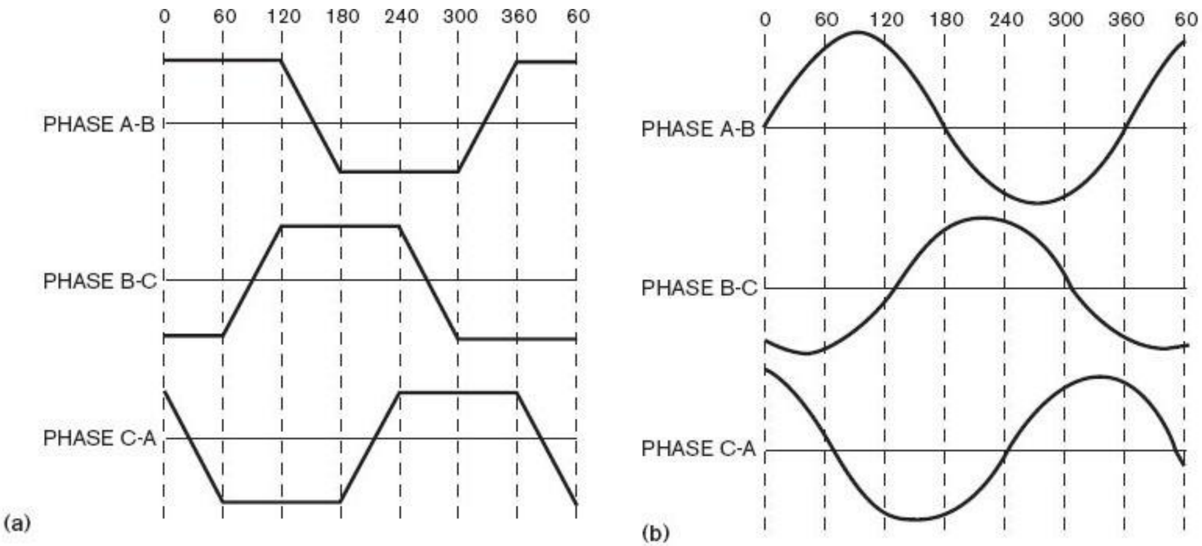
- Hall sensor or EMF to detect rotation



Motor Controller

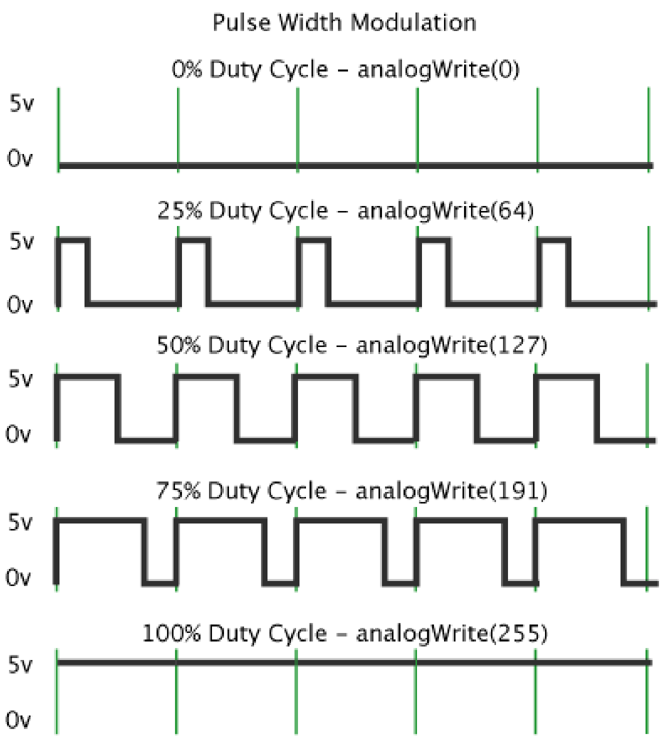
- Micro controller estimates rotation and generates PWM signal
- AC signal generator (inverter) generates motor phases

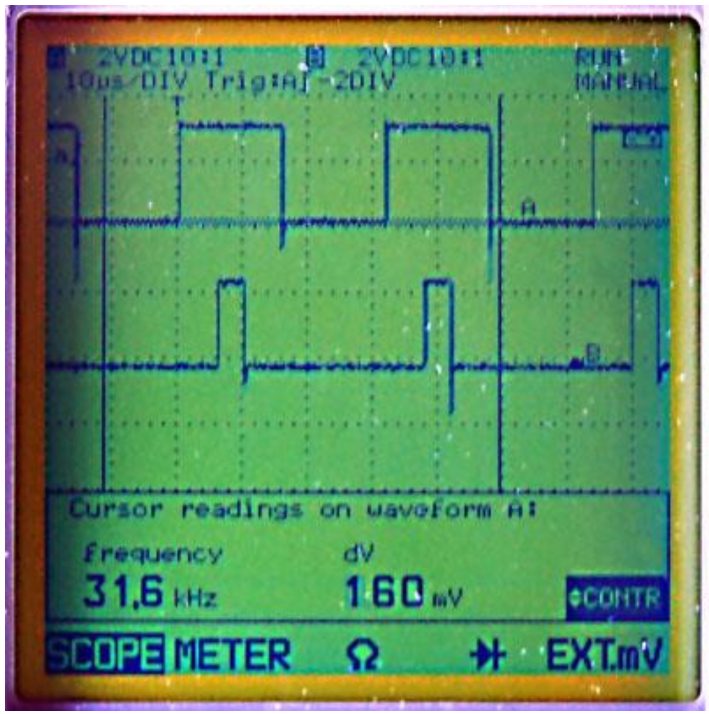




Pulse Width Modulation (PWM)

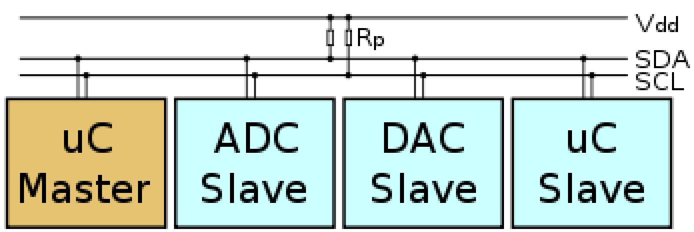
- Protocol used to control motor speed
- Remote controls typically output PWM

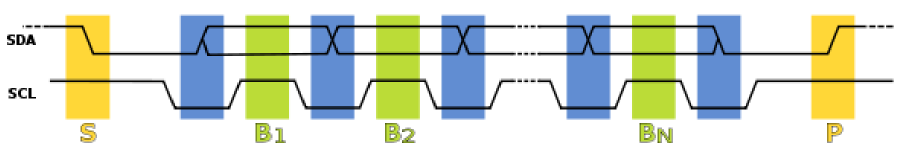




I2C Protocol

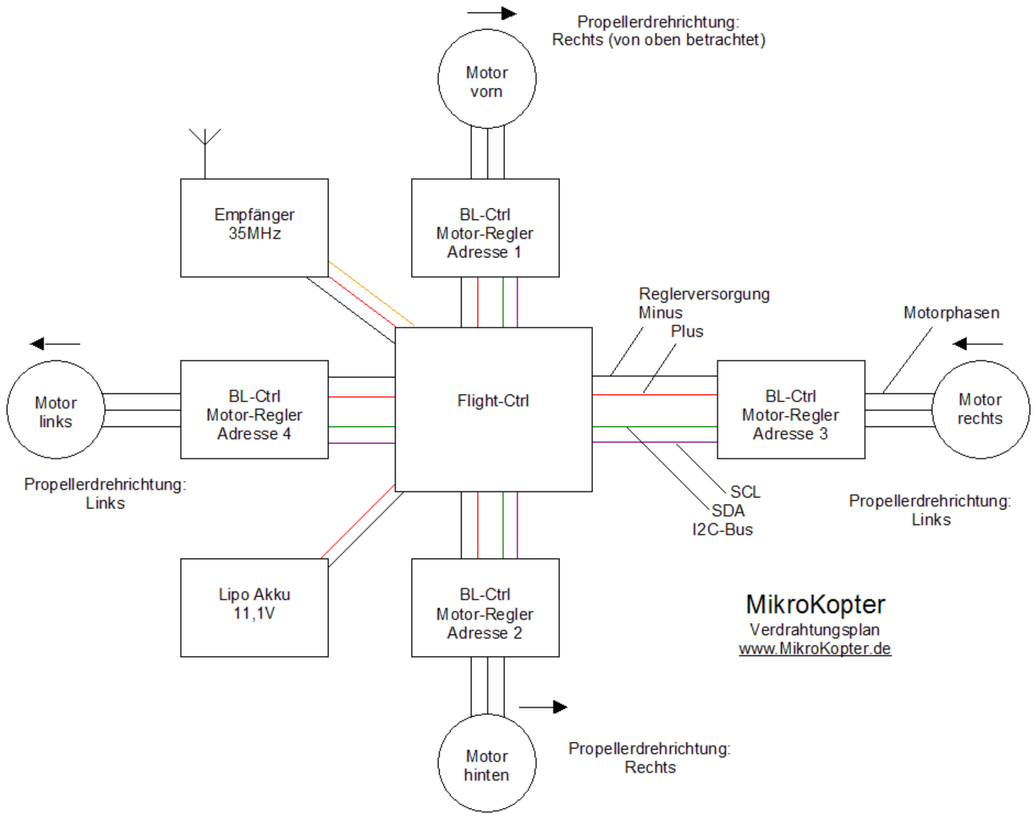
- Serial data line (SDA) + serial clock line (SCL)
- All devices connected in parallel
- 7-10 bit address, 100-3400 kbit/s speed
- Used by Mikrocopter for motor control

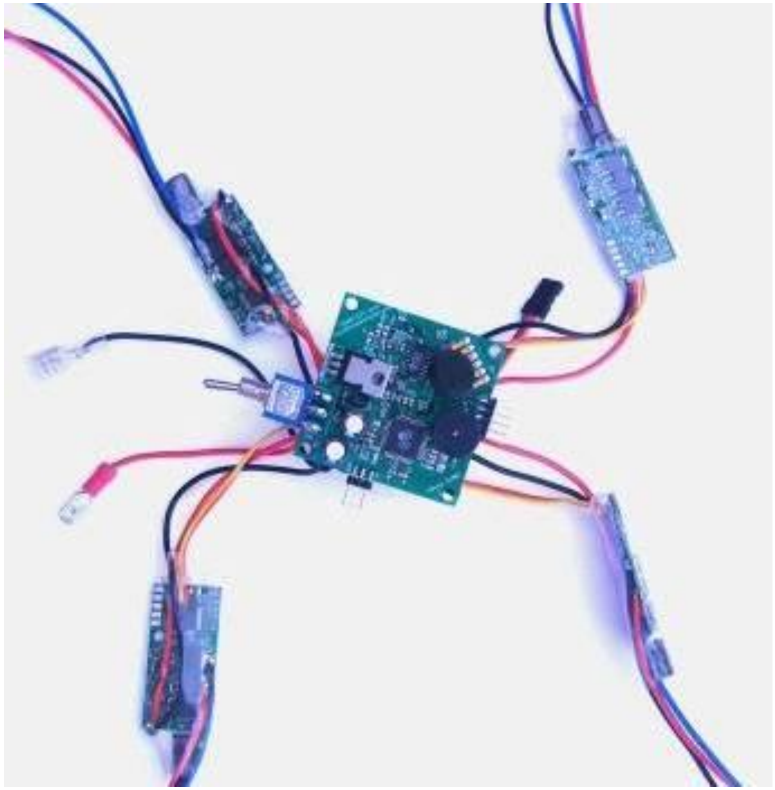




Attitude + Motor Controller Boards

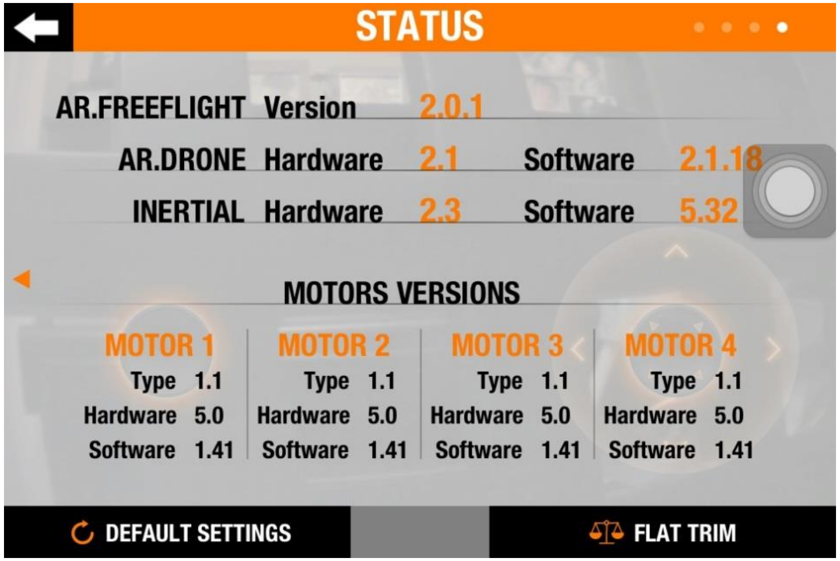
- Example: Mikrokopter Platform



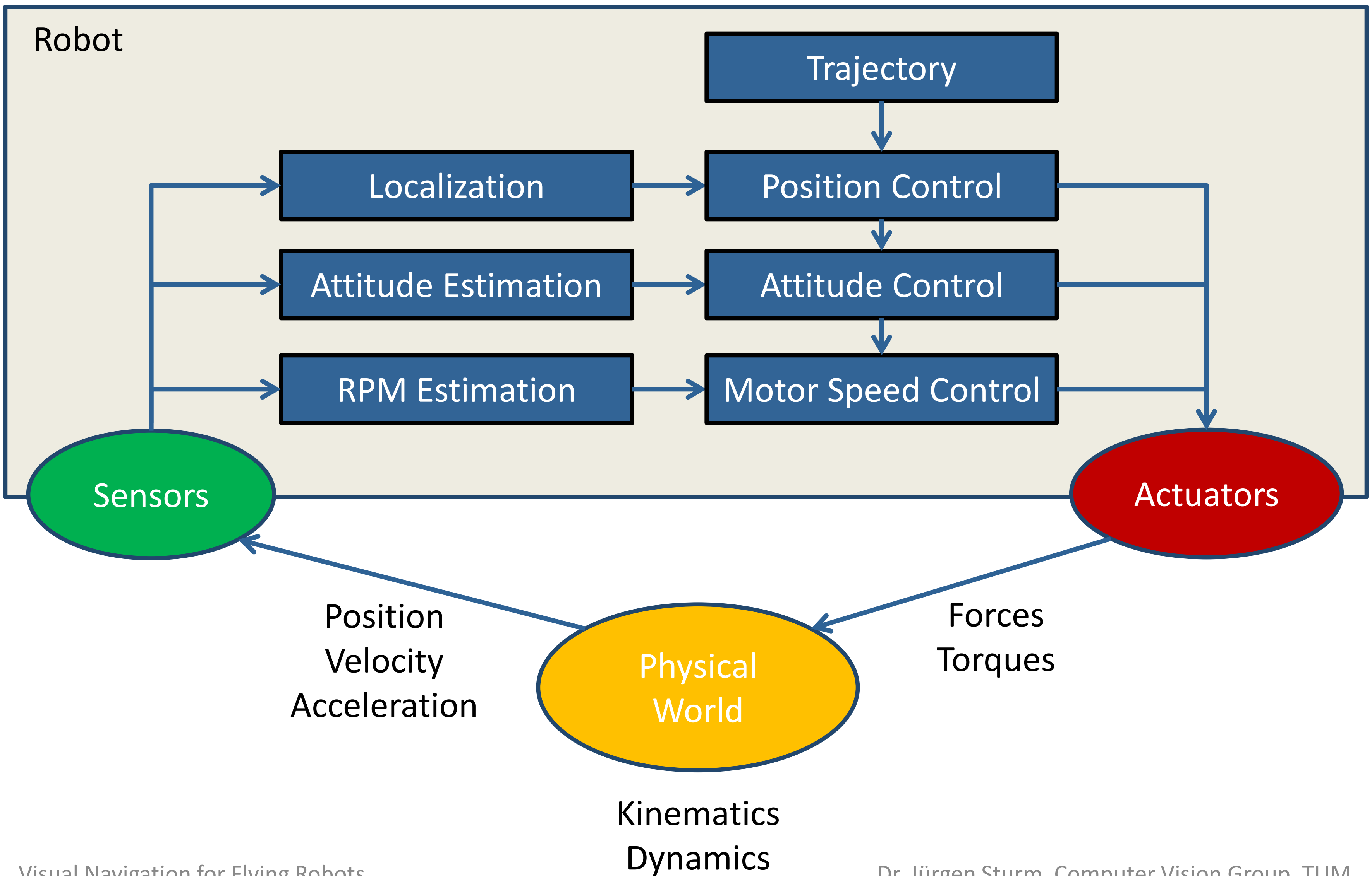


Attitude + Motor Controller Boards

- Example: Ardrone Platform



Control Architecture



Kinematics and Dynamics

- Kinematics
  - Describes the motion of rigid bodies
  - Position, velocity, acceleration
- Dynamics
  - Actuators induce forces and torques
  - Forces induce linear acceleration
  - Torques induce angular acceleration
- What types of forces do you know?
- What types of torques do you know?

Example: 1D Kinematics

- State $\mathbf{x} = \begin{pmatrix} x & \dot{x} & \ddot{x} \end{pmatrix}^\top \in \mathbb{R}^3$
- Action $u \in \mathbb{R}$
- Linear process model

$$\mathbf{x}_t = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_t$$

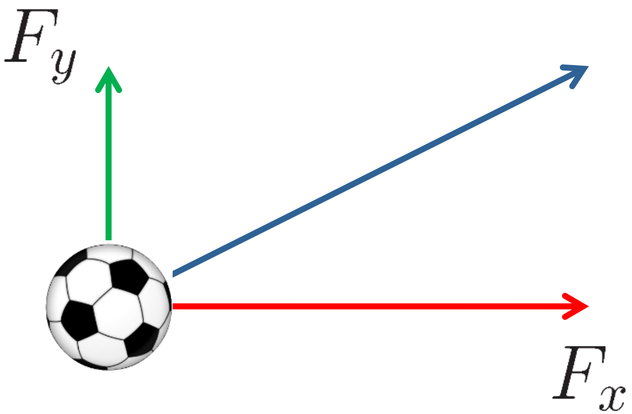
- Kalman filter
- How many states do we need for 3D?

