Rotorcraft

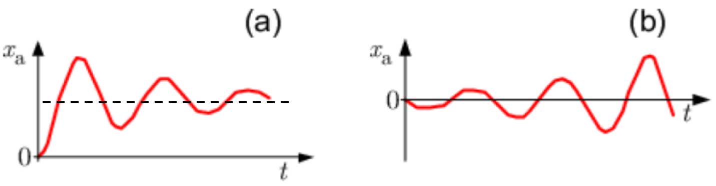
Lecture 4: Control of Rotorcraft

Konrad Rudin

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Why control of rotorcraft?

- **In general rotorcraft are unstable systems like:**
  - Bicycles, inverted pendulums, etc…



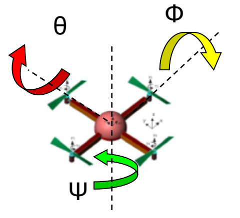
STABLE UNSTABLE

- We want the rotorcraft to execute our desire!
  - Attitude control
  - Trajectory tracking
  - Obstacle avoidance
  - Automatic landing

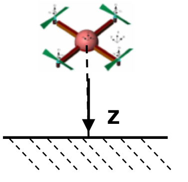
Out of Ctrl

What do we control?

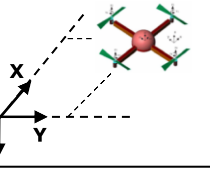
- **Low level controllers**
  - Attitude (3D orientation of the system)
    - **Roll (Φ) → Rotation about x axis**
    - **Pitch (θ) → Rotation about y axis**
    - **Yaw (Ψ) → Rotation about z axis**



  - Altitude (vertical distance to the ground)
    - **Z**

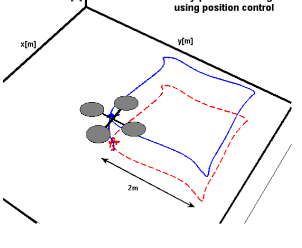


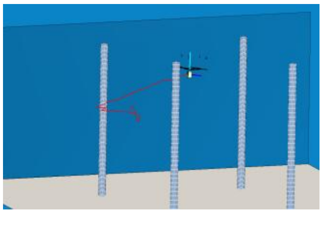
  - Position (2D coordinates in an inertial frame)
    - **X, Y**



What do we control?

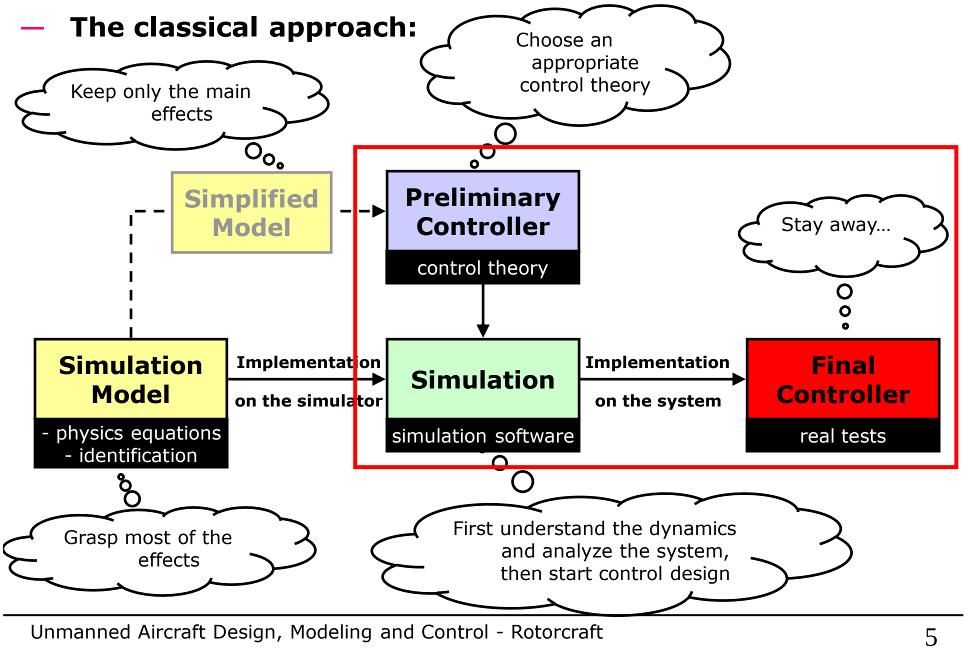
- **High level controllers**
  - Obstacle avoidance (Sense & Avoid)
  - Trajectory tracking
  - …





