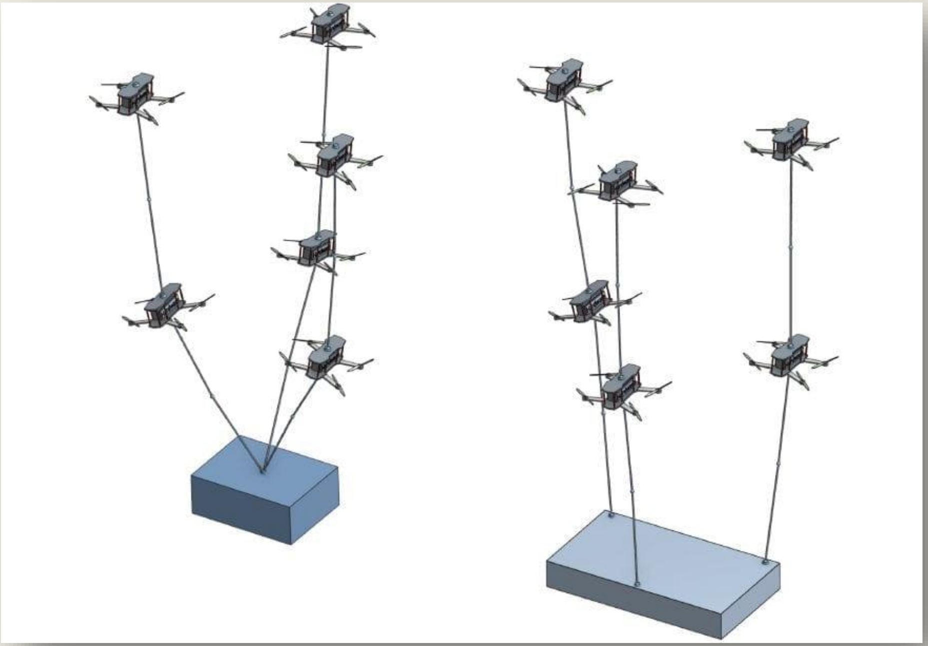
MODELING A QUADCOPTER AND SIMPLE PENDULUM

Christian Llanes

Motivation

- **Goal:** Autonomous drone cooperation
  - *Building structures*
  - *Transporting packages*

- **Requires:** Robust control algorithms or model the dynamics for the coupled system.



[Muhammad Usama, “Stack Approach to Cooperative Drone Lifting”, DroneBelow]

Outline

- Quadcopter Dynamics
- Quadcopter and Pendulum Free Body Diagram
- Lagrangian Mechanics
- Quadcopter and Pendulum Equations of Motion
- Simulation Results

Quadcopter dynamics

States

$$\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{E}, \boldsymbol{\omega}]^T$$
$$\boldsymbol{p} = [p_x, p_y, p_z]$$
$$\boldsymbol{v} = [v_x, v_y, v_z]$$
$$\boldsymbol{E} = [\phi, \theta, \psi]$$
$$\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$$

Controls

$$\boldsymbol{u} = [\tau, M_x, M_y, M_z]^T$$

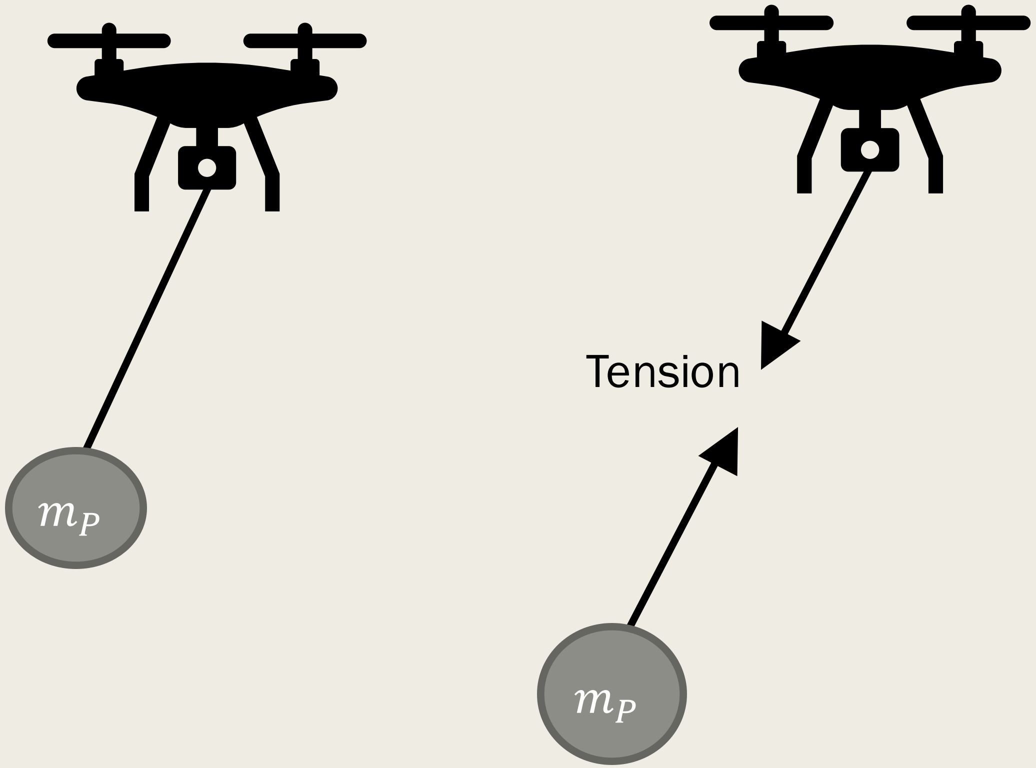
Equations of Motion for a Rigid Body

$$\dot{\boldsymbol{p}} = \boldsymbol{v}$$
$$\dot{\boldsymbol{v}} = g\vec{e}_z - \frac{1}{m}\boldsymbol{R}_B^E \tau \vec{b}_z$$
$$\dot{\boldsymbol{E}} = \boldsymbol{\Gamma}(\boldsymbol{E})^{-1}\omega$$
$$\dot{\omega} = \boldsymbol{J}^{-1}(M - \omega \times \boldsymbol{J}\omega)$$

