Drone modeling

Aerodynamics, Dynamics, Control

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA
ROMANIA

December 11, 2018

Agenda

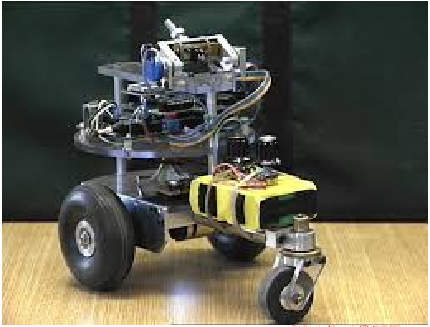
- Basic principles of aerodynamics
- How propellers work
- Drone design and flight principles
- Dynamic modeling
- Control

What have we seen so far

Articulated robots

Other types of robots

Next two lectures



Quadrotor drones

What is a quadrotor?

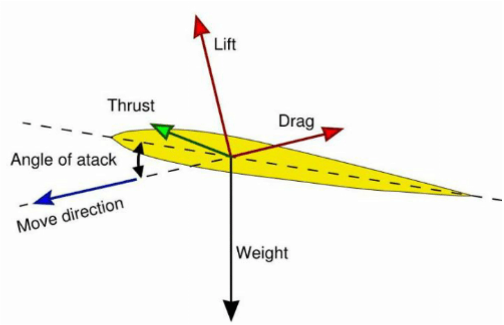
Quadrotor drones

What is a quadrotor?

Why four rotors?

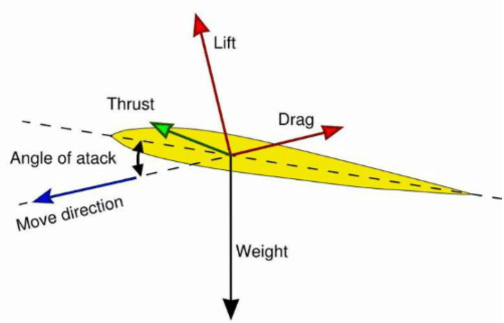
Aerodynamics

Thrust-Lift-Drag



Aerodynamics

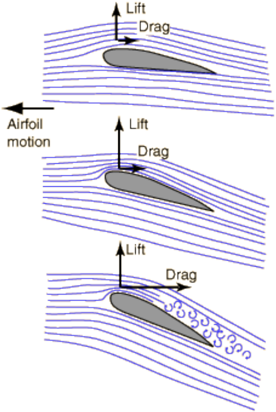
Thrust-Lift-Drag



Thrust, Lift and Drag are related to each other and to the design of the airfoil

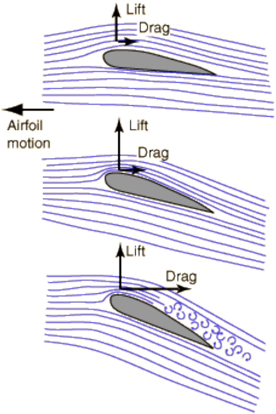
Aerodynamics

Angle of attack



Aerodynamics

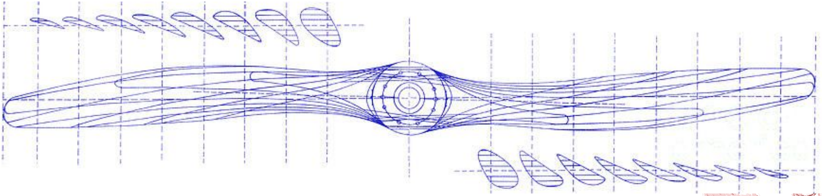
Angle of attack



The ratio of lift to drag are also related to the 'angle of attack'

