香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系 DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

Introduction to Aerial Robotics
Lecture 3

Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

21 February 2023

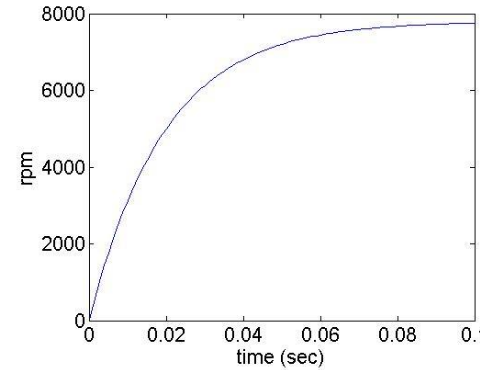
Outline

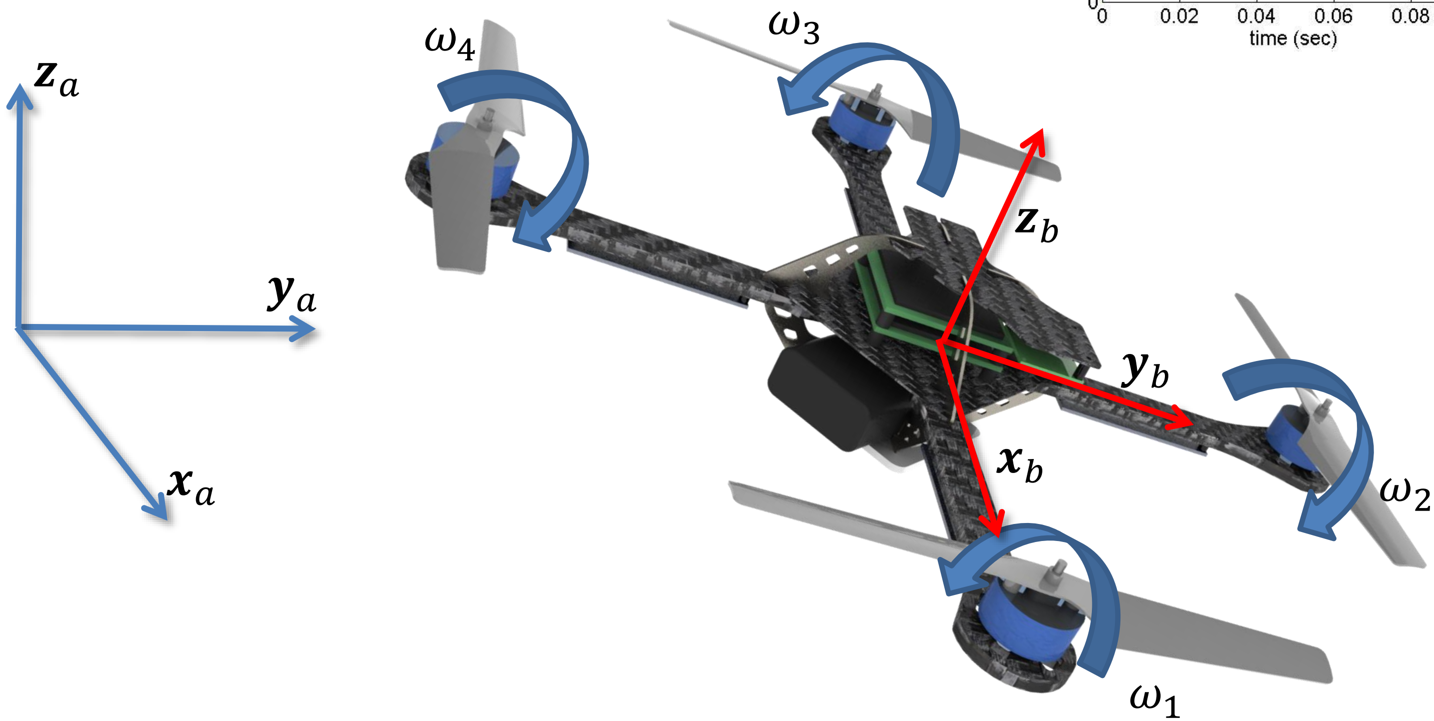
- Quadrotor Dynamics
- Control Basics
- Quadrotor Control
- Trajectory Generation

Quadrotor Dynamics

Quadrotor Dynamics

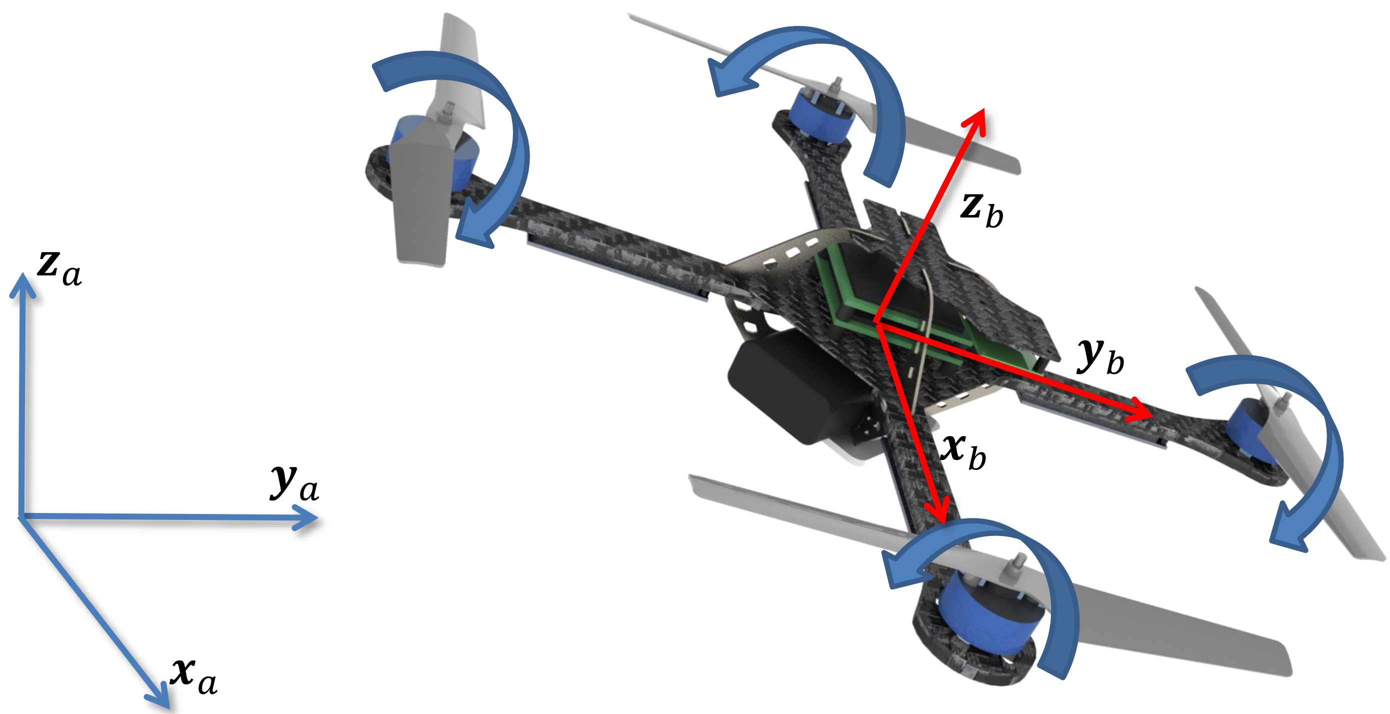
- Motor model: $\dot{\omega}_i = k_m(\omega_i^{des} - \omega_i)$
- Thrust from individual motor: $F_i = k_F \omega_i^2$
- Moment from individual motor: $M_i = k_M \omega_i^2$





Quadrotor Dynamics

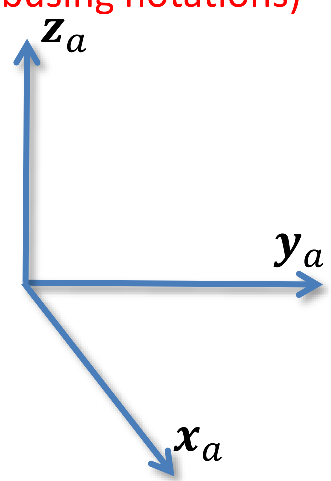
- Z-X-Y Euler Angles: $\boldsymbol{R}_{ab} = \boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_x(\phi) \cdot \boldsymbol{R}_y(\theta)$
- Sequence of three rotations about body-fixed axes
- What are the singularities?

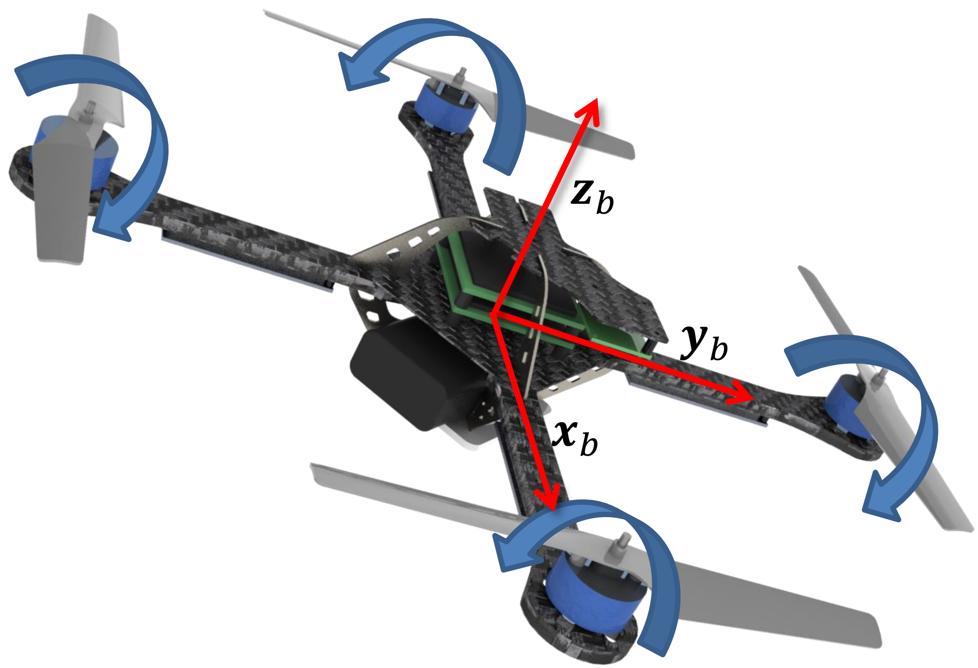


Quadrotor Dynamics

- $\boldsymbol{R}_{ab} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$
- $\boxed{\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$

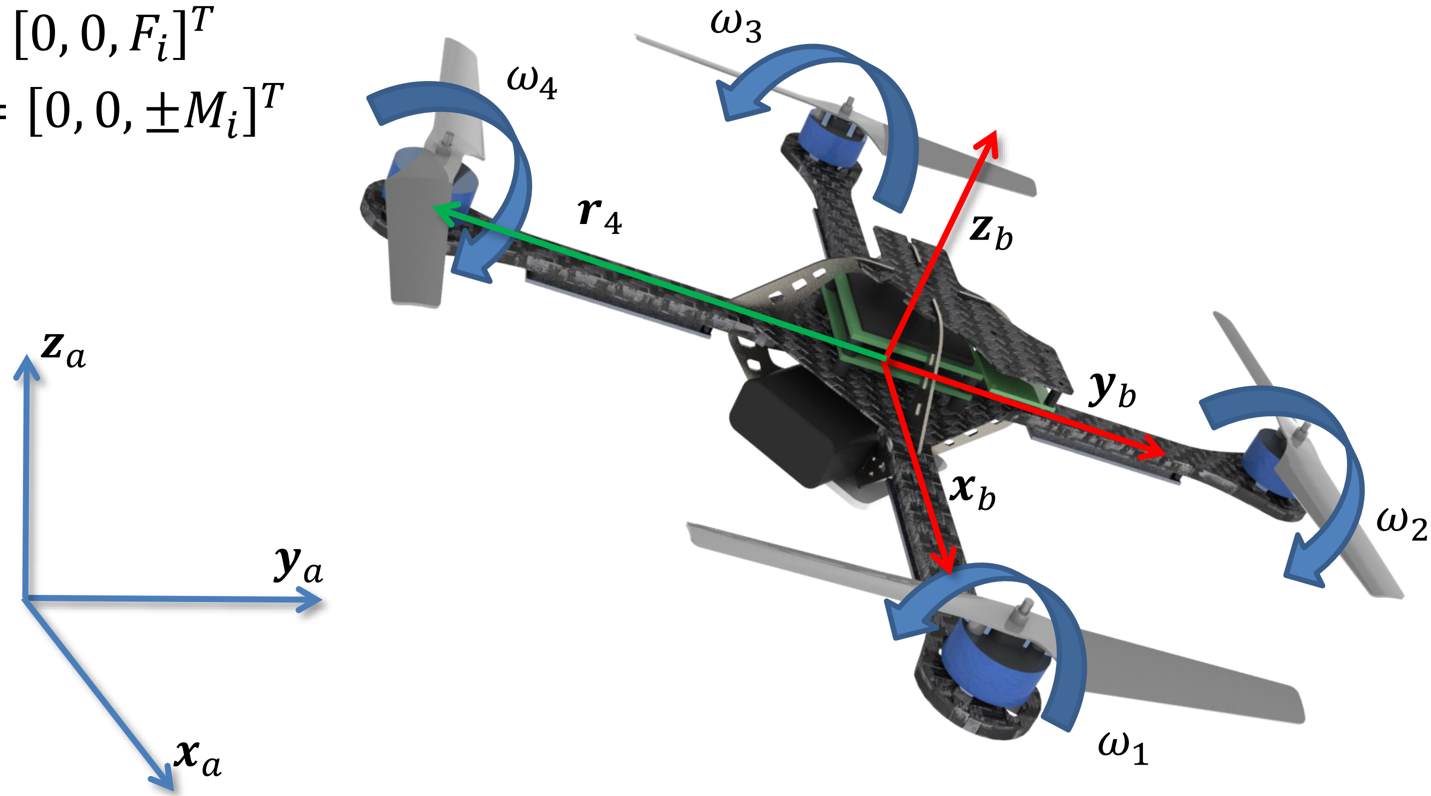
Instantaneous body angular velocity viewed in the body frame (sorry for abusing notations)





