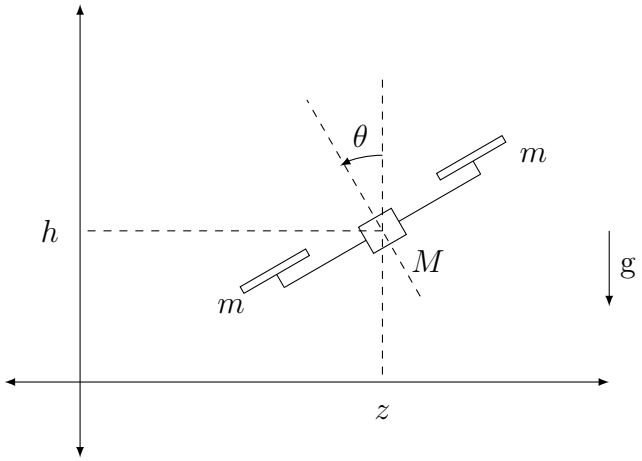
Pole Placement control design for a planar bicopter **(Graduate Version)**

Equations of motion

Consider a bicopter comprised of a central platform that contains batteries, sensors, and electronics, and two motor-propeller assemblies on the side. We will consider the motion of this bicopter confined to a single vertical plane that contains the axes of rotation of the two propellers.

The planar dynamics of the bicopter are expressed in terms of its angle of tilt, θ, altitude, h, and lateral position, z. Under simplifying assumptions, the differential equations that model the motion are given by



$$\ddot{h}(t) = -g + \frac{1}{M+2m}\cos(\theta(t))F(t)$$

$$\ddot{z}(t) = -\frac{\mu}{M+2m}\dot{z}(t) - \frac{1}{M+2m}\sin(\theta(t))F(t)$$

$$\ddot{\theta}(t) = \frac{1}{2md^2+J}\tau(t)$$

where $M = 1$ kg is the mass of the central platform, $m = 0.25$ kg is the mass of each motor-propeller assembly, $d = 0.3$ m is the distance between the center of mass of the bicopter and the axis of rotation of the propeller motor, $J = 0.0042\ \text{kg}\,\text{m}^2$ is the moment of inertia of the central platform, $g = 9.81\ \text{m}\,\text{s}^{-2}$ denotes the gravitational acceleration, and $\mu = 0.1\ \text{kg}\,\text{s}^{-1}$ denotes the drag coefficient that models momentum drag as a force acting on the center of mass of the bicopter, proportional to it's velocity, that is, $F_{drag} = -\mu\dot{z}$. In the model above,

F denotes the total thrust generated by the two propellers, and τ denotes the total torque generated by the two propellers.

The total thrust is given by $F = f_r + f_l$, where f_r and f_l are thrusts generated by the right and the left propellers, respectively, and the total torque is given by $\tau = d(f_r - f_l)$. Thus, given F and τ, we can easily compute f_r and f_l.

The motor thrust is proportional to the square of the motor angular velocity. A simplified model for the dynamics of the motor-propeller system is given by

$$\begin{aligned} J_m \dot{\varpi}_r(t) &= -b_m \varpi_r(t) + k_v v_r(t) \\ f_r &= k_f \varpi_r^2(t) \\ J_m \dot{\varpi}_l(t) &= -b_m \varpi_l(t) + k_v v_l(t) \\ f_l &= k_f \varpi_l^2(t) \end{aligned}$$

where $J_m = 0.004\ \mathrm{kg\,m^2}$ depends on the moment of inertia of the propeller and the rotor; $b_m = 0.04\ \mathrm{kg\,m^2\,s^{-1}}$ depends on damping in the motor, the back-EMF constant, the torque constant, the winding resistance, and the propeller drag; $k_v = 12\ \mathrm{kg\,m^2\,s^{-2}\,V^{-1}}$ depends on the torque constant and the winding resistance of the motor; $k_f = 1.2 \times 10^{-06}\ \mathrm{kg\,m}$ depends on the thrust constant of the propeller and the gear ratio of the motor-gearbox assembly; ϕ denotes the angular position of the motor; and $-10\mathrm{V} \leq v \leq 10\mathrm{V}$ denotes the voltage applied to the motor.