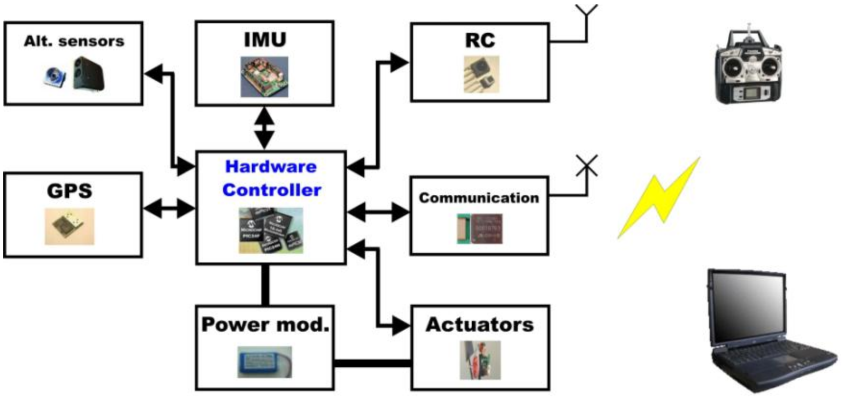
How do we control?

– **The classical approach:**



Hardware

– **Typical block diagram of a miniature flying robot**



Attitude sensors

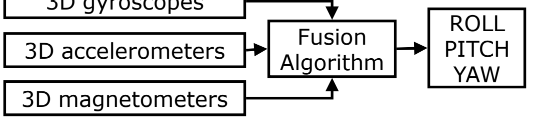
- **Inertial sensors**
  - Inertial Measurement Unit (IMU) is generally composed of:
    - **3 rate Gyroscopes** → provide rate of turn of a body (p,q,r) [rad/s]
    - **3 Accelerometers** → provide linear acceleration [m/s^2]
    - **3 Magnetometers** → provide magnetic field measurement [T]
    - **1 Thermometer** → temperature (for gyro.+accel. compensation)
  - Output: Attitude and body angular rates

Courtesy of Xsens

Low-cost IMU
~ 2000 CHF

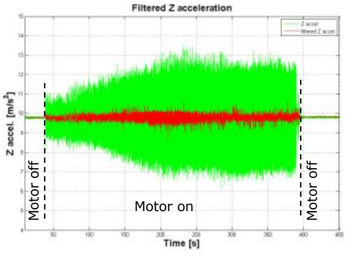
Attitude sensors

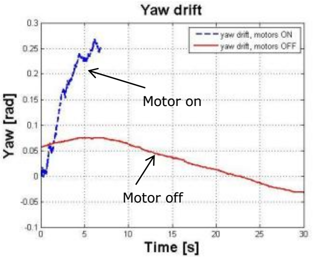
- IMU does not measure the attitude directly, need a estimator
  - Gyroscopes: Measure the angular velocity
    - Integration gives the attitude
      - Due to integration of noise there will be a drift on the attitude
  - Accelerometer: Assumption $a_{acc} \approx g$ (near hover condition)
    - Can be used as a measure of some parts of the attitude (known gravity direction in the inertial frame)
      - Yaw angles rest undetermined
  - Magnetometer: Measures earth magnetic field
    - Can be used as a measure of the yaw angle



Attitude sensors

- **Inertial sensors**
  - Inertial Measurement Unit (IMU) is generally selected for its:
    - **Angular resolution** → 0.05° or less
    - **Static accuracy** → <0.5° or less
    - **Dynamic accuracy** → <2° or less (depends on motion)
    - **Mass !**

