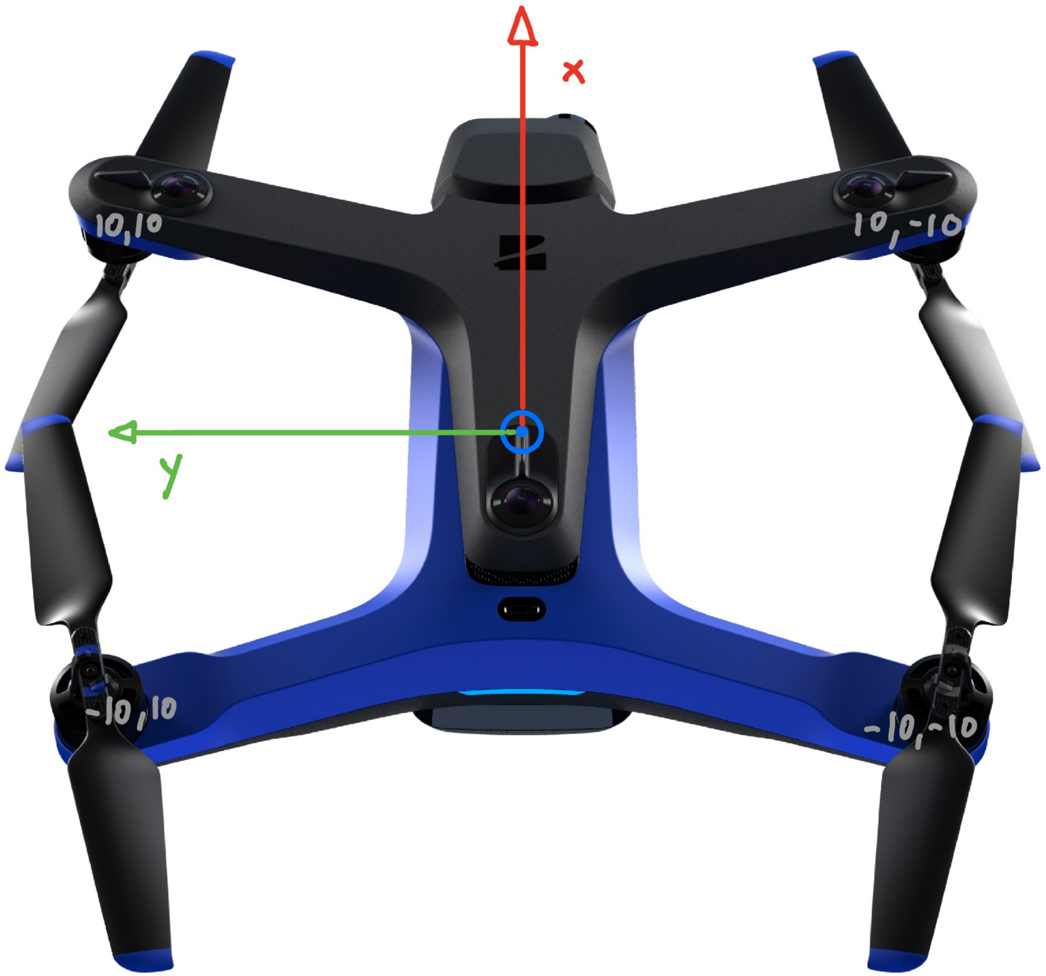
CS 3630!

Lecture 25:
Drone Actions

Actions for Quadrotor drones

1. Definitions
2. Hover
3. Forward Flight
4. Maximum Thrust
5. Drag
6. Kinematics
7. Simulation
8. Code Example
9. Dynamics
10. Gyroscopic effects

Definitions



- **Body frame** *B*: FLU = Forward-Left-Up
- **Navigation Frame** *N: ENU = East-North-Up*

- the vehicle's position $r^n \doteq [x, y, z]^T$,
- its linear velocity $v^n = \dot{r}^n \doteq [u, v, w]^T$,
- the attitude $R_b^n \doteq [i^b, j^b, k^b] \in SO(3)$, a 3×3 rotation matrix the navigation frame $\mathcal{N}$,
- the body angular velocity $\omega^b \doteq [p, q, r]^T$.