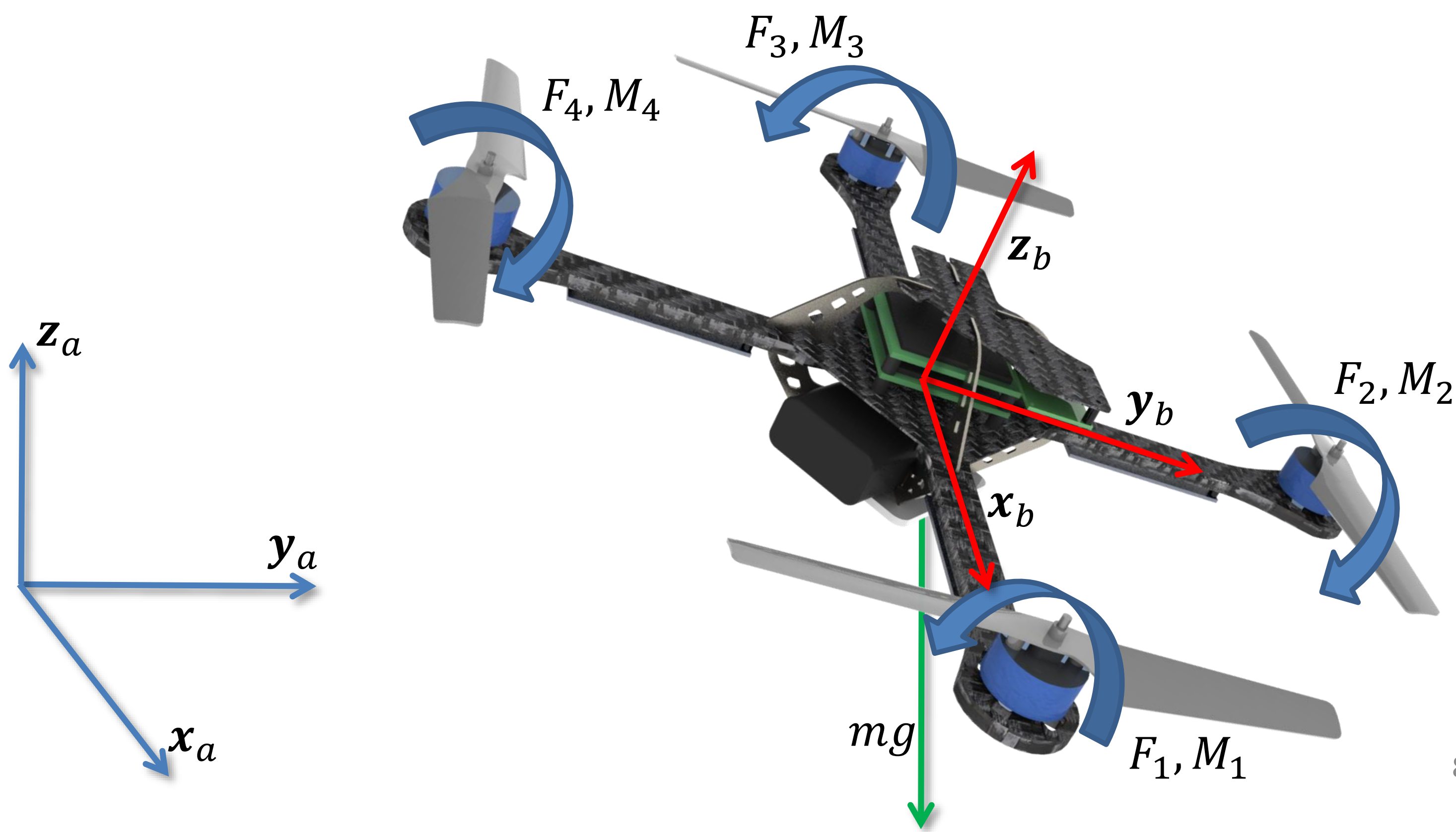
Quadrotor Dynamics

- Consider body frame
  - $\boldsymbol{F} = \boldsymbol{F}_1 + \boldsymbol{F}_2 + \boldsymbol{F}_3 + \boldsymbol{F}_4 - \boldsymbol{R}_{ab}^T[0, 0, mg]^T$
  - $\boldsymbol{M} = \boldsymbol{r}_1 \times \boldsymbol{F}_1 + \boldsymbol{r}_2 \times \boldsymbol{F}_2 + \boldsymbol{r}_3 \times \boldsymbol{F}_3 + \boldsymbol{r}_4 \times \boldsymbol{F}_4 + \boldsymbol{M}_1 + \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_4$
  - $\boldsymbol{F}_i = [0, 0, F_i]^T$
  - $\boldsymbol{M}_i = [0, 0, \pm M_i]^T$

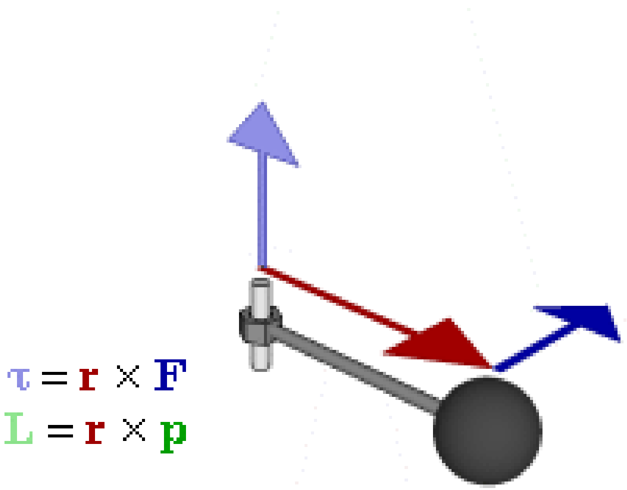


Newton-Euler Equations

- Newton Equation: $m\ddot{\boldsymbol{p}}^a = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}_{ab} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$



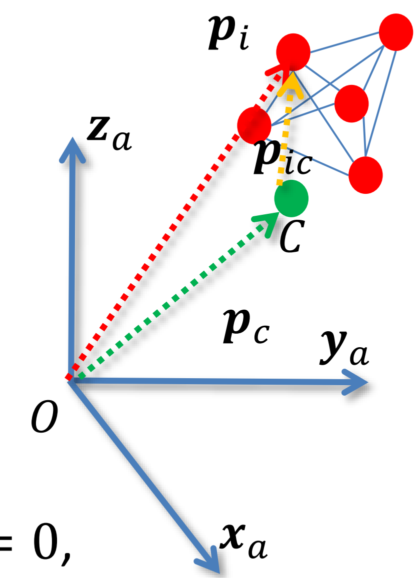
Newton-Euler Equations



Relationship between force (F), torque/moment of force(τ), momentum (p), and angular momentum (L) vectors in a rotating system. r is the position vector.

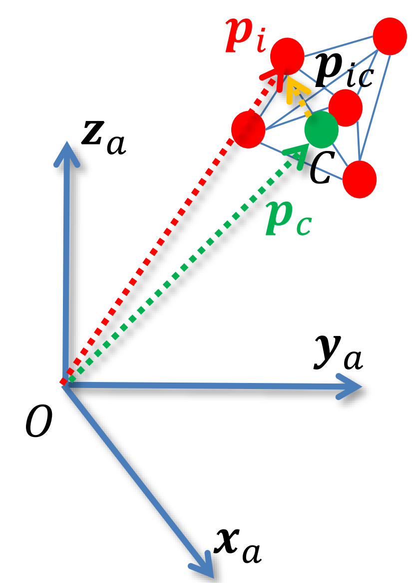
Newton-Euler Equations

- The rigid body as a collection of particles
  - Center of mass (CoM): $\boldsymbol{p}_c$
  - Position of the i-th particle to CoM: $\boldsymbol{p}_{ic} = \boldsymbol{p}_i - \boldsymbol{p}_c$
  - Velocity of the i-th particle to CoM: $\boldsymbol{v}_{ic} = \dot{\boldsymbol{p}}_i - \dot{\boldsymbol{p}}_c$
    $= \boldsymbol{v}_i - \boldsymbol{v}_c$
  - Angular momentum of the i-th particle:
    $$\boldsymbol{H}_i = \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$$
- Angular momentum of the rigid body:
  - $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$
  - Since: $\sum m_i \boldsymbol{p}_{ic} = \sum m_i(\boldsymbol{p}_i - \boldsymbol{p}_c) = \sum m_i \boldsymbol{p}_i - \boldsymbol{p}_c \sum m_i = 0$,
  - We have: $\sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_c = (\sum m_i \boldsymbol{p}_{ic}) \times \boldsymbol{v}_c = 0$
  - Therefore: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i - \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_c = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_{ic}$
  - Since: $\boldsymbol{v}_{ic} = \boldsymbol{\omega} \times \boldsymbol{p}_{ic}$,
  - We have: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times (\boldsymbol{\omega} \times m_i \boldsymbol{p}_{ic}) = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$



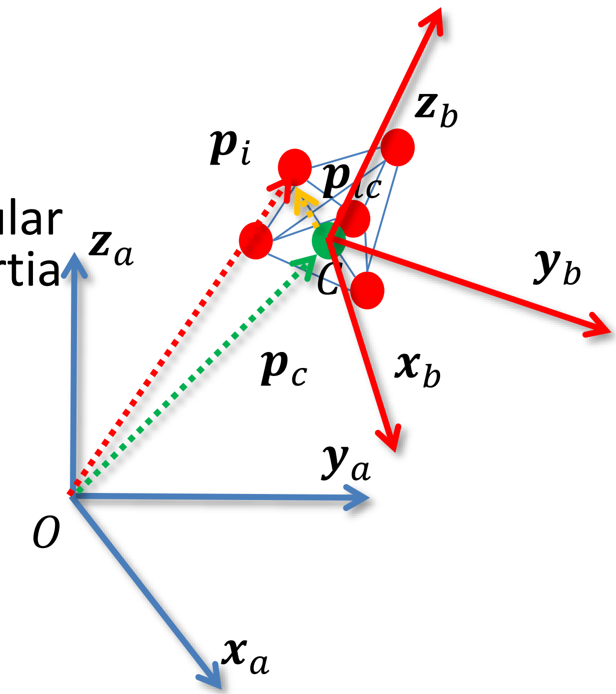
Newton-Euler Equations

- Rotational dynamics
  - Angular momentum: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$
  - Take the derivative: $\dot{\boldsymbol{H}} = \sum \dot{\boldsymbol{p}}_{ic} \times m_i \boldsymbol{v}_i + \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i$
  - Since $\sum \dot{\boldsymbol{p}}_{ic} \times m_i \boldsymbol{v}_i = \sum \boldsymbol{v}_i \times m_i \boldsymbol{v}_i - \boldsymbol{v}_c \times m_i \boldsymbol{v}_i = \sum -\boldsymbol{v}_c \times m_i \boldsymbol{v}_i = -\boldsymbol{v}_c \times \frac{d}{dt} \sum m_i \boldsymbol{p}_i = -\boldsymbol{v}_c \times \frac{d}{dt} \boldsymbol{p}_c \sum m_i = 0$
  - We have $\dot{\boldsymbol{H}} = \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i$
  - Referring to Newton's second law: $\boldsymbol{F}_i + \sum_{i \neq j} \boldsymbol{F}_{ij} = m_i \dot{\boldsymbol{v}}_i$
  - $\dot{\boldsymbol{H}} = \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i = \sum \boldsymbol{p}_{ic} \times (\boldsymbol{F}_i + \sum_{i \neq j} \boldsymbol{F}_{ij}) = \sum \boldsymbol{p}_{ic} \times \boldsymbol{F}_i$
  - We also know that the external moment: $\boldsymbol{M} = \sum \boldsymbol{p}_{ic} \times \boldsymbol{F}_i$
  - We have the rotational dynamics: $\boldsymbol{M} = \dot{\boldsymbol{H}}$



Newton-Euler Equations

- Finishing the work on rotational dynamics
  - Given: $\boldsymbol{H} = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$
  - And using the fact: $(\boldsymbol{R}\boldsymbol{a}) \times (\boldsymbol{R}\boldsymbol{b}) = \boldsymbol{R}(\boldsymbol{a} \times \boldsymbol{b})$
    - $\boldsymbol{R}$: rotation matrix
    - $\boldsymbol{a}, \boldsymbol{b}$: vectors
  - We can transform the representation of the angular momentum to the body frame with constant inertia matrix:



$$\boldsymbol{H} = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$$

$$= -\sum \boldsymbol{R}_{ab}\boldsymbol{p}_{ic}^b \times \left(m_i \boldsymbol{R}_{ab}\boldsymbol{p}_{ic}^b \times \boldsymbol{R}_{ab}\boldsymbol{\omega}^b\right)$$

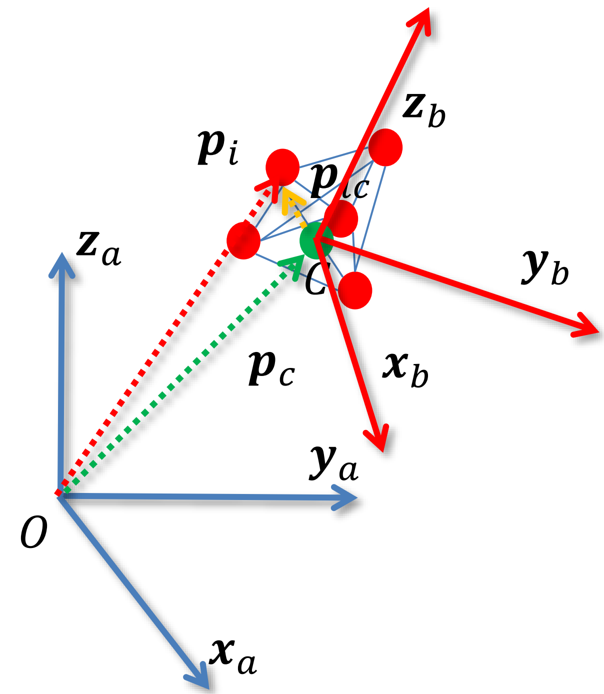
$$= -\boldsymbol{R}_{ab} \sum \boldsymbol{p}_{ic}^b \times \left(m_i \boldsymbol{p}_{ic}^b \times \boldsymbol{\omega}^b\right)$$

$$= -\boldsymbol{R}_{ab} \sum m_i \cdot \boldsymbol{p}_{ic}^b \times \left(\widehat{\boldsymbol{p}}_{ic}^b \cdot \boldsymbol{\omega}^b\right)$$

$$= \boldsymbol{R}_{ab}\left(-\sum m_i \cdot \widehat{\boldsymbol{p}}_{ic}^b \cdot \widehat{\boldsymbol{p}}_{ic}^b\right) \cdot \boldsymbol{\omega}^b = \boldsymbol{R}_{ab}\left(\boldsymbol{I}^b \boldsymbol{\omega}^b\right)$$