Question 1 To simplify the control design, express the dynamics in terms of the total thrust and the total torque using

$$\begin{aligned} \dot{F}(t) &= \dot{f}_r(t) + \dot{f}_l(t) \\ \dot{\tau}(t) &= d\dot{f}_r(t) - d\dot{f}_l(t) \end{aligned}$$

to get a dynamical system of the form $\dot{x} = f(x, u)$ and $y = \phi(x, u)$, where we will treat the voltages, v_r and v_l, as our control inputs and the position, z, the altitude, h, the tilt, θ, and the motor angular velocities, ϖ_r and ϖ_l, as our measured outputs, and the state vector is $x = \left[h, \dot{h}, z, \dot{z}, \theta, \dot{\theta}, F, \tau\right]$

Question 2: Find an equilibrium point (x^*, u^*) so that the bicopter is hovering at $z^* = 0$, at an altitude of $h^* = 10\,m$, with zero tilt, $\theta^* = 0$. Also find the corresponding equilibrium output, y^*.

Question 3: Linearize the dynamics of the bicopter around the equilibrium trajectory to yield a MIMO linear system of the form

$$\begin{aligned} \dot{x}_\delta &= Ax_\delta + Bu'_\delta, \\ y_\delta &= Cx_\delta + Du'_\delta, \end{aligned}$$

where $x_\delta = x - x^*$, $u'_\delta = u - u^*$, and $y_\delta = y - y^*$.

Hint: The `jacobian` and the `subs` commands from the Symbolic Math Toolbox and the `ss` command from Control Systems Toolbox

To simplify the next step, express the linear system in terms of a new control input u_δ where

$$u_\delta(1) = u'_\delta(1) + u'_\delta(2)$$
$$u_\delta(2) = u'_\delta(1) - u'_\delta(2)$$

Question 4: Use the modified linearized dynamics to generate a MIMO transfer function from the inputs $u_\delta(1) = v_{r\delta} + v_{l\delta}$ and $u_\delta(2) = v_{r\delta} - v_{l\delta}$ to the outputs h_δ, z_δ, and θ_δ.

Hint: The `tf` command from the Control Systems Toolbox can convert state-space models to transfer function models.

Question 5: At this stage, you should be able to express the motion of the bicopter in terms of the following transfer functions (the others are all zero)

$$G_{HU_1}(s) \coloneqq \frac{H(s)}{U_1(s)} \qquad G_{ZU_2}(s) \coloneqq \frac{Z(s)}{U_2(s)} \qquad G_{\Theta U_2}(s) \coloneqq \frac{\Theta(s)}{U_2(s)}$$
$$G_{\Omega_l U_1}(s) \coloneqq \frac{\Omega_l(s)}{U_1(s)} \qquad G_{\Omega_l U_2}(s) \coloneqq \frac{\Omega_l(s)}{U_2(s)} \qquad G_{\Omega_r U_1}(s) \coloneqq \frac{\Omega_r(s)}{U_1(s)} \qquad G_{\Omega_r U_2}(s) \coloneqq \frac{\Omega_r(s)}{U_2(s)}$$

Write down the above transfer functions. These will just be the corresponding elements of the MIMO transfer function.

Question 6: We want to design the motor voltages v_r and v_l so that the bicopter hovers above a desired z at a desired h with zero tilt.

Before we get there, note that the first modified control, $u_{1\delta} = v_{r\delta} + v_{l\delta}$, only affects the linearized height, h_δ, of the bicopter (the relationship between h_δ and $u_{1\delta}$ (which is proportional to F_δ) will be referred to as the **longitudinal** dynamics) and the second modified control, $u_{2\delta} = v_{r\delta} - v_{l\delta}$, only affects the linearized lateral position, z_δ and the linearized tilt θ_δ (i.e., the **lateral** dynamics). This is somewhat counterintuitive. **Tilt should reduce thrust in the vertical direction, and as a result, should make the bicopter fall, but that effect is lost in linearization.** We will plow ahead however, and see if we can design controllers that achieve what we want.

Since $u_{1\delta}$ only affects the height, we can design a controller that computes the *desired* $u_{1\delta}$ to keep the bicopter at the desired height. Since the same control input, $u_{2\delta}$, affects the linearized lateral position and the linearized tilt, to get the desired position and zero tilt, we cannot simply design two different transfer functions for $u_{2\delta}$. We need a successive loop closure design.