Expanded Equations of Motion

$$\dot{p}_x = v_x$$
$$\dot{p}_y = v_y$$
$$\dot{p}_z = v_z$$
$$\dot{v}_x = -\frac{\tau}{m}[\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi]$$
$$\dot{v}_y = -\frac{\tau}{m}[\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi]$$
$$\dot{v}_z = g - \frac{\tau}{m}[\cos\phi\cos\theta]$$
$$\dot{\phi} = \omega_x + \omega_y\sin\phi\tan\theta + \omega_z\cos\phi\tan\theta$$
$$\dot{\theta} = \omega_y\cos\phi - \omega_z\sin\phi$$
$$\dot{\psi} = \omega_y\sin\phi\sec\theta + \omega_z\cos\phi\sec\theta$$
$$\dot{\omega}_x = \frac{(I_y - I_z)}{I_x}\omega_y\omega_z + \frac{M_x}{I_x}$$
$$\dot{\omega}_y = \frac{(I_z - I_x)}{I_y}\omega_x\omega_z + \frac{M_y}{I_y}$$
$$\dot{\omega}_z = \frac{(I_x - I_y)}{I_z}\omega_x\omega_y + \frac{M_z}{I_z}$$

A Simple Pendulum attached to a Quadcopter



Newtonian Mechanics

$$\sum F_P = m_P \vec{a}_P$$

$$\sum F_Q = m_Q \vec{a}_Q$$

$$\sum M_Q = J \vec{\alpha}_Q$$

Quite a complex method of computing the Equations of Motion…

Lagrangian Mechanics

- Hamilton's principle
  - *The trajectory of* N *generalized coordinates* $\boldsymbol{q} = (q_1, q_2, q_3, \dots, q_N)$ *for time* $t \in [t_1, t_2]$ *is a stationary point of the action functional which is equivalent to the variation is equal to zero:*

$$\delta \int_{t_1}^{t_2} \mathcal{L}\big(t, \boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\big) dt = 0$$

  - *The Lagrangian function for a system is* $\mathcal{L}\big(t, \boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\big) = T\big(t, \boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\big) - V(\boldsymbol{q}(t), t)$ *where* T *is the kinetic energy and* V *is the potential energy of the system.*
- Calculus of Variations
  - *The variation of the Lagrangian function* $\mathcal{L}\big(t, \boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\big)$ *is zero if the generalized coordinates* $q_i(t)$ *and generalized velocities* $\dot{q}_i(t)$ *satisfy the Euler-Lagrange equations:*

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0$$

Lagrangian Mechanics

- For systems with non-conservative forces that are not potentials.
  - *Introduce virtual work* $\delta W = \sum Q_i \delta q_i$ *from generalized forces* Q_i *for the generalized coordinates* $\boldsymbol{q} = (q_1, q_2, q_3, \dots, q_N)$*. The extended Hamilton's principle is therefore,*

$$\int_{t_1}^{t_2} (\delta \mathcal{L} + \delta W) dt = 0$$

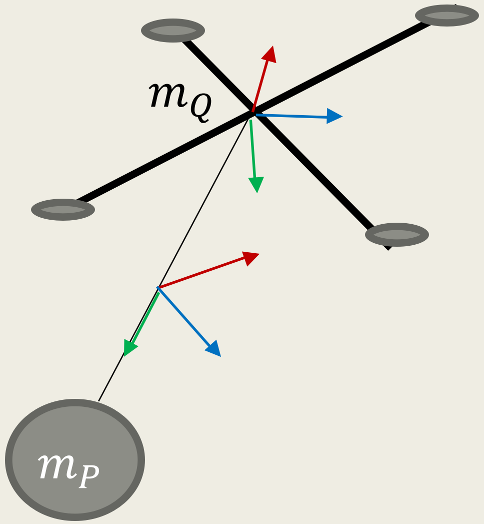
The variation can only be zero if the integrand is zero which is equivalent to:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = Q_i$$

The generalized forces and moments are projections of the non-conservative forces in the generalized coordinates for P *particles:*

$$Q_i = \sum_{j=1}^{P} \vec{F}_j \cdot \frac{\partial \vec{v}_j}{\partial \dot{q}_j}$$

A Simple Pendulum attached to a Quadcopter



- ❑ Quadcopter frame 1-2-3 (roll-pitch-yaw)
- ❑ Pendulum frame 1-2 (roll-pitch)

- ➢ Position of quadrotor, $\boldsymbol{r}_Q$
- ➢ Velocity of quadrotor, $\boldsymbol{v}_Q = \frac{d\boldsymbol{r}_Q}{dt} = \{v_x, v_y, v_z\}$
- ➢ Position of pendulum, $\boldsymbol{r}_P = \boldsymbol{r}_Q + DCM^T(\alpha,\beta)\begin{bmatrix}0\\0\\L\end{bmatrix} = \begin{pmatrix} L\,\mathrm{Cos}[\alpha[t]]\,\mathrm{Sin}[\beta[t]] + x[t] \\ -L\,\mathrm{Sin}[\alpha[t]] + y[t] \\ L\,\mathrm{Cos}[\alpha[t]]\,\mathrm{Cos}[\beta[t]] + z[t] \end{pmatrix}$
- ➢ Velocity of pendulum, $\boldsymbol{v}_P = \frac{d\boldsymbol{r}_P}{dt} = \begin{pmatrix} x'[t] - L\,\mathrm{Sin}[\alpha[t]]\,\mathrm{Sin}[\beta[t]]\,\alpha'[t] + L\,\mathrm{Cos}[\alpha[t]]\,\mathrm{Cos}[\beta[t]]\,\beta'[t] \\ y'[t] - L\,\mathrm{Cos}[\alpha[t]]\,\alpha'[t] \\ z'[t] - L\,\mathrm{Cos}[\beta[t]]\,\mathrm{Sin}[\alpha[t]]\,\alpha'[t] - L\,\mathrm{Cos}[\alpha[t]]\,\mathrm{Sin}[\beta[t]]\,\beta'[t] \end{pmatrix}$
- ➢ Assume quadrotor is symmetric so that moment of inertia tensor $J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$