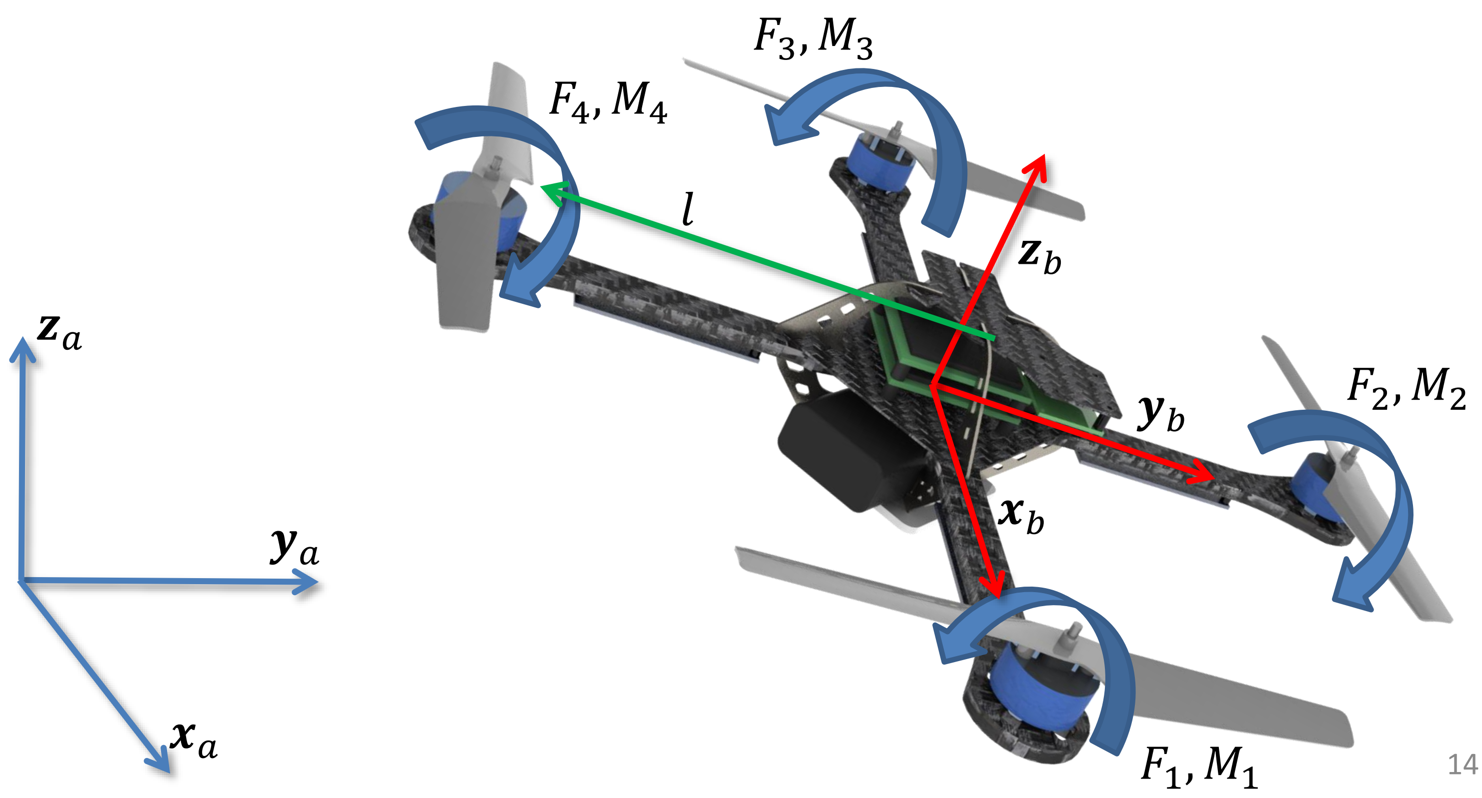
Newton-Euler Equations

- Finishing the work on rotational dynamics
  - Given $\boldsymbol{H} = \boldsymbol{R}_{ab}(\boldsymbol{I}^b\boldsymbol{\omega}^b)$
  - Take the derivative: $\dot{\boldsymbol{H}} = \dot{\boldsymbol{R}}_{ab}\boldsymbol{I}^b\boldsymbol{\omega}^b + \boldsymbol{R}_{ab}\boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b = \boldsymbol{R}_{ab}\widehat{\boldsymbol{\omega}}^b\boldsymbol{I}^b\boldsymbol{\omega}^b + \boldsymbol{R}_{ab}\boldsymbol{I}^b\dot{\omega}^b = \boldsymbol{R}_{ab}(\boldsymbol{\omega}^b \times (\boldsymbol{I}^b\boldsymbol{\omega}^b) + \boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b)$
  - Also transform the moment into body frame: $\boldsymbol{M} = \boldsymbol{R}_{ab}\boldsymbol{M}^b$
  - Finally: $\boldsymbol{M}^b = \boldsymbol{\omega}^b \times (\boldsymbol{I}^b\boldsymbol{\omega}^b) + \boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b$



Newton-Euler Equations

- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$

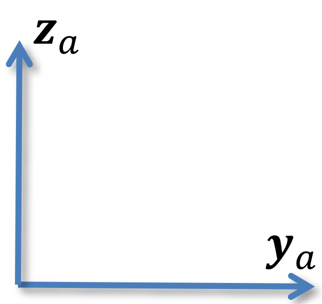


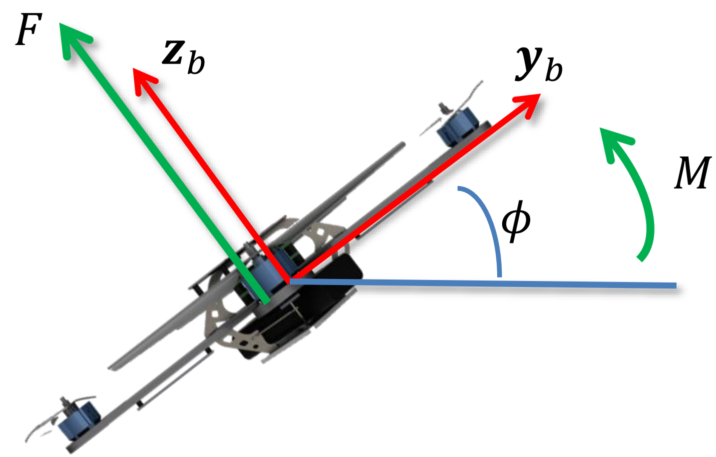
Quadrotor Dynamics

- Motor model: $\dot{\omega}_i = k_m(\omega_i^{des} - \omega_i)$
- Thrust from individual motor: $F_i = k_F\omega_i^2$
- Moment from individual motor: $M_i = k_M\omega_i^2$
- Newton Equation: $m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}\begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$
- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$

A Planar Quadrotor

- $$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} F \\ M \end{bmatrix}$$





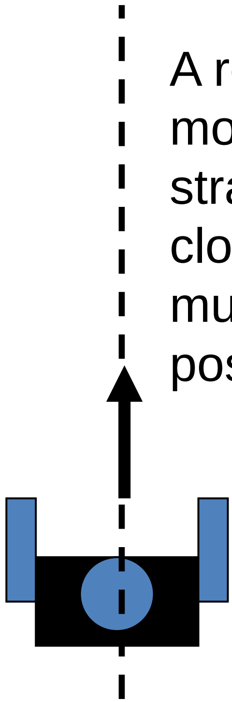
ω_1

Control System Design

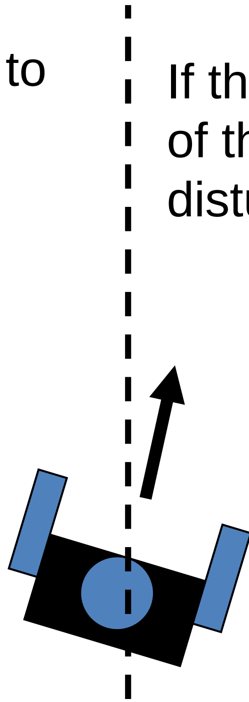
Control System Design

- Introduction of control systems
- Linear Time Invariant (LTI) systems
  - Simple first-order system
  - Simple second-order system
- Controller design
  - Gain tuning
  - Model-based control

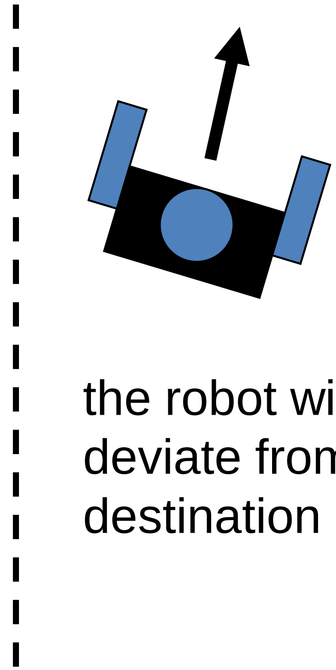
Introduction of control system



A robot needs to move in a straight line or close to it as much as possible



If the movement of the robot gets disturbed,



the robot will deviate from its destination

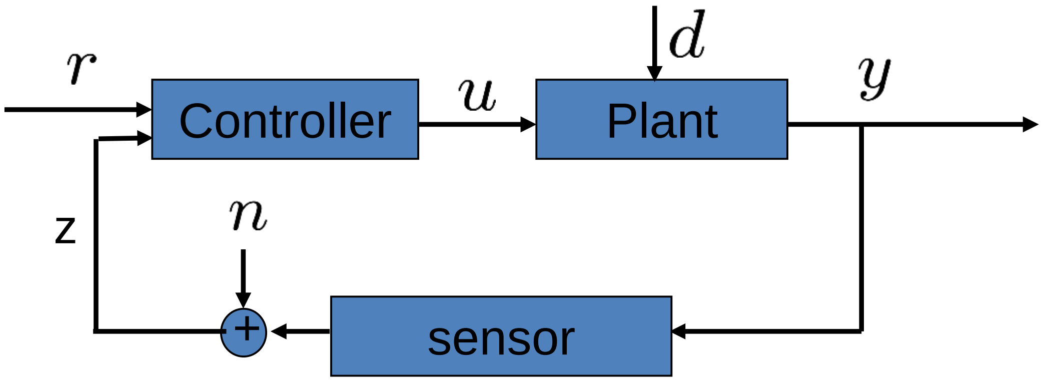
❖ Therefore we need to have a controller to control its movement in real time based on its movement and the destination

Introduction of control system

- Open loop control
  - Move the robot in a pre-determined way
  - Example: walking with your eyes closed
- Closed loop (feedback) control
  - Use the output (i.e. the location of the robot) to adjust the input (i.e. the direction and may be speed) to the movement of the robot
  - We also call it feedback control, since we make the control decision based on the output feedback
  - Example: walking with your eyes open
- We want to stabilize a system with closed loop control

Introduction of control system

- One objective of control is to make the plant stable and track a given reference signal as precise and swift as possible



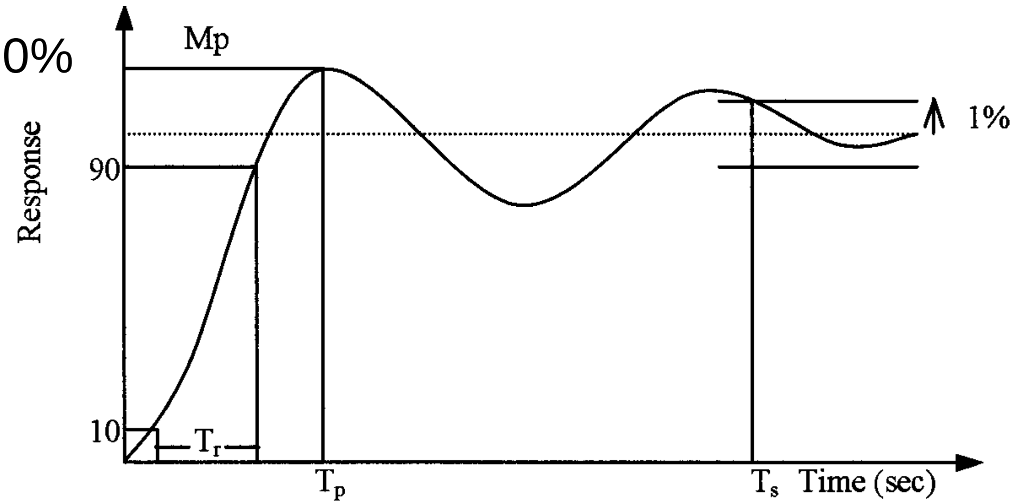
r: reference input
z: state feedback
u: controller output
d: plant disturbance
y: output
n: communication noise

- A controller is simply a computation unit that computes the "optimal" or "desired" input to the plant

"Feedback is a method of controlling a system by inserting into it the result of its past performance"

Introduction of control system

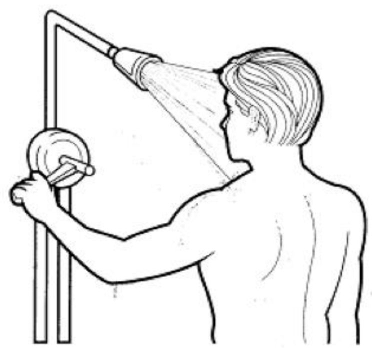
- Rise time:
  - Time it takes from 10% to 90%
- Steady-state error
- Overshoot
  - Percentage by which peak exceeds final value