Hover

$$F_z^b = \sum_{i=1}^{4} f_i.$$

- Assume weight = 1kg
- g = 10 m/s^2
- Need to provide 10N of thrust!

- $f_i = 0N$ for $i \in 1..4$: downwards acceleration at $-10\frac{m}{s^2}$.
- $f_i = 2.5N$ for $i \in 1..4$: stable hover $0\frac{m}{s^2}$.
- $f_i = 5N$ for $i \in 1..4$: upwards acceleration at $10\frac{m}{s^2}$.

Forward Flight

$$F^n = R_b^n \begin{bmatrix} 0 \\ 0 \\ F_z^b \end{bmatrix} = \hat{z}_b^n F_z^b$$

$$F^n = \begin{bmatrix} 0 \\ \sin\theta \\ \cos\theta \end{bmatrix} F_z^b.$$

- Force is always up *in body frame*
- Need to rotate to navigation frame
- Thrust is always aligned with body z-axis expressed in navigation frame
- Maintain altitude:

$$\cos\theta F_z^b == 10N.$$

- That means:

$$F_y^n = \sin\theta F_z^b = \sin\theta \frac{10N}{\cos\theta} = \tan\theta \cdot 10N$$

Maximum Thrust

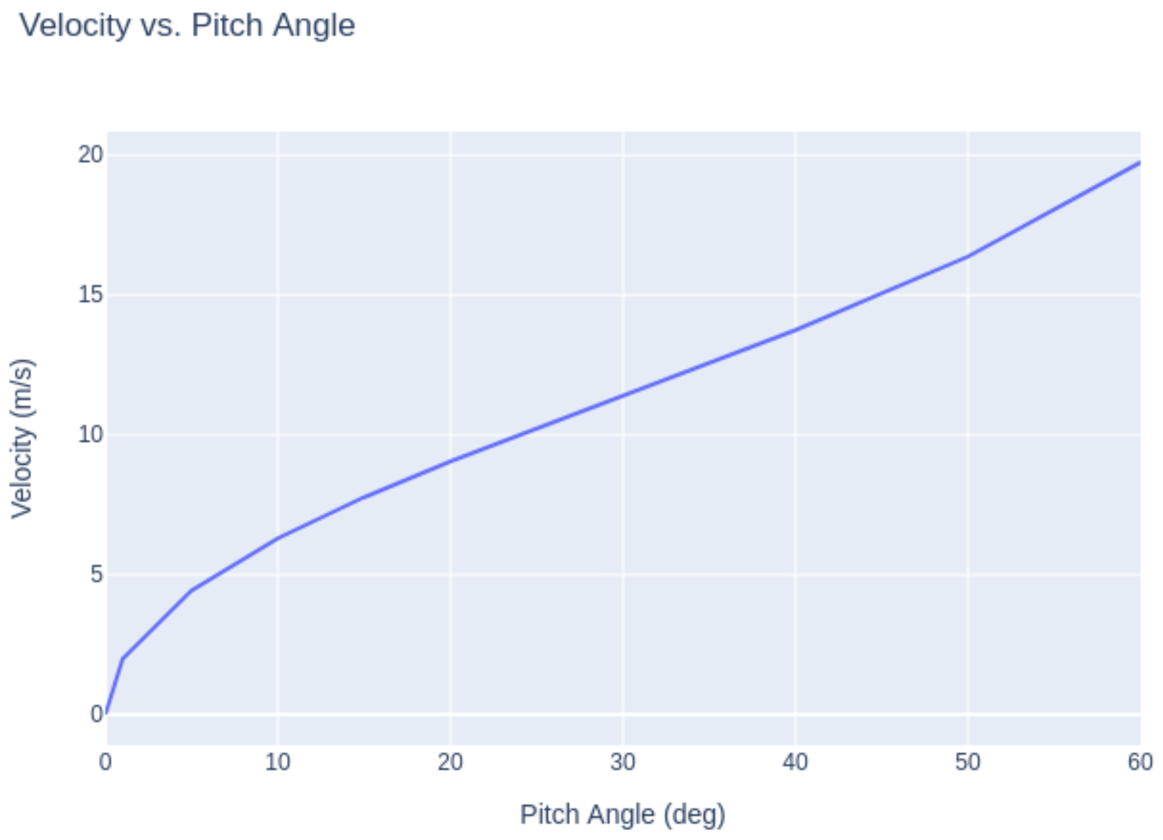
- Assume maximum thrust is 5N per rotor, i.e., 20N total!
- That means:

$$F_z^b = \frac{10N}{\cos\theta} \leq 20N \rightarrow \cos\theta \geq 0.5 \rightarrow -60^\circ \leq \theta \leq 60^\circ$$

Drag

- Air resistance increases quadratically with velocity
- Max velocity 20 m/s
- In book: calculate velocity while maintaining level flight:

$$v \approx 15\sqrt{\tan\theta}$$



Kinematics (position)

- Easy kinematics: derivative of position is velocity

$$\dot{\boldsymbol{r}}^n = \boldsymbol{v}^n.$$

Angular velocity

- First, let us define **angular velocity**
- A three-vector defined in the body frame
- Axis-angle interpretation: velocity $\|\omega^b\|$ around axis ω^b
- Example: 10 degrees/sec around Y-axis (pitch down):

$$\omega^b = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} 10 \frac{\pi}{180}.$$

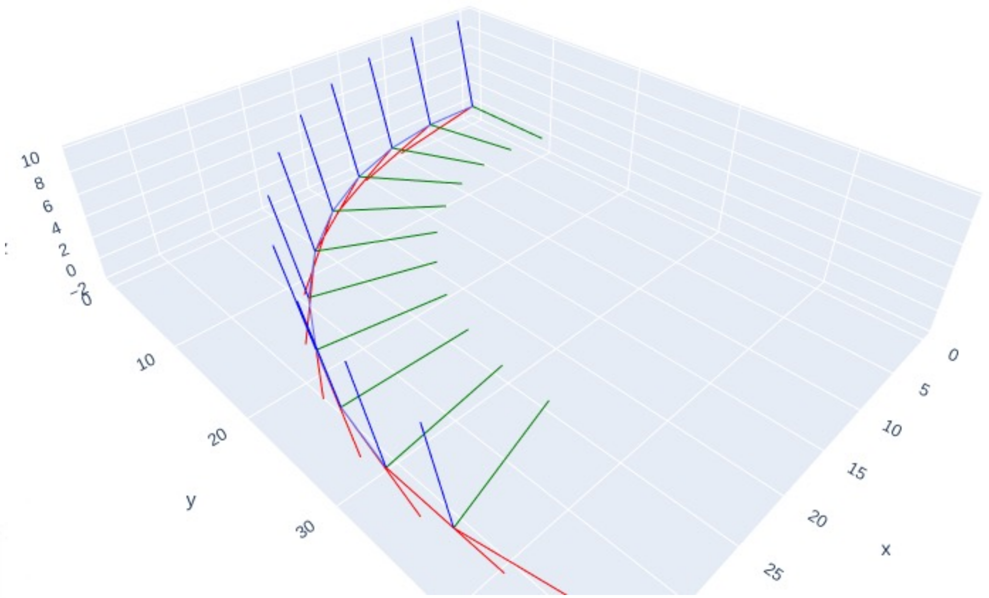
Kinematics (attitude)

- Not so easy: derivative of attitude is…

$$\dot{R}_b^n = R_b^n \hat{\omega}^b.$$

$$\hat{\omega}^b \doteq \begin{bmatrix} & -\omega_z^b & \omega_y^b \\ \omega_z^b & & -\omega_x^b \\ -\omega_y^b & \omega_x^b & \end{bmatrix}.$$

Simulation



- Forward integration
- Position:

$$r_{k+1}^{n} = r_{k}^{n} + d_{k+1}^{k}[v^{n}(t), \Delta t]$$

- Attitude:

$$R_{b,k+1}^{n} = R_{b,k}^{n} R_{k+1}^{k}[\omega^{b}(t), \Delta t]$$

Simulation (position)