Lagrangian Mechanics for the Quadrotor and Simple Pendulum

- Define the 8 generalized coordinates of the system:
  - $\boldsymbol{q} = (x, y, z, \phi, \theta, \psi, \alpha, \beta)$
- Define the kinetic and potential energy functions:

$$T = \frac{1}{2} m_P \boldsymbol{v}_P^T \boldsymbol{v}_P + \frac{1}{2} m_Q \boldsymbol{v}_Q^T \boldsymbol{v}_Q + \frac{1}{2} \boldsymbol{\omega}_Q^T \boldsymbol{J} \boldsymbol{\omega}_Q$$

$$V = -m_P g z_P - m_Q g z_Q$$

- The Lagrangian is then,

$$\mathcal{L} = T - V = \frac{1}{2} m_P \boldsymbol{v}_P^T \boldsymbol{v}_P + \frac{1}{2} m_Q \boldsymbol{v}_Q^T \boldsymbol{v}_Q + \frac{1}{2} \boldsymbol{\omega}_Q^T \boldsymbol{J} \boldsymbol{\omega}_Q + m_P g z_P + m_Q g z_Q$$

Lagrange's Equation

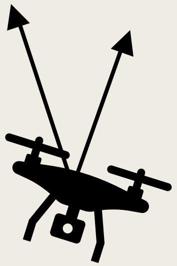
- Generalized coordinates are : $[x, y, z, \phi, \theta, \psi, \alpha, \beta]$
- Generalized velocities: $[v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \dot{\alpha}, \dot{\beta}]$
- Use Mathematica to solve the 8 Euler-Lagrange equations for the time derivative of the generalized velocities.

```
LEq1 = Simplify[(D[D[T, x'[t]], t] - D[T, x[t]] + D[V, x[t]])[[1]][[1]] == F.vQuad1 /. KDE /. KDED];
LEq2 = Simplify[(D[D[T, y'[t]], t] - D[T, y[t]] + D[V, y[t]])[[1]][[1]] == F.vQuad2 /. KDE /. KDED];
LEq3 = Simplify[(D[D[T, z'[t]], t] - D[T, z[t]] + D[V, z[t]])[[1]][[1]] == F.vQuad3 /. KDE /. KDED];
LEq4 = Simplify[(D[D[T, ϕ'[t]], t] - D[T, ϕ[t]] + D[V, ϕ[t]])[[1]][[1]] == (M.wQuad1) [[1]] /. KDE /. KDED];
LEq5 = Simplify[(D[D[T, θ'[t]], t] - D[T, θ[t]] + D[V, θ[t]])[[1]][[1]] == (M.wQuad2 )[[1]] /. KDE /. KDED];
LEq6 = Simplify[(D[D[T, ψ'[t]], t] - D[T, ψ[t]] + D[V, ψ[t]])[[1]][[1]] ==  (M.wQuad3)[[1]] /. KDE /. KDED];
LEq7 = Simplify[(D[D[T, α'[t]], t] - D[T, α[t]] + D[V, α[t]])[[1]][[1]] == 0 /. KDE /. KDED];
LEq8 = Simplify[(D[D[T, β'[t]], t] - D[T, β[t]] + D[V, β[t]])[[1]][[1]] == 0 /. KDE /. KDED];
LEqs = Simplify[Solve[{LEq1, LEq2, LEq3, LEq4, LEq5, LEq6, LEq7, LEq8}, {u1'[t], u2'[t], u3'[t], u4'[t], u5'[t], u6'[t], u7'[t], u8'[t]}]]
```

```
{{u1'[t] → 1/(mQ^2 (mP + mQ)) (-1/16 mQ Thrust (Sin[ϕ[t]] (-8 mP Cos[ψ[t]] Sin[2 α[t]] Sin[β[t]] + (14 mP + 16 mQ - 2 mP Cos[2 α[t]] + mP Cos[2 (α[t] - β[t])] + 2 mP Cos[2 β[t]] + mP Cos[2 (α[t] + β[t])]) Sin[ψ[t]]) +
    Cos[ϕ[t]] Sin[θ[t]] ((14 mP + 16 mQ - 2 mP Cos[2 α[t]] + mP Cos[2 (α[t] - β[t])] + 2 mP Cos[2 β[t]] + mP Cos[2 (α[t] + β[t])]) Cos[ψ[t]] + 8 mP Sin[2 α[t]] Sin[β[t]] Sin[ψ[t]]) -
    8 mP Cos[α[t]]^2 (Cos[θ[t]] Cos[ϕ[t]] Sin[2 β[t]] + Sin[β[t]]^2 (Cos[ϕ[t]] Cos[ψ[t]] Sin[θ[t]] + Sin[ϕ[t]] Sin[ψ[t]]))) + L mP mQ^2 Cos[α[t]] Sin[β[t]] u7[t]^2 + L mP mQ^2 Cos[α[t]]^3 Sin[β[t]] u8[t]^2),