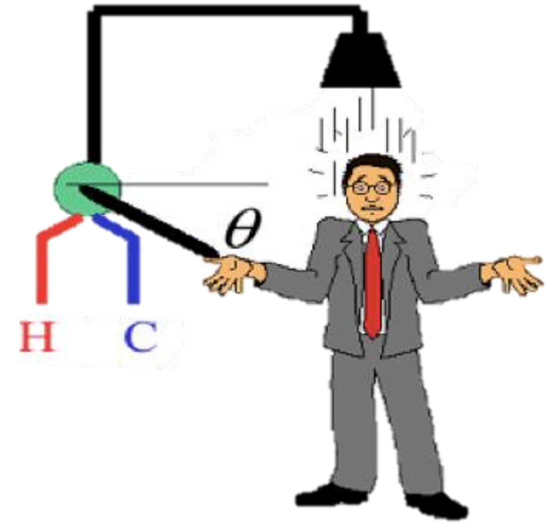


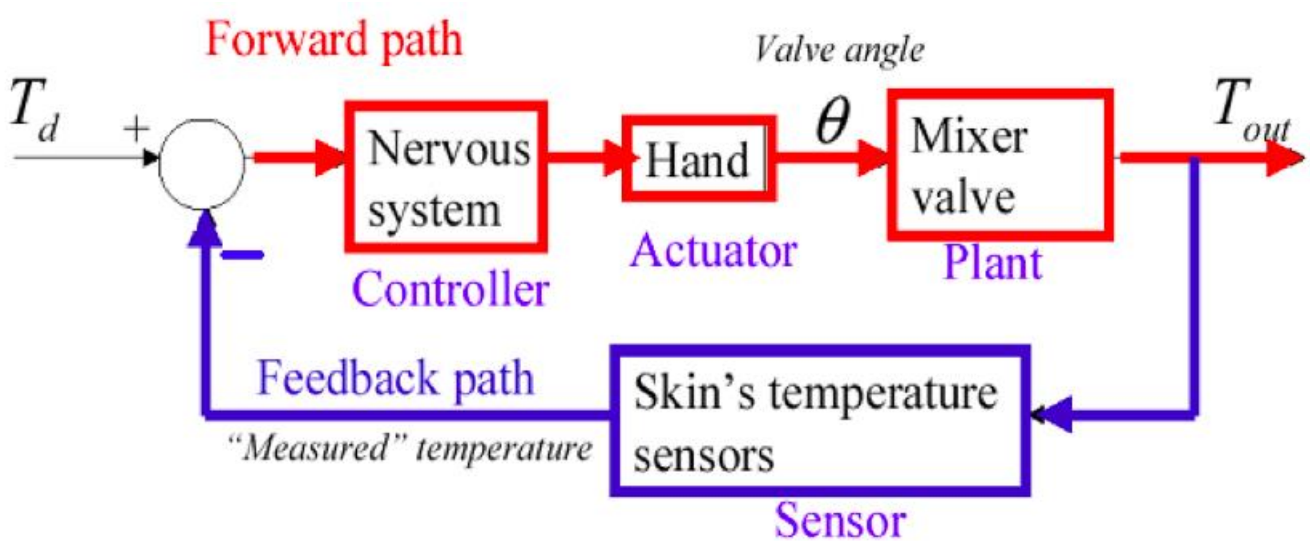
- Settling time
  - Time it takes to reach 1% of final value
- A good control system has small rise time, overshoot, settling time and steady-state error

Introduction of control system

When taking a shower

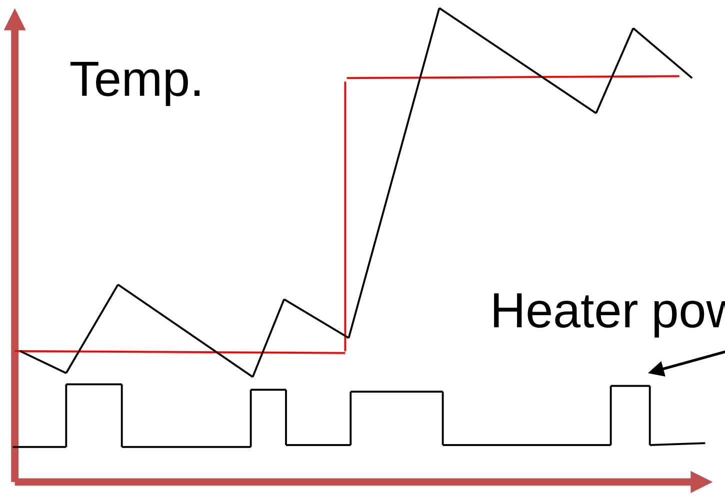






Introduction of control system

- Example: shower water temperature control
  - Turn the heater on if T_{water} is below certain value
  - Turn the heater off if T_{water} is above certain value
- Simple
- Transition is not smooth



Control of a simple first-order system

- Problem

  State, input

  $$x, u \in \boldsymbol{R}$$

  Kinematic plant model

  $$\dot{x} = u$$

  Want x to follow trajectory $x^{des}(t)$

- General Approach

  Define error, $e(t) = x^{des}(t) - x(t)$

  Want $e(t)$ to converge exponentially to zero

- Strategy

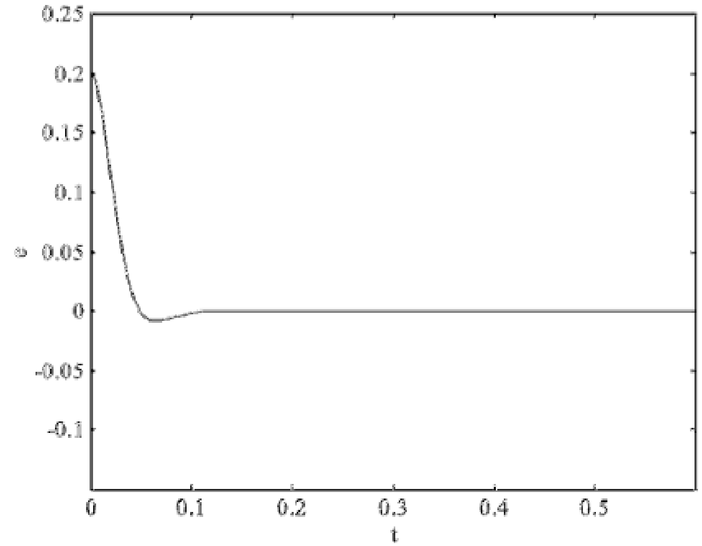
  Find u such that

  $$\dot{e} + K_p e = 0 \qquad K_p > 0$$

  $$u(t) = \dot{x}^{des}(t) + K_p e(t)$$

  Feed forward   Proportional

Control of a simple second-order system



- Problem
  - State, input

    $$x, u \in \boldsymbol{R}$$

  - Kinematic plant model

    $$\ddot{x} = u$$

  - Want x to follow trajectory $x^{des}(t)$
- General Approach
  - Define error, $e(t) = x^{des}(t) - x(t)$
  - Want $e(t)$ to converge exponentially to zero
- Strategy
  - Find u such that

    $$\ddot{e} + K_d\dot{e} + K_p e = 0 \qquad K_d, K_p > 0$$

    $$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)$$

Feed forward   Derivative   Proportional

PD control and PID control

- PD control

$$u(t) = \ddot{x}^{des}(t) + K_d \dot{e}(t) + K_p e(t)$$

Proportional control acts like a spring (capacitance) response

Derivative control is a viscous dashpot (resistance) response

Large derivative gain makes the system overdamped and the system converges slowly

- PID control

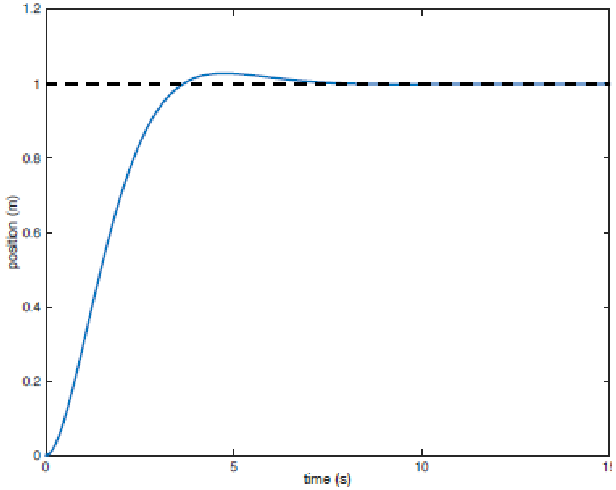
$$u(t) = \ddot{x}^{des}(t) + K_d \dot{e}(t) + K_p e(t) + K_I \int_0^t e(\tau) d\tau$$

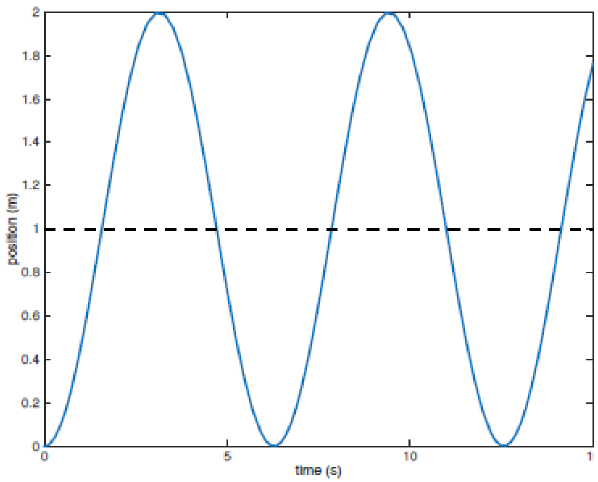
Integral

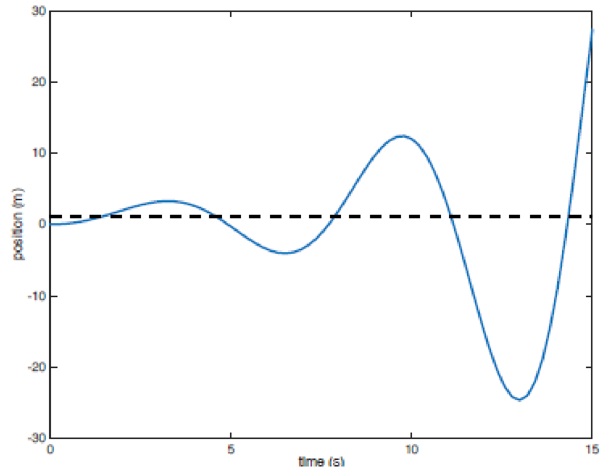
PID control generates a third-order closed-loop system

Integral control makes the steady-state error go to zero

Gain Tuning





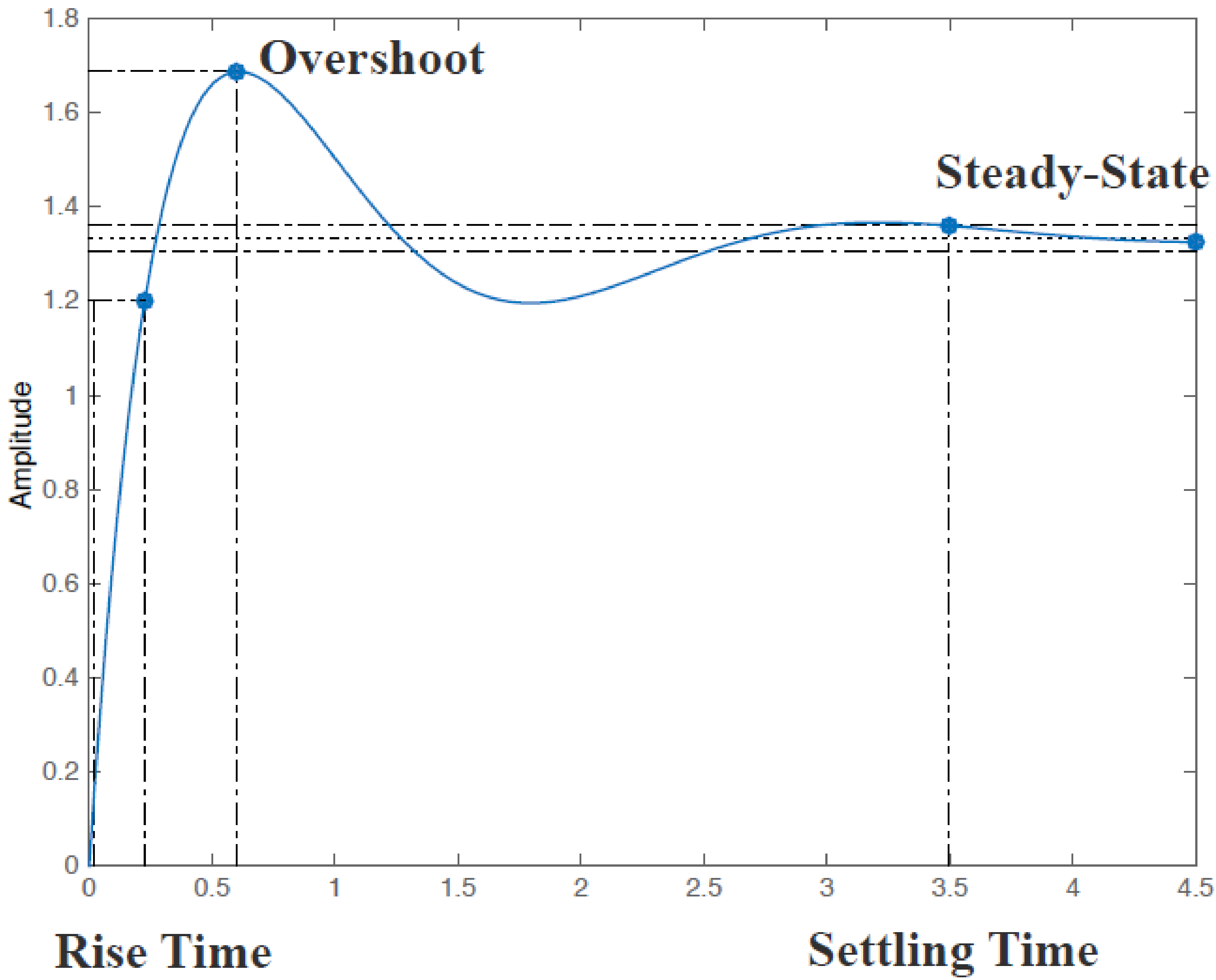


Stable
(converge)

Marginally Stable
(oscillate)

Unstable
(diverge)

Manual Tuning

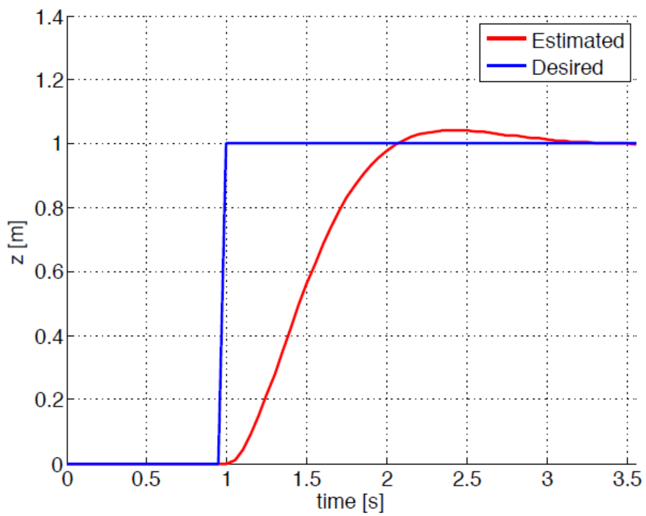


Manual Tuning

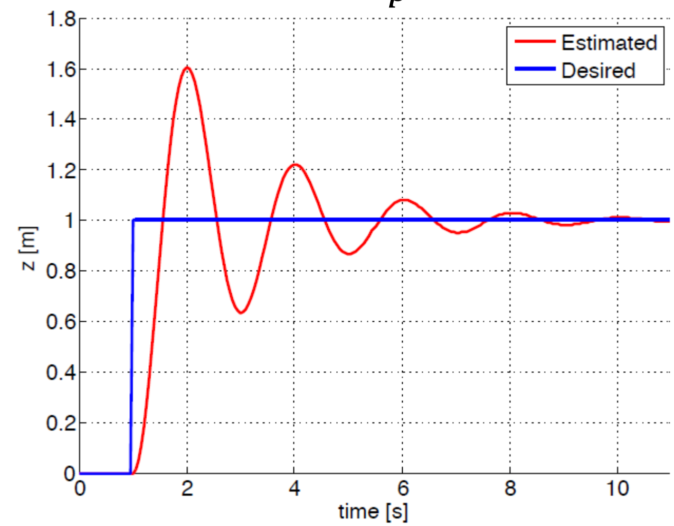
Parameter Increased	$K_p \uparrow$	$K_d \uparrow$	$K_i \uparrow$
Rise Time	Decrease	-	Decrease
Overshoot	Increase	Decrease	Increase
Settling Time	-	Decrease	Increase
Steady-State Error	Decrease	-	Eliminate