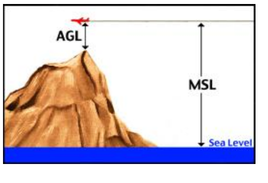




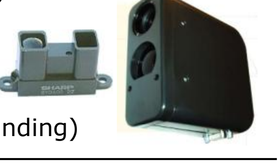
Altitude sensors

- **Absolute pressure sensors (MSL, Mean Sea Level)**
  - Measures the change in pressure due to altitude
  - Low cost, low weight
  - High noise

$$p = p_0 (1 - 0.0065 \frac{h}{T_0})^{5.2561}$$

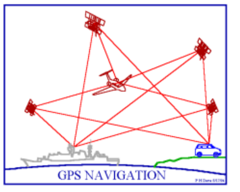


- **Range finders (AGL, Above Ground Level)**
  - Measures the distance to the ground
  - High accuracy
  - Used for distances close to the ground (and landing)

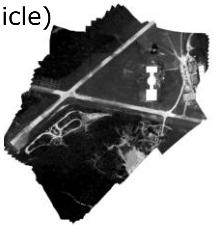


Position sensors

- **Global Positioning System (GPS, Galileo…)**
  - Estimates absolute position (latitude, longitude and altitude)
  - Low accuracy (2-5m)
  - Very small receivers available
  - Used only outdoor

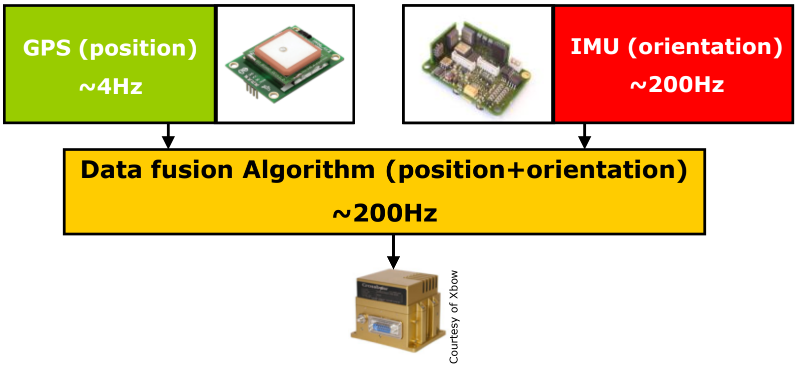


- **Vision Based Positioning**
  - Estimates relative position (X, Y, Z: ground-vehicle)
  - High precision (cm)
  - Requires powerful processing
  - Used indoor or outdoor



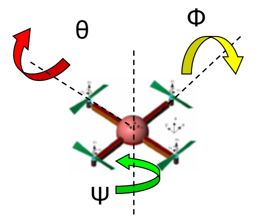
Attitude+Position sensor modules

- **GPS Aided Inertial Systems**
  - Combination of IMU and GPS
  - Provides estimation of the 6DoF

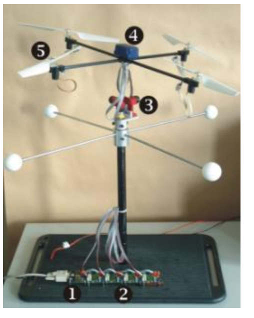


AHRS: Attitude and Heading Reference Systems **(6 DoF)**

Quadrotor control example

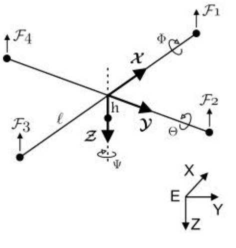


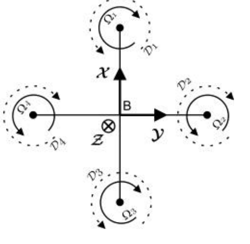
- **Objective:**
  - Control the orientation of a quadrotor
- **Steps:**
  1. Check the equations
  2. Simplify the model (when model-based control is used)
  3. Design a controller
  4. Simulate the Model + Controller
  5. Tune the parameters
  6. Implement & test on the system



Check the equations

- Position and attitude states 6
- Motor speeds 4





- Underactuated system
  - Cannot control all states independently
  - Fly maneuvers
- Motor Dynamics faster than system dynamics
  - Assume motor speeds are algebraic variables