  u2'[t] → -1/(2 mQ (mP + mQ)) (Thrust (mP Cos[β[t]] Cos[θ[t]] Cos[ϕ[t]] Sin[2 α[t]] + Cos[ϕ[t]] Sin[θ[t]] (mP Cos[ψ[t]] Sin[2 α[t]] Sin[β[t]] + (mP + 2 mQ + mP Cos[2 α[t]]) Sin[ψ[t]]) - Sin[ϕ[t]] ((mP + 2 mQ + mP Cos[2 α[t]]) Cos[ψ[t]] - mP Sin[2 α[t]] Sin[β[t]] Sin[ψ[t]])) +
    2 L mP mQ Sin[α[t]] u7[t]^2 + 2 L mP mQ Cos[α[t]]^2 Sin[α[t]] u8[t]^2),

  u3'[t] → 1/(4 mQ (mP + mQ)) (4 g mP mQ + 4 g mQ^2 - 3 mP Thrust Cos[θ[t]] Cos[ϕ[t]] - 4 mQ Thrust Cos[θ[t]] Cos[ϕ[t]] + mP Thrust Cos[α[t]]^2 Cos[θ[t]] Cos[ϕ[t]] + mP Thrust Cos[β[t]]^2 Cos[θ[t]] Cos[ϕ[t]] + mP Thrust Cos[α[t]]^2 Cos[β[t]]^2 Cos[θ[t]] Cos[ϕ[t]] -
    mP Thrust Cos[θ[t]] Cos[ϕ[t]] Sin[α[t]]^2 - mP Thrust Cos[β[t]]^2 Cos[θ[t]] Cos[ϕ[t]] Sin[α[t]]^2 - mP Thrust Cos[θ[t]] Cos[ϕ[t]] Sin[β[t]]^2 - mP Thrust Cos[α[t]]^2 Cos[θ[t]] Cos[ϕ[t]] Sin[β[t]]^2 +
    mP Thrust Cos[θ[t]] Cos[ϕ[t]] Sin[α[t]]^2 Sin[β[t]]^2 + mP Thrust Cos[ϕ[t]] Cos[ψ[t]] Sin[2 β[t]] Sin[θ[t]] + mP Thrust Cos[α[t]]^2 Cos[ϕ[t]] Cos[ψ[t]] Sin[2 β[t]] Sin[θ[t]] -
    mP Thrust Cos[ϕ[t]] Cos[ψ[t]] Sin[α[t]]^2 Sin[2 β[t]] Sin[θ[t]] + 2 mP Thrust Cos[β[t]] Cos[ψ[t]] Sin[2 α[t]] Sin[ϕ[t]] - 2 mP Thrust Cos[β[t]] Cos[ϕ[t]] Sin[2 α[t]] Sin[θ[t]] Sin[ψ[t]] +
    mP Thrust Sin[2 β[t]] Sin[ϕ[t]] Sin[ψ[t]] + mP Thrust Cos[α[t]]^2 Sin[2 β[t]] Sin[ϕ[t]] Sin[ψ[t]] - mP Thrust Sin[α[t]]^2 Sin[2 β[t]] Sin[ϕ[t]] Sin[ψ[t]] + 4 L mP mQ Cos[α[t]] Cos[β[t]] u7[t]^2 + 4 L mP mQ Cos[α[t]]^3 Cos[β[t]] u8[t]^2),

  u4'[t] → (Mx + (Iyy - Izz) u5[t] × u6[t])/Ixx, u5'[t] → (My + (-Ixx + Izz) u4[t] × u6[t])/Iyy, u6'[t] → (Mz + (Ixx - Iyy) u4[t] × u5[t])/Izz,

  u7'[t] → -(Sin[α[t]] (Thrust (Cos[β[t]] Cos[θ[t]] Cos[ϕ[t]] + Cos[ϕ[t]] Sin[θ[t]] (Cos[ψ[t]] Sin[β[t]] + Cot[α[t]] Sin[ψ[t]]) + Sin[ϕ[t]] (-Cos[ψ[t]] Cot[α[t]] + Sin[β[t]] Sin[ψ[t]])) + L mQ Cos[α[t]] u8[t]^2))/(L mQ),

  u8'[t] → (Thrust Sec[α[t]] (-Cos[θ[t]] Cos[ϕ[t]] Sin[β[t]] + Cos[β[t]] (Cos[ϕ[t]] Cos[ψ[t]] Sin[θ[t]] + Sin[ϕ[t]] Sin[ψ[t]])))/(L mQ) + 2 Tan[α[t]] u7[t] × u8[t]}}
```

Simulation in MATLAB

(f_x, f_y, f_z)



$\rightarrow (\theta_{cmd}, \phi_{cmd}, T_{cmd})$

- Designing a controller:
  - $f_x = k_p \tanh(\sigma(p_{xdes} - p_x)) - k_v\, v_x$
  - $f_y = k_p \tanh(\sigma(p_{ydes} - p_y)) - k_v\, v_y$
  - $f_z = k_p \tanh(\sigma(p_{zdes} - p_z)) - (m_Q + m_P)g - k_v\, v_z$

  - $\omega_{x_{cmd}} = k_{p_{rollrate}}(\phi_{cmd} - \phi)$
  - $\omega_{y_{cmd}} = k_{p_{pitchrate}}(\theta_{cmd} - \theta)$
  - $\omega_{z_{cmd}} = -k_{p_{yawrate}}\psi$

  - $M_x = k_{p_{roll}}(\omega_{x_{cmd}} - \omega_x)$
  - $M_y = k_{p_{pitch}}(\omega_{y_{cmd}} - \omega_y)$
  - $M_z = k_{p_{yaw}}(\omega_{z_{cmd}} - \omega_z)$