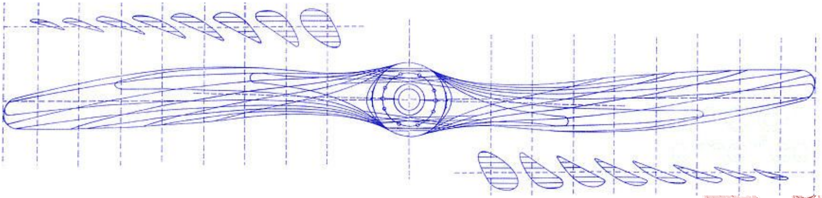
Aerodynamics

Propellers



Aerodynamics

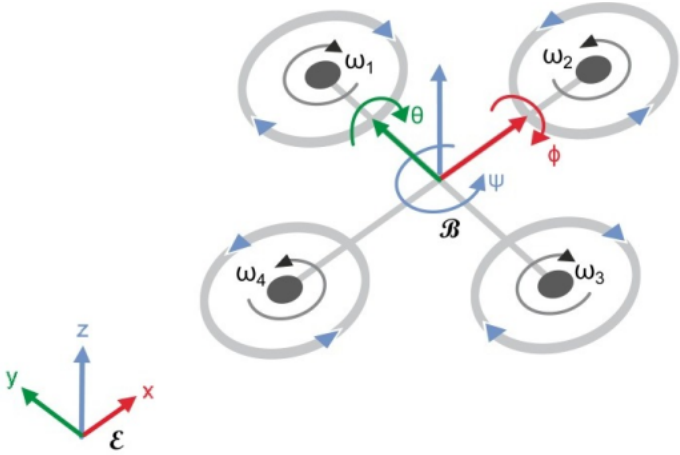
Propellers



A propeller is many airwings with different angles of attack

Quadrotor

Systems of reference and degrees of freedom



Quadrotor

Propeller forces and torques

Connecting with the aerodynamics of airwings and propellers, each rotors produces a lifting force (F_i) and a torque (τ_i).

Quadrotor

Propeller forces and torques

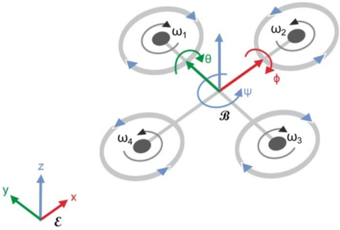
Connecting with the aerodynamics of airwings and propellers, each rotors produces a lifting force (F_i) and a torque (τ_i).

Both of these have to do with the design of the propeller, and are proportional to the square of the angular velocity of the propeller.

$$F_i = b\omega_i^2$$
$$\tau_i = d\omega_i^2$$

Quadrotor

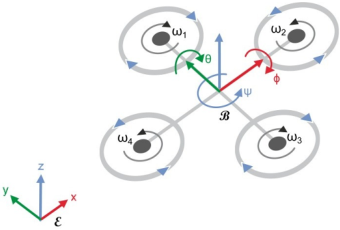
Systems of reference and degrees of freedom



- x, y, z: translation along x, y, z axes of the fixed frame
- ϕ, θ, ψ: rotation around x, y, z axes of the fixed frame

Quadrotor

Systems of reference and degrees of freedom



- x, y, z: translation along x, y, z axes of the fixed frame
- ϕ, θ, ψ: rotation around x, y, z axes of the fixed frame

These are known as euler rotations (roll, pitch, yaw)

$$S = [\xi, \eta]^T\text{, where: } \xi = [x, y, z]^T \text{ and } \eta = [\phi, \theta, \psi]^T$$

Dynamic modeling

Transformation

$$R_B^E(\eta) = \begin{bmatrix} c(\theta)c(\psi) & c(\psi)s(\theta)s(\phi) - c(\phi)s(\psi) & s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) \\ c(\theta)c(\psi) & c(\phi)c(\psi) + s(\theta)s(\phi)s(\psi) & c(\phi)s(\theta)s(\psi) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix}$$

where:

$$c(\theta) = cos(\theta)$$
$$s(\psi) = sin(\psi)$$

Quadrotor

Achieving flight

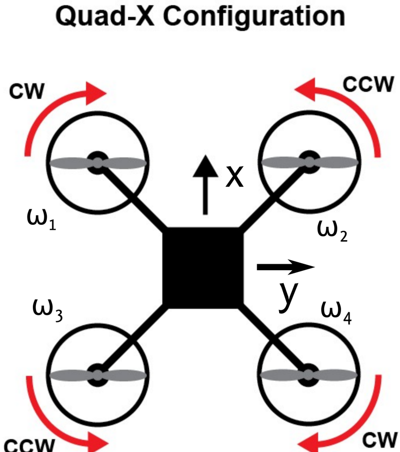


Figure 1

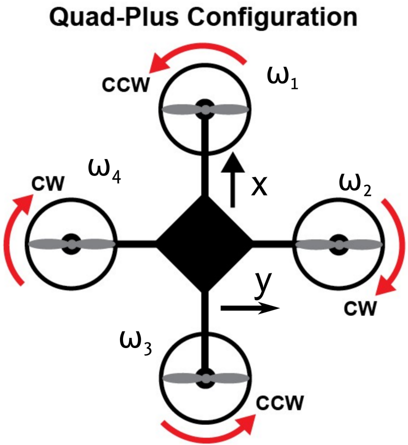


Figure 2

Quadrotor

Achieving flight in X configuration

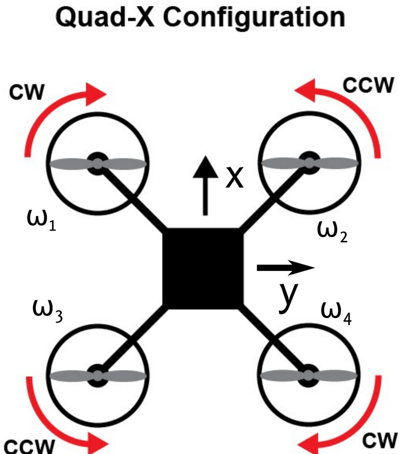


Figure 1

Quadrotor

Achieving flight in X configuration

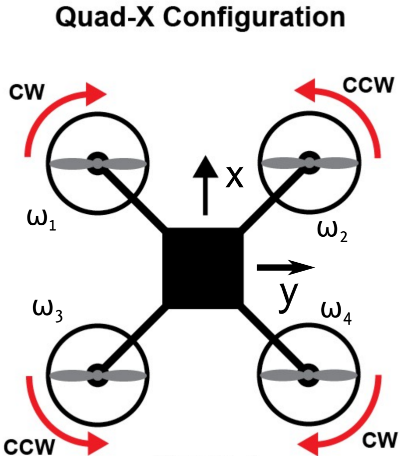


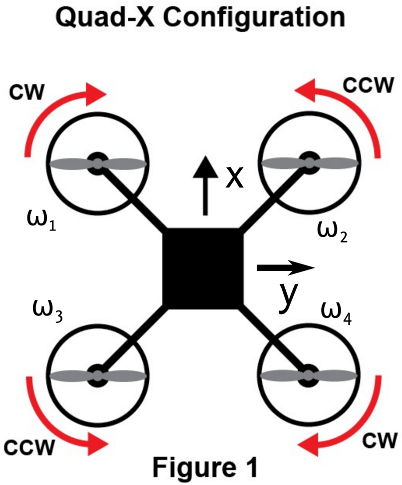
Figure 1

- Translation on z:

$$\omega_1 = \omega_2 = \omega_3 = \omega_4$$

Quadrotor

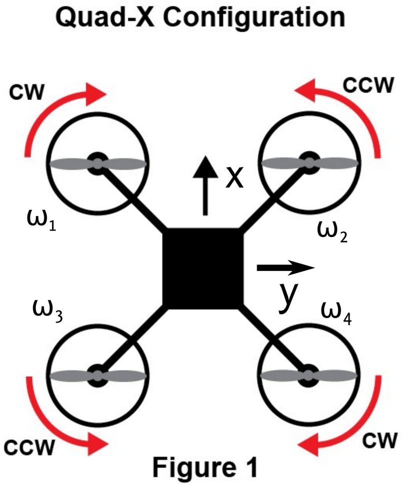
Achieving flight in X configuration



- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$

Quadrotor

Achieving flight in X configuration



- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$

Quadrotor

Achieving flight in X configuration

Quad-X Configuration

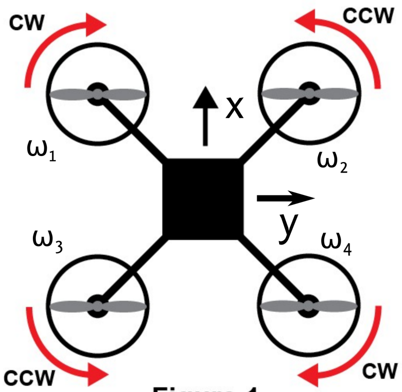


Figure 1

- Translation on z:

  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:

  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:

  $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z:

Quadrotor

Achieving flight in X configuration

Quad-X Configuration

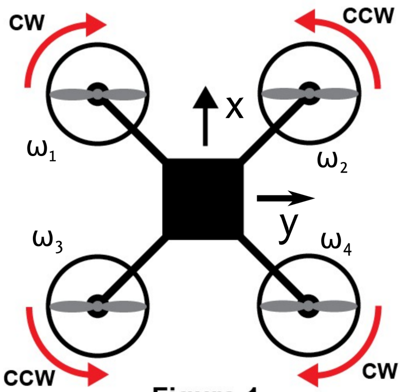


Figure 1

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$

Quadrotor

Achieving flight in X configuration

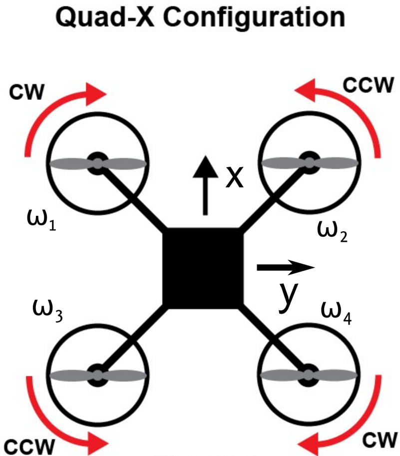


Figure 1

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y:
  $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y

Quadrotor

Achieving flight in X configuration

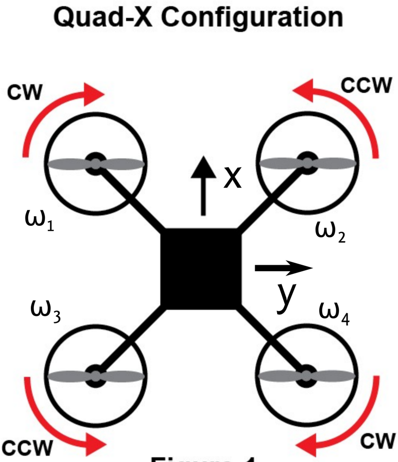


Figure 1

- Translation on z: $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x: $\omega_1 = \omega_3, \omega_2 = \omega_4$
- Rotation around y: $\omega_1 = \omega_2, \omega_3 = \omega_4$
- Rotation around z: $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y
- Translation on y: Rotation around x

Quadrotor

Achieving flight in X configuration

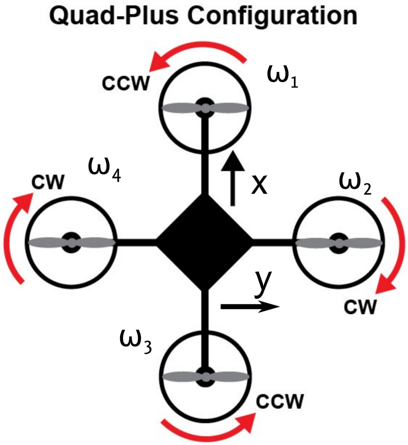


Figure 2

Quadrotor

Achieving flight in X configuration

- Translation on z:

  $\omega_1 = \omega_2 = \omega_3 = \omega_4$

Quad-Plus Configuration

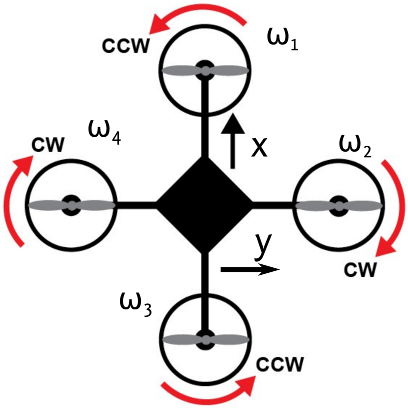


Figure 2

Quadrotor

Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$

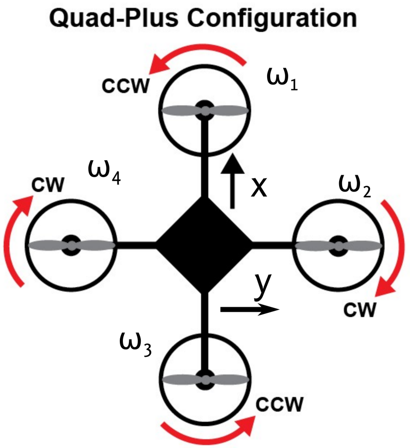


Figure 2

Quadrotor

Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$

Quad-Plus Configuration

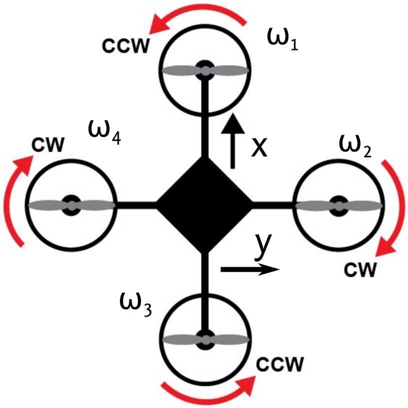
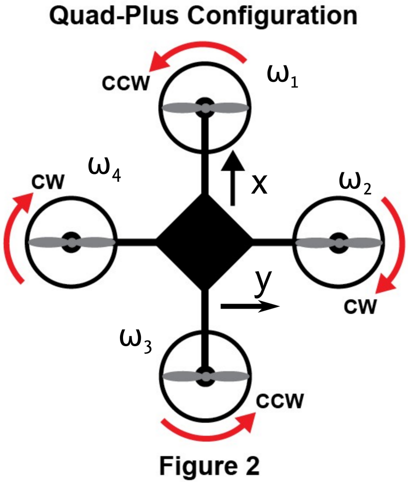


Figure 2

Quadrotor

Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$



Quadrotor

Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y

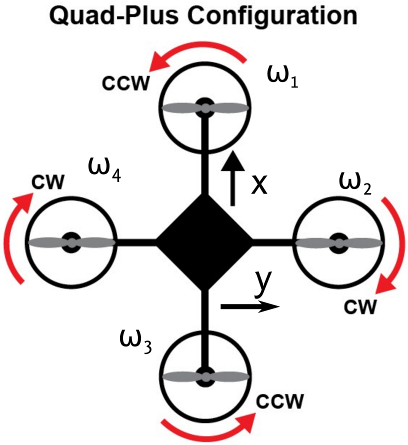


Figure 2

Quadrotor

Achieving flight in X configuration

- Translation on z:
  $\omega_1 = \omega_2 = \omega_3 = \omega_4$
- Rotation around x:
  $\omega_1 = \omega_3, \omega_2 \neq \omega_4$
- Rotation around y:
  $\omega_1 \neq \omega_3, \omega_2 = \omega_4$
- Rotation around z:
  $\omega_1 = \omega_4, \omega_2 = \omega_3$
- Translation on x: Rotation around y
- Translation on y: Rotation around x