Check the equations

- Rotational dynamics near hover

$$\begin{cases} J_{xx}\dot{p} = qr(J_{yy} - J_{zz}) - J_R q\Omega_r + lb(\Omega_4^2 - \Omega_2^2) \\ J_{yy}\dot{q} = pr(J_{zz} - J_{xx}) + J_R p\Omega_r + lb(\Omega_3^2 - \Omega_1^2) \\ J_{zz}\dot{r} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases}$$

$$\begin{cases} J_r \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{cases}$$

- Full control of all moments
- Not dependend on position states

Check the equations

- Translational dynamics in hover

$$\begin{cases} m\dot{u} = -\sin\theta mg \\ m\dot{v} = \sin\varphi\cos\theta mg \\ m\dot{w} = \cos\varphi\cos\theta mg - b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{cases}$$

$$\dot{X} = R_{(\varphi,\theta,\psi)} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

- Only control the body z acceleration
  - Use attitude to control position

Virtual control input

- Define new control inputs to simplify the equations
  - Get linear decoupled inputs
  - Thrust input along body z axis

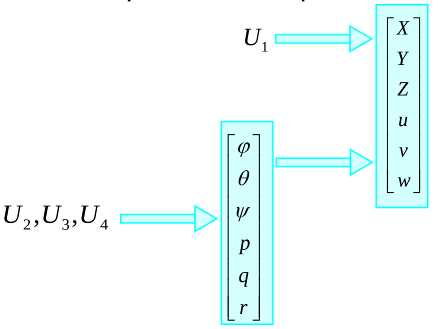
$$\begin{cases} m\dot{u} = -\sin\theta mg \\ m\dot{v} = \sin\varphi\cos\theta mg \\ m\dot{w} = \cos\varphi\cos\theta mg - U_1 \end{cases} \qquad U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$

  - Moments along each axis

$$\begin{cases} J_{xx}\dot{p} = qr(J_{yy} - J_{zz}) - J_R q\Omega_r + U_2 \\ J_{yy}\dot{q} = pr(J_{zz} - J_{xx}) + J_R p\Omega_r + U_3 \\ J_{zz}\dot{r} = U_4 \end{cases} \qquad \begin{aligned} U_2 &= lb(\Omega_4^2 - \Omega_2^2) \\ U_3 &= lb(\Omega_3^2 - \Omega_1^2) \\ U_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{aligned}$$

Group the equations

- Rotational dynamics independent of translational dynamics



- Control cascaded system with a hierarchical controller
  - Calculate desired attitude and thrust out of position error
  - Calculate control moment out of attitude error

Simplified model for attitude control

- **Inner loop:** Attitude control
- **Hover condition:** $\varphi=\theta=\psi=p=q=r=0 \quad \Omega_r=U_2=U_3=U_4=0$

$$\begin{cases} J_{xx}\dot{p} = qr(J_{yy}-J_{zz}) - J_R q\Omega_r + U_2 \\ J_{yy}\dot{q} = pr(J_{zz}-J_{xx}) + J_R p\Omega_r + U_3 \\ J_{zz}\dot{r} = U_4 \end{cases}$$

$$\left\{ J_r \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right.$$

$$\ddot{\varphi} = \frac{1}{I_{xx}} U_2$$

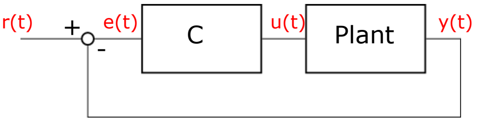
$$\ddot{\theta} = \frac{1}{I_{yy}} U_3$$

$$\ddot{\psi} = \frac{1}{I_{zz}} U_4$$

This model has no coupling
→ We use 3 individual controller

PID controller

- PID control strategy
  - Widely used in industry
  - Generic
  - Non model based
  - Consist of three terms
    - P-Proportional
    - I-Integral
    - D-Derivative
  - Exists different tuning methods
    - E.g. Ziegler-Nichols method
    - Pole placement



$$u(t) = k_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right)$$

PID Attitude control

- Roll subsystem

$$\ddot{\varphi} = \frac{1}{I_{xx}} U_2$$

- P controller can't stabilize the system

$$\ddot{\varphi} = \frac{1}{I_{xx}} k_p (\varphi_{des} - \varphi)$$

- Use a PD controller

$$\ddot{\varphi} = \frac{1}{I_{xx}} k_p ((\varphi_{des} - \varphi) + T_d (\dot{\varphi}_{des} - \dot{\varphi})$$