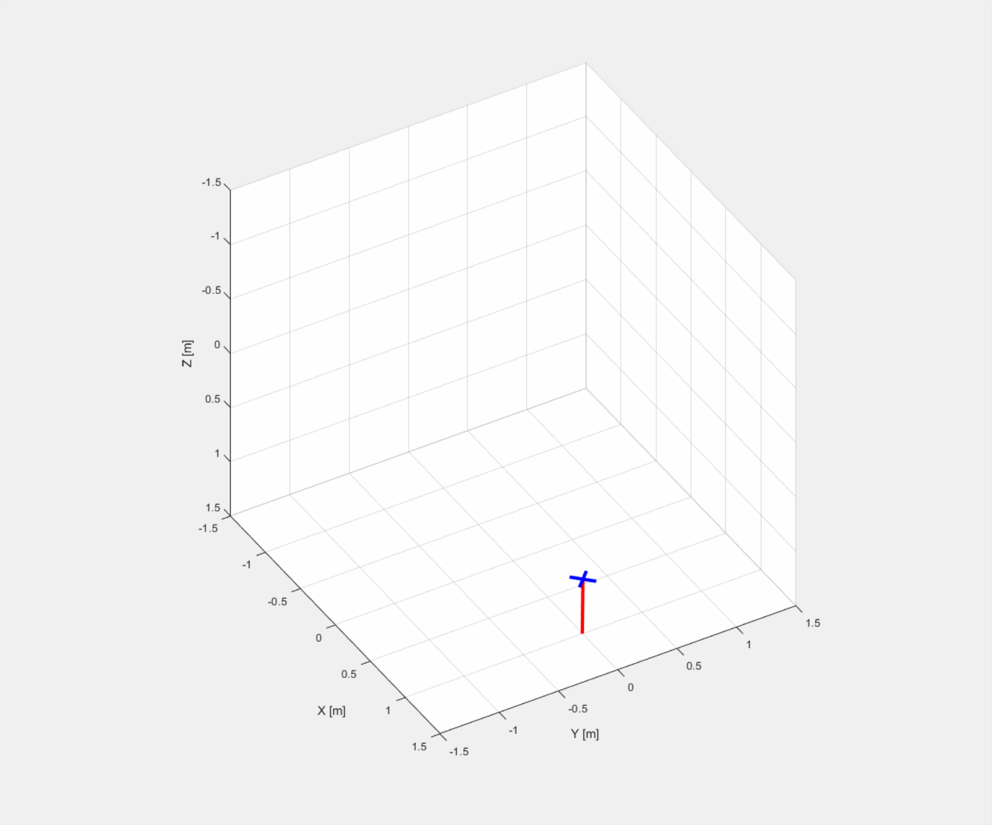
Simulation in MATLAB

- Implement the dynamic update equations with the controller from the previous slide.