Manual Tuning

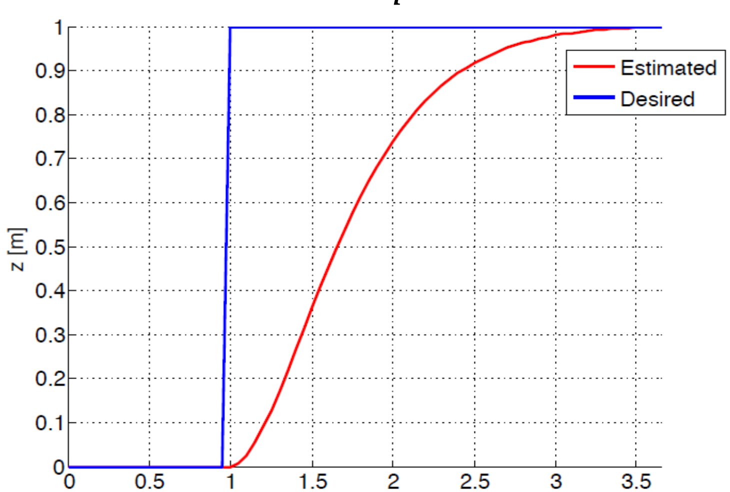
PD controller



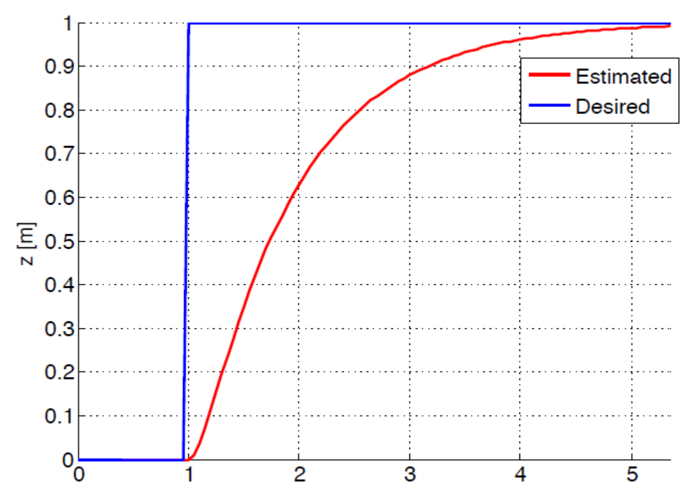
High K_p(overshoot)



Low K_p(soft response)



High K_d(overdamped)



Ziegler-Nichols Method

- Heuristic for tuning gains
    1. Set $K_i = K_d = 0$
    2. Increase K_p until ultimate gain, K_u, when output starts to oscillate
    3. Find the oscillation period T_u at K_u
    4. Set gains according to:

Controller	K_p	K_d	K_i
P	$0.5K_u$	-	-
PD	$0.8K_u$	$K_pT_u/8$	-
PID	$0.6K_u$	$K_pT_u/8$	$2K_p/T_u$

Model-based control

- Consider a general second-order model

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = u(\mathrm{t})$$

- Disadvantages of PID or PD control schemes

$$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)$$

  - Performance will depend on the model
  - Need to tune gains to maximize performance

- Model based control law

$$u(t) = \overbrace{\hat{m}(\underbrace{\ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)}_{\text{Servo: feedforward + PD feedback}}) + \hat{b}\dot{x}(t) + \hat{k}x(t)}^{\text{Model based}}$$

  - Model based part
    - Cancel the dynamics of the system
    - Specific to the model
  - Servo based part
    - Use PID or PD with feedforward to drive errors to zero
    - Independent of the model of the system

Model-based control

- Model based control law

Model based (estimates)

$$u(t) = \hat{m}\underbrace{(\ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t))}_{\text{Servo: feedforward + PD feedback}} + \hat{b}\dot{x}(t) + \hat{k}x(t)$$

- Advantage
  - Decomposes the control law into
    - Model-dependent part (depends on the knowledge of the model)
    - Model-independent part (servo control, gains are independent of the model)
- Disadvantage
  - Based on estimates of model parameters
    - Ideal performance

$$\ddot{e} + K_d\dot{e} + K_p e = 0$$

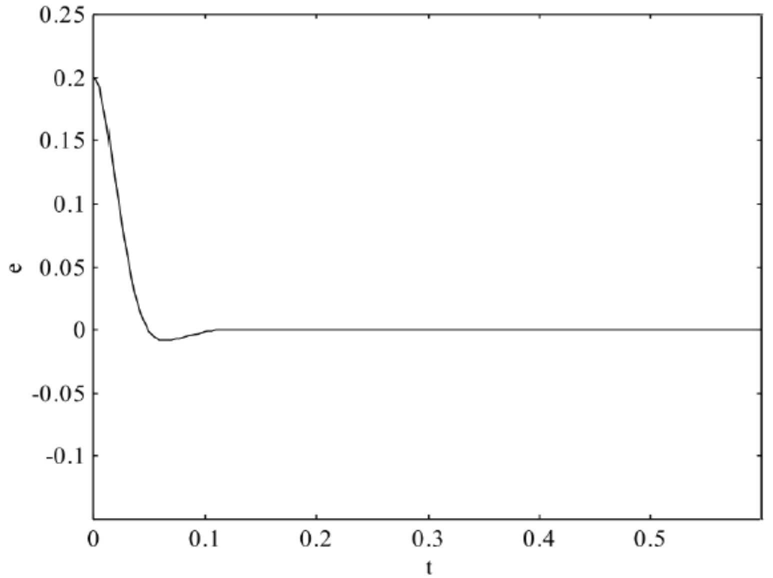
    - Actual performance

$$\ddot{e} + K_d\dot{e} + K_p e = \left(\frac{m}{\hat{m}} - 1\right)\ddot{x} + \frac{(b - \hat{b})}{\hat{m}}\dot{x} + \frac{(k - \hat{k})}{\hat{m}}x$$

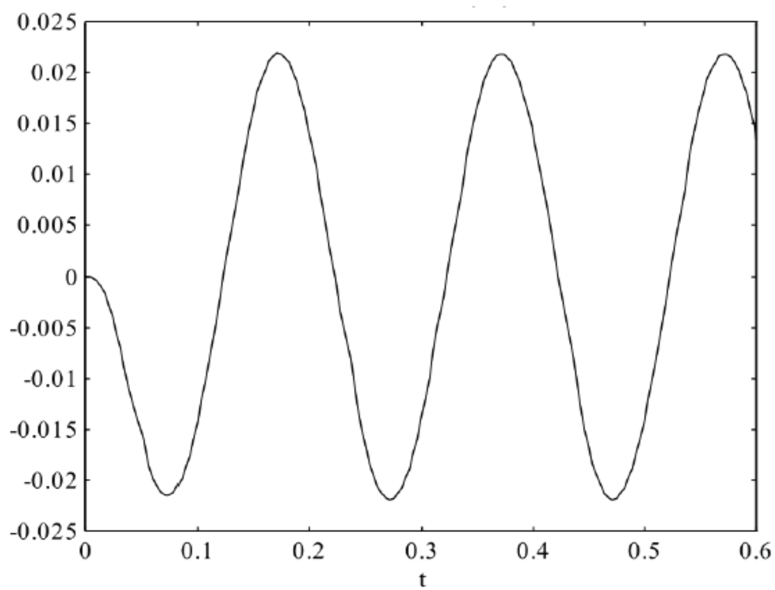
Model-based control

- Performance

$$u(t) = \hat{m}(\ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)) + \hat{b}\dot{x}(t) + \hat{k}x(t)$$



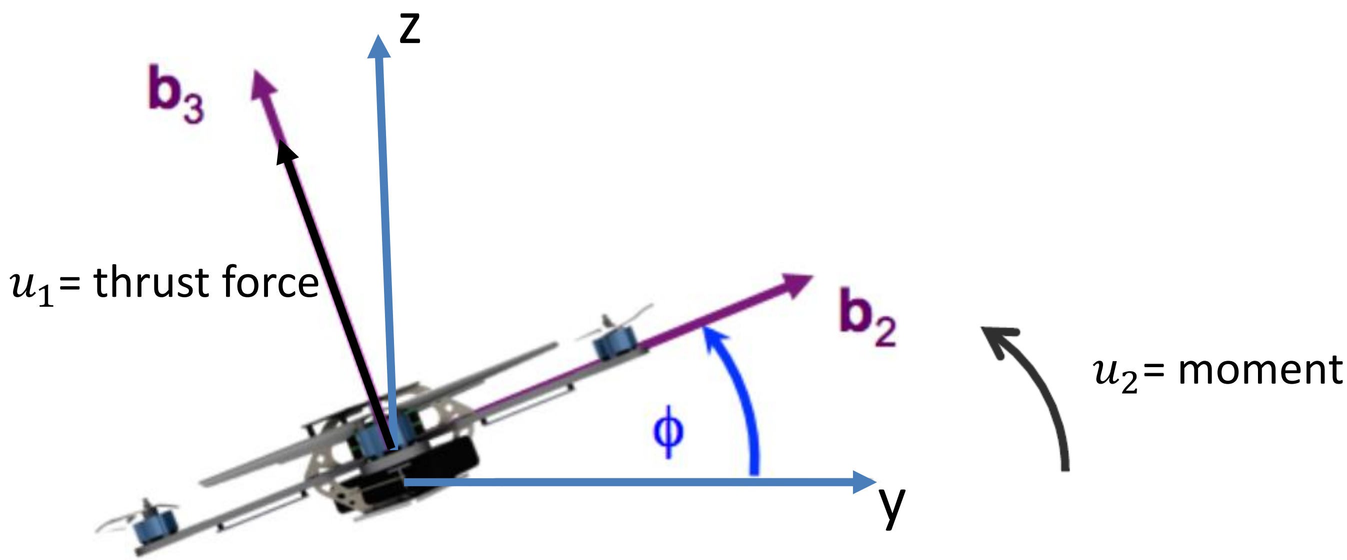
Perfect model



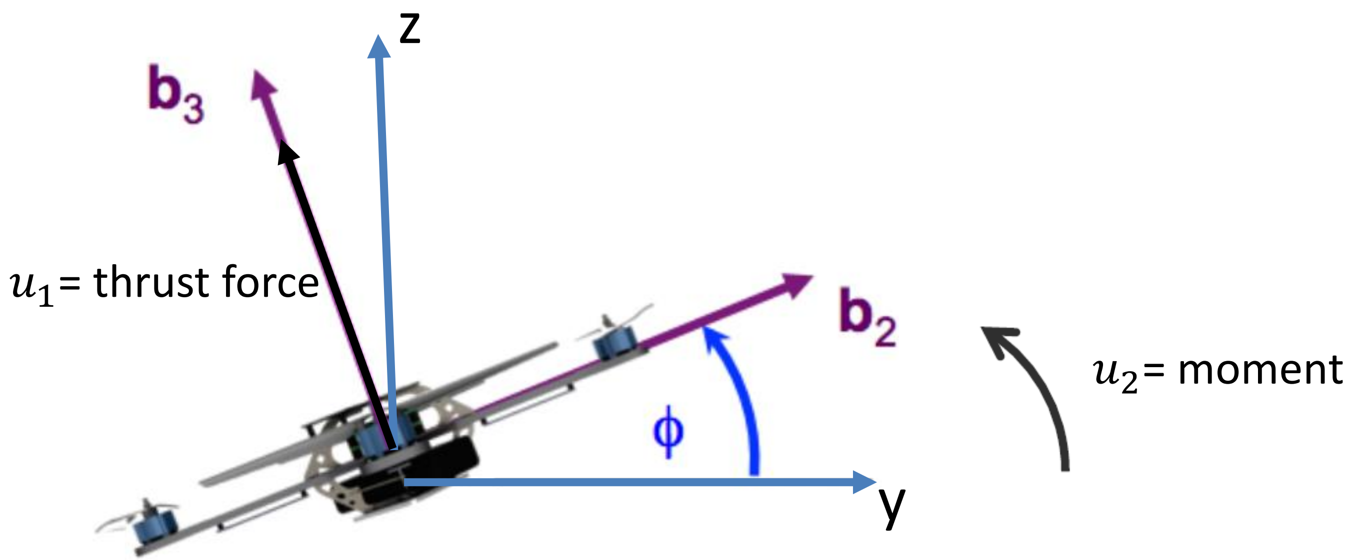
Imperfect model, 10% errors in parameters

Quadrotor Control

Application to Quadrotors



Planar Quadrotor Model



$$\sum F_y = -u_1 \sin(\phi) = m\ddot{y}$$

$$\sum F_z = -mg + u_1 \cos(\phi) = m\ddot{z}$$

$$M = u_2 = I_{xx}\ddot{\phi}$$

$$\Rightarrow \begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Linearized Dynamic Model

- Nonlinear dynamics

$$\ddot{y} = -\frac{u_1}{m} sin(\phi)$$
$$\ddot{z} = -g + \frac{u_1}{m} cos(\phi)$$
$$\ddot{\phi} = \frac{u_2}{I_{xx}}$$