Dynamics - Essential Equations

- Force (Kraft)

$$m\ddot{\mathbf{x}} = \sum_i F_i$$

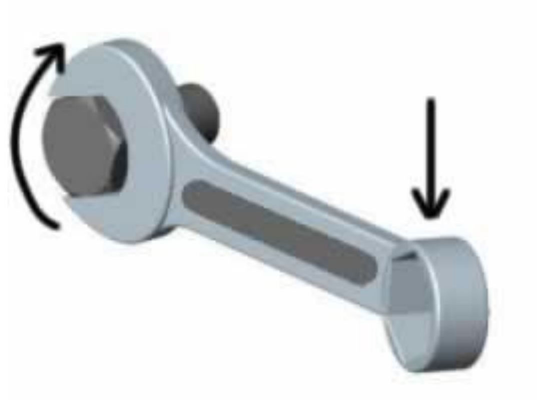
linear acceleration



- Torque (Drehmoment)

$$J\boldsymbol{\alpha} = \sum_i \boldsymbol{\tau}_i$$

angular acceleration

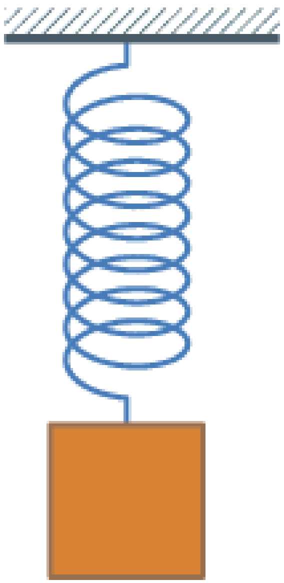


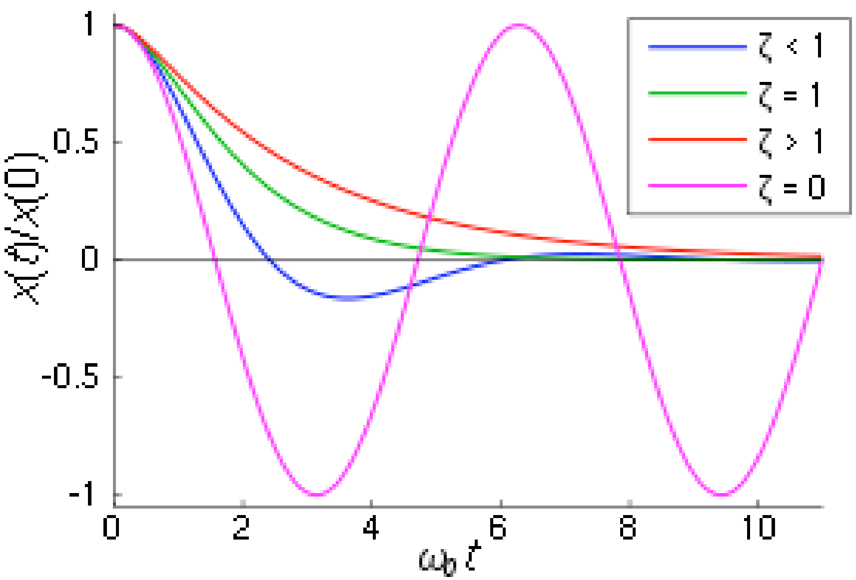
Forces

- Gravity $F_{\text{grav}} = mg$
- Friction
  - Stiction (static friction) $F_{\text{stiction}} = c_s \text{sign } \dot{x}$
  - Damping (viscous friction) $F_{\text{damping}} = D\dot{x}$
  - Air drag $F_{\text{airdrag}} = c_W A \frac{1}{2} \rho \dot{x}$
- Spring $F_{\text{spring}} = K(x - x_{\text{eq}})$
- Magnetic force
- …

Example: Spring-Damper System

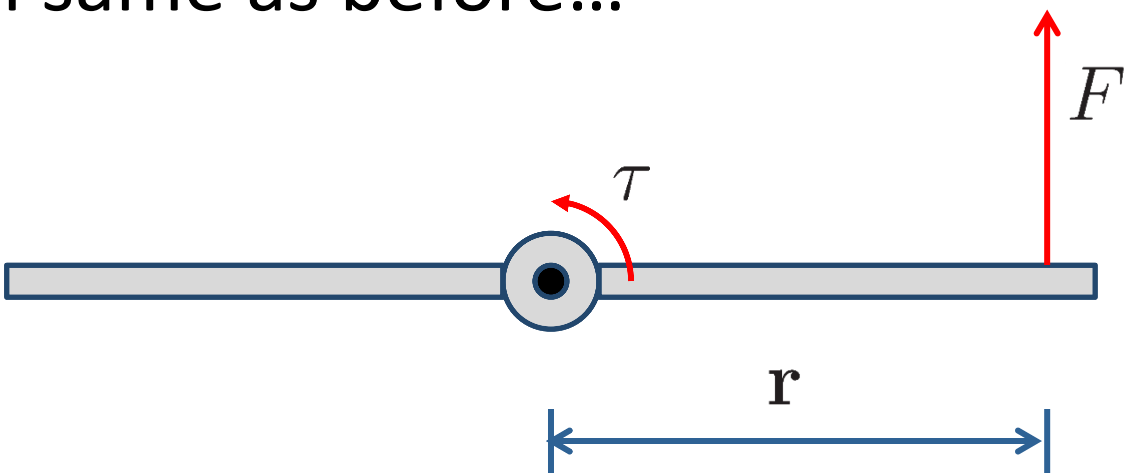
- Combination of spring and damper
- Forces $F = F_{\text{damping}} + F_{\text{spring}}$
- Resulting dynamics $m\ddot{x} = D\dot{x} + K(x - x_{\text{eq}})$





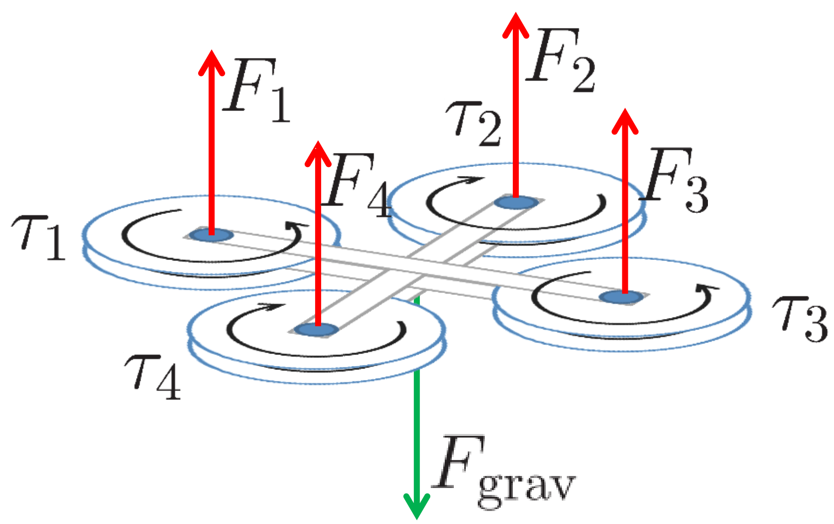
Torques

- Definition $\boldsymbol{\tau} = F \times \mathbf{r}$
- Torques sum up $\boldsymbol{\tau}_{\text{net}} = \sum \boldsymbol{\tau}_i$
- Torque results in angular acceleration $\boldsymbol{\tau} = J\boldsymbol{\alpha}$ (with $\boldsymbol{\alpha} = \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t}$, J moment of inertia)
- Friction same as before…



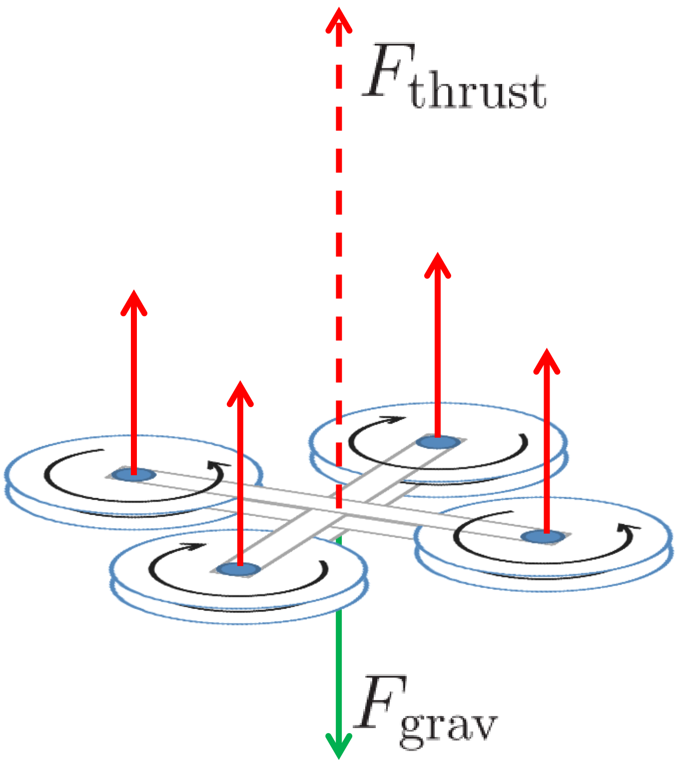
Dynamics of a Quadrocopter

- Each propeller induces force and torque by accelerating air
- Gravity pulls quadrocopter downwards



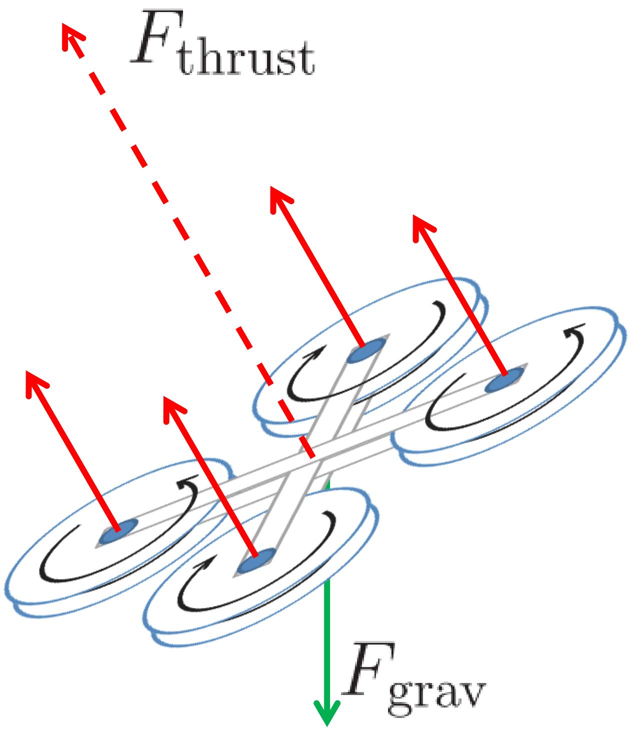
Vertical Acceleration

- Thrust $F_{\text{thrust}} = F_1 + F_2 + F_3 + F_4$



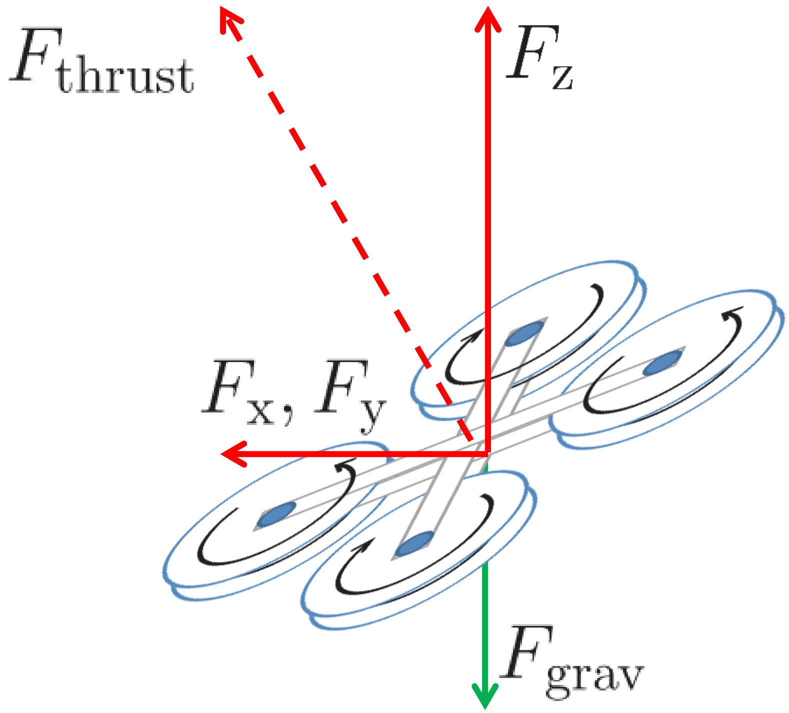
Vertical and Horizontal Acceleration

- Thrust $F_{\text{thrust}} = F_1 + F_2 + F_3 + F_4$



Vertical and Horizontal Acceleration

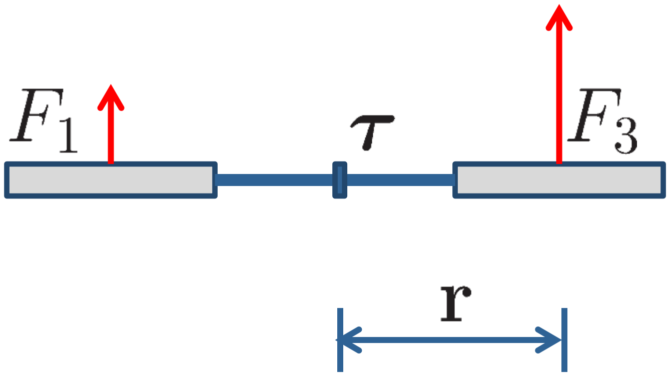
- Thrust $F_{\text{thrust}} = F_1 + F_2 + F_3 + F_4$
- Acceleration $\ddot{\mathbf{x}}_{\text{global}} = (R_{RPY} F_{\text{thrust}} - F_{\text{grav}})/m$



attitude R_{RPY}

Pitch (and Roll)

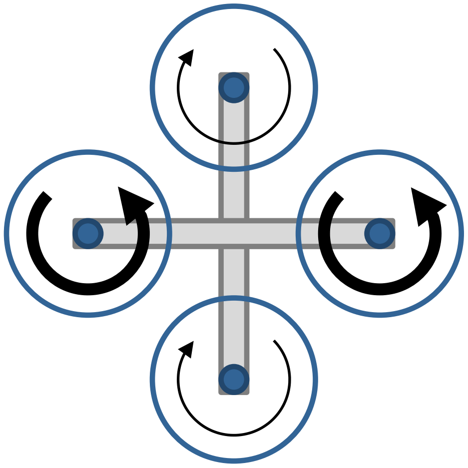
- Attitude changes when opposite motors generate unequal thrust
- Induced torque $\boldsymbol{\tau} = (F_1 - F_3) \times \mathbf{r}$
- Induced angular acceleration $\boldsymbol{\alpha} = J^{-1}\boldsymbol{\tau}$



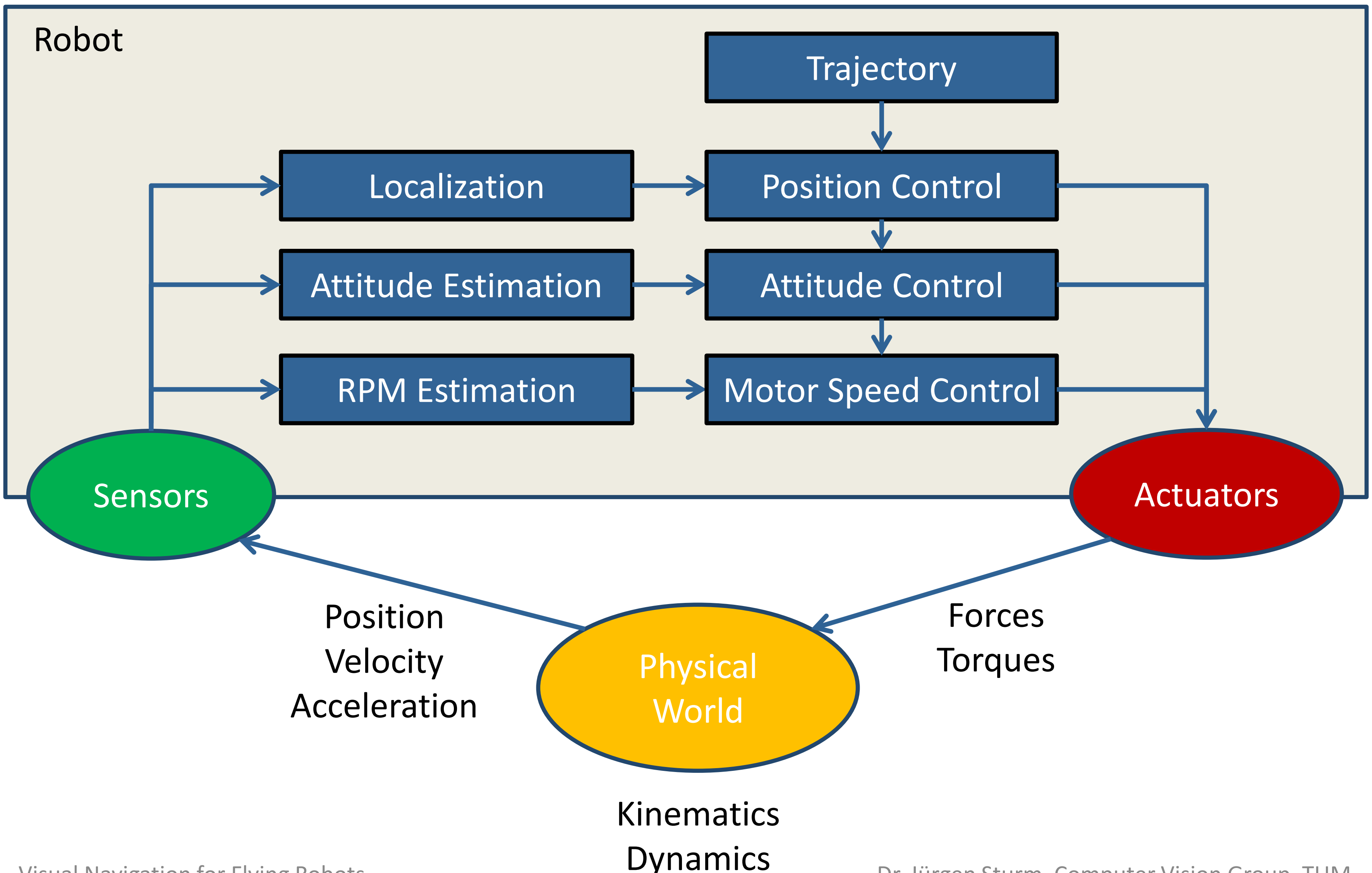
Side view of quadrocopter

Yaw

- Each propeller induces torque due to rotation and the interaction with the air
- Induced torque $\tau = \tau_1 - \tau_2 + \tau_3 - \tau_4$
- Induced angular acceleration