- For the following simulations assume, $m_Q = 1kg, \mathrm{L} = 0.5\mathrm{m}$

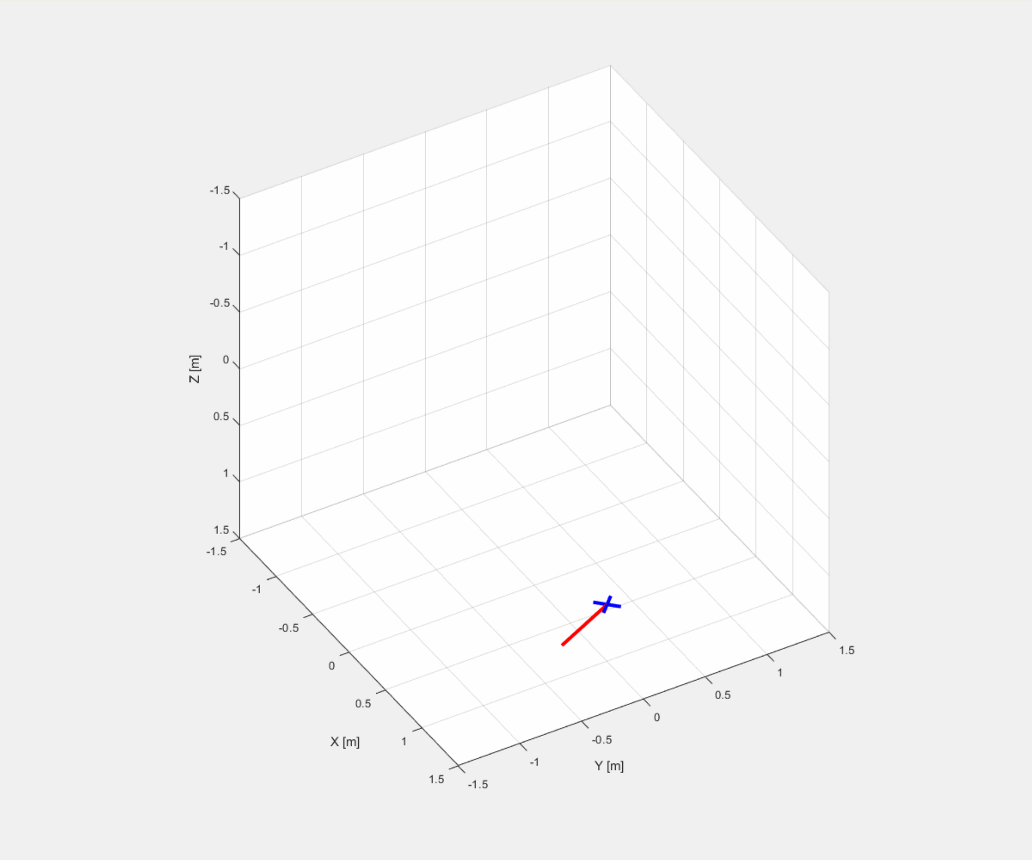
Simulation in MATLAB

- Elliptical trajectory with, $m_P = 0.1kg, \alpha(0) = 0, \beta(0) = 0$



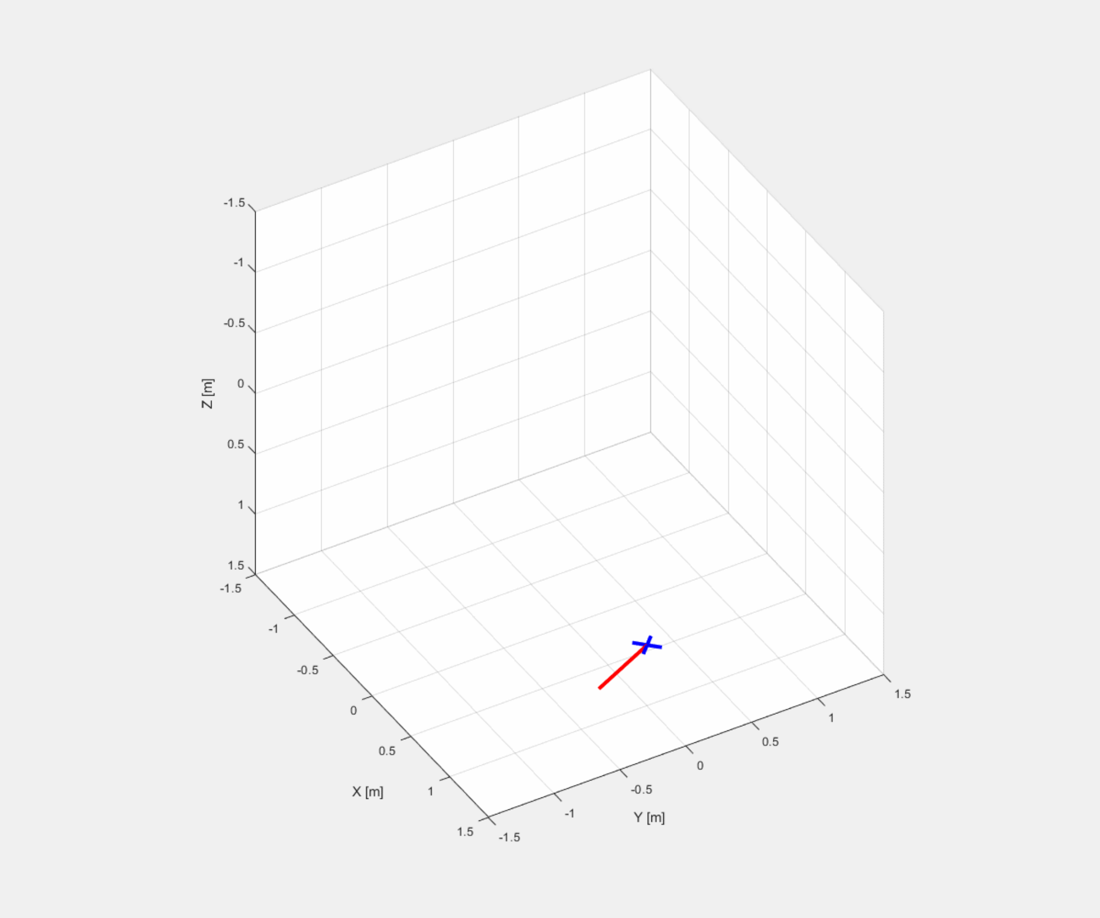
Simulation in MATLAB

- Elliptical trajectory with, $m_P = 0.1kg, \alpha(0) = 1.2rad, \beta(0) = 1.2rad$



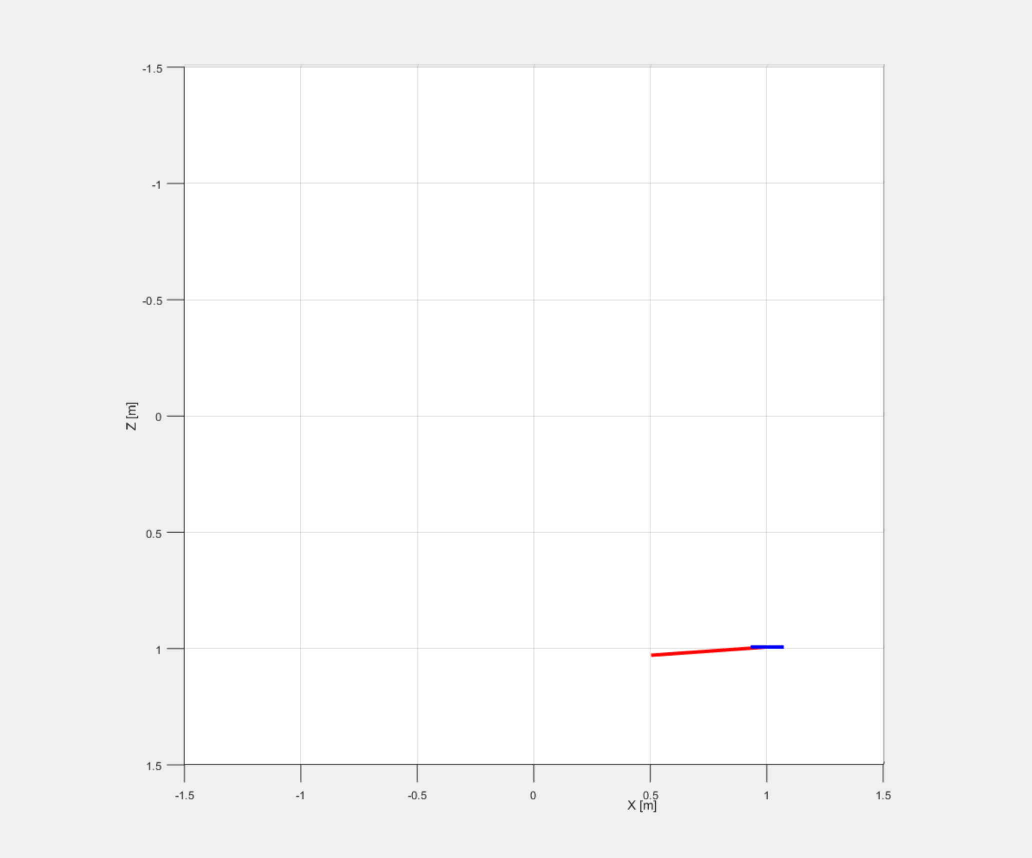
Simulation in MATLAB

- Elliptical trajectory with, $m_P = 0.5kg, \alpha(0) = 1.2rad, \beta(0) = 1.2rad$