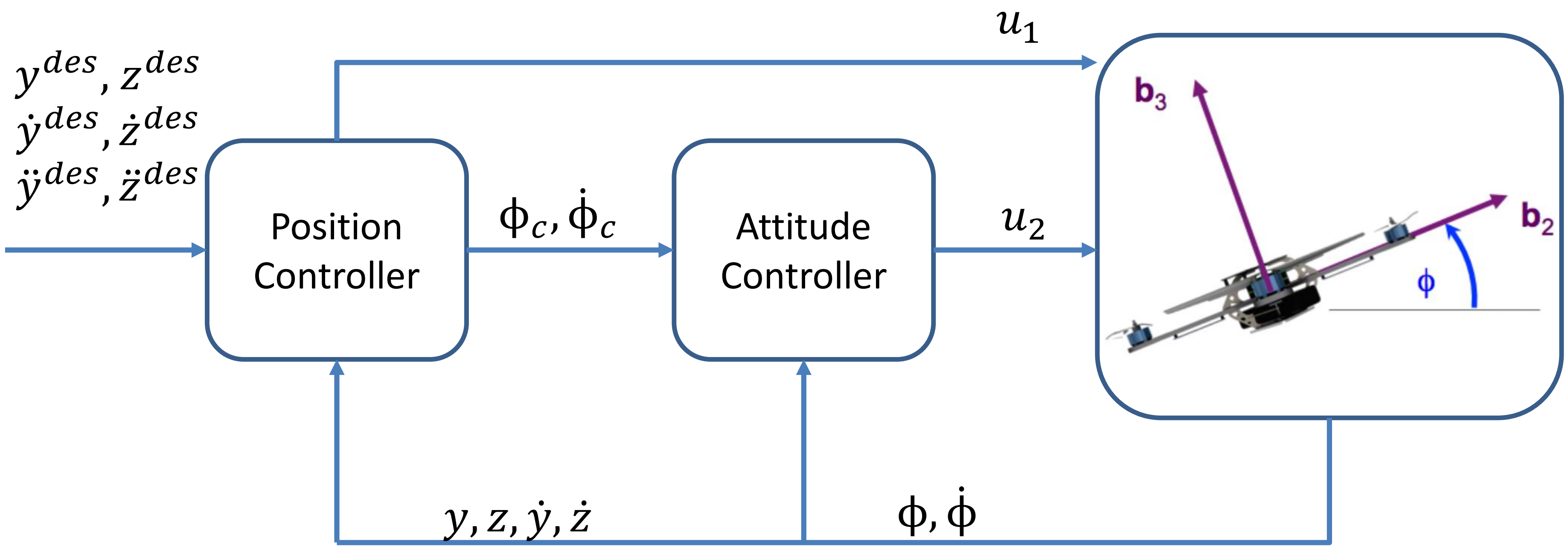
- Equilibrium hover configuration

$$y_0, z_0, \phi_0 = 0, u_{1,0} = mg, u_{2,0} = 0$$

- Linearized dynamics

$$\boxed{\ddot{y} = -g\phi}$$ Cascaded second order system
$$\ddot{z} = -g + \frac{u_1}{m}$$
$$\boxed{\ddot{\phi} = \frac{u_2}{I_{xx}}}$$ A simple second order system

Control Structure



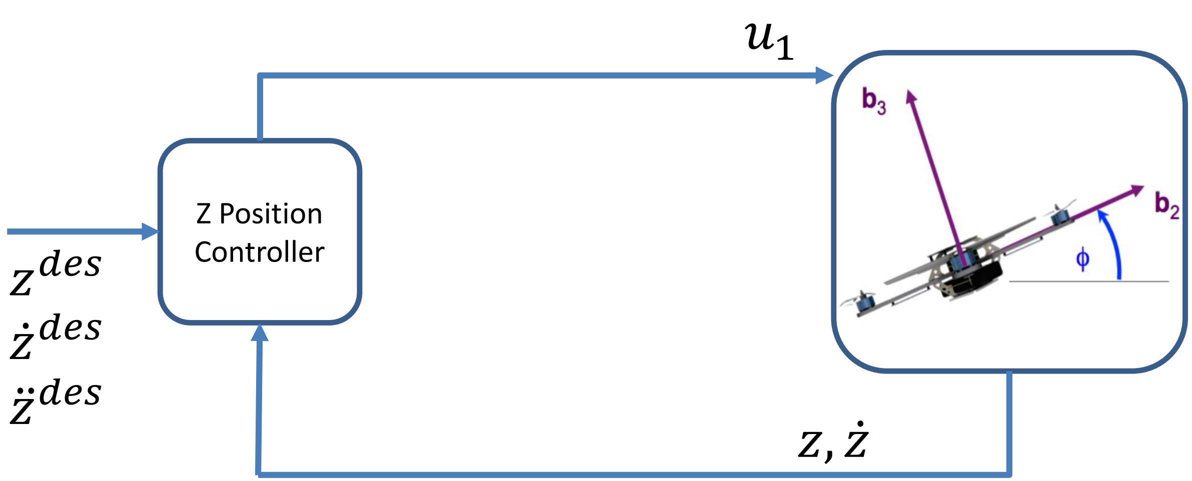
Control Equations

- Z-position control

PD: $$\ddot{z}_c = \ddot{z}^{des} + K_{d,z}(\dot{z}^{des} - \dot{z}) + K_{p,z}(z^{des} - z)$$

Model: $$\ddot{z} = -g + \frac{u_1}{m}$$

$$u_1 = m(g + \ddot{z}^{des} + K_{d,z}(\dot{z}^{des} - \dot{z}) + K_{p,z}(z^{des} - z))$$



Linearized Dynamic Model

- Y-position control

PD: $$\ddot{y}_c = \ddot{y}^{des} + K_{d,y}(\dot{y}^{des} - \dot{y}) + K_{p,y}(y^{des} - y)$$

Model: $$\ddot{y} = -g\phi$$

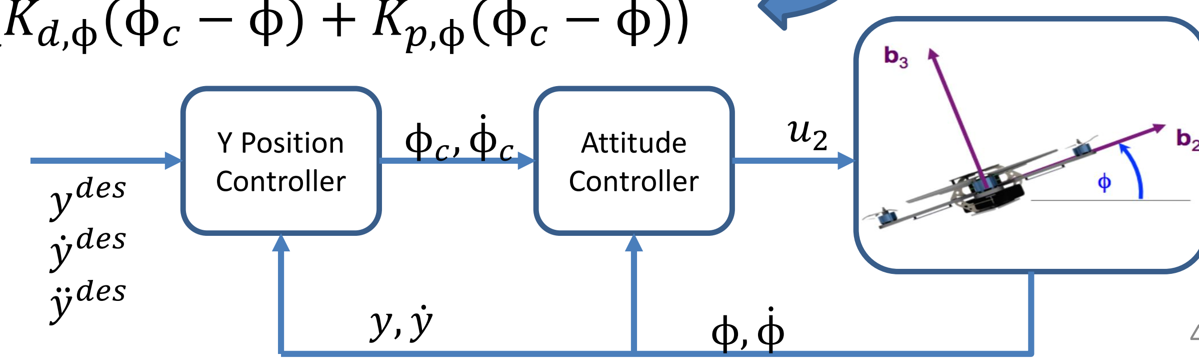
$$\phi_c = -\frac{1}{g}(\ddot{y}^{des} + K_{d,y}(\dot{y}^{des} - \dot{y}) + K_{p,y}(y^{des} - y))$$

- Attitude control

PD: $$\ddot{\phi}_c = K_{d,\phi}(\dot{\phi}_c - \dot{\phi}) + K_{p,\phi}(\phi_c - \phi)$$

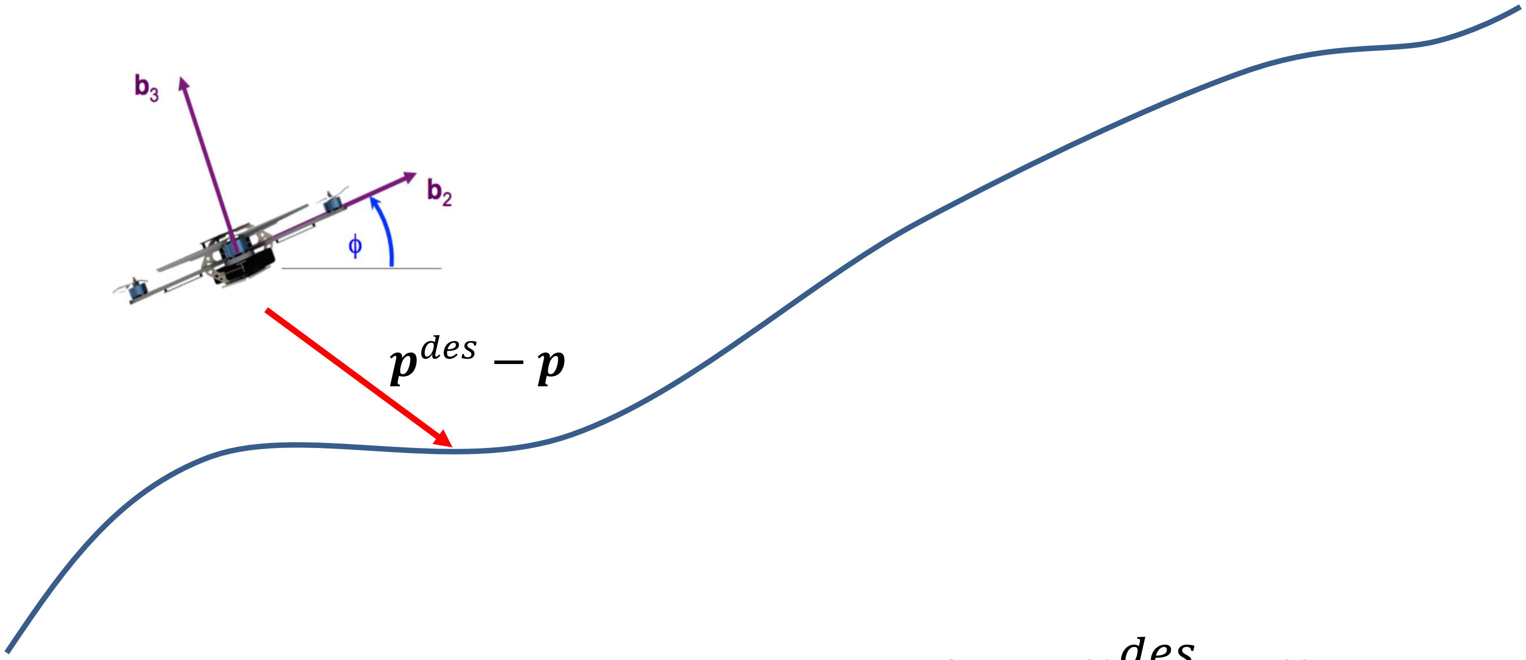
Model: $$\ddot{\phi} = \frac{u_2}{I_{xx}}$$

$$u_2 = I_{xx}(K_{d,\phi}(\dot{\phi}_c - \dot{\phi}) + K_{p,\phi}(\phi_c - \phi))$$



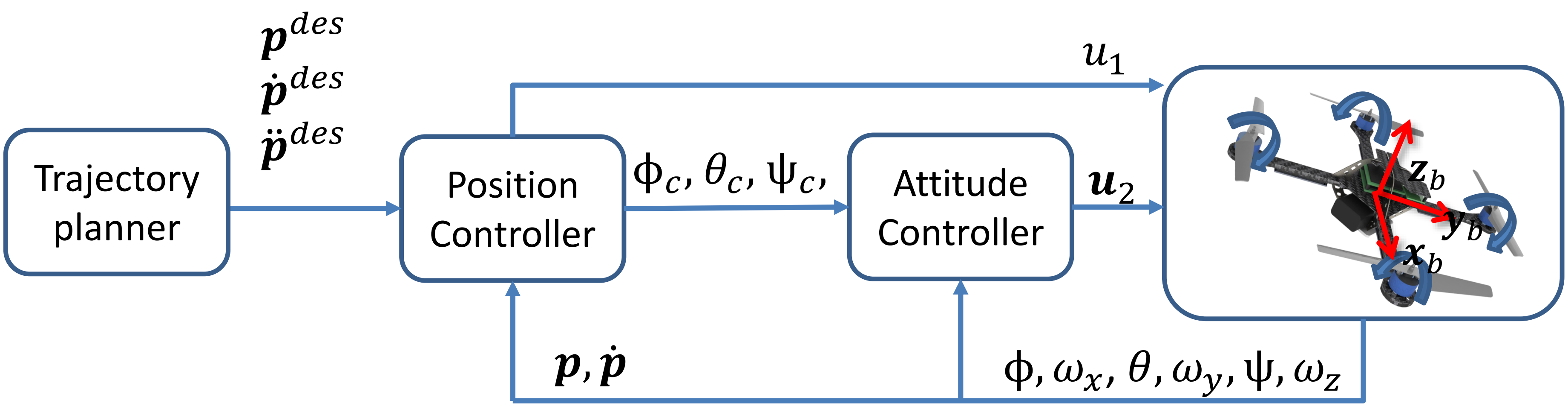
Trajectory Tracking

Given $\boldsymbol{p}^{des}$,$\dot{\boldsymbol{p}}^{des}$,$\ddot{\boldsymbol{p}}^{des}$



$$\boldsymbol{e_p} = \boldsymbol{p}^{des} - \boldsymbol{p}$$
$$\boldsymbol{e_v} = \dot{\boldsymbol{p}}^{des} - \dot{\boldsymbol{p}}$$
$$\ddot{\boldsymbol{p}}_c = \ddot{\boldsymbol{p}}^{des} + \boldsymbol{K_d e_v} + \boldsymbol{K_p e_p}$$

3-D Quadrotor



- Nonlinear dynamics

Newton Equation: $$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}\begin{bmatrix} 0 \\ 0 \\ \boxed{F_1 + F_2 + F_3 + F_4} \end{bmatrix} \quad u_1$$

Euler Equation: $$\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \boxed{\begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}} \quad \boldsymbol{u}_2$$

3-D Quadrotor

- Linearization

Equilibrium hover $(\phi_0 \sim 0, \theta_0 \sim 0, u_{1,0} \sim mg)$

Newton equation

$$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}\begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

$$\Rightarrow$$

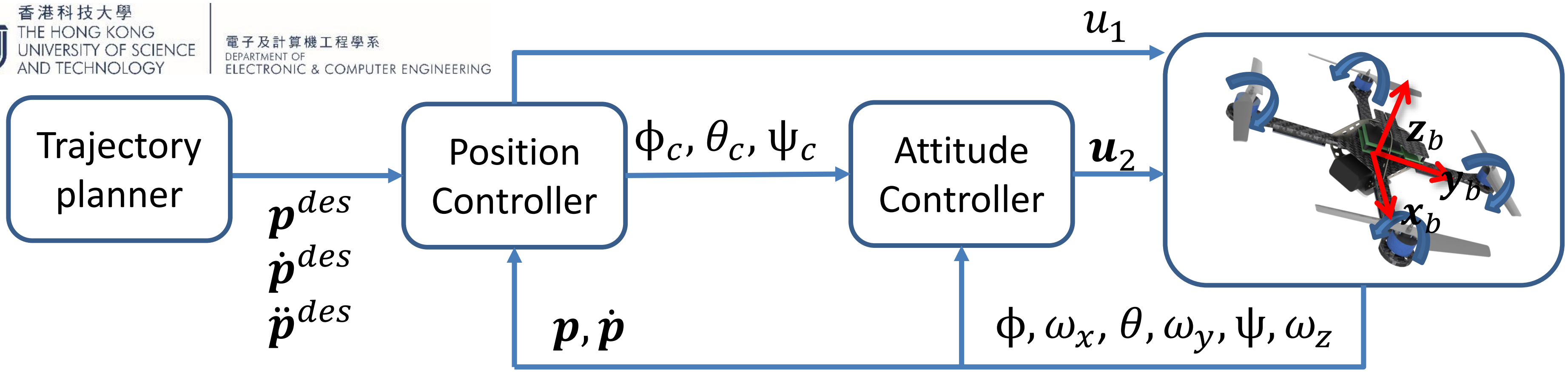
$$\begin{aligned} \ddot{\boldsymbol{p}}_1 &= \ddot{x} = g(\theta cos\psi + \phi sin\psi) \\ \ddot{\boldsymbol{p}}_2 &= \ddot{y} = g(\theta sin\psi - \phi cos\psi) \\ \ddot{\boldsymbol{p}}_3 &= \ddot{z} = -g + \frac{u_1}{m} \end{aligned}$$