Cascaded Control



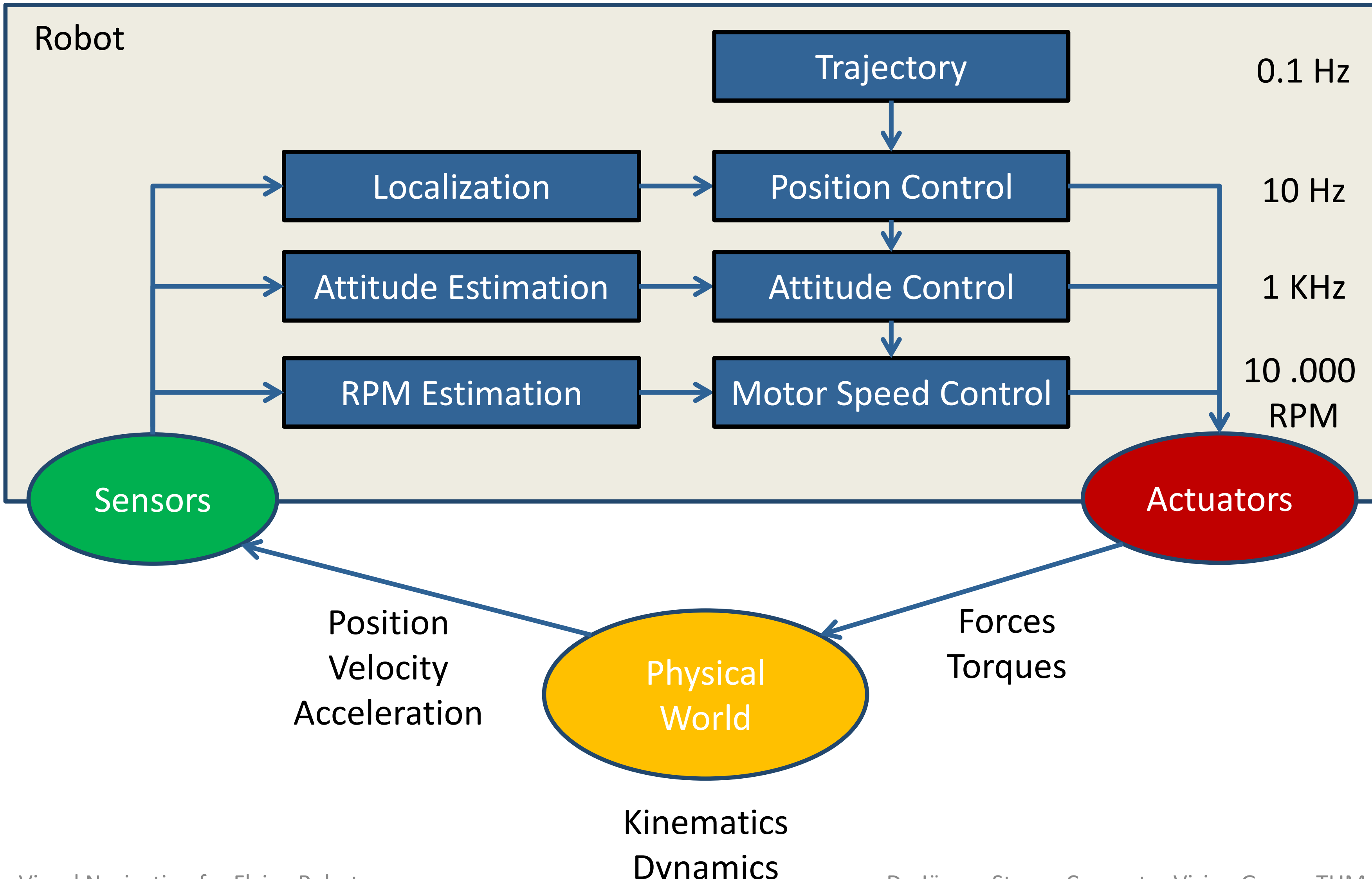
Assumptions of Cascaded Control

- Dynamics of inner loops is so fast that it is not visible from outer loops
- Dynamics of outer loops is so slow that it appears as static to the inner loops

Cascaded Control Example

- Motor control happens on motor boards (controls every motor tick)
- Attitude control implemented on micro-controller with hard real-time (at 1000 Hz)
- Position control (at 10 – 250 Hz)
- Trajectory (waypoint) control (at 0.1 – 1 Hz)

Cascaded Control

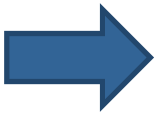


Feedback Control

- Given:
  - Goal state x_{des}
  - Measured state (feedback) z
- Wanted:
  - Control signal u to reach goal state

- How to compute the control signal?

Feedback Control - Generic Idea

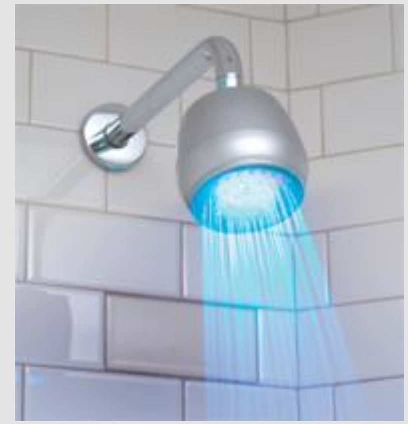


Desired
value
35°

Feedback Control - Generic Idea

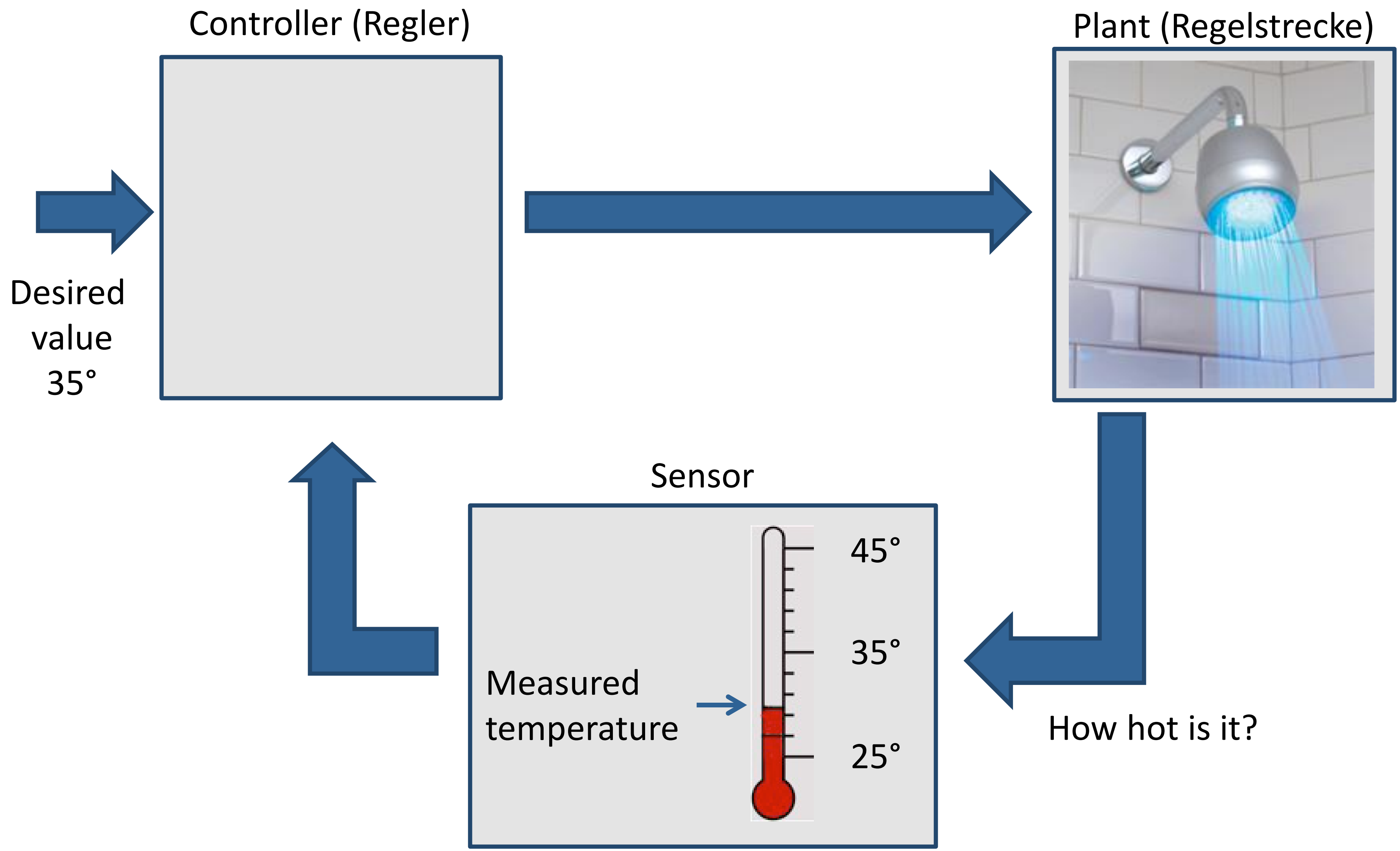
Controller (Regler)

Plant (Regelstrecke)