- Constant oscillation around desired attitude

$$\varphi(t) = a \sin(\sqrt{\frac{k_p}{I_{xx}}} t + \phi) + \varphi_{des}$$

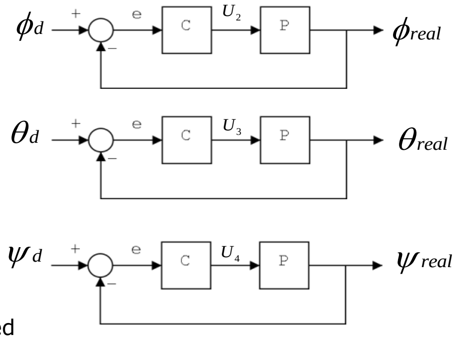
- Underdamped, critically damped or overdamped system

$$\lim_{t \to \infty} \varphi(t) = \varphi_{des}$$

Attitude control design

- We have 3 separate control loops:

$$U_1 = T_{des}$$
$$U_2 = (\varphi_d - \varphi) K_{pRoll} - \dot{\varphi} K_{pRoll} T_{dRoll}$$
$$U_3 = (\theta_d - \theta) K_{pPitch} - \dot{\theta} K_{pPitch} T_{dPitch}$$
$$U_4 = (\psi_d - \psi) K_{pYaw} - \dot{\psi} K_{pYaw} T_{dYaw}$$



- Choose K_p to meet desired convergence rate
- Choose T_d to get a overdamped system

C: controller, P: plant

Control design

- We need to control Roll, Pitch and Yaw
  - We have 3 virtual control inputs (U_2, U_3, U_4)
  - U_1 is virtual thrust control input → additional free input
  - Need to know the propeller speeds
- Recall $T_i = b\Omega_i^2 \quad \Rightarrow \quad \textit{The thrust force}$

$$D_i = d\Omega_i^2 \quad \Rightarrow \quad \textit{The drag moment}$$

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$
$$U_2 = lb(-\Omega_2^2 + \Omega_4^2)$$
$$U_3 = lb(\Omega_1^2 - \Omega_3^2)$$
$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

Control design

- We rewrite the control inputs (U_i) in matrix form then we extract the desired speed to send to the motors:

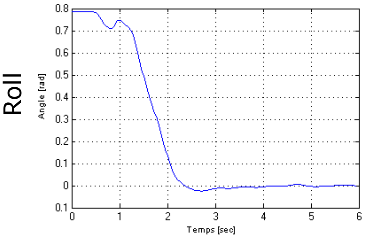
$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ lb & 0 & -lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} \frac{1}{4}\frac{1}{b} & 0 & \frac{1}{2}\frac{1}{lb} & -\frac{1}{4}\frac{1}{d} \\ \frac{1}{4}\frac{1}{b} & -\frac{1}{2}\frac{1}{lb} & 0 & \frac{1}{4}\frac{1}{d} \\ \frac{1}{4}\frac{1}{b} & 0 & -\frac{1}{2}\frac{1}{lb} & -\frac{1}{4}\frac{1}{d} \\ \frac{1}{4}\frac{1}{b} & \frac{1}{2}\frac{1}{lb} & 0 & \frac{1}{4}\frac{1}{d} \end{bmatrix} \\ \begin{array}{cccc} \text{Thrust} & \text{Roll} & \text{Pitch} & \text{Yaw} \end{array} \end{array} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

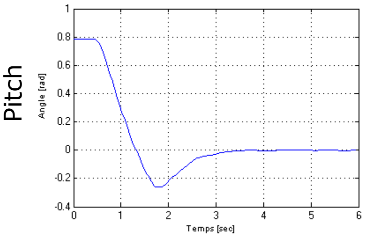
- We see that the (Ω_i) is the sum of the contribution of each of the 4 control loops
- Saturate virtual inputs to get positive motor speeds!

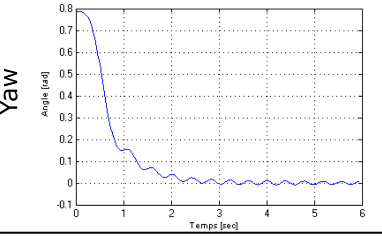
Control design

— Controller simulation

— The nonlinear system has to stabilize the orientation from $\pi/4$ initial condition.