Euler angle derivative

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \Rightarrow \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Euler Equation:

$$\boldsymbol{I} \cdot \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$$



- Position control

PID: $$\ddot{\boldsymbol{p}}_{i,c} = \ddot{\boldsymbol{p}}_i^{\,des} + K_{d,i}\left(\dot{\boldsymbol{p}}_i^{\,des} - \dot{\boldsymbol{p}}_i\right) + K_{p,i}\left(\boldsymbol{p}_i^{\,des} - \boldsymbol{p}_i\right)$$

Model: $$u_1 = m(g + \ddot{\boldsymbol{p}}_{3,c})$$

(Newton Equation)

$$\phi_c = \frac{1}{g}\left(\ddot{\boldsymbol{p}}_{1,c} sin\boxed{\psi} - \ddot{\boldsymbol{p}}_{2,c} \cos\boxed{\psi}\right) \qquad \theta_c = \frac{1}{g}\left(\ddot{\boldsymbol{p}}_{1,c} cos\boxed{\psi} + \ddot{\boldsymbol{p}}_{2,c} sin\boxed{\psi}\right)$$

These are current yaw, not the commanded yaw

- Attitude control

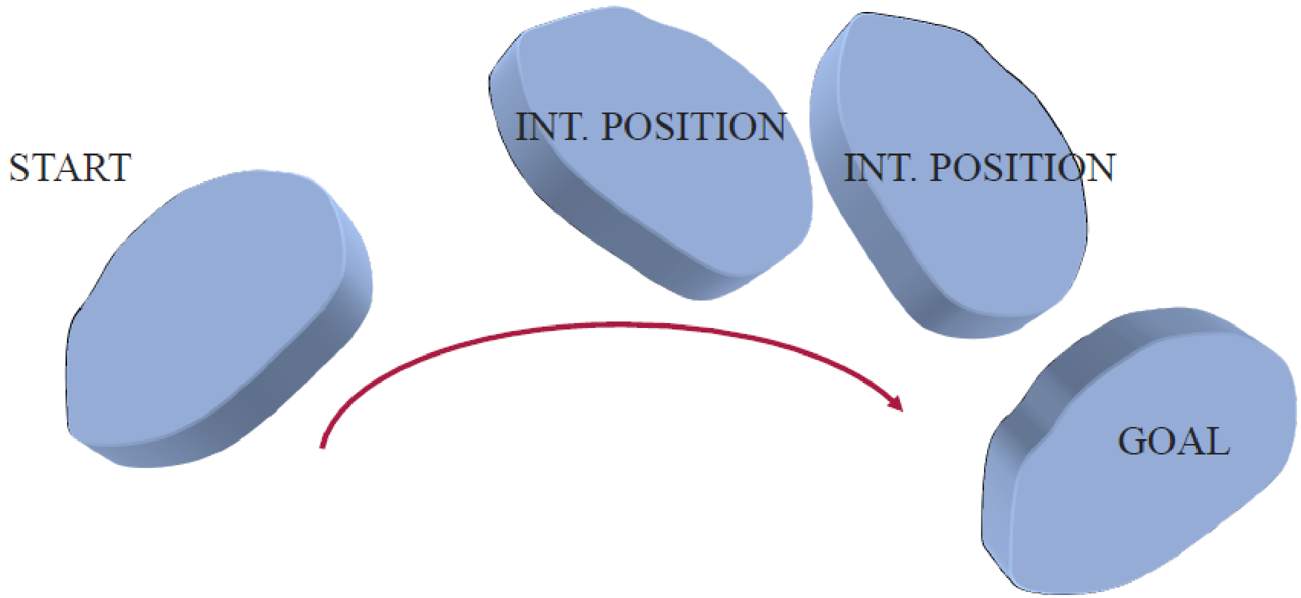
PID: $$\begin{bmatrix} \ddot{\phi}_c \\ \ddot{\theta}_c \\ \ddot{\psi}_c \end{bmatrix} = \begin{bmatrix} K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(\dot{\phi}_c - \dot{\phi}) \\ K_{p,\theta}(\theta_c - \theta) + K_{d,\phi}(\dot{\theta}_c - \dot{\theta}) \\ K_{p,\psi}(\psi_c - \psi) + K_{d,\psi}(\dot{\psi}_c - \dot{\psi}) \end{bmatrix}$$

Model: $$\boldsymbol{u}_2 = \boldsymbol{I} \cdot \begin{bmatrix} \ddot{\phi}_c \\ \ddot{\theta}_c \\ \ddot{\psi}_c \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$ (Euler Equation)

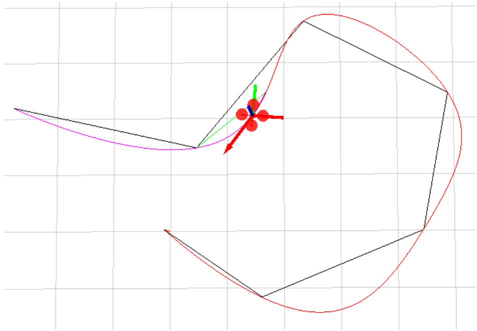
Trajectory Generation

Smooth 3D Trajectories

- Smooth trajectory is beneficial for autonomous flight
  - Smooth trajectories respect the continuous nature of aerial robots
  - The robot should not stop at turns



Smooth 3D Trajectories



- General setup
  - Start, goal positions (orientations)
  - Waypoint positions (orientations)
    - Waypoints can be found by path planning (A*, RRT*, etc)
    - To be covered in the next lecture
  - Smoothness criterion
    - Generally translates into minimizing rate of change of "input"
- Question: How to make sure that a trajectory can be tracked by the quadrotor?

Differential Flatness

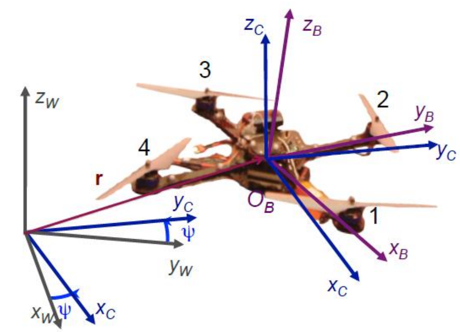
- The states and the inputs of a quadrotor can be written as algebraic functions of four carefully selected flat outputs and their derivatives
  - Enables automated generation of trajectories
  - Any smooth trajectory in the space of flat outputs (with reasonably bounded derivatives) can be followed by the under-actuated quadrotor
  - A possible choice:
    - $\boldsymbol{\sigma} = [x, y, z, \psi]^T$
  - Trajectory in the space of flat outputs:
    - $\boldsymbol{\sigma}(t) = [T_0, T_M] \to \mathbb{R}^3 \times SO(2)$

Differential Flatness

- Quadrotor states
  - Position, orientation, linear velocity, angular velocity

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \boxed{\omega_x, \omega_y, \omega_z}]^T$$

Body angular velocity viewed in the body frame



  - Equation of motions:

$$m\ddot{\mathbf{p}} = -mg\mathbf{z}_W + u_1\mathbf{z}_B.$$

$$\boldsymbol{\omega}_B = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad \dot{\boldsymbol{\omega}}_B = \boldsymbol{I}^{-1}\left[-\boldsymbol{\omega}_B \times \boldsymbol{I} \cdot \boldsymbol{\omega}_B + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right]$$

- Position, velocity, and acceleration are simply derivatives of the flat outputs

Differential Flatness

- Orientation
  - Quadrotor state:

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

  - From the equation of motion:

$$\mathbf{z}_B = \frac{\mathbf{t}}{\|\mathbf{t}\|}, \mathbf{t} = [\ddot{\boldsymbol{\sigma}}_1, \ddot{\boldsymbol{\sigma}}_2, \ddot{\boldsymbol{\sigma}}_3 + g]^T$$

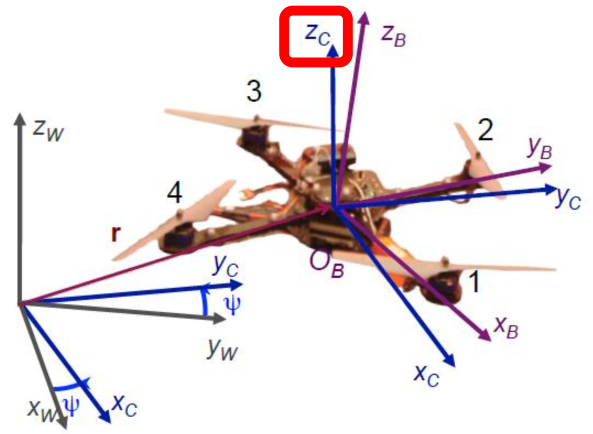
  - Define the yaw vector (Z-X-Y Euler):

$$\mathbf{x}_C = [cos\boldsymbol{\sigma}_4, sin\boldsymbol{\sigma}_4, 0]^T$$

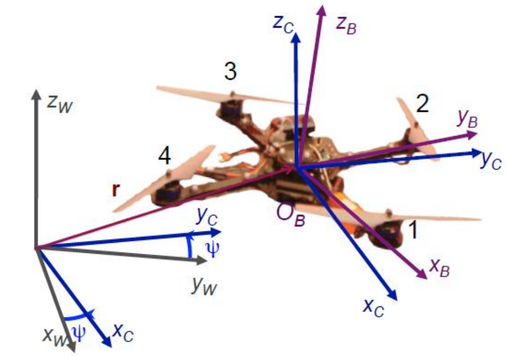
  - Orientation can be expressed in terms of flat outputs

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{\|\mathbf{z}_B \times \mathbf{x}_C\|}, \quad \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \quad \boldsymbol{R}_B = [\mathbf{x}_B \ \ \mathbf{y}_B \ \ \mathbf{z}_B]$$

A "no tilt" frame



Differential Flatness



- Angular velocity
  - Quadrotor state:

    $$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_z, \omega_y, \omega_z]^T$$

  - Take the derivative of the equation of motion

    $$m\ddot{\boldsymbol{p}} = -mg\mathbf{z}_W + u_1\mathbf{z}_B. \longrightarrow m\dot{\boldsymbol{a}} = \dot{u}_1\mathbf{z}_B + \boxed{\boldsymbol{\omega}_{BW}} \times u_1\,\mathbf{z}_B$$

    Body angular velocity viewed in the world frame

  - Quadrotors only have vertical thrust:

    $$\dot{u}_1 = \mathbf{z}_B \cdot m\dot{\boldsymbol{a}}$$

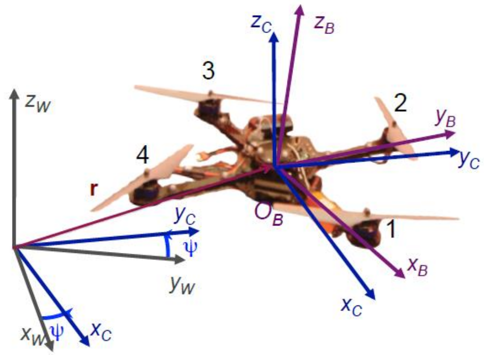
  - We have:

    $$\mathbf{h}_\omega = \boldsymbol{\omega}_{BW} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\boldsymbol{a}} - (\mathbf{z}_B \cdot \dot{\boldsymbol{a}})\mathbf{z}_B).$$

Differential Flatness

- Angular velocity
  - Quadrotor state:

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$



  - We have:

$$\mathbf{h}_\omega = \boldsymbol{\omega}_{BW} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\boldsymbol{a}} - (\mathbf{z}_B \cdot \dot{\boldsymbol{a}})\mathbf{z}_B).$$

    - This is the projection of $\frac{m}{u_1}\dot{\boldsymbol{a}}$ onto the $x_B - y_B$ plane
  - We know that:

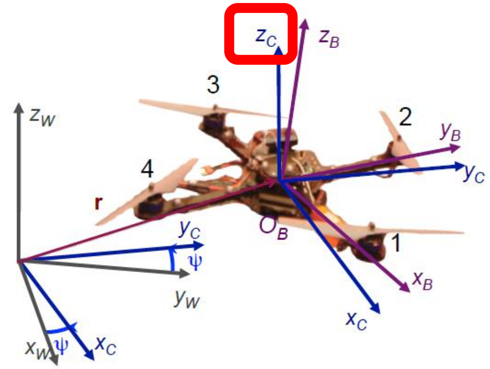
$$\boldsymbol{\omega}_{BW} = \omega_x \mathbf{x}_B + \omega_y \mathbf{y}_B + \omega_z \mathbf{z}_B$$

  - Angular velocities along x_B and y_B directions can be found as:

$$\omega_x = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \quad \omega_y = \mathbf{h}_\omega \cdot \mathbf{x}_B$$

Differential Flatness

A "no tilt" frame



- Angular velocity
  - Quadrotor state:

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

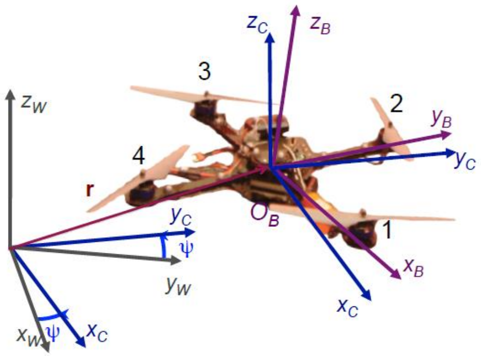
  - We have:

$$\mathbf{h}_\omega = \boldsymbol{\omega}_{BW} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\boldsymbol{a}} - (\mathbf{z}_B \cdot \dot{\boldsymbol{a}})\mathbf{z}_B).$$

    - This is the projection of $\frac{m}{u_1}\dot{\boldsymbol{a}}$ onto the $x_B - y_B$ plane
  - Since $\boldsymbol{\omega}_{BW} = \boldsymbol{\omega}_{BC} + \boldsymbol{\omega}_{CW}$, where $\boldsymbol{\omega}_{BC}$ has no $\mathbf{z}_B$ component:

$$\omega_z = \boldsymbol{\omega}_{BW} \cdot \mathbf{z}_B = \boldsymbol{\omega}_{CW} \cdot \mathbf{z}_B = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B.$$