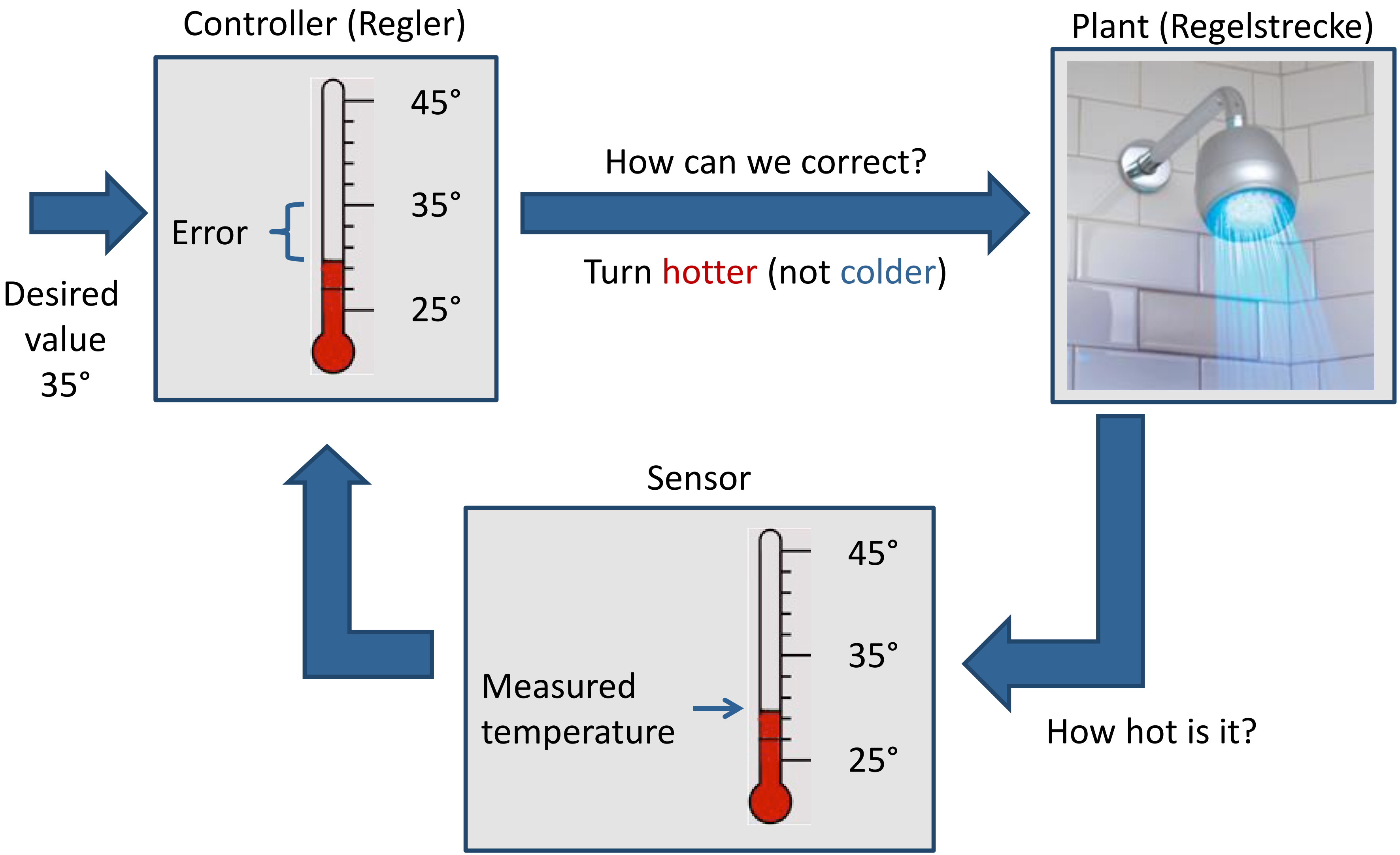


Desired value 35°

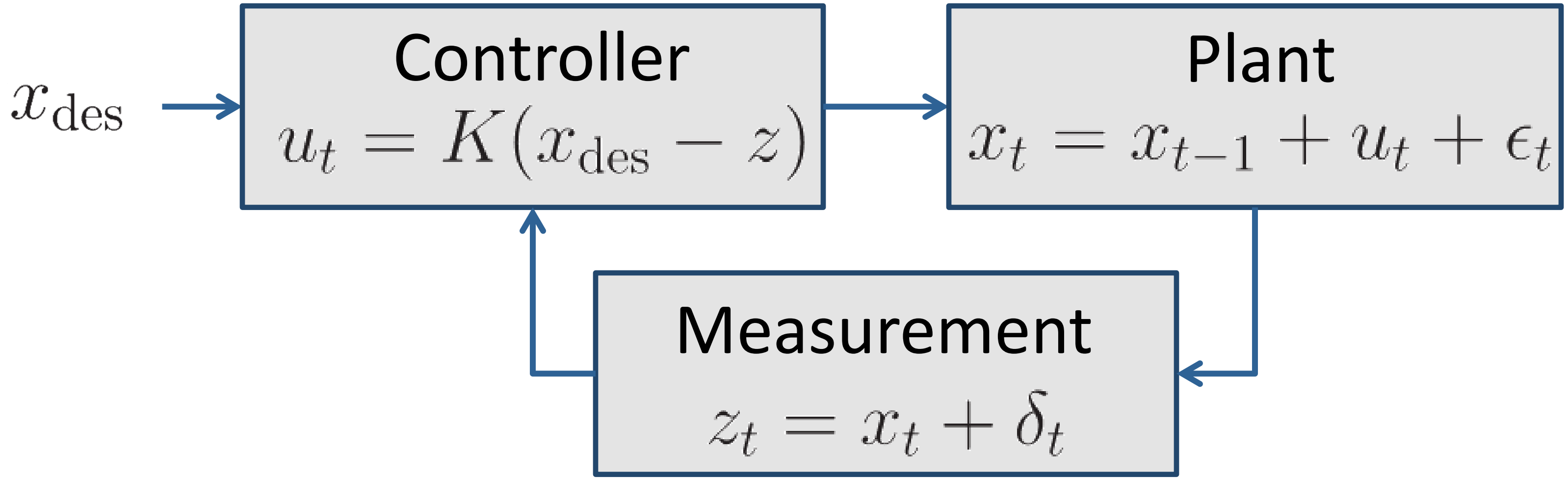
Feedback Control - Generic Idea

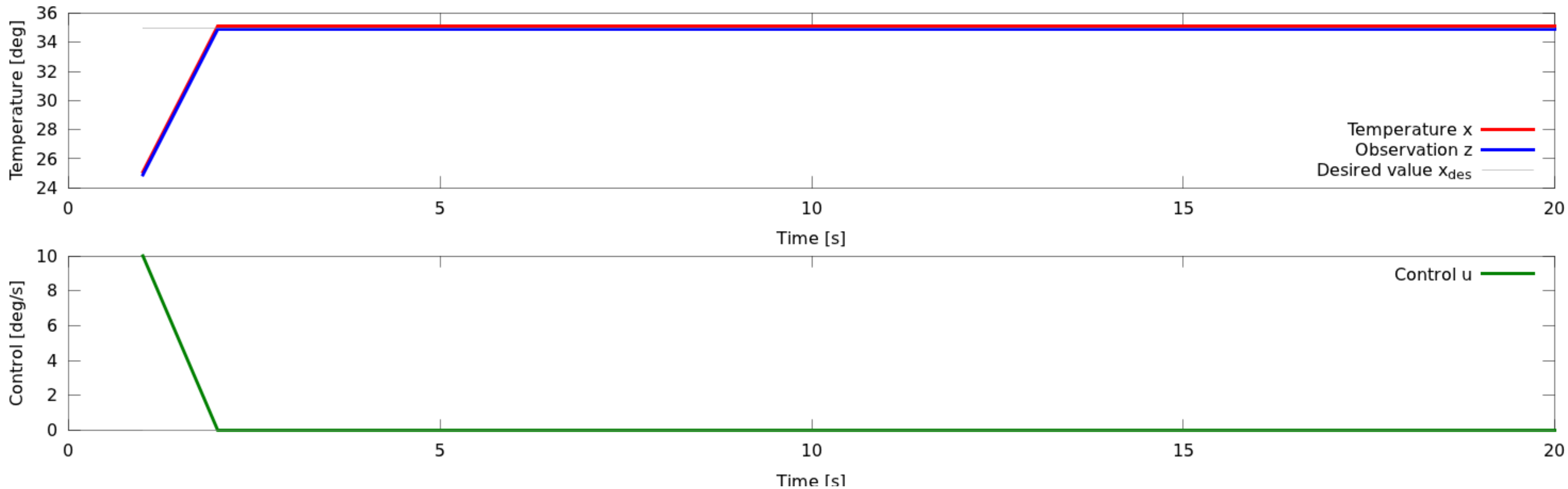


Feedback Control - Generic Idea



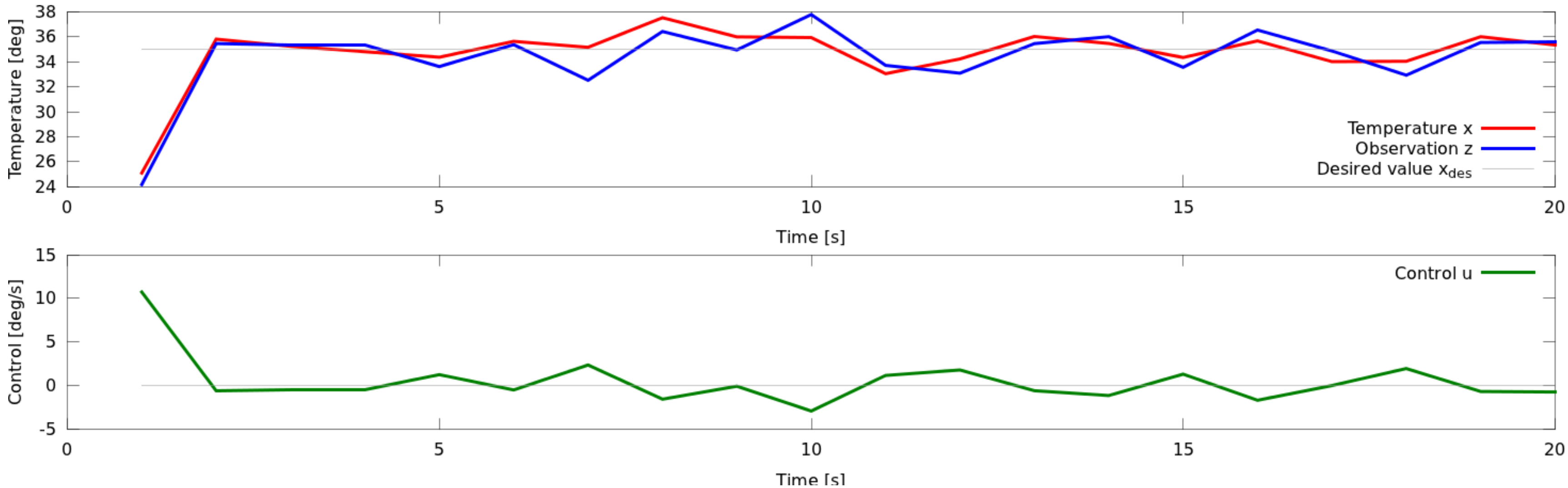
Feedback Control - Example





Measurement Noise

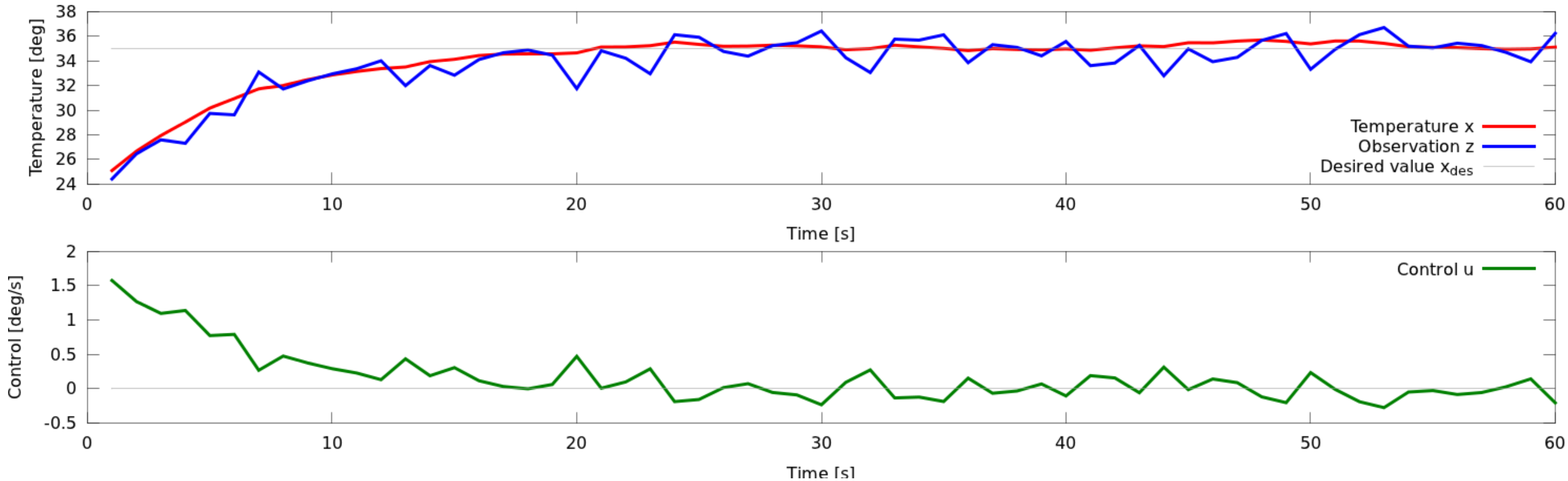
- What effect has noise in the measurements?



- Poor performance for K=1
- How can we fix this?

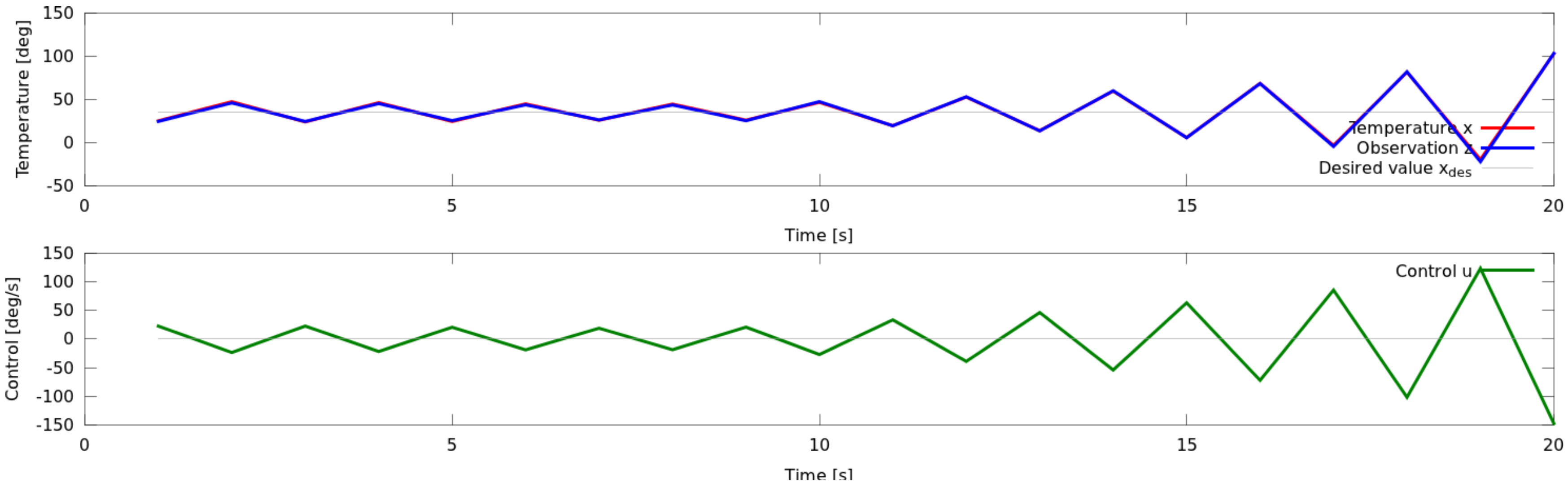
Proper Control with Measurement Noise

- Lower the gain… (K=0.15)



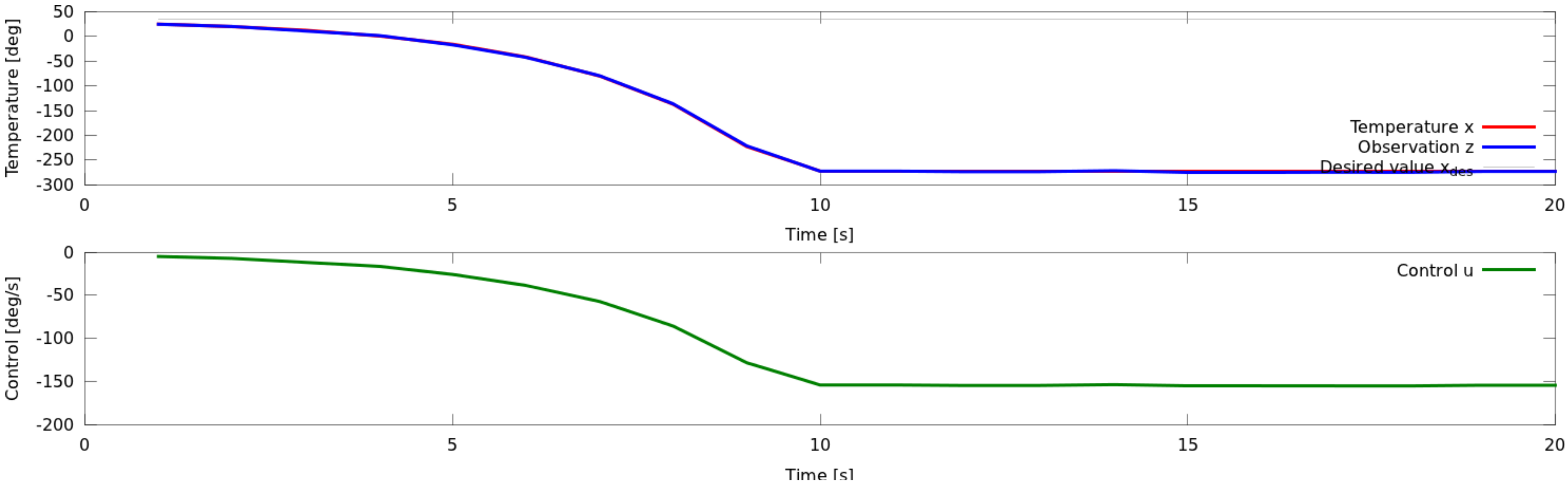
What do High Gains do?

- High gains are always problematic (K=2.15)



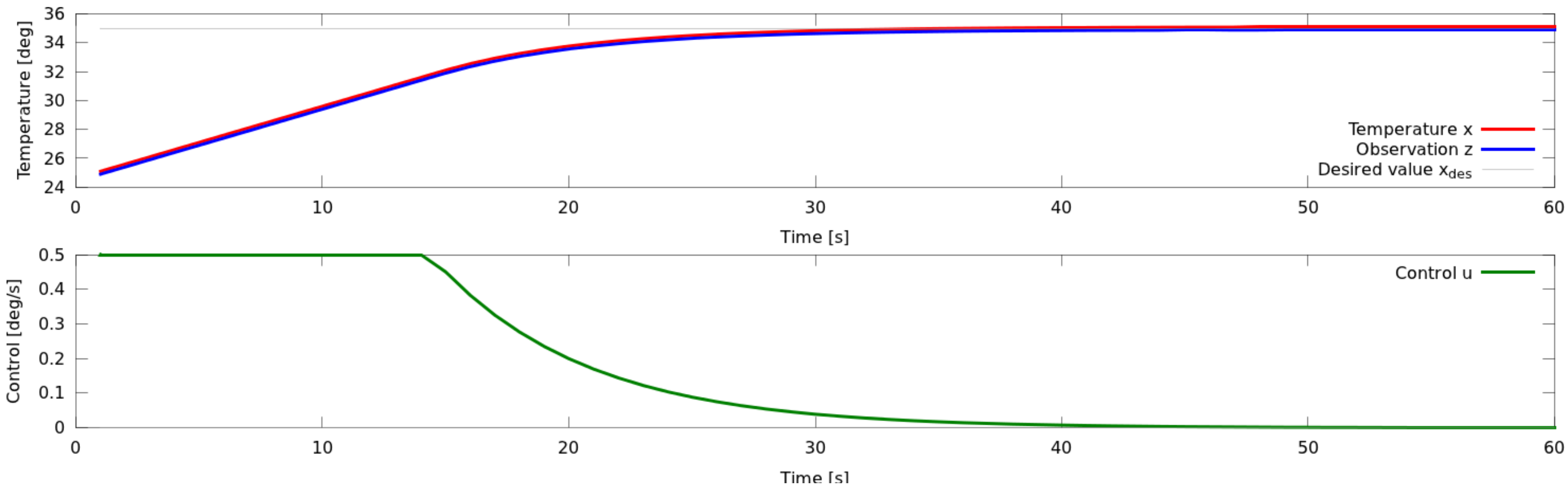
What happens if sign is messed up?

- Check K=-0.5

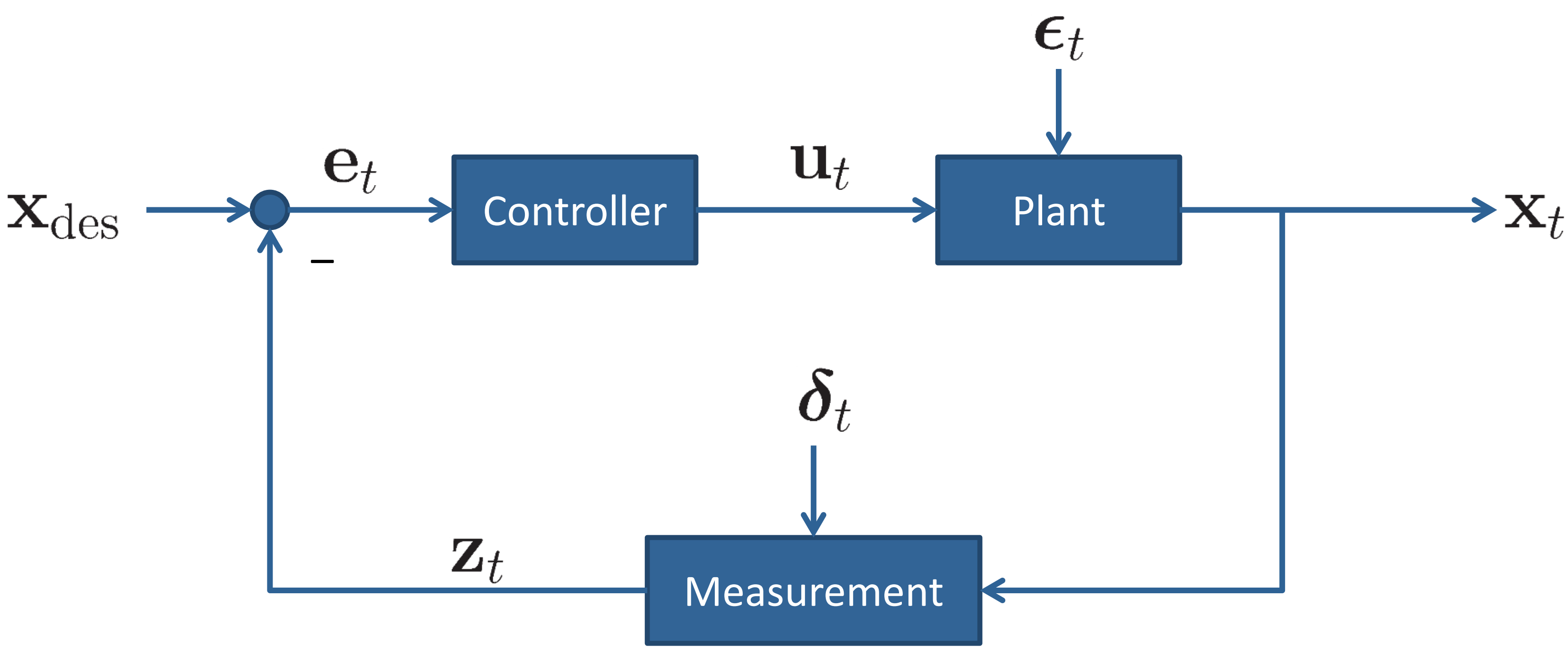


Saturation

- In practice, often the set of admissible controls u is bounded
- This is called (control) saturation