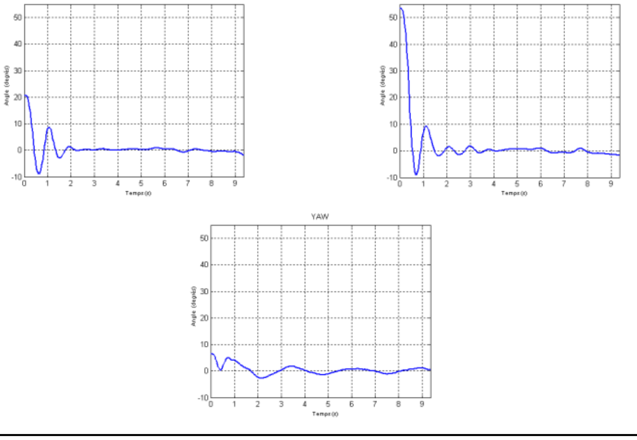


- Roll and Pitch: P=0.8, D=0.4
- Yaw: P=0.8, D=0.5

Control design

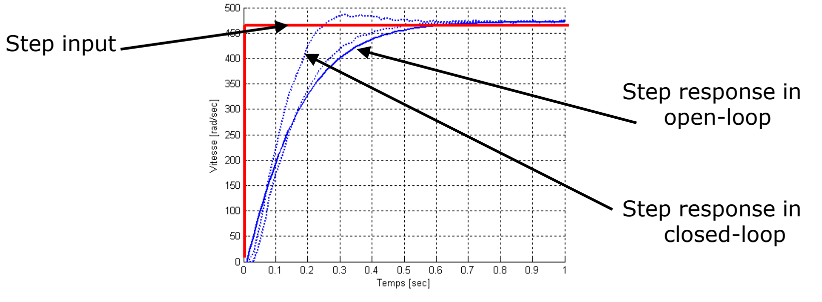
— Experimentation on the system with motor dynamics

— The angles are stabilized in less than 2 seconds, but the system response is too soft. This is partly due to the actuator bandwidth.



Control design

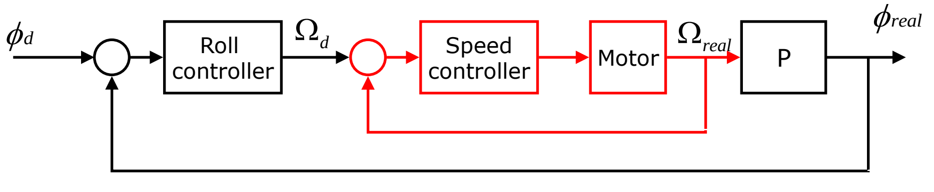
- **Actuator dynamics**
  - The actuators (rotors) have their own dynamics, this means that the propeller speed desired by the controller is not reached instantaneously.
  - If these dynamics are not fast enough, they can have an effect on the stability of the overall system.

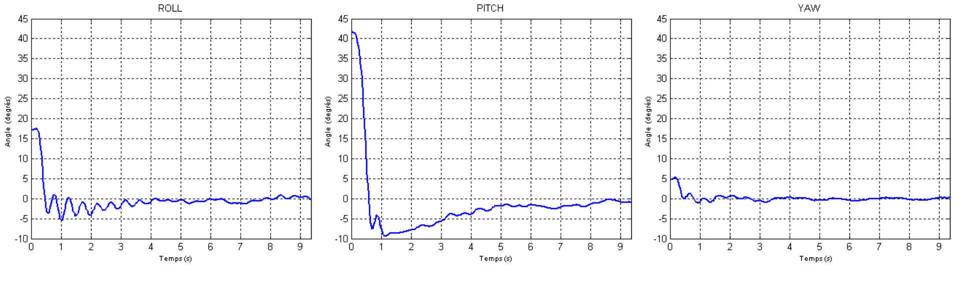


  - **Closing the loop locally on the motor speed, enhances the actuator bandwidth.**

