2D dynamics of a quadcopter drone with a tethered payload

Calum, Julia, Ruchira

03/10/2025

1 Introduction

This coursework project explores the safe control of autonomous systems with neural network (NN) components.

We aim to apply a neural network based controller to an aerial winched delivery system, consisting of a quadcopter drone which lowers a package (herein referred to as the payload). Aerial winched delivery has been pioneered by companies like Zipline[1] which uses a drone to lower packages on a tether via a motorised winch.

A benefit of this type of system is that it could enable the package to be delivered in a way that ensures a human collecting the package does not come within a close distance of dangerous rotating parts such as propellers on the drone. However, the system must be safe and there are certain risks posed which include payload swing, slack or broken tether, winch overload, and collisions with the environment.

In order to develop our control system, we first model the dynamics of our combined quadcopter and payload system. This is described fully in the following sections.

2 Dynamics

2.1 Summary and Assumptions

Below are the full equations of motion for the 2D dynamics of a quadcopter drone and a tethered payload, derived from first principles.

[1] www.zipline.com/technology

The x coordinate is defined positive to the right, and the z coordinate positive upwards.

Simplifying Assumptions

1. **Planar motion:** Both quadcopter and payload move only in the x–z plane; there is no yaw or roll motion.
2. **Rigid symmetric quadcopter:** The quadcopter is a single rigid body with its mass concentrated at its centre of mass.
3. **Massless, inextensible rope:** Always taut ($Q > 0$) with no elasticity or mass. Rope length ℓ varies only via the commanded feed rate u_ℓ.
4. **Axial thrust:** Each rotor produces thrust only along its body axis (no drag or gyroscopic coupling). Thrust changes instantaneously.
5. **No aerodynamic drag:** Air resistance on the quadcopter and payload is neglected.
6. **No external disturbances:** Wind and other perturbations are ignored (*may be added later for realism*).
7. **Quasi-static rope iteration:** The rope tension Q depends on the quad acceleration, which itself depends on Q. Instead of solving implicitly, we iterate once or twice per timestep, which is accurate for $dt < 0.05$ s.

2.2 State and Controls

$$\boldsymbol{s} = \begin{bmatrix} x_q & z_q & \theta & \dot{x}_q & \dot{z}_q & \dot{\theta} & \ell & \phi & \dot{\ell} & \dot{\phi} \end{bmatrix}^\top, \quad \boldsymbol{u} = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 & u_\ell \end{bmatrix}^\top$$

where x_q, z_q are the quadcopter centre of mass (COM) coordinates, θ is the pitch angle, ℓ is the rope length, and ϕ is the rope swing angle from vertical. Dots denote time derivatives.

2.3 Parameters

$$m_q,\ I_q,\ m_p,\ g,\ d$$

denote the quad mass, pitch inertia, payload mass, gravitational acceleration, and the rotor arm length (distance from COM to rotor thrust line), respectively.

2.4 Rope Geometry and Kinematics

Unit vectors along and perpendicular to the rope are defined as:

$$\boldsymbol{u}(\phi) = \begin{bmatrix} \sin\phi \\ -\cos\phi \end{bmatrix}, \qquad \boldsymbol{t}(\phi) = \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix},$$

so that $\boldsymbol{u}$ points from the quad to the payload (downward when $\phi = 0$) and $\boldsymbol{t}$ is tangent to the payload's swing, perpendicular to $\boldsymbol{u}$.

Payload position:

$$\boldsymbol{p}_p = \begin{bmatrix} x_q \\ z_q \end{bmatrix} + \ell\,\boldsymbol{u}(\phi).$$

Differentiating:

$$\dot{\boldsymbol{p}}_p = \begin{bmatrix} \dot{x}_q \\ \dot{z}_q \end{bmatrix} + \dot{\ell}\,\boldsymbol{u} + \ell\,\dot{\phi}\,\boldsymbol{t},$$

$$\ddot{\boldsymbol{p}}_p = \begin{bmatrix} \ddot{x}_q \\ \ddot{z}_q \end{bmatrix} + (\ddot{\ell} - \ell\dot{\phi}^2)\,\boldsymbol{u} + (2\dot{\ell}\dot{\phi} + \ell\ddot{\phi})\,\boldsymbol{t}.$$

2.5 Free-Body Diagram (FBD)

Below are free-body diagrams of the system, showing the forces acting on the quadcopter and the payload. A smaller side by side figure of each system separately is shown below.