Differential Flatness



- Summary
  - Quadrotor state:

    $$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

  - Flat outputs:
    - $\boldsymbol{\sigma} = [x, y, z, \psi]^T$

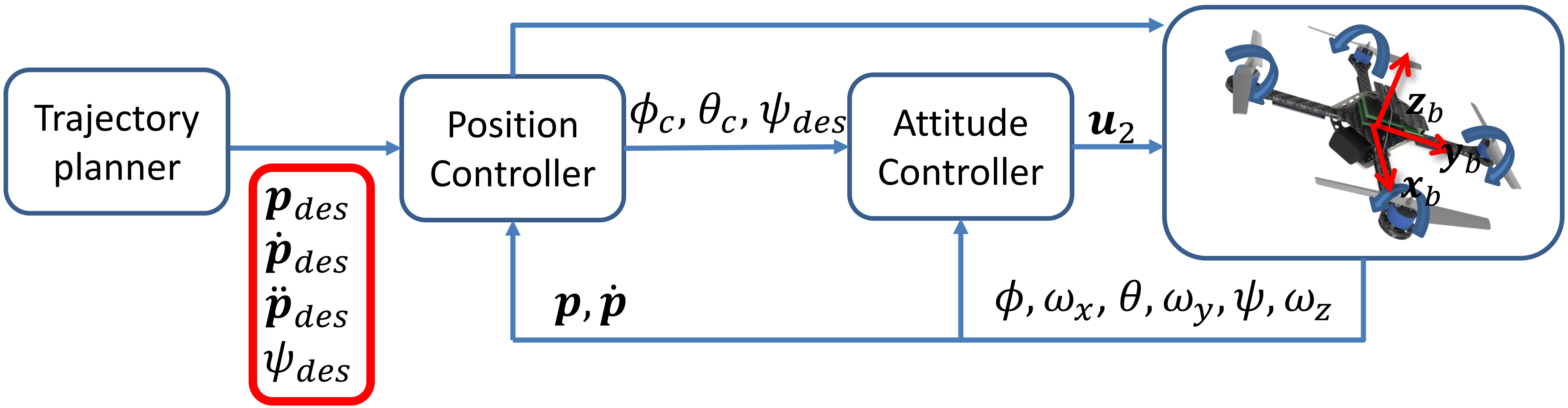
How about Force and Moment Input ($u_1, \boldsymbol{u}_2$)?

  - Position, velocity, acceleration
    - Derivatives of flat outputs
  - Orientation

    $$\mathbf{x}_C = [cos\sigma_4, sin\sigma_4, 0]^T \longrightarrow \boldsymbol{R}_B = [\mathbf{x}_B \ \ \mathbf{y}_B \ \ \mathbf{z}_B]$$

  - Angular velocity

    $$\omega_x = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \quad \omega_y = \mathbf{h}_\omega \cdot \mathbf{x}_B, \quad \omega_z = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B$$



Nonlinear dynamics

Newton Equation: $$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$

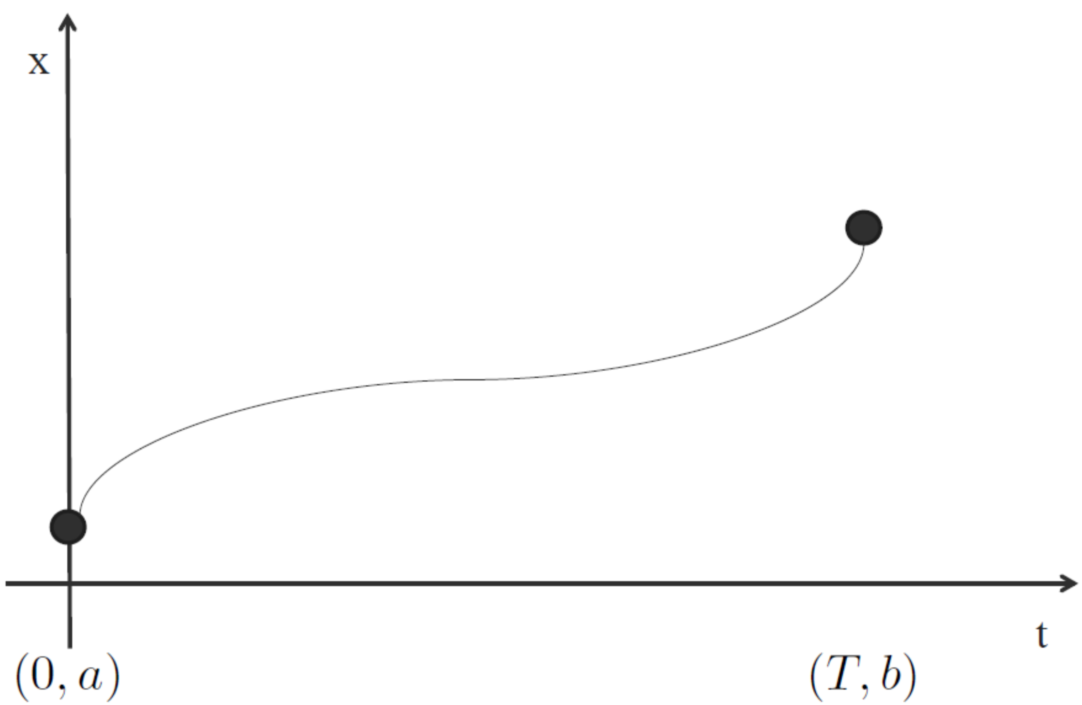
Euler Equation: $$\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$$

Polynomial Trajectories

- Flat outputs:
  - $\boldsymbol{\sigma} = [x, y, z, \psi]^T$
- Trajectory in the space of flat outputs:
  - $\boldsymbol{\sigma}(t) = [T_0, T_M] \rightarrow \mathbb{R}^3 \times SO(2)$
- Polynomial functions can be used to specify trajectories in the space of flat outputs
  - Easy determination of smoothness criterion with polynomial orders
  - Easy and closed form calculation of derivatives
  - Decoupled trajectory generation in three dimensions

Smooth 1D Trajectory

- Design a trajectory $x(t)$ such that:
  - $x(0) = a$
  - $x(T) = b$



Smooth 1D Trajectory

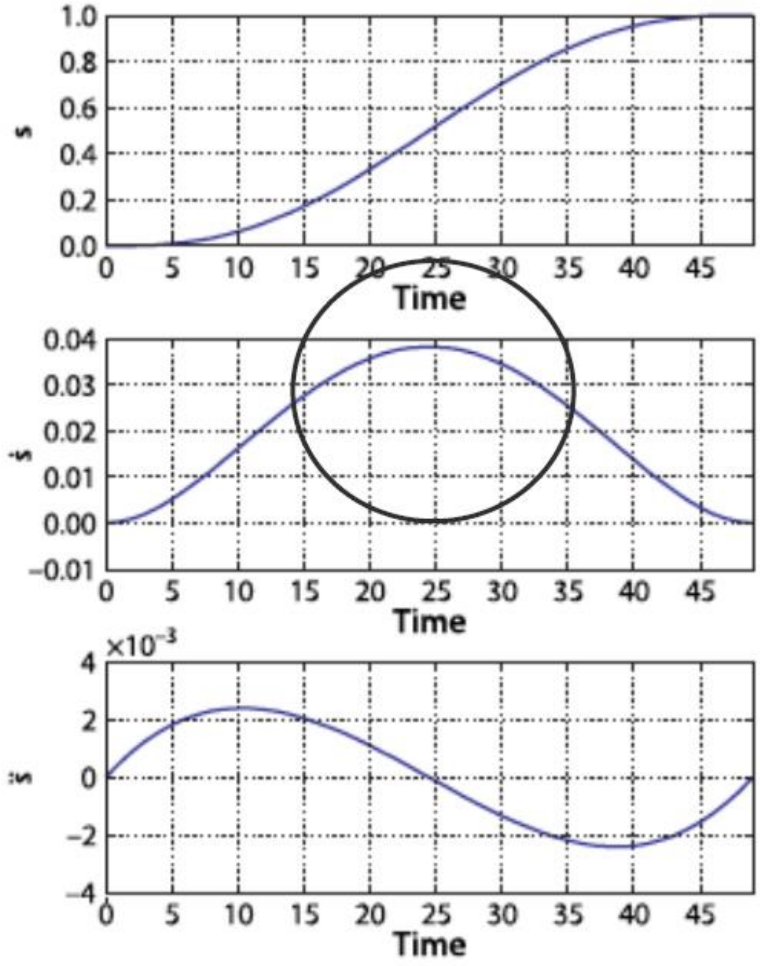
- 5^{th} order polynomial trajectory:
  - $x(t) = c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0$
- Boundary conditions

	Position	Velocity	Acceleration
t = 0	a	0	0
t = T	b	0	0

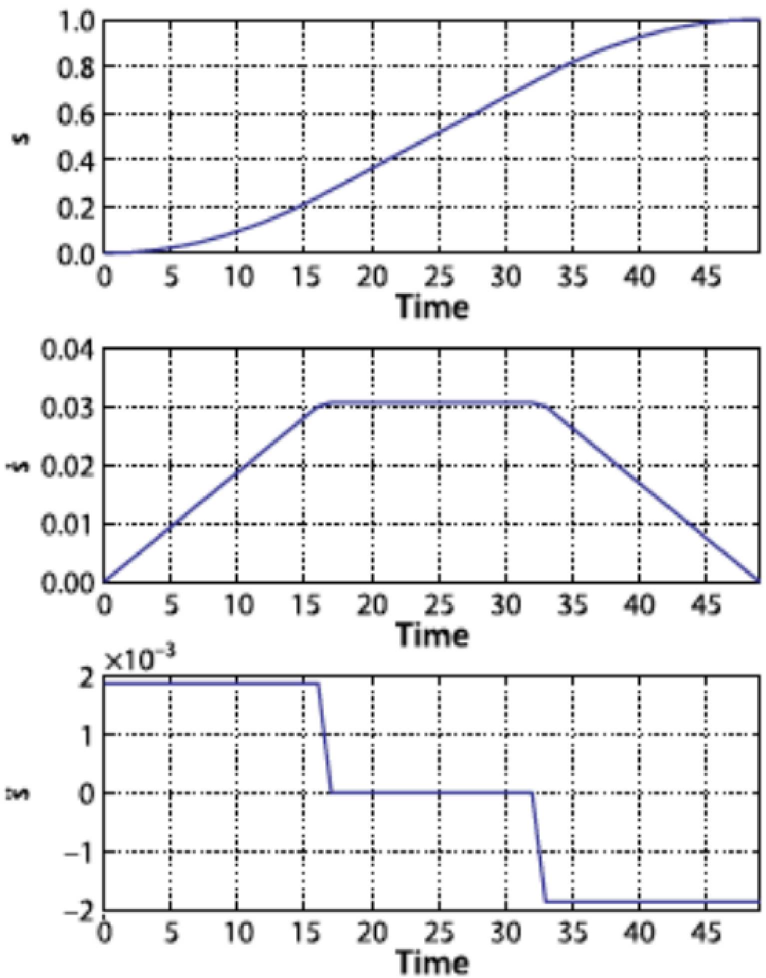
- Solve:

$$\begin{bmatrix} a \\ b \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}$$

Smooth 1D Trajectory

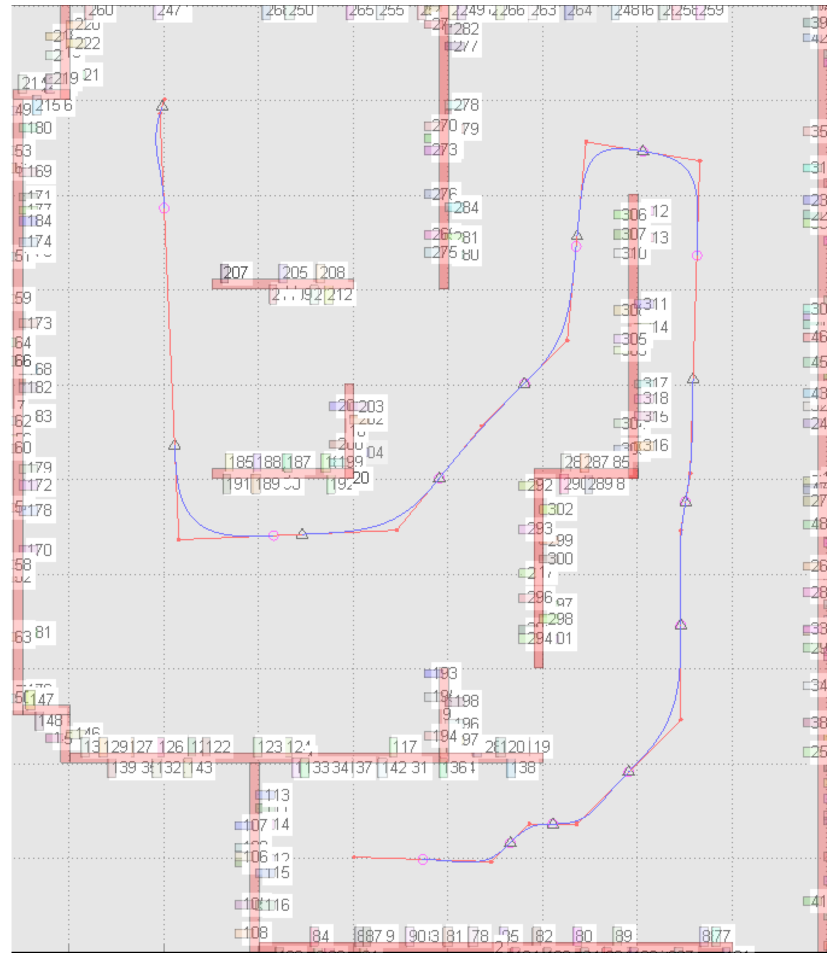


Bang-Bang Trajectory



Smooth Multi-Segment Trajectory

- Smooth the corners of straight line segments
- Preferred constant velocity motion at v
- Preferred zero acceleration
- Requires special handling of short segments



Smooth 1D Trajectory

- Generate each 5^{th} order polynomial independently:
  - $x(t) = c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0$
- Boundary conditions

	Position	Velocity	Acceleration
t = 0	a	v_0	0
t = T	b	v_T	0

- Solve:

$$\begin{bmatrix} a \\ b \\ v_0 \\ v_T \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}$$

Next Lecture…

- Continuation on trajectory generation
- Path planning

Logistics

- Project 1, phase 1 is released (02/21)
  - Due on 3/3. Early submission is encouraged.