Control design

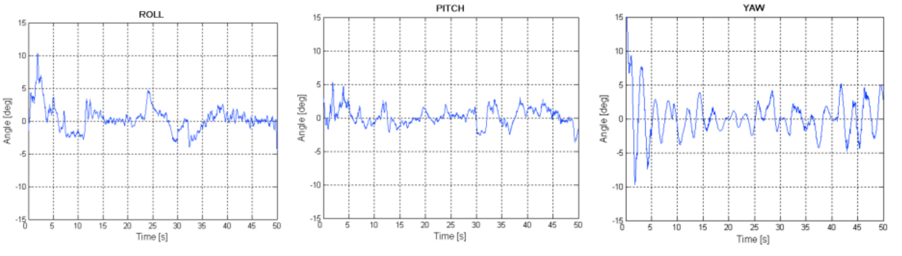
- Adding a speed control loop enhances the overall stability!





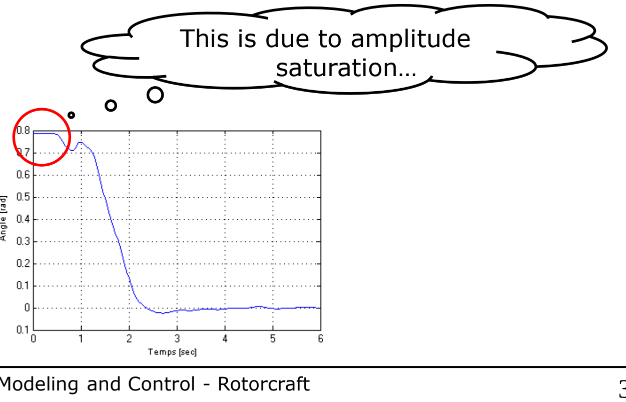
Control testing

- **A real flight...**



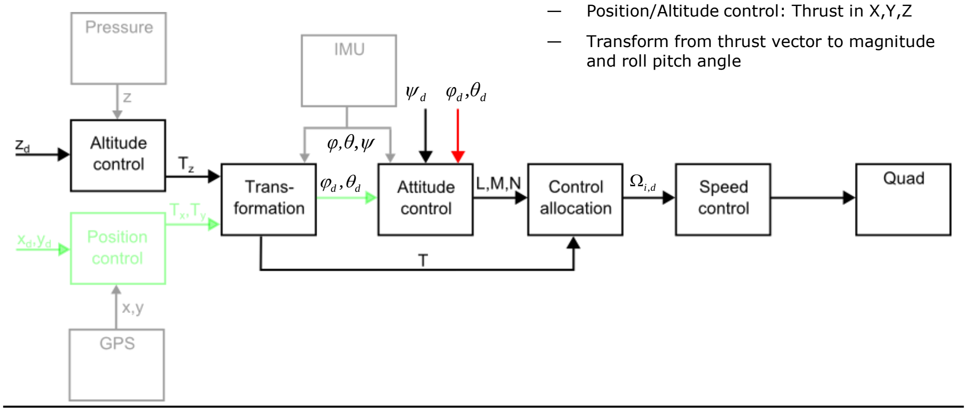
Control limitations

- **The main limitations come from the saturation of the:**
  - Actuation bandwidth (e.g. can't react quicker to disturbances)
  - Actuation amplitude (e.g. can't spin the propellers faster)
    - **A quadrotor with actuators' amplitude saturation is no more controllable!!**



Full control chart

- **A possible control flow**
  - Altitude or full position mode
- Altitude mode
  - Altitude control: Thrust in Z
  - Increase thrust according to orientation
  - Attitude control: Body moments
- Position mode
  - Position/Altitude control: Thrust in X,Y,Z
  - Transform from thrust vector to magnitude and roll pitch angle



Altitude control mode

- Calculate inertial thrust in z direction
  - Inertial height system dynamics

$$\ddot{z} = \frac{1}{m} T_z + g$$

  - Use a PD controller to control the height

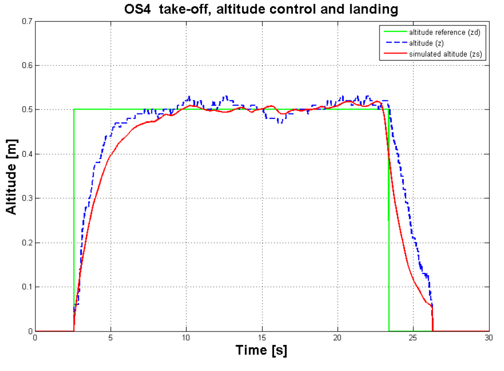
$$T_z = k_p (z_d - z) + k_d (\dot{z}_d - \dot{z}) - mg$$