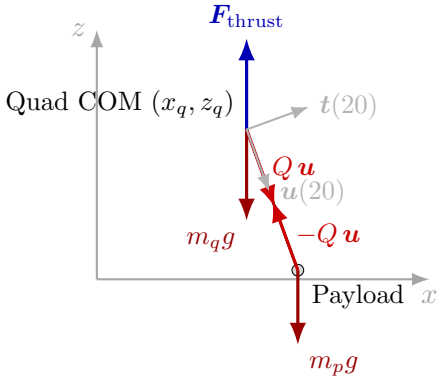

Figure 1: Free-body diagram showing the main forces on the quadcopter and payload.

(a) Quadcopter FBD (b) Payload FBD

Figure 2: Free-body diagrams showing the main forces on the quadcopter and payload.

2.6 Equations of Motion

(a) Quadcopter Translation:

$$m_q \begin{bmatrix} \ddot{x}_q \\ \ddot{z}_q \end{bmatrix} = \boldsymbol{F}_{\text{thrust}} + Q\,\boldsymbol{u} - m_q g \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Note: $Q\boldsymbol{u}$ acts downward because $\boldsymbol{u}$ points from the quad toward the payload; its sign convention is embedded in $\boldsymbol{u}$.

(b) Quadcopter Rotation:

$$I_q\,\ddot{\theta} = \tau,$$

where I_q is the pitch-axis inertia of the quadcopter.

(c) Payload Translation:

$$m_p\,\ddot{\boldsymbol{p}}_p = -m_p g \begin{bmatrix} 0 \\ 1 \end{bmatrix} - Q\,\boldsymbol{u}.$$

The rope acts on the payload in the opposite direction to that acting on the quad (Newton's third law).

2.7 Projection of Payload Motion

Projecting the payload equation onto $\boldsymbol{t}(\phi)$ (tangential) and $\boldsymbol{u}(\phi)$ (radial) directions gives:

(i) Tangential (swing) motion:

$$\ell\,\ddot{\phi} + 2\,\dot{\ell}\,\dot{\phi} = -\,g\,\sin\phi + \begin{bmatrix} \ddot{x}_q \\ \ddot{z}_q \end{bmatrix} \cdot \boldsymbol{t}(\phi).$$

(ii) Radial (tension) direction:

$$m_p(\ddot{\ell} - \ell\dot{\phi}^2) = m_p g\cos\phi - Q + m_p \begin{bmatrix} \ddot{x}_q \\ \ddot{z}_q \end{bmatrix} \cdot \boldsymbol{u}(\phi),$$

so that the rope tension is

$$\boxed{Q = m_p \left[g\cos\phi + \begin{bmatrix} \ddot{x}_q \\ \ddot{z}_q \end{bmatrix} \cdot \boldsymbol{u}(\phi) - (\ddot{\ell} - \ell\dot{\phi}^2) \right].}$$

2.8 Rope Feed (Winch) Model

$$\dot{\ell} = \begin{cases} u_\ell, & \text{algebraic model,} \\ v, & \text{first-order model } (\dot{v} = \omega_\ell(u_\ell - v)). \end{cases}$$

2.9 Summary of State Derivatives

$$\dot{\boldsymbol{s}} = \begin{bmatrix} \dot{x}_q \\ \dot{z}_q \\ \dot{\theta} \\ \ddot{x}_q \\ \ddot{z}_q \\ \ddot{\theta} \\ \dot{\ell} \\ \dot{\phi} \\ \ddot{\ell} \\ \ddot{\phi} \end{bmatrix}.$$

2.10 Verification Checks

- Fixed pivot, constant length: $\ddot{\phi} = -\frac{g}{\ell}\sin\phi$.
- Hover equilibrium: $T_{\text{tot}} = (m_q + m_p)g, \quad Q = m_p g.$

3 Simulation Model and Control Implementation

3.1 Numerical Simulation Framework

The system of equations derived in Section **??** describes a coupled set of nonlinear second-order differential equations for the quadcopter and payload

states. These equations cannot be solved analytically, hence they are integrated numerically over discrete timesteps Δt using a Runge–Kutta method of order 4 (RK4).