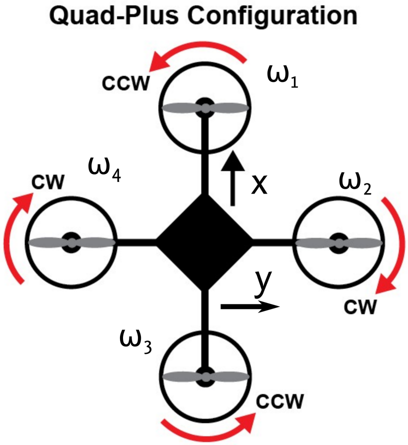
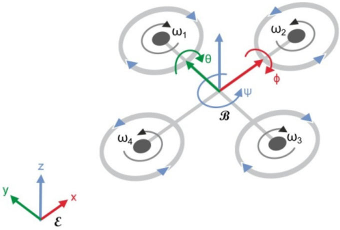


Figure 2

Dynamic modeling

The Lagrangian

Remember:



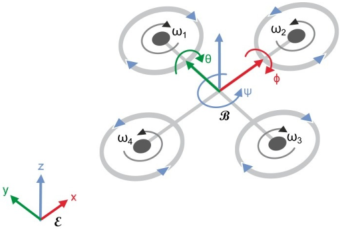
- x, y, z: translation along x, y, z axes of the fixed frame
- ϕ, θ, ψ: rotation around x, y, z axes of the fixed frame

$S = [\xi, \eta]^T$, where: $\xi = [x, y, z]^T$ and $\eta = [\phi, \theta, \psi]^T$

Dynamic modeling

The Lagrangian

Remember:



- x, y, z: translation along x, y, z axes of the fixed frame
- ϕ, θ, ψ: rotation around x, y, z axes of the fixed frame

$S = [\xi, \eta]^T$, where: $\xi = [x, y, z]^T$ and $\eta = [\phi, \theta, \psi]^T$
Therefore:

$$L(S, \dot{S}) = K_{lin} + K_{rot} - P = \frac{1}{2}\left(m\dot{\xi}^T\dot{\xi} + \dot{\eta}^T J(\eta)\dot{\eta}\right) - mgz$$

Dynamic modeling

The Langrangian

$$L(S, \dot{S}) = K_{lin} + K_{rot} - P = \frac{1}{2}\left(m\dot{\xi}^T\dot{\xi} + \dot{\eta}^T J(\eta)\dot{\eta}\right) - mgz$$

Where $J(\eta)$ is the moments of inertia expressed in the fixed frame.

Dynamic modeling

The Langrangian

$$L(S, \dot{S}) = K_{lin} + K_{rot} - P = \frac{1}{2}\left(m\dot{\xi}^T\dot{\xi} + \dot{\eta}^T J(\eta)\dot{\eta}\right) - mgz$$

Where $J(\eta)$ is the moments of inertia expressed in the fixed frame.

By differenciating, we can calculate the equation of motion of the quadrotor

Dynamic modeling

The Langrangian

$$L(S, \dot{S}) = K_{lin} + K_{rot} - P = \frac{1}{2}\left(m\dot{\xi}^T\dot{\xi} + \dot{\eta}^T J(\eta)\dot{\eta}\right) - mgz$$

Where $J(\eta)$ is the moments of inertia expressed in the fixed frame.

By differenciating, we can calculate the equation of motion of the quadrotor

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{S}} - \frac{\partial L}{\partial S} = \begin{bmatrix} F_{transl} \\ F_{rot} \end{bmatrix}$$

Dynamic modeling

The Langrangian

Since the translation kinetic and potential energies depend only on $\xi and \dot{\xi}$ and the rotational kinetic energy only on $\dot{\eta}$, we can break this down to two decoupled systems of equations.

$$\frac{d}{dt}\frac{\partial\left(K_{transl}+P\right)}{\partial\dot{S}}-\frac{\partial\left(K_{transl}+P\right)}{\partial S}=F_{transl}$$

$$\frac{d}{dt}\frac{\partial K_{rot}}{\partial\dot{S}}-\frac{\partial K_{rot}}{\partial S}=F_{rot}$$