- Approximation of exact integration = integration scheme
- Simplest: *Euler's method:*

$$r_{k+1}^{n} = r_{k}^{n} + d_{k+1}^{k}[v^{n}(t), \Delta t]$$

$$d_{k+1}^{k}[v^{n}(t), \Delta t] \approx v^{n}(t_k)\Delta t$$

$$\boxed{r_{k+1}^{n} = r_{k}^{n} + v^{n}(t_k)\Delta t}$$

- Other schemes: backward Euler, trapezoidal method..

Simulation (attitude)

- A bit more complex
- Euler equivalent = Rodrigues' formula with constant angular velocity

$$R^n_{b,k+1} = R^n_{b,k} R^k_{k+1}[\omega^b(t), \Delta t]$$

$$R^k_{k+1}[\omega^b(t), \Delta t] \approx I + \sin\theta K + (1 - \cos\theta) K^2$$

$$\theta = \|\omega^b_k\| \Delta t$$

$$K = \hat{\omega}^b_k / \|\omega^b_k\|$$

Simulation (attitude, first order)

- For small rotation angles, we can approximate the Euler step:

$$R_{k+1}^{k}[\omega^b(t), \Delta t] \approx I + \sin\theta K \approx I + \hat{\omega}_k^b \Delta t = \begin{bmatrix} 1 & -\omega_z^b \Delta t & \omega_y^b \Delta t \\ \omega_z^b \Delta t & 1 & -\omega_x^b \Delta t \\ -\omega_y^b \Delta t & \omega_x^b \Delta t & 1 \end{bmatrix}$$

- So, finally:

$$R_{b,k+1}^{n} = R_{b,k}^{n} R_{k+1}^{k}[\omega^b(t), \Delta t] \approx R_{b,k}^{n} \begin{bmatrix} 1 & -\omega_z^b \Delta t & \omega_y^b \Delta t \\ \omega_z^b \Delta t & 1 & -\omega_x^b \Delta t \\ -\omega_y^b \Delta t & \omega_x^b \Delta t & 1 \end{bmatrix}$$

Code Example: terminal fwd. velocity + yaw

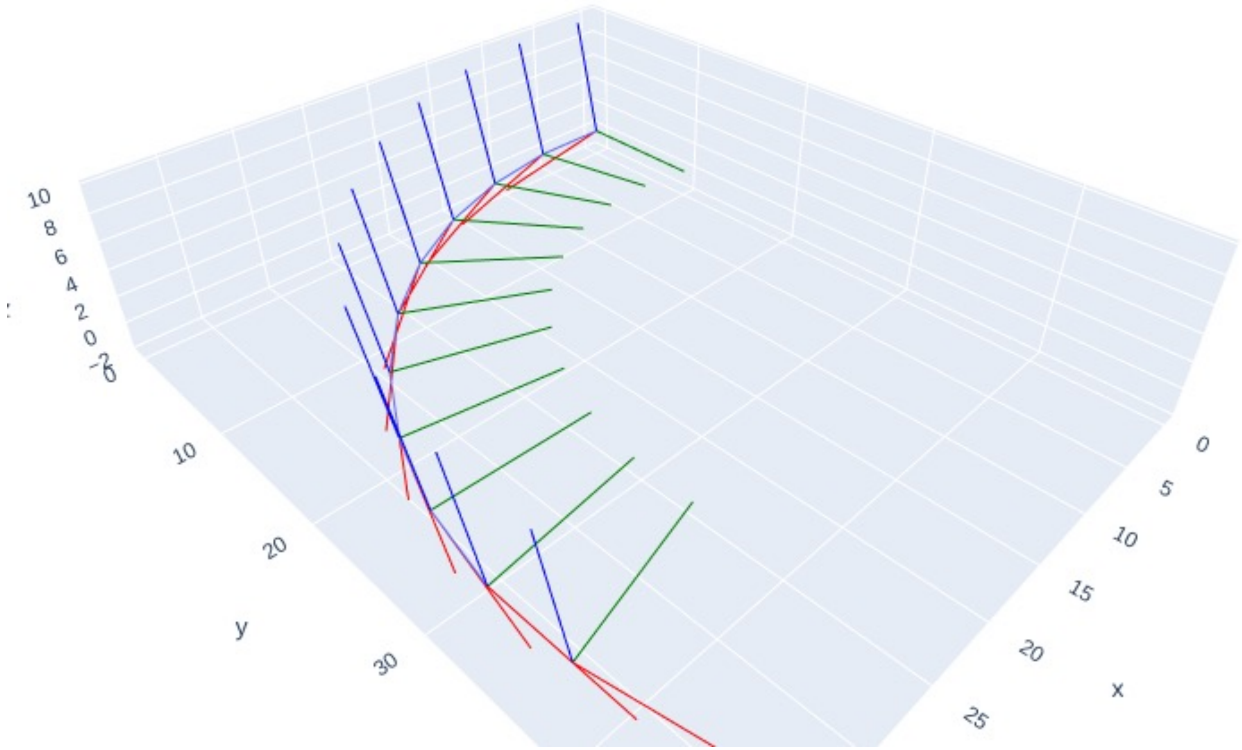
- Position:

$$r^n_{k+1} = r^n_k + d^k_{k+1}[v^n(t), \Delta t]$$

- Attitude:

$$R^n_{b,k+1} = R^n_{b,k} R^k_{k+1}[\omega^b(t), \Delta t]$$

```
# integrate forward
for k in range(K):
    vn = nRb[:,:,k] @ vb
    vn[2] = 0
    rn[:,k+1] = rn[:,k] + vn * delta_t
    delta_R = gtsam.Rot3.Expmap(wb * delta_t)
    nRb[:,:,k+1] = nRb[:,:,k] @ delta_R.matrix()
```



Dynamics

- Dynamics is harder
- Positional is easy:

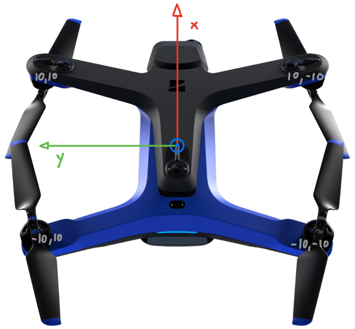
$$F^n = m\dot{v}^n$$

- Attitude is harder:

$$\tau^b \approx I\dot{\omega}^b$$

- Mass: resists force
- Same resistance in all axes

- 3x3 Inertial matrix *I*
- How much do we resist torque?
- Typical: small-small-big



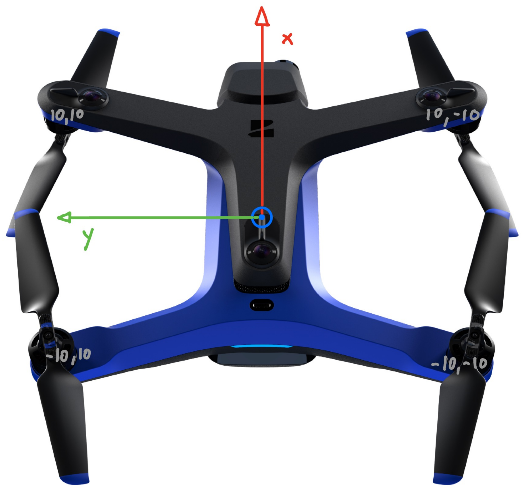
Creating forces and torques

- Linear force:

$$F_z^b = \sum_i f_i.$$

- Angular torque:

$$\tau^b = \begin{bmatrix} l(f_1 - f_2 - f_3 + f_4) \\ l(f_1 + f_2 - f_3 - f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}$$



Gyroscopic effects*

- For large angular velocities:

$$\tau^b = I\dot{\omega}^b - \omega^b \times I\omega^b$$

Summary

1. Definitions
2. Hover
3. Forward Flight
4. Maximum Thrust
5. Drag
6. Kinematics
7. Simulation
8. Code Example
9. Dynamics
10. Gyroscopic effects