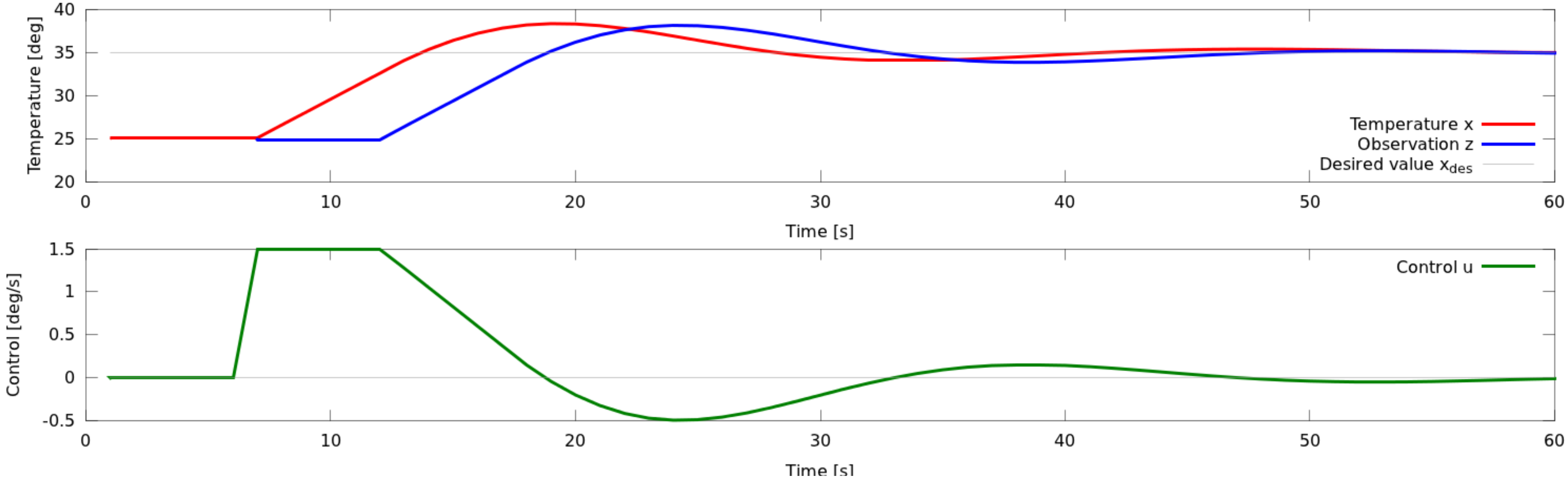


Block Diagram



Delays

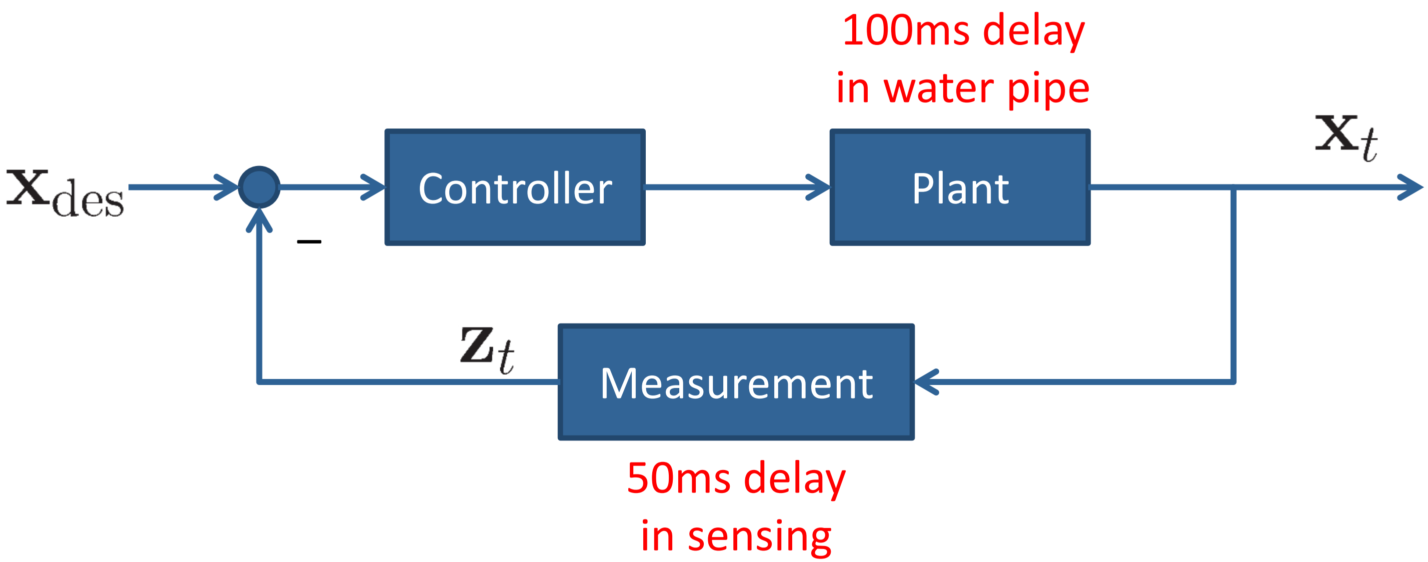
- In practice most systems have delays
- Can lead to overshoots/oscillations/de-stabilization



- One solution: lower gains (why is this bad?)

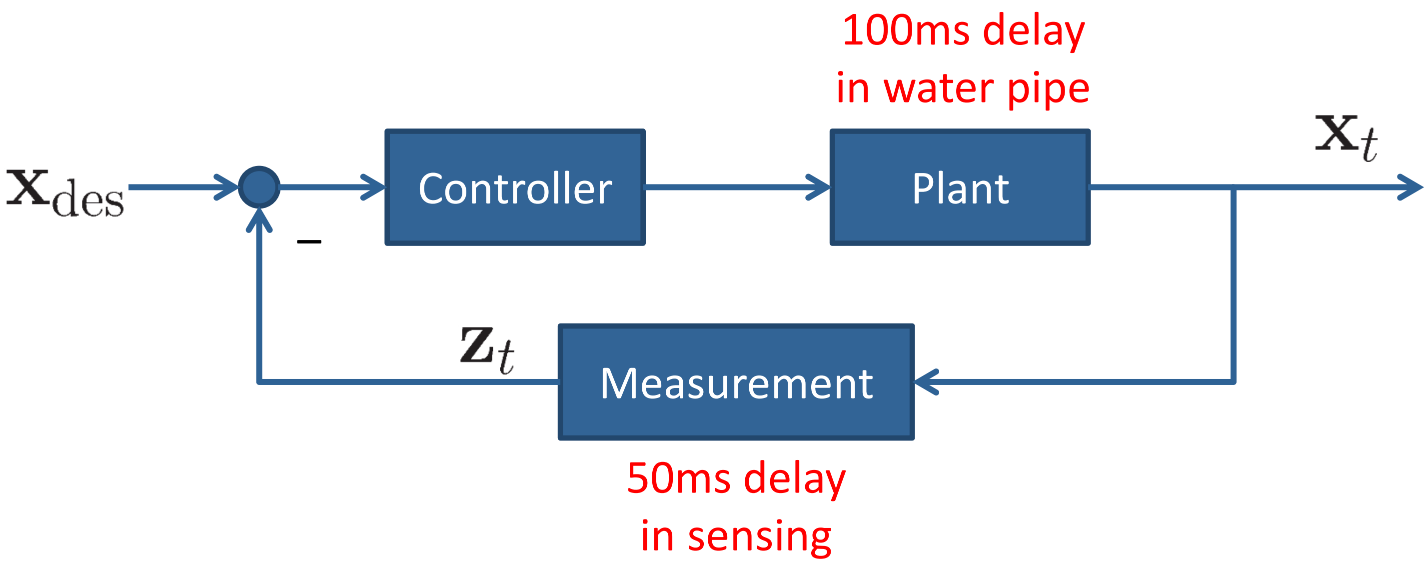
Delays

- What is the total dead time of this system?



Delays

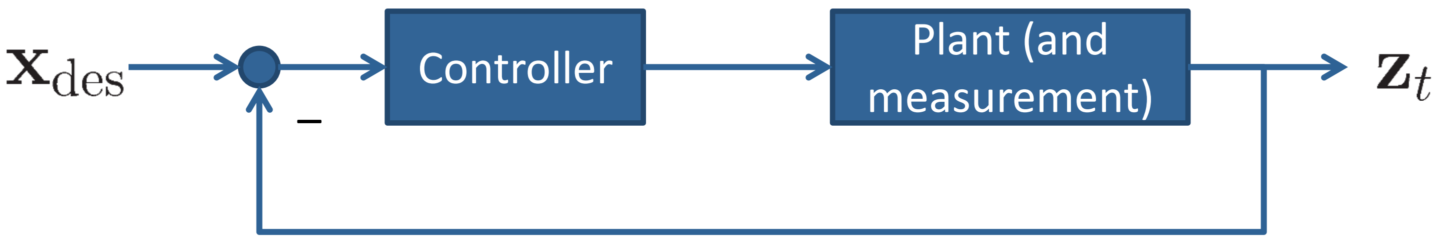
- What is the total dead time of this system?



- Can we distinguish delays in the measurement from delays in actuation?

Delays

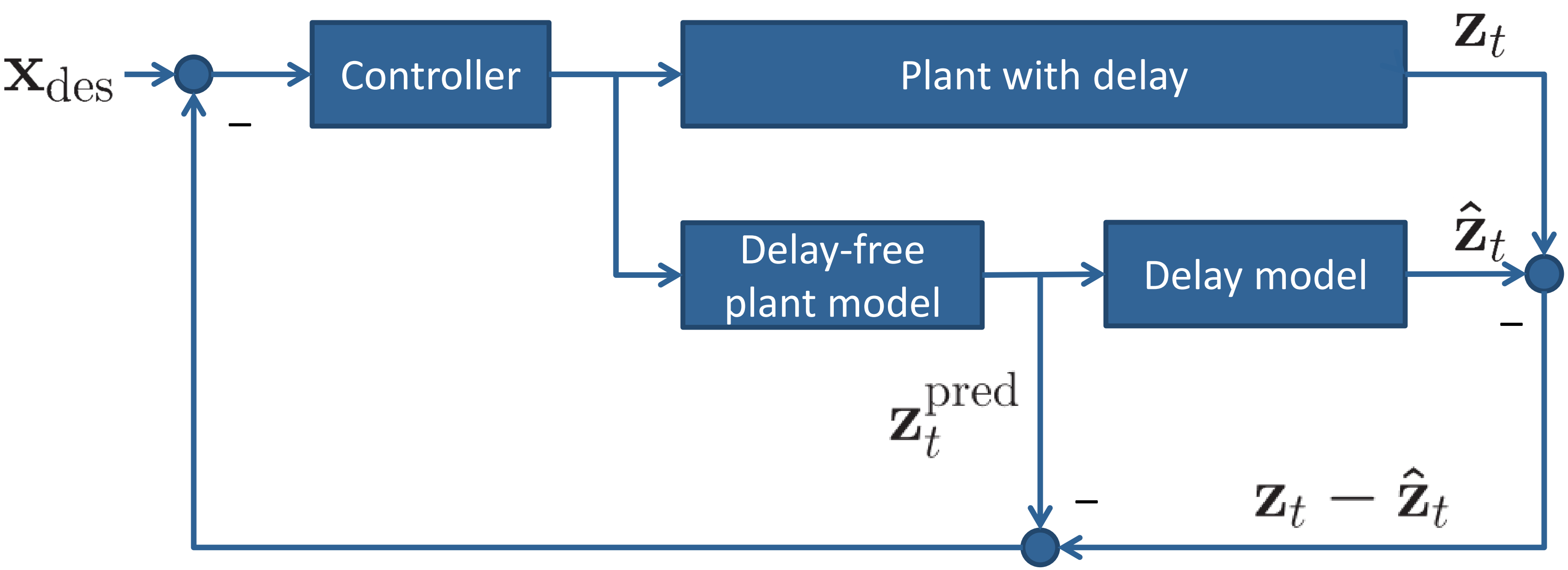
- What is the total dead time of this system?



- Can we distinguish delays in the measurement from delays in actuation? No!

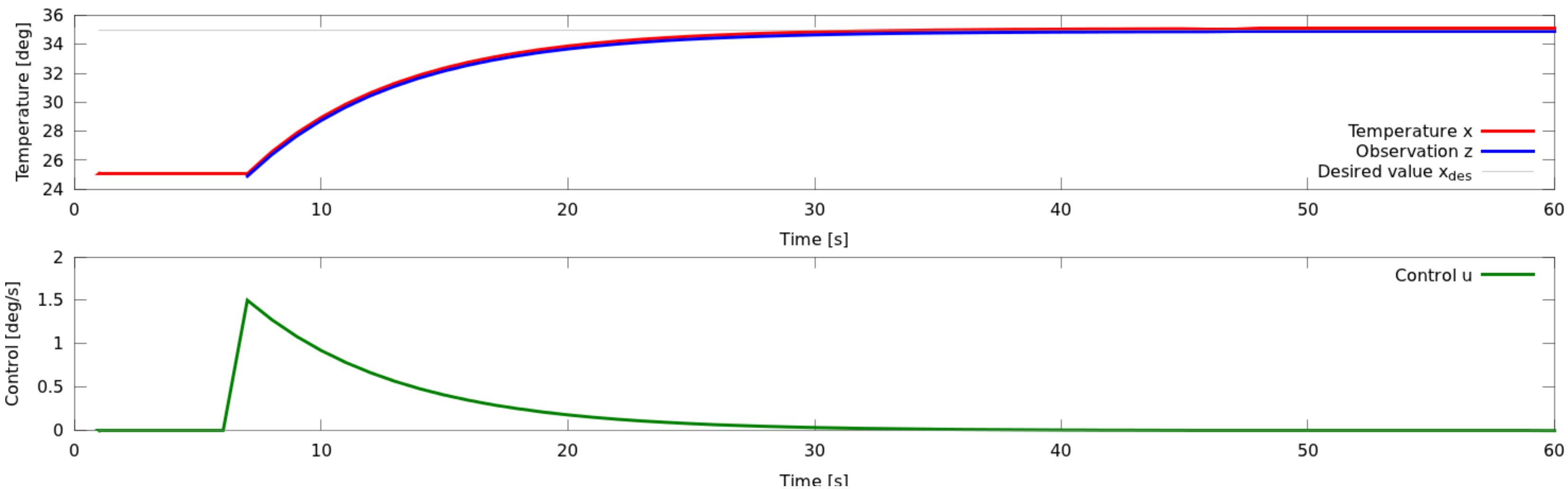
Smith Predictor

- Allows for higher gains
- Requires (accurate) model of plant



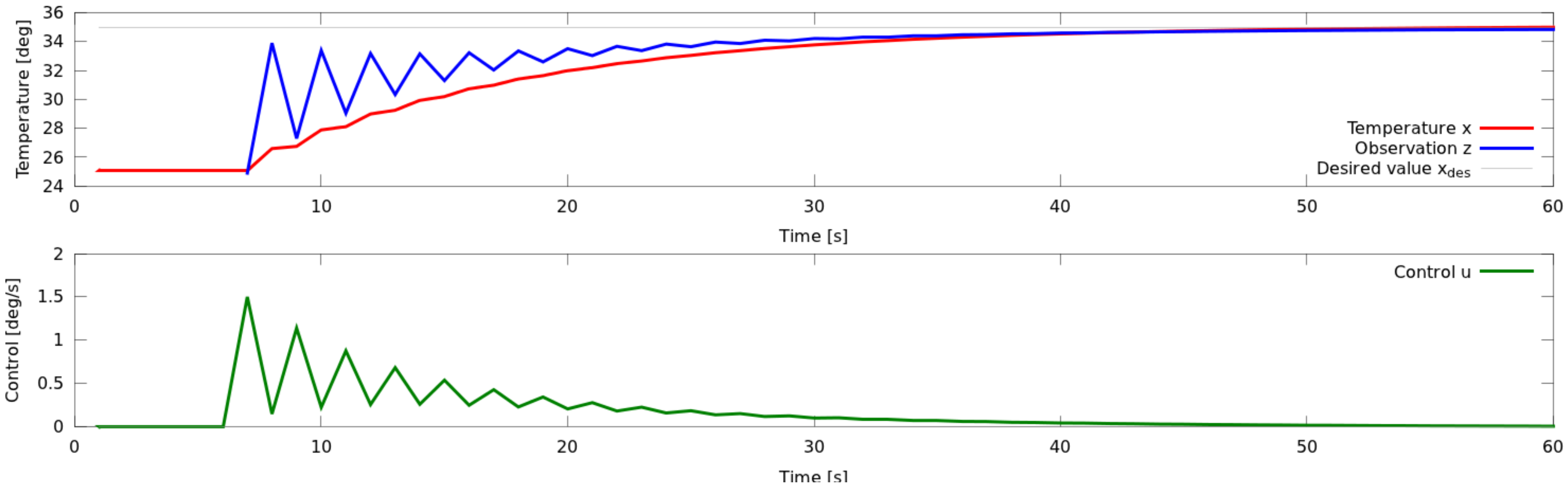
Smith Predictor

- Plant model is available
- 5 seconds delay
- Results in perfect compensation
- Why is this unrealistic in practice?



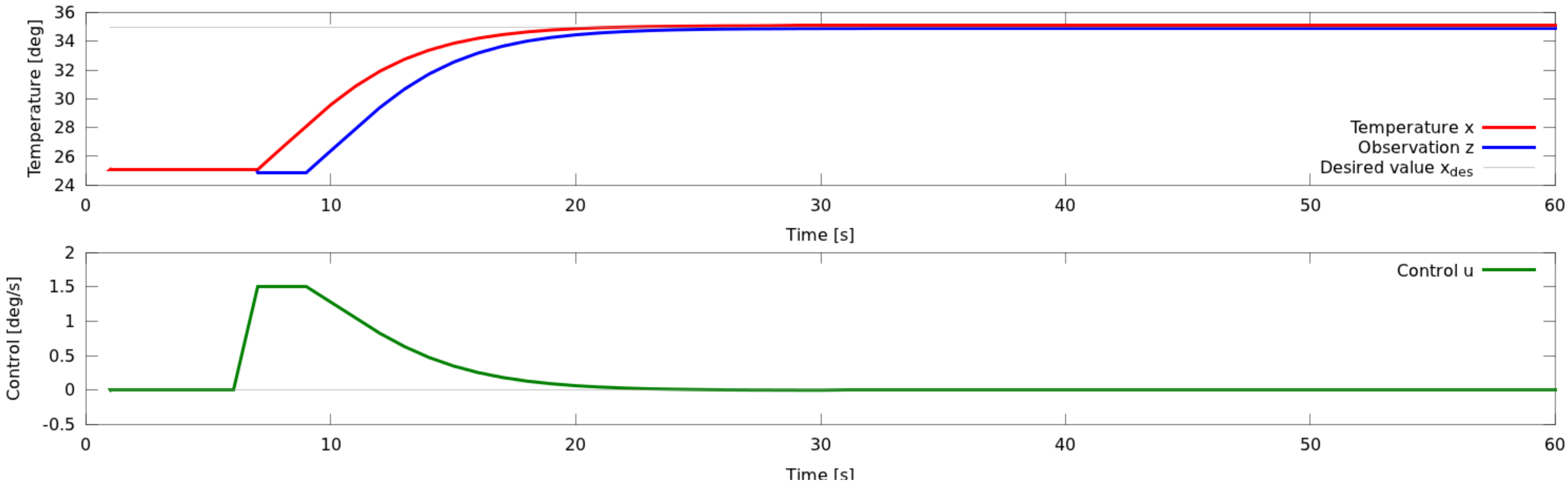
Smith Predictor

- Time delay (and plant model) is often not known accurately (or changes over time)
- What happens if time delay is **over**estimated?