Dynamic modeling

The Langrangian

The first equation is easy to differenciate:

$$F_{transl} = m\ddot{\xi} + [0, 0, mg]^t$$

The second one is a bit more 'stiff', due to the moments of inertia:

$$F_{rot} = J(\eta)\ddot{\eta} + \dot{J}(\eta)\dot{\eta} - \frac{1}{2}\frac{d}{d\eta}\left(\dot{\eta}^T J(\eta)\dot{\eta}\right) = J(\eta)\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta}$$

Dynamic modeling

The forces

If we consider a hover flight, with very small changes in orientation, we need to consider a vertical force to achieve this:

$$F_{transl} = [0, 0, U_{coll}]^T$$

Dynamic modeling

The forces

If we consider a hover flight, with very small changes in orientation, we need to consider a vertical force to achieve this:

$$F_{transl} = [0, 0, U_{coll}]^T$$

We need to express this force on the frame attached on the drone, therefore:

$$F_{transl} = R_B^E \, [0, 0, U_{coll}]^T$$

Dynamic modeling

The torques

Since we usually express rotations on the frame attached on the drone, the rotational forces (torques), do not need to be 'transformed' in the drones body frame. Therefore:

$$F_{rot} = [U_\phi, U_\theta, U_\psi]^T$$

Dynamic modeling

Putting it all together

Therefore, if we plug our forces in the dynamic model equation, we have:

$$\ddot{x} = [c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi)]\,\frac{U_{coll}}{m}$$
$$\ddot{y} = [c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi)]\,\frac{U_{coll}}{m}$$
$$\ddot{z} = -g + c(\phi)c(\theta)\frac{U_{coll}}{m}$$
$$\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = J^{-1}(\eta)\left(\begin{bmatrix}U_\phi\\ U_\theta\\ U_\psi\end{bmatrix} - C(\eta,\dot{\eta})\eta\right)$$

Dynamic modeling

Simplifications

Since we consider the hovering motion, this means that the rotations are small, therefore, for α very small we have:

$$cos(\alpha) \approx 1$$
$$sin(\alpha) \approx \alpha$$

Dynamic modeling

Simplifications

Since we consider the hovering motion, this means that the rotations are small, therefore, for α very small we have:

$$cos(\alpha) \approx 1$$
$$sin(\alpha) \approx \alpha$$

The collective force, can be also expressed as $U_{coll} = mg + \Delta U_{coll}$

Dynamic modeling

Simplifications

Since we consider the hovering motion, this means that the rotations are small, therefore, for α very small we have:

$$cos(\alpha) \approx 1$$
$$sin(\alpha) \approx \alpha$$

The collective force, can be also expressed as $U_{coll} = mg + \Delta U_{coll}$ Therefore, our equations of motion become:

$$\ddot{x} = \theta g$$
$$\ddot{y} = -\phi g$$
$$\ddot{z} = \frac{\Delta U_{coll}}{m}$$

$$\ddot{\phi} = \frac{1}{I_x} U_\phi$$
$$\ddot{\theta} = \frac{1}{I_y} U_\theta$$
$$\ddot{\psi} = \frac{1}{I_z} U_\psi$$

Dynamic modeling

What are the control inputs?

We need to consider again how does a drone achieve flight. For the plus configuration:

$$U_{coll} = F_1 + F_2 + F_3 + F_4$$

$$U_\phi = l(F_1 - F_3)$$

$$U_\theta = l(F_2 - F_4)$$

$$U_\psi = \tau_1 + \tau_3 - \tau_2 - \tau_4$$

Quad-Plus Configuration

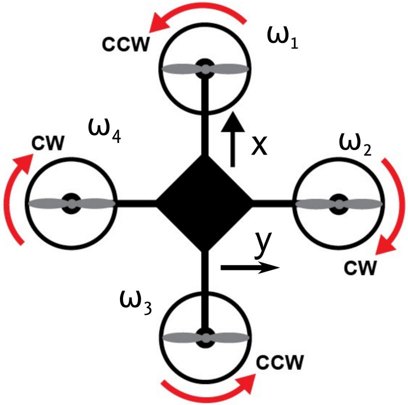


Figure 2

Dynamic modeling

What are the control inputs?