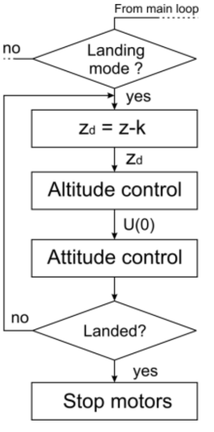
  - Transform to thrust

$$T = \frac{-T_z}{\cos\varphi \cos\theta}$$

Altitude control mode

- **Altitude control**





Position control mode

- Calculate thrust in each direction of the inertial frame
  - Inertial position system dynamics

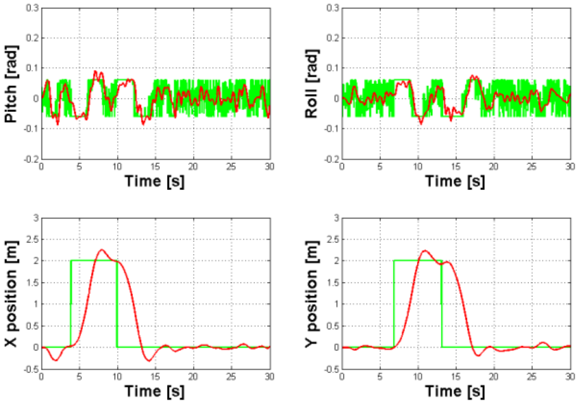
$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

  - Use 3 separate PD controller for each direction
  - Transform to desired thrust, roll and pitch

$$T_d = \sqrt{T_x^2 + T_y^2 + T_z^2} \qquad \frac{1}{\mathrm{T_d}} R^T(z,\psi) \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} -\sin\theta_d \cos\varphi_d \\ \sin\varphi_d \\ \cos\theta_d \cos\varphi_d \end{bmatrix}$$

Full control

- **Position control and Way-point following**
  - The helicopter must tilt to move forward



Nonlinear attitude control

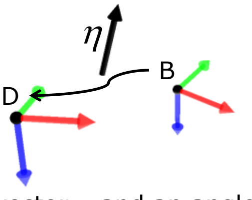
- Linear PD controller only works near hover conditions
  - Neglected body gyroscopic effects
  - Linearized relation between tait bryan angles derivatives and body angular rates

$$\begin{cases} J_{xx}\dot{p} = qr(J_{yy} - J_{zz}) - J_R q\Omega_r + U_2 \\ J_{yy}\dot{q} = pr(J_{zz} - J_{xx}) + J_R p\Omega_r + U_3 \\ J_{zz}\dot{r} = U_4 \end{cases} \qquad \left\{ J_r \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right.$$

- Nonlinear attitude controllers includes these effects
  - E.g. nonlinear hierarchical controller
  - Outer loop: Attitude error defines a desired body angular rate
  - Inner loop: Body angular rate error defines desired moments

Nonlinear attitude control

- Attitude error control
  - Define the attitude error as the rotation between the desired body frame and the current body frame



  - A rotation can be represented by a vector η and an angle μ
  - Desired body angular rate must be parallel to attitude error vector

$$\omega_d = k_p \sin(\frac{\mu}{2})\eta$$

Nonlinear attitude control

- Body angular rates error
  - Substract body gyro effect
  - Use P controller for the body angular rates error

$$\begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} -qr(J_{yy} - J_{zz}) + k_{p1}(p_d - p) \\ -pr(J_{zz} - J_{xx}) + k_{p2}(q_d - q) \\ k_{p3}(r_d - r) \end{bmatrix}$$

REFERENCES

- R. Prouty: **Helicopter Performance, Stability and Control**
- P. Castillo et Al: **Modeling and Control of Mini-Flying Machines**
- M. Krstic & Kokotovic: **Nonlinear and Adaptive Control Design**