At each timestep, the following operations are performed:

1. Evaluate the current state vector $\boldsymbol{s}(t)$.
2. Compute the forces and accelerations from the dynamic model:

$$\dot{\boldsymbol{s}} = f(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{p})$$

   where $\boldsymbol{u}$ are the control inputs and $\boldsymbol{p}$ the physical parameters loaded from the configuration file (`config.yaml`).

3. Apply the Runge–Kutta integration scheme:

$$\boldsymbol{s}(t + \Delta t) = \boldsymbol{s}(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

   with

$$\begin{aligned} k_1 &= f(\boldsymbol{s}_t), \\ k_2 &= f(\boldsymbol{s}_t + \tfrac{1}{2}\Delta t k_1), \\ k_3 &= f(\boldsymbol{s}_t + \tfrac{1}{2}\Delta t k_2), \\ k_4 &= f(\boldsymbol{s}_t + \Delta t k_3). \end{aligned}$$

4. Update the rope tension Q using a quasi-static iteration:

$$Q^{(i+1)} = m_p\big(g\cos\phi + \boldsymbol{a}_q \cdot \boldsymbol{u} - (\ddot{\ell} - \ell\dot{\phi}^2)\big)$$

   This step resolves the algebraic coupling between Q and the quadcopter acceleration $\boldsymbol{a}_q$. Convergence is typically achieved in one or two iterations if $\Delta t < 0.05$ s.

The simulation is implemented in modular Python files:

- `dynamics.py`: defines the continuous-time state derivatives and force relationships.
- `simulation.py`: executes the RK4 integration loop and logs the system states.
- `pid_control.py`: provides the feedback controller implementation.

3.2 PID Control Architecture

The control objective is to stabilise the quadcopter and regulate the payload position during rope feed operations. A cascade control structure is used: an outer loop controls position, and inner loops stabilise the vehicle's attitude and velocity.

Each controlled state variable x_i has an associated PID controller defined by:

$$u_i(t) = K_{p,i}e_i(t) + K_{i,i}\int_0^t e_i(\tau)\,d\tau + K_{d,i}\frac{de_i(t)}{dt}$$

where $e_i(t) = x_{i,\text{ref}}(t) - x_i(t)$ is the error between the reference and measured state, and $K_{p,i}$, $K_{i,i}$, $K_{d,i}$ are the proportional, integral, and derivative gains respectively.

For the present implementation, individual PID controllers are defined for:

u_x : horizontal position control of the quadcopter (desired x_q)
u_z : vertical position control of the quadcopter (desired z_q)
u_θ : pitch angle stabilisation (desired $\theta = 0$)
u_ϕ : payload swing angle damping (desired $\phi = 0$)
u_ℓ : rope feed control (desired rope length profile $\ell_{\text{ref}}(t)$)

Each controller outputs a force or rate command that is converted to actuator inputs through a mixing matrix:

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = M \begin{bmatrix} u_z \\ u_\theta \end{bmatrix}$$

where M is the thrust allocation matrix determined by rotor geometry (arm length d and configuration).

3.3 Anti-Windup and Saturation Handling

To prevent integral windup under actuator saturation, the integral term is frozen whenever the control output exceeds predefined bounds:

$$\text{if } (u_i > u_{\max} \text{ and } e_i > 0) \quad \text{or} \quad (u_i < u_{\min} \text{ and } e_i < 0), \quad \dot{I}_i = 0.$$

This ensures stability and smooth recovery when the system re-enters the linear operating range.

3.4 Model Validation and Extensions

The simulation could be extended to include:

- Environmental disturbances (e.g. wind gusts, modelled as random force perturbations on the quadcopter).
- Aerodynamic drag or rope damping effects.
- Sensor and actuator delay models.