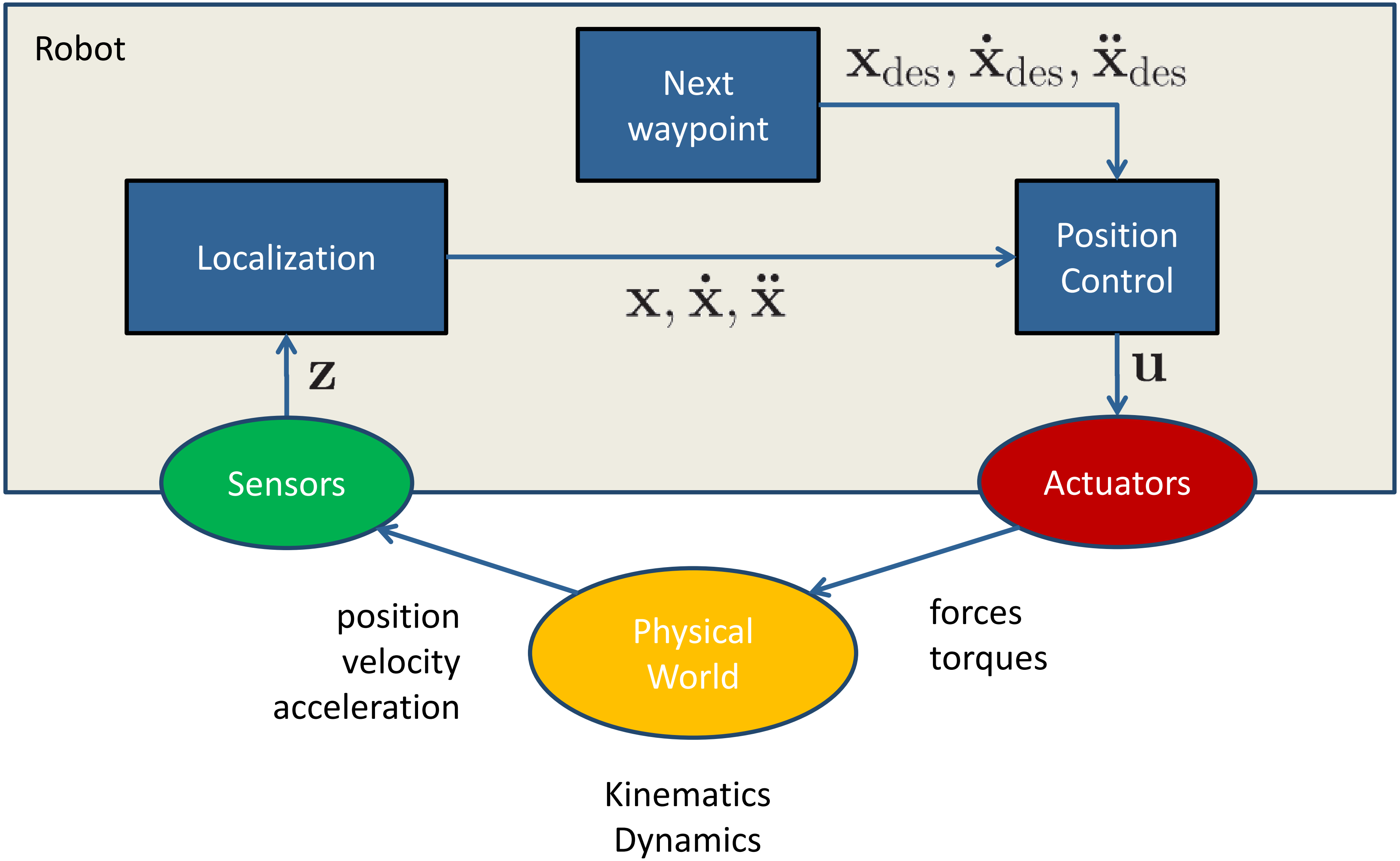


Smith Predictor

- Time delay (and plant model) is often not known accurately (or changes over time)
- What happens if time delay is **underestimated?**



Position Control

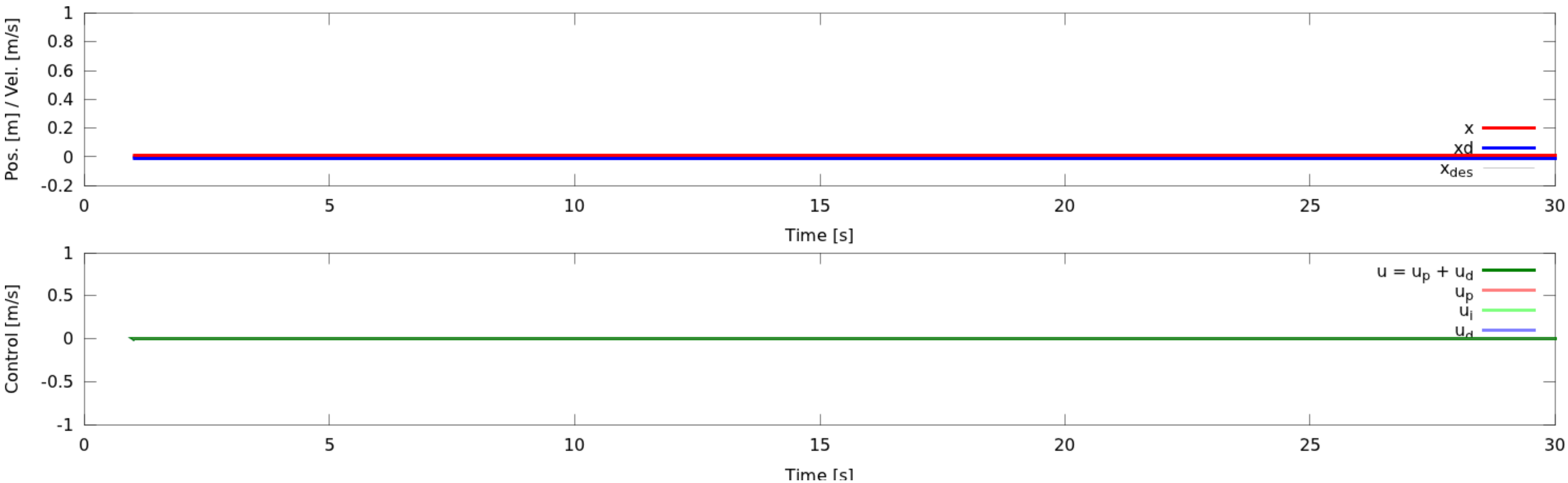


Rigid Body Kinematics

- Consider a rigid body
- Free floating in 1D space, no gravity
- How does this system evolve over time?

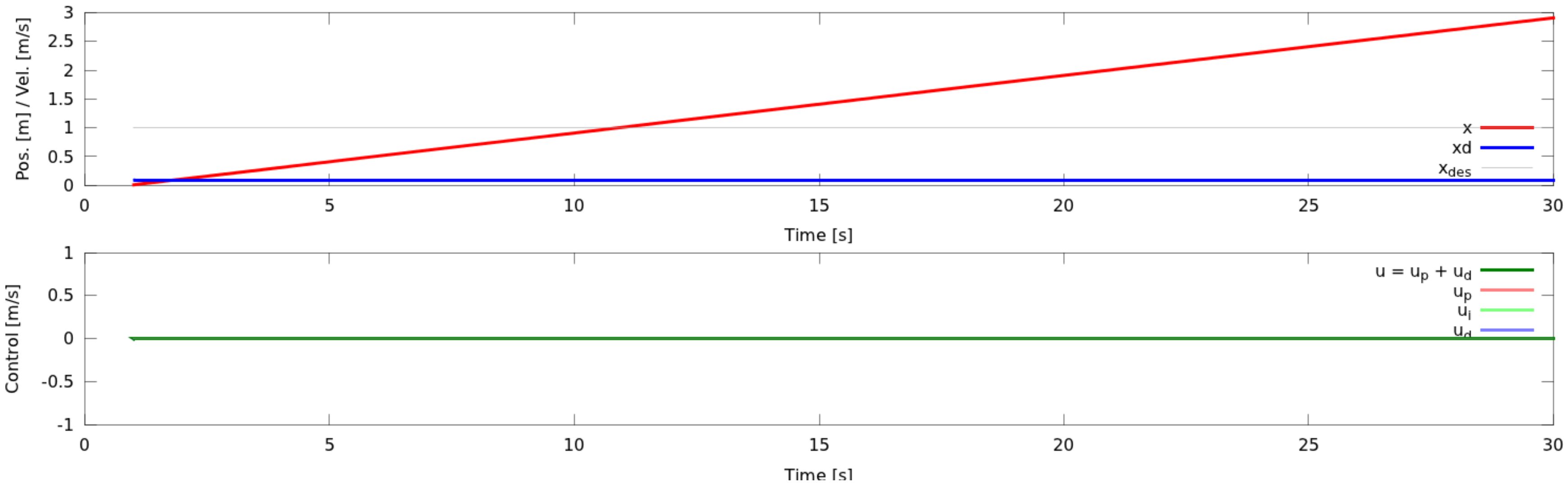
Rigid Body Kinematics

- Consider a rigid body
- Free floating in 1D space, no gravity
- How does this system evolve over time?
- Example: $x_0 = 0, \dot{x}_0 = 0$



Rigid Body Kinematics

- Consider a rigid body
- Free floating in 1D space, no gravity
- How does this system evolve over time?
- Example: $x_0 = 0, \dot{x}_0 = 0.1$



Rigid Body Kinematics

- Consider a rigid body
- Free floating in 1D space, no gravity
- In each time instant, we can apply a force F
- Results in acceleration $\ddot{x} = F/m$
- Desired position $x_{\text{des}} = 1$

P Control

- What happens for this control law?

$$u_t = K(x_{\text{des}} - x_{t-1})$$

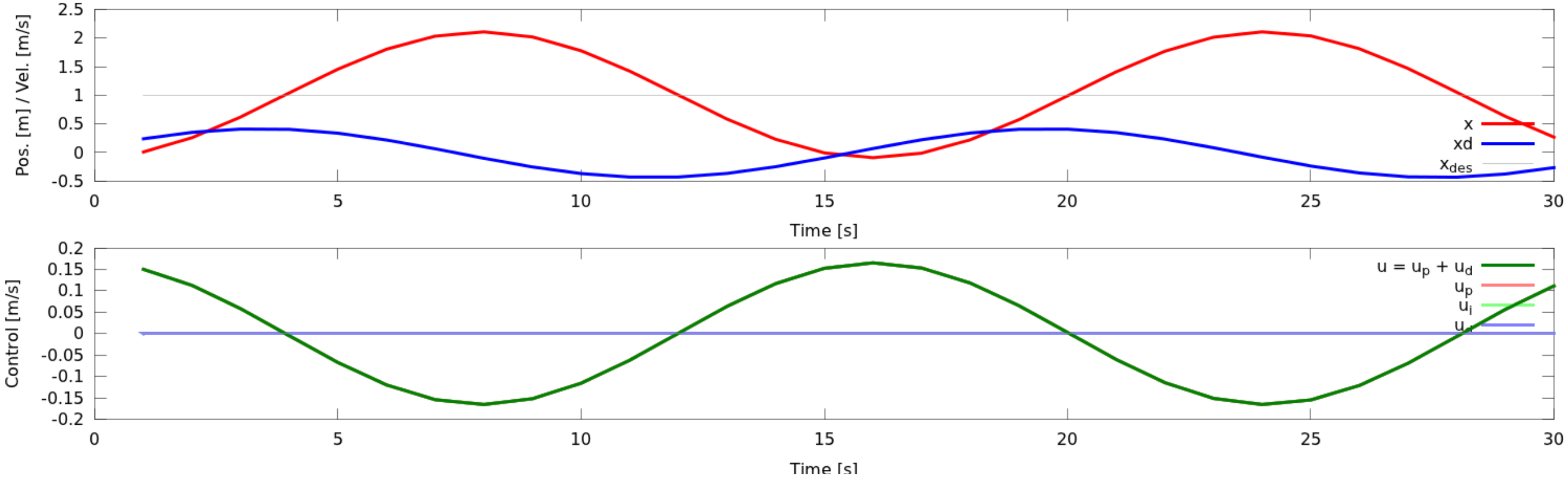
- This is called proportional control

P Control

- What happens for this control law?

$$u_t = K(x_{\text{des}} - x_{t-1})$$

- This is called proportional control

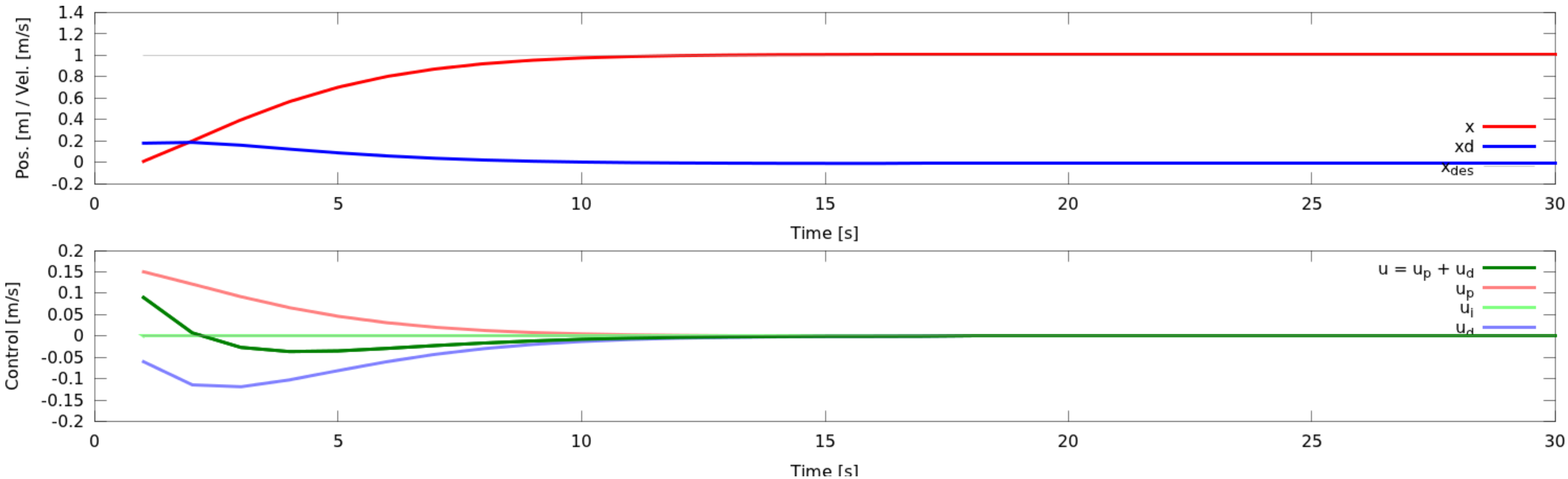


PD Control

- What happens for this control law?

$$u_t = K_P(x_{\text{des}} - x_{t-1}) + K_D(\dot{x}_{\text{des}} - \dot{x}_{t-1})$$

- Proportional-Derivative control

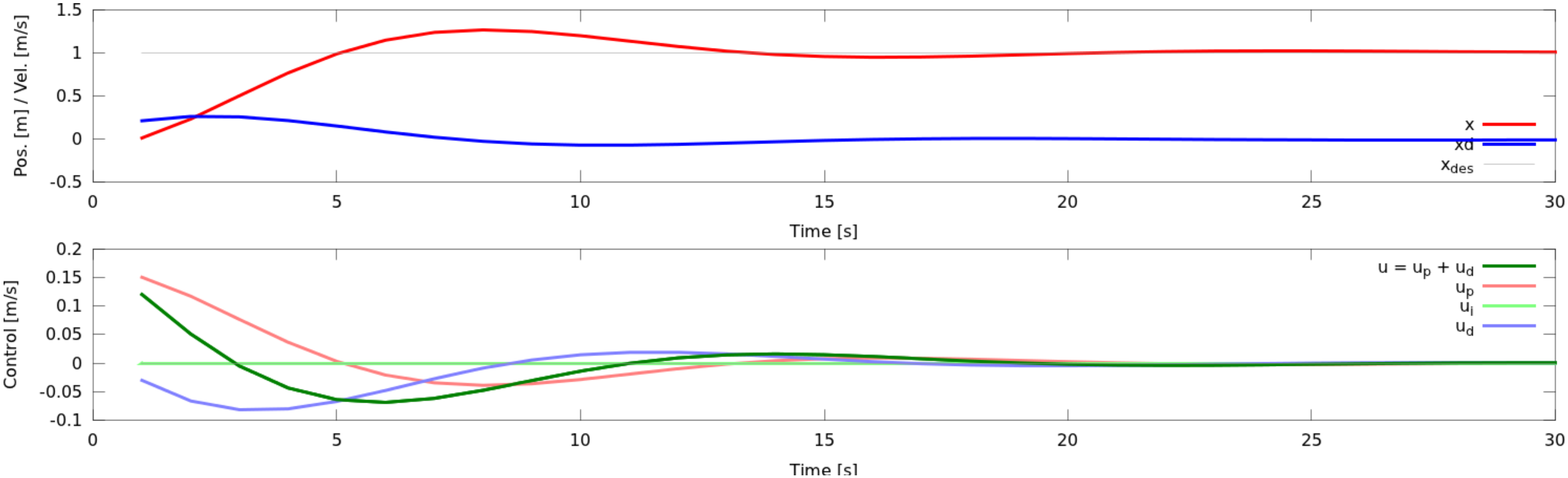


PD Control

- What happens for this control law?

$$u_t = K_P(x_{\text{des}} - x_{t-1}) + K_D(\dot{x}_{\text{des}} - \dot{x}_{t-1})$$

- What if we set **higher** gains?