For the cross configurations

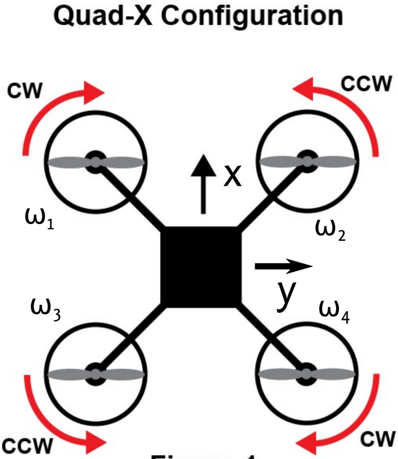


Figure 1

$$U_{coll} = F_1 + F_2 + F_3 + F_4$$

$$U_\phi = \frac{\sqrt{2}}{2} l (F_1 + F_4 - F_2 - F_3)$$

$$U_\theta = \frac{\sqrt{2}}{2} l (F_1 + F_2 - F_3 - F_4)$$

$$U_\psi = \tau_1 + \tau_3 - \tau_2 - \tau_4$$

Quadrotor

Control

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

- We made several assumptions to end up with a linear model

Quadrotor

Control

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

- We made several assumptions to end up with a linear model
- We did not include motor dynamics

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

- We made several assumptions to end up with a linear model
- We did not include motor dynamics
- We did not include aerodynamics

Quadrotor

Control

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

- We made several assumptions to end up with a linear model
- We did not include motor dynamics
- We did not include aerodynamics
- When flying outdoors, there are huge disturbances (wind)

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

- We made several assumptions to end up with a linear model
- We did not include motor dynamics
- We did not include aerodynamics
- When flying outdoors, there are huge disturbances (wind)
- A quadrotor is underactuated

Quadrotor

Control

If we can derive the dynamic model of a quadrotor, why do you think control is difficult?

- We made several assumptions to end up with a linear model
- We did not include motor dynamics
- We did not include aerodynamics
- When flying outdoors, there are huge disturbances (wind)
- A quadrotor is underactuated

For simple slow motions though, we can try to control it with this simple model

Quadrotor

Control

The model we have calculated is already linearized, so we can apply state feedback control. Our states are and inputs are:

$$x = \left[\xi, \dot{\xi}\right]^T$$

$$u = \left[U_{coll}, U_{\phi}, U_{\theta}, U_{\psi}\right]^T$$

The model can be writen in state space as:

$$\dot{x} = Ax + Bu$$

Quadrotor

State feedback

By calculating the matrices **A** and **B**, we can calculate a feedback gain matrix **K**, that will stabilize the system. We do that using standard pole positioning. Our system has 12 states and 4 inputs, so the gain matrix will have dimensions 4×12:

$$u = -Kx$$

Quadrotor

State feedback

By calculating the matrices **A** and **B**, we can calculate a feedback gain matrix **K**, that will stabilize the system. We do that using standard pole positioning. Our system has 12 states and 4 inputs, so the gain matrix will have dimensions 4×12:

$$u = -Kx$$

We then add an integrator to track a specific setpoint, in our case a position in space:

$$u = -Kx + u_o$$
$$u_o = [u_z, u_y, u_x, 0]^T$$

where:

$$u_x = \frac{ki_x}{s}\left(r_x - x\right), u_y = \frac{ki_y}{s}\left(r_y - y\right), u_z = \frac{ki_z}{s}\left(r_z - z\right)$$

Further reading/watching

Very cool TED talk on drone modeling:
https://www.youtube.com/watch?v=w2itwFJCgFQ

Questions?