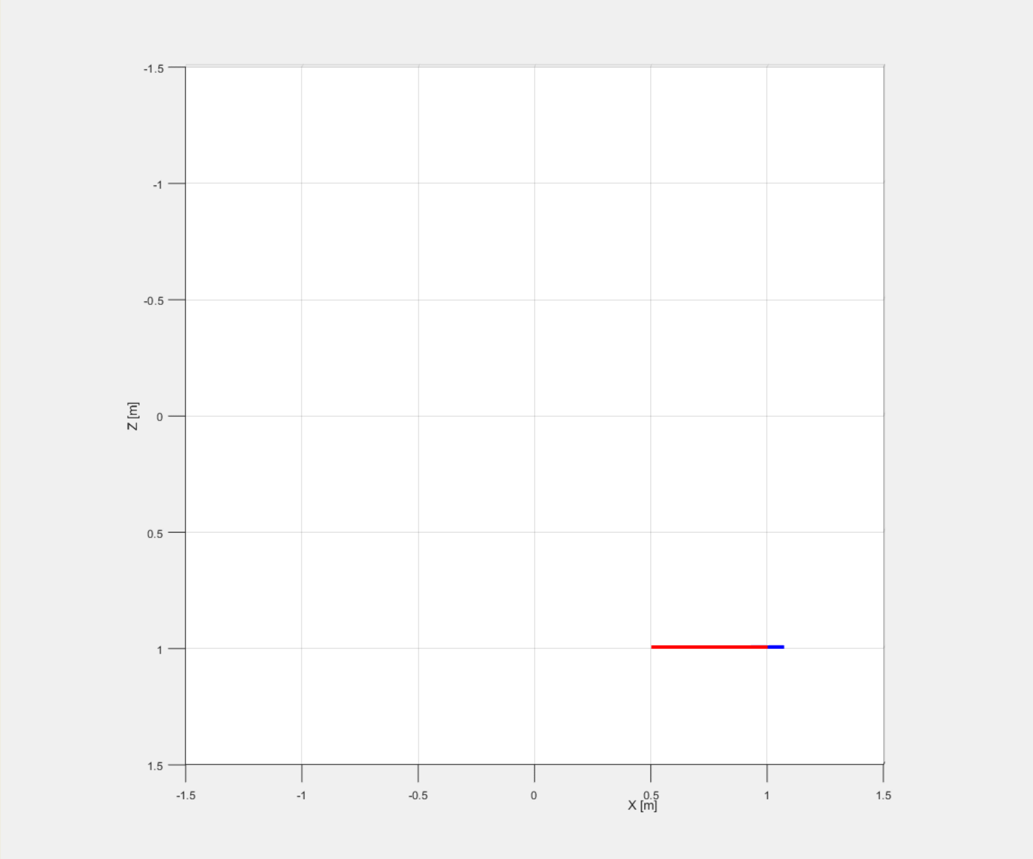
Simulation in MATLAB

- Static setpoint with, $m_P = 0.5kg, \alpha(0) = 0, \beta(0) = -1.5rad$



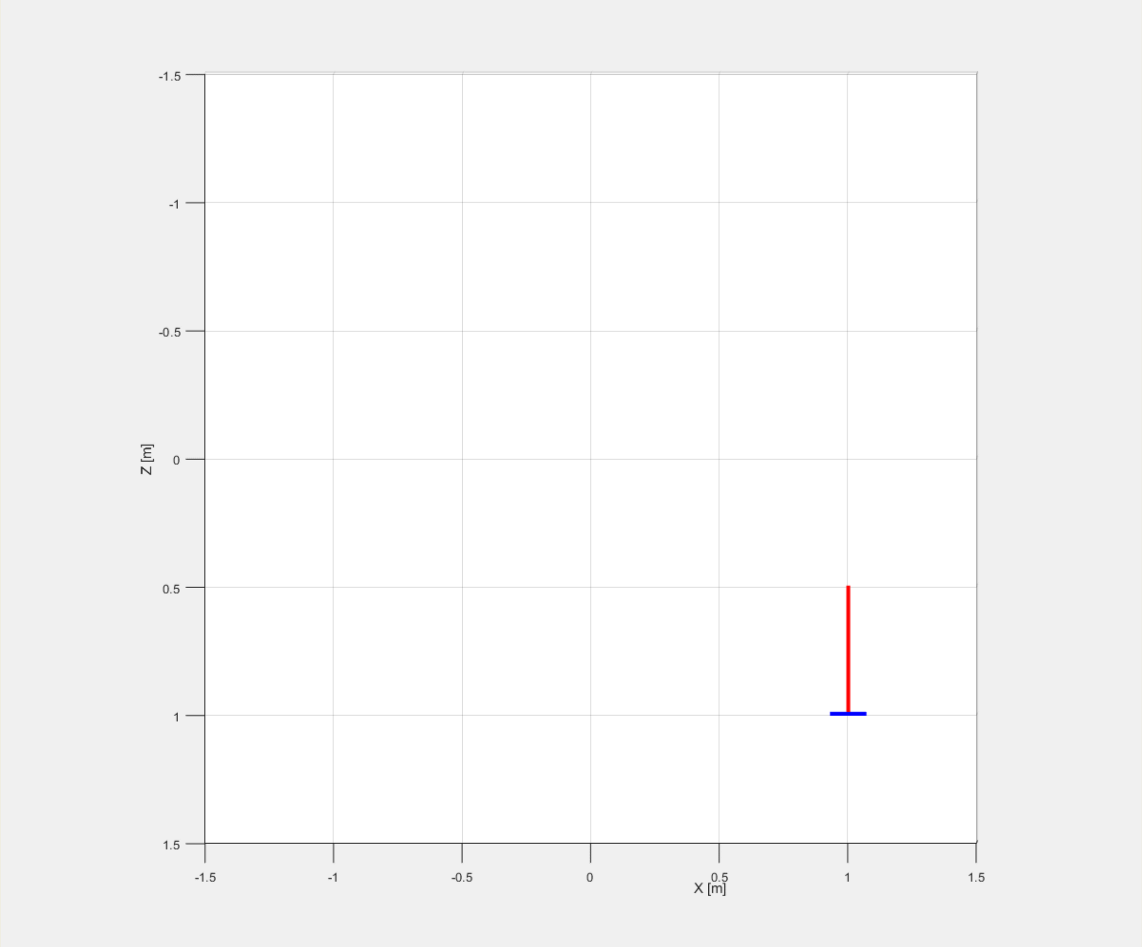
Simulation in MATLAB

- Static setpoint with, $m_P = 1.0kg, \alpha(0) = 0, \beta(0) = -1.571rad$



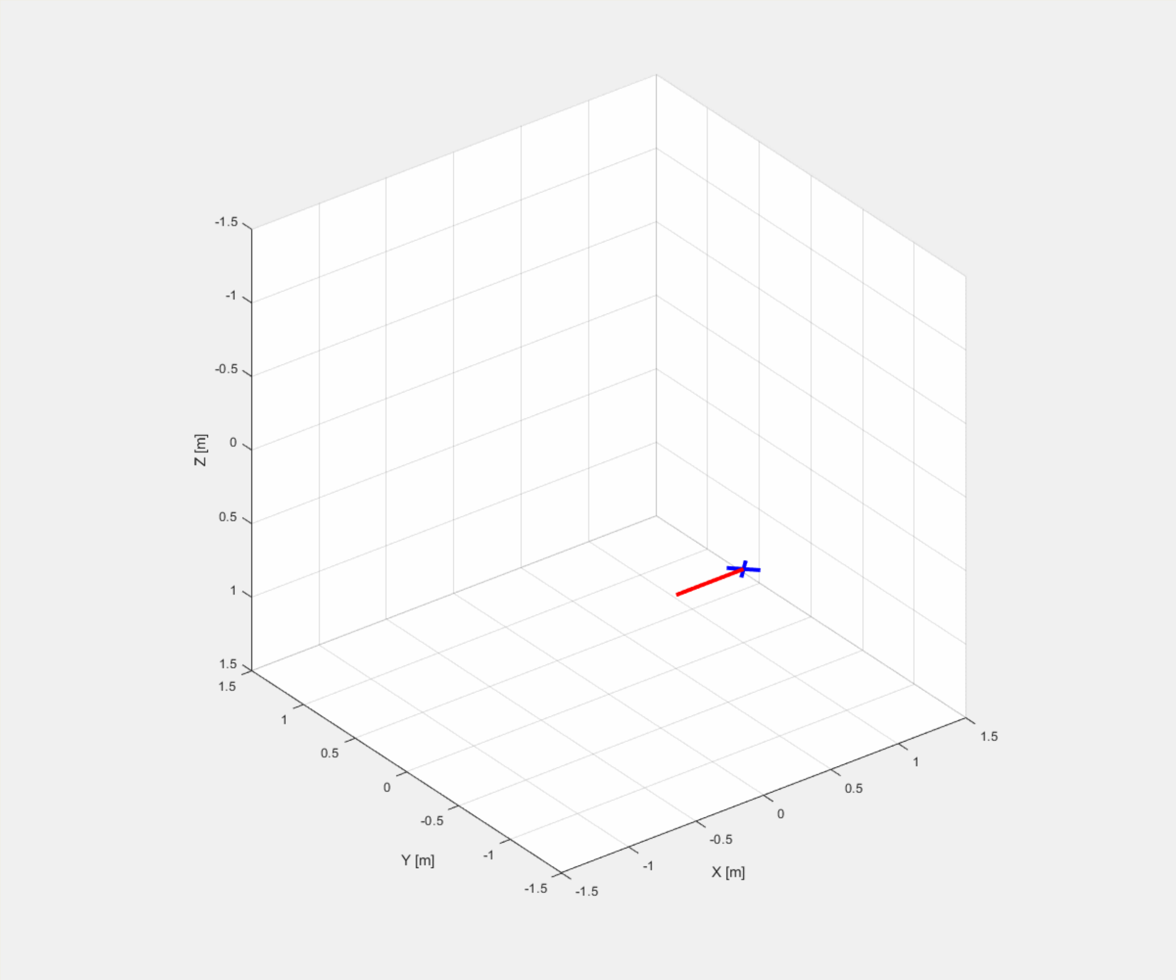
Simulation in MATLAB

- Static setpoint with, $m_P = 1.0kg, \alpha(0) = 0, \beta(0) = -3.14rad$



Simulation in MATLAB

■ Static setpoint with, $m_P = 0.3kg, \alpha(0) = 0, \beta(0) = -1.571rad$



Trajectory Planning with Dynamics of Quadrotor + Pendulum

- The controller in the simulations for this presentation is only a feedback controller. It does not account for the dynamics of the pendulum.

- Better controllers that account for the trajectory of the pendulum:
  - *Model Predictive Control*
  - *Differential Dynamic Programming (Solve the Hamilton-Jacobi-Bellman PDE)*