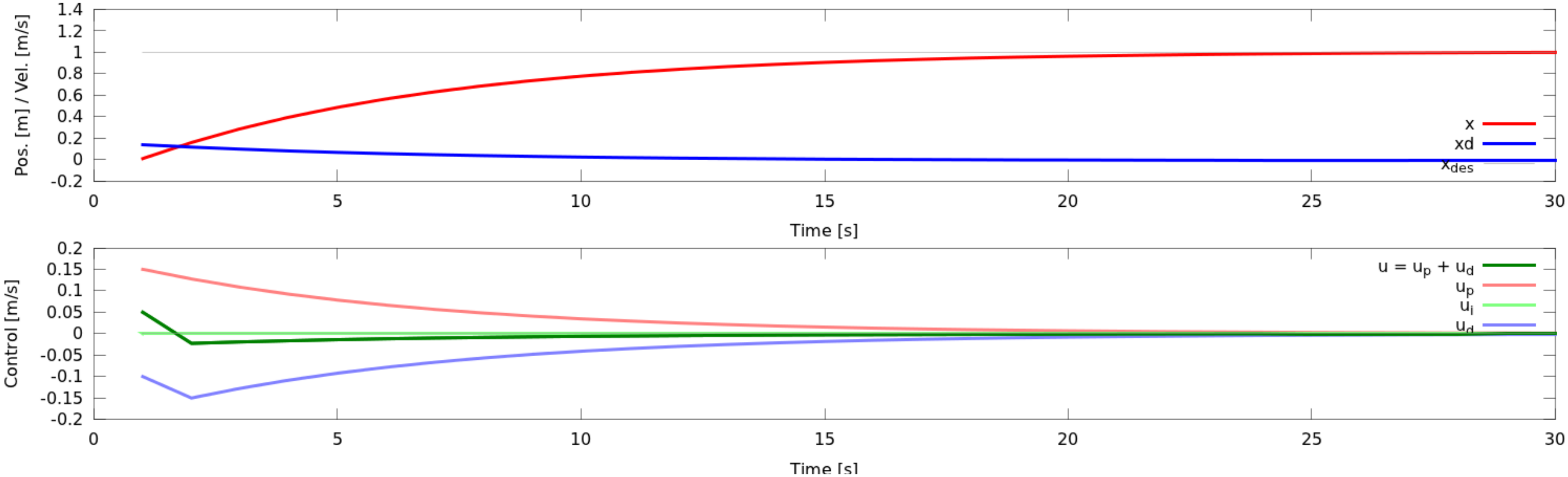


PD Control

- What happens for this control law?

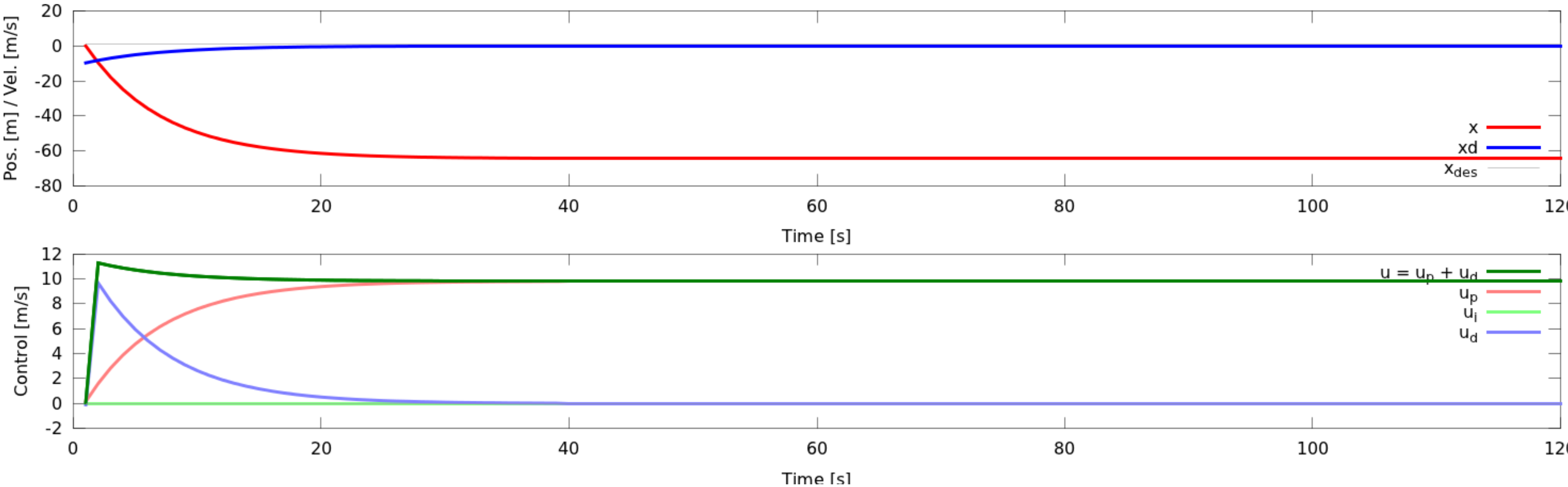
$$u_t = K_P(x_{\text{des}} - x_{t-1}) + K_D(\dot{x}_{\text{des}} - \dot{x}_{t-1})$$

- What if we set **lower** gains?



PD Control

- What happens when we add gravity?

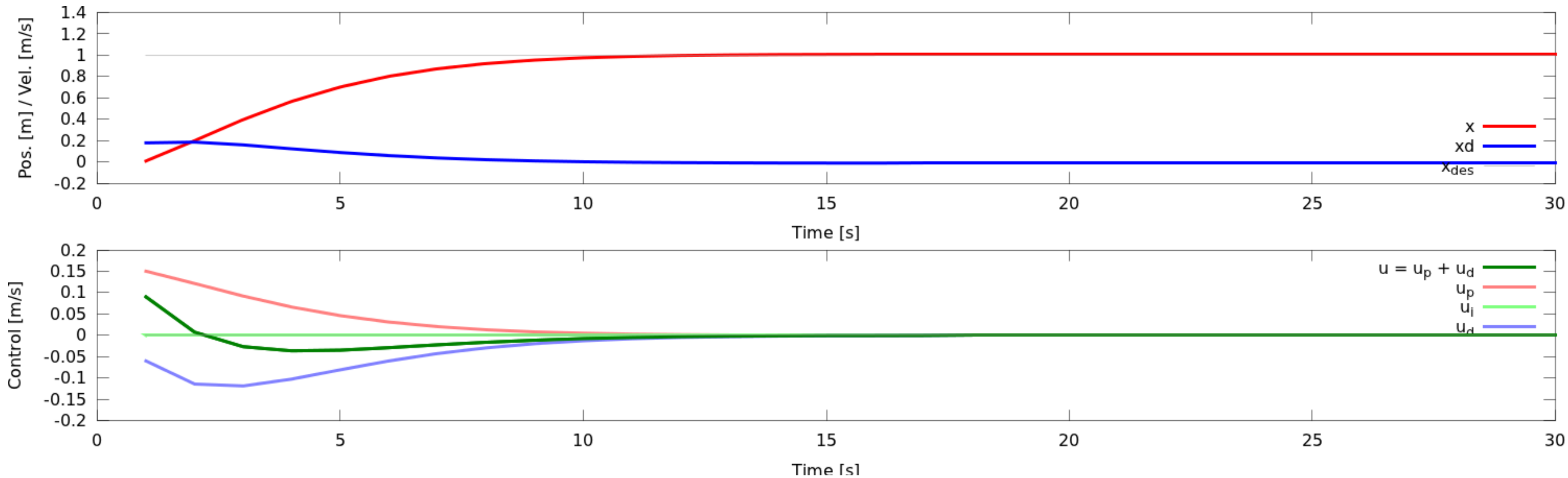


Gravity compensation

- Add as an additional term in the control law

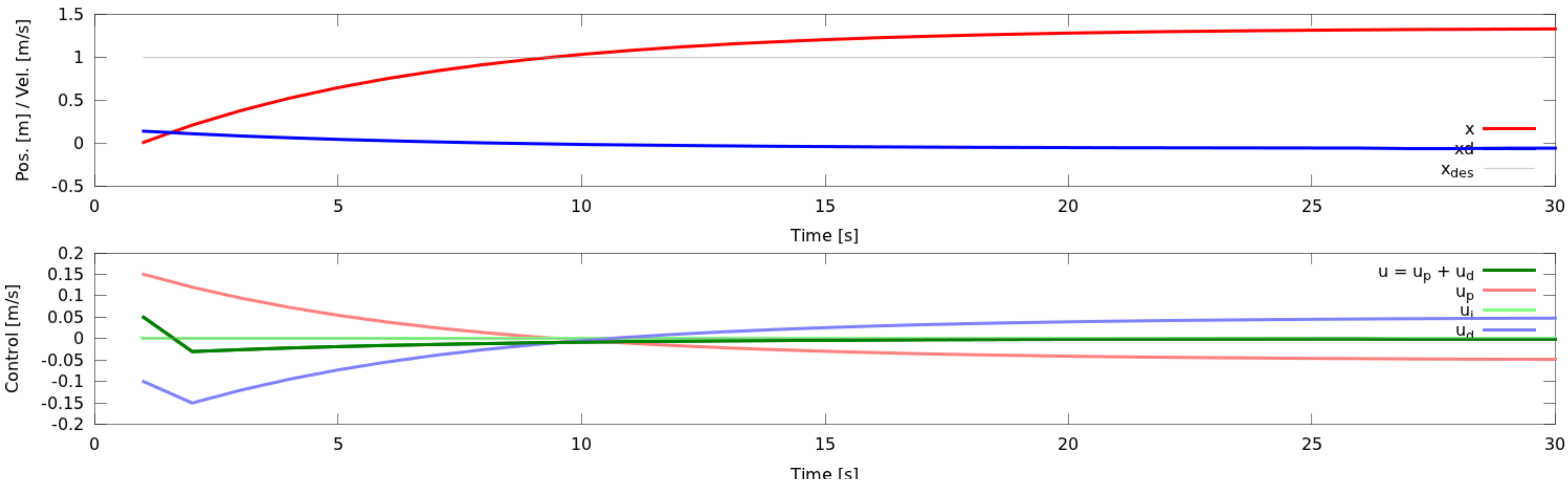
$$u_t = K_P(x_{\text{des}} - x_{t-1}) + K_D(\dot{x}_{\text{des}} - \dot{x}_{t-1}) + F_{\text{grav}}$$

- Any known (inverse) dynamics can be included



PD Control

- What happens when we have systematic errors? (control/sensor noise with non-zero mean)
- Example: unbalanced quadrocopter, wind, …
- Does the robot ever reach its desired location?

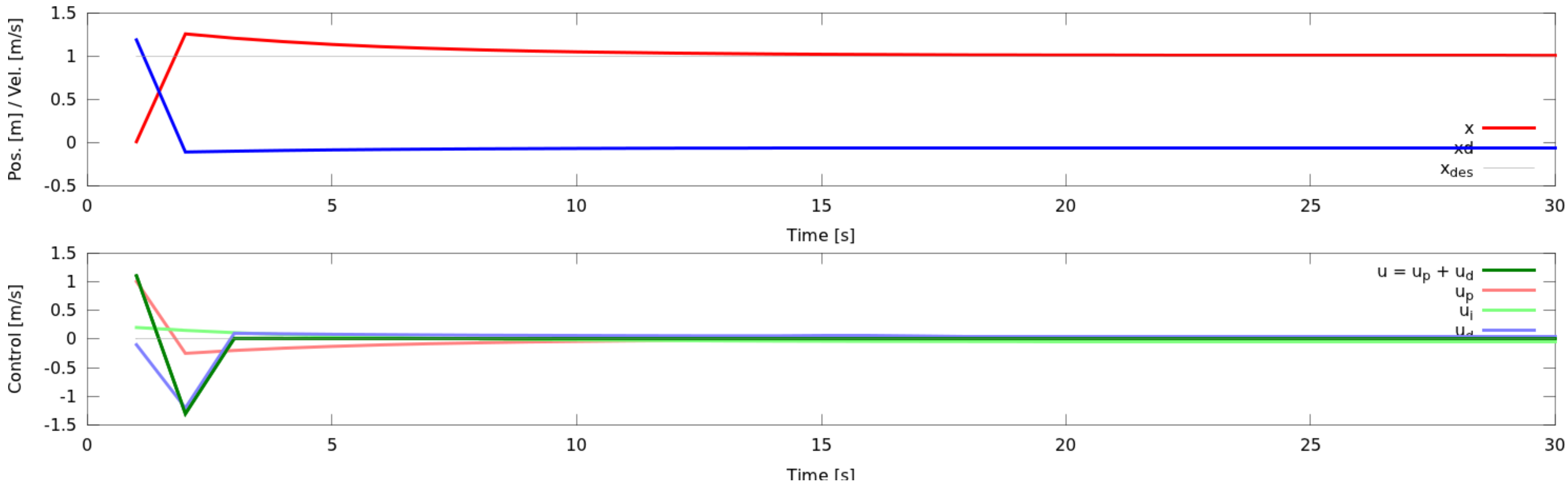


PID Control

- Idea: Estimate the system error (bias) by integrating the error

$$u_t = K_P(x_{\text{des}} - x_t) + K_D(\dot{x}_{\text{des}} - \dot{x}_t) + K_I \int_{-\infty}^{t} x_{des} - x_t \mathrm{d}t$$

- Proportional+Derivative+Integral Control



PID Control

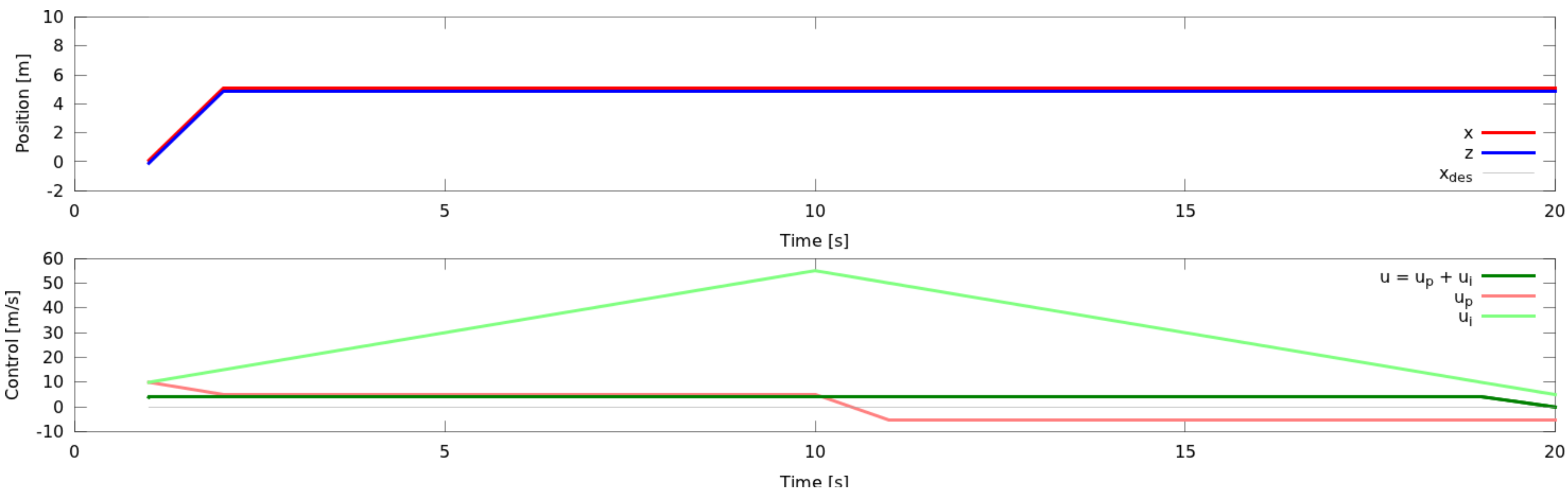
- Idea: Estimate the system error (bias) by integrating the error

$$u_t = K_P(x_{\text{des}} - x_t) + K_D(\dot{x}_{\text{des}} - \dot{x}_t) + K_I \int_{-\infty}^{t} x_{des} - x_t \mathrm{d}t$$

- Proportional+Derivative+Integral Control
- For steady state systems, this can be reasonable
- Otherwise, it may create havoc or even disaster (wind-up effect)

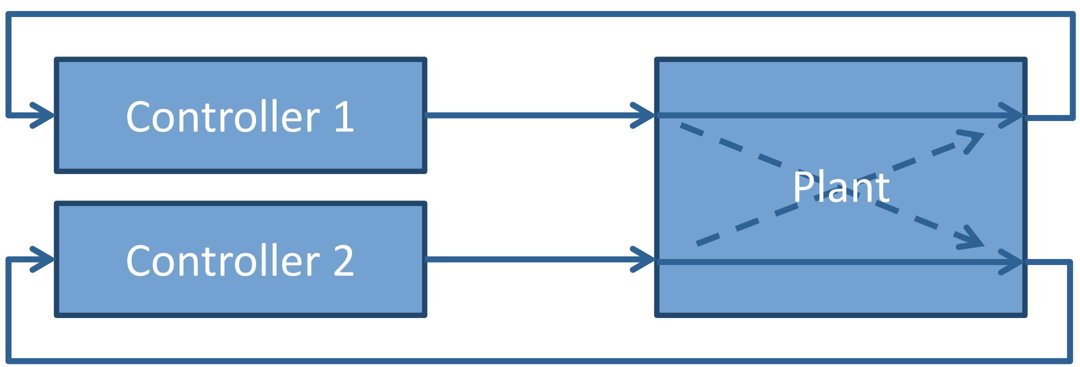
Example: Wind-up effect

- Quadrocopter gets stuck in a tree → does not reach steady state
- What is the effect on the I-term?



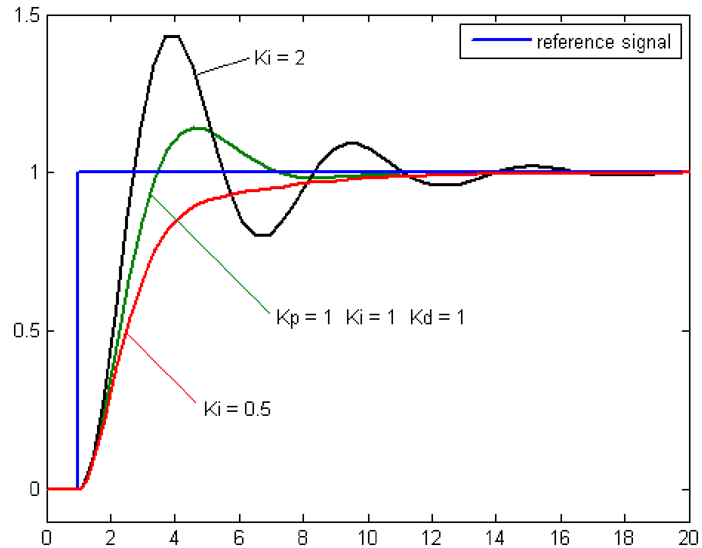
De-coupled Control

- So far, we considered only single-input, single-output systems (SISO)
- Real systems have multiple inputs + outputs
- MIMO (multiple-input, multiple-output)
- In practice, control is often de-coupled



How to Choose the Coefficients?

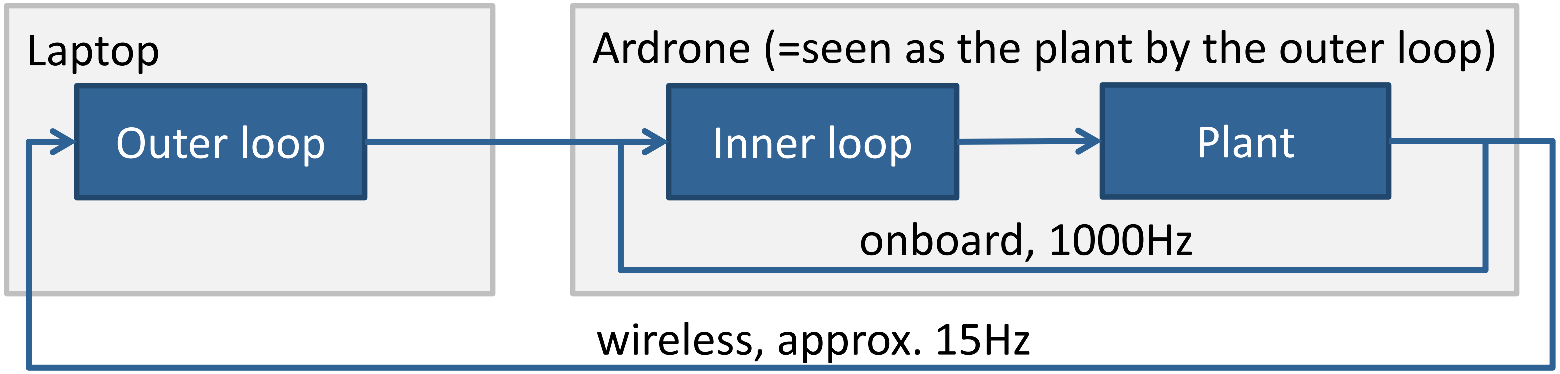
- Gains too large: overshooting, oscillations
- Gains too small: long time to converge
- Heuristic methods exist
- In practice, often tuned manually



Example: Ardrone

Cascaded control

- Inner loop runs on embedded PC and stabilizes flight
- Outer loop runs externally and implements position control



Ardrone: Inner Control Loop

- Plant input: motor torques

$$\mathbf{u}_{\text{inner}} = \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \end{pmatrix}^\top$$

- Plant output: roll, pitch, yaw rate, z velocity

$$\mathbf{x}_{\text{inner}} = \begin{pmatrix} \omega_x & \omega_y & \omega_z & z \end{pmatrix}^\top$$

attitude
(measured using gyro +
accelerometer)

altitude
(measured using ultrasonic
distance sensor + attitude)

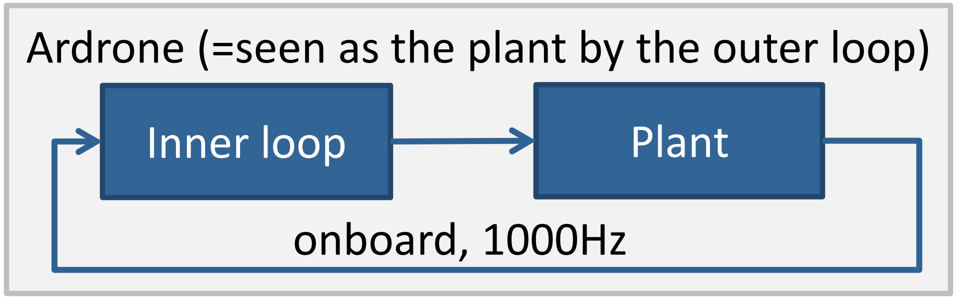
Ardrone: Inner Control Loop

- Plant input: motor torques

$$\mathbf{u}_{\text{inner}} = \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \end{pmatrix}^\top$$

- Plant output: roll, pitch, yaw rate, z velocity

$$\mathbf{x}_{\text{inner}} = \begin{pmatrix} \omega_x & \omega_y & \omega_z & z \end{pmatrix}^\top$$

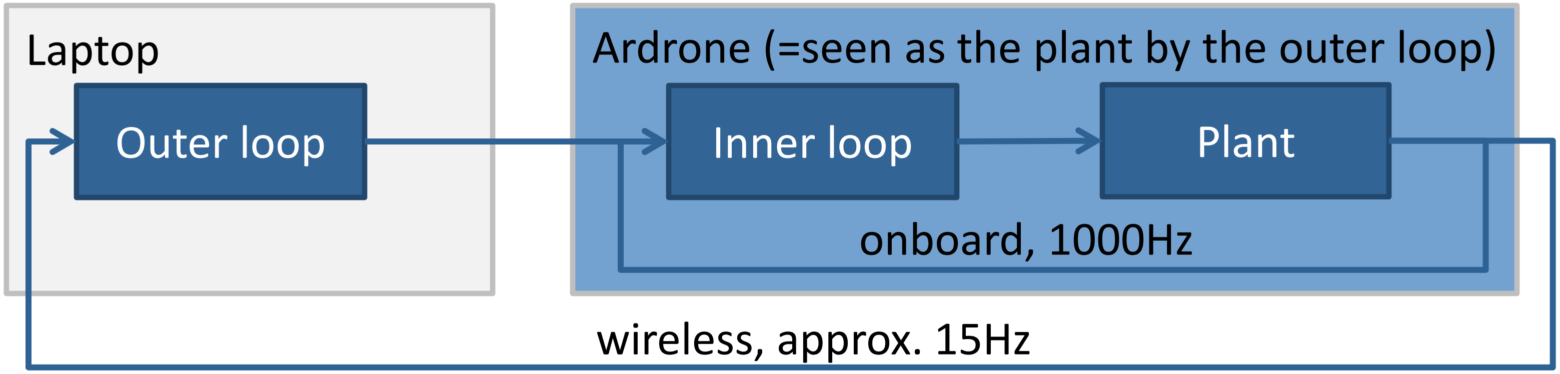


Ardrone: Outer Control Loop

- Outer loop sees inner loop as a plant (black box)
- Plant input: roll, pitch, yaw rate, z velocity

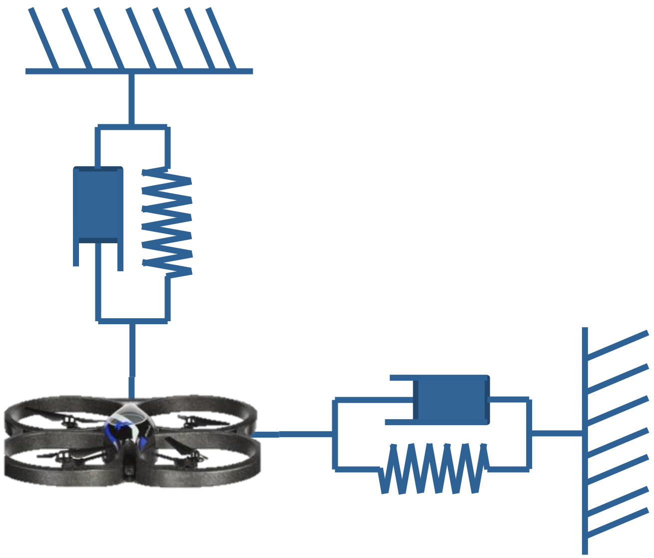
$$\mathbf{u}_{\text{outer}} = \begin{pmatrix} \omega_x & \omega_y & \dot{\omega}_z & \dot{z} \end{pmatrix}^\top$$

- Plant output: $\mathbf{x}_{\text{outer}} = \begin{pmatrix} x & y & z & \psi \end{pmatrix}^\top$



Mechanical Equivalent

- PD Control is equivalent to adding spring-dampers between the desired values and the current position

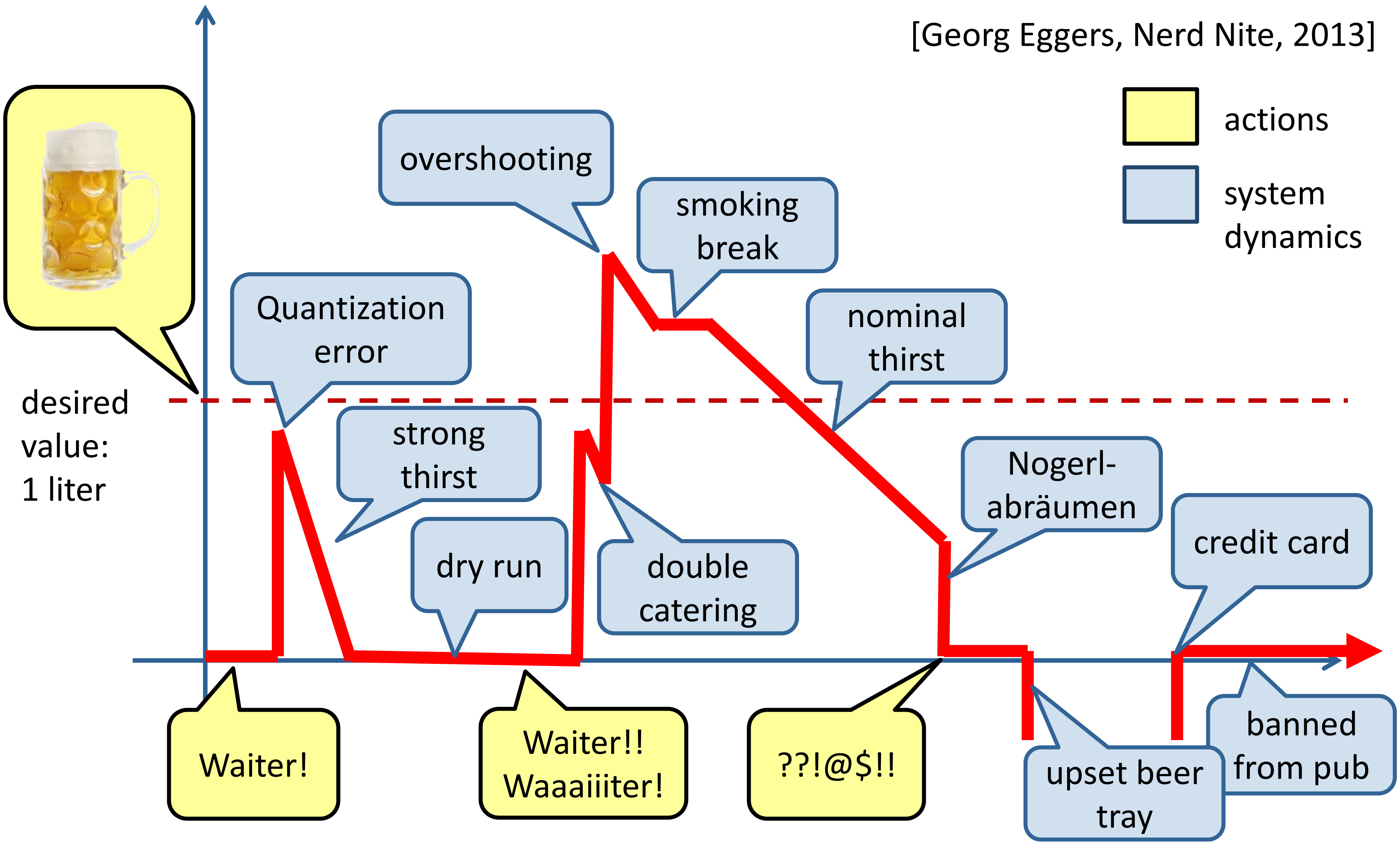


PID Control – Summary

PID is the most used control technique in practice

- P control → simple proportional control, often enough
- PI control → can compensate for bias (e.g., wind)
- PD control → can be used to reduce overshoot (e.g., when acceleration is controlled)
- PID control → all of the above

PID Control – Beergarden Example



Advanced Control Techniques

What other control techniques do exist?

- Adaptive control
- Robust control
- Optimal control
- Linear-quadratic regulator (LQR)
- Reinforcement learning
- Inverse reinforcement learning
- ... and many more

Optimal Control

- Find the controller that provides the best performance
- Need to define a measure of performance
- What would be a good performance measure?
  - Minimize the error?
  - Minimize the controls?
  - Combination of both?

Linear Quadratic Regulator

Given:

- Discrete-time **linear** system

$$x_{k+1} = Ax_k + Bu_k$$

- **Quadratic** cost function

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k \right)$$

Goal: Find the controller with the lowest cost → LQR control

Linear Quadratic Regulator

- Advantage:
  Cost matrix has an intuitive interpretation

- Disadvantage:
  Typically no closed form solution

- Often solved numerically
- Only for small planning horizon

Reinforcement Learning

- Note that in principle, any cost function can be used
- Sometimes, it is easier to specify a reward function $r(x_t, u_t)$
- Example:

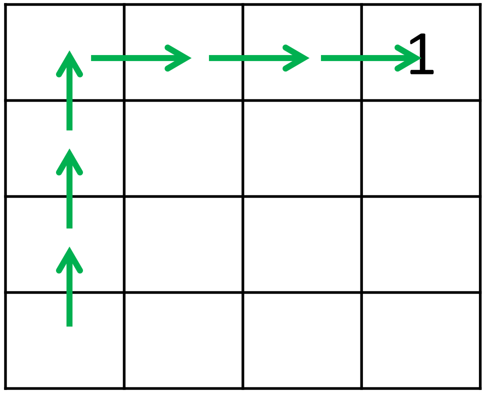
$$r(x_t, u_t) = \begin{cases} 1 & \text{if } x_{\text{des}} = x_t \\ 0 & otherwise \end{cases}$$

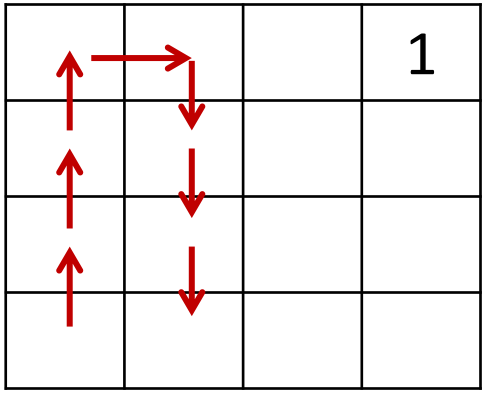
			1

Reinforcement Learning

- Reward function $r(x_t, u_t)$
- Episode (=trajectory) $\tau = (x_1, u_1, \ldots, x_n, u_n)$

- Reward of an episode $R(\tau) = \sum_t r(x_t, u_t)$



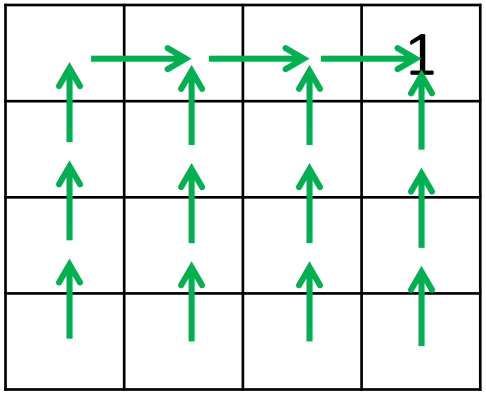


Reinforcement Learning

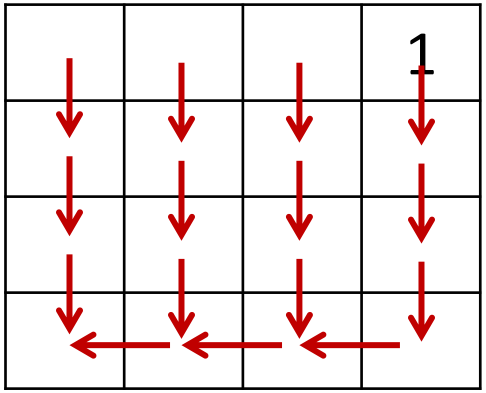
- A policy (=controller) defines which action to take in a particular state

$$\pi(x) = u$$

π_1



π_2



Policy Evaluation

- The optimal policy maximizes the expected future reward
- How can we estimate the expected future reward?
- How can we find the optimal policy?
- Closer look at two methods
  - Q learning
  - Policy gradient methods

Reinforcement Learning

Q learning

- Learn the value function for each state-action pair
- Needs compact representation of value function (e.g., neural networks)
- Examples: TD-Gammon (mid-90’s), Inverted pendulum, …

Reinforcement Learning

Policy gradient methods

- Policy is parameterized
- Analytic gradient typically not available
- Simulation-based optimization

[Lupashin et al., ICRA 2010]

Reinforcement Learning

Policy gradient methods

- Policy is parameterized
- Analytic gradient typically not available
- Simulation-based optimization

[Schoellig et al., ACC 2012]

Inverse Reinforcement Learning

- Parameterized reward function
- Learn these parameters from expert demonstrations and refine
- Example: [Abbeel and Ng, ICML 2010]

Lessons Learned Today

- Brushless Motors
- Motor Controllers
- Cascaded Control
- PID Control
- Advanced Control Techniques