To simplify the design even further we can separate the motor angular velocities from the motor voltages to reduce the degree of the transfer functions involved. Notice that the last

two outputs, $\varpi_{r\delta}$ and $\varpi_{l\delta}$, in our 2-input 5-output linearized system are independent of all other states and outputs, with the transfer functions

$$\begin{bmatrix} \Omega_r(s) \\ \Omega_l(s) \end{bmatrix} = \begin{bmatrix} G_{\Omega_r U_1}(s) & G_{\Omega_r U_2}(s) \\ G_{\Omega_l U_1}(s) & G_{\Omega_l U_2}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}$$

Using the relationship above and the specific values of the transfer functions we found, it is possible to construct transfer functions

$$G_{\Omega_F U_1}(s) \coloneqq \frac{\Omega_F(s)}{U_1(s)} \qquad G_{\Omega_\tau U_2}(s) \coloneqq \frac{\Omega_\tau(s)}{U_1(s)}$$

where Ω_F is the Laplace transform of $\varpi_{F\delta} = \varpi_{l\delta} + \varpi_{r\delta}$ and Ω_τ is the laplace transform of $\varpi_{\tau\delta} = \varpi_{l\delta} - \varpi_{r\delta}$.

We will then follow the following steps:

1. Design a controller that computes the *desired* $\varpi_{F\delta}$ to keep the bicopter at the desired height. To do this, you will need to compute the transfer function $G_{H\Omega_F}$ using G_{HU_1} and $G_{\Omega_F U_1}$. This will be the **altitude loop**. The `minreal` MATLAB function will come in handy to evaluate any pole zero cancellations that might reduce the order of the transfer function.

2. Treat the linearized tilt angle θ_δ as the control input and design a controller that generates the *desired* tilt angles so that the linearized lateral position, z_δ, tracks the reference lateral position signal, z_r. This will be our **outer (or position) loop**. To implement the outer loop, we need the DC gain of the inner closed-loop and the transfer function $G_{Z\Theta}$ computed using G_{ZU_2} and $G_{\Theta U_2}$.

3. Treat $\varpi_{\tau\delta}$ as a control input to make the linearized tilt, θ_δ, track the desired tilt angles, θ_r, generated by the outer loop controller. This will be our **inner (or tilt) loop** that computes the *desired* $\varpi_{\tau\delta}$. To design this, you will need to compute the transfer function $G_{\Theta\Omega_\tau}$ using $G_{\Theta U_2}$ and $G_{\Omega_\tau U_2}$.

4. Once we have the *desired* $\varpi_{F\delta}$ and $\varpi_{\tau\delta}$, we can compute the *desired* $\varpi_{r\delta}$ and $\varpi_{l\delta}$. From there, **using the value of y^* we computed during linearization, we can compute the *desired* ϖ_r and ϖ_l.** Then, treat v_r and v_l as control inputs to design a controller for each motor so that the actual ϖ_r and ϖ_l track the desired ϖ_r and ϖ_l. This is our **motor loop**. To design the motor loop you will need to compute y^* (specifically, ϖ_l^* and ϖ_r^*) and the transfer functions $G_{\Omega_r V_r}$ and $G_{\Omega_l V_l}$.

Draw a block diagram of the overall control system.

Control Design Objective: The final closed-loop transfer function from z_r to z_δ needs to have a rise time of approximately 3 s and a settling time less than 8 s.

Motor control

Question 7: Design an angular velocity controller for each motor with appropriate design specifications to compute voltage commands for tracking a desired angular velocity. **Justify your choice of performance specifications.**

Longitudinal Control Design

Question 8 (Altitude loop control via pole placement): Replacing the motor closed-loop by its DC gain, design a controller to generate linearized thrust commands so that the rise time, in response to a unit step height input, is approximately $2\,s$ and the damping ratio is approximately 0.707. Show your design process (**trial and error or MATLAB PIDTuner are not acceptable design process if that is all you are doing)**, show step response plots for the closed-loop and the output of the `stepinfo` and `isstable` commands.

Lateral Control Design

Question 9. (Lateral control via pole placement): For both the inner and outer loop designs below, show your design process (**(trial and error or MATLAB PIDTuner are not acceptable design process if that is all you are doing)**, show step response plots for the closed-loop and the output of the `stepinfo` and `isstable` commands.

Question 9.1. Replacing all internal closed-loops (if any) by their DC gains, design the inner loop lateral controller with appropriate performance specifications. **Justify your choice of performance specifications.**

Question 9.2. Replacing all internal closed-loops (if any) by their DC gains, design the outer loop lateral controller so that the closed-loop outer-loop system satisfies appropriate performance specifications. **Justify your choice of performance specifications.**

Simulation

Question 10 (Linear Simulation): Implement the successive loop closure lateral controller along with the longitudinal controller in Simulink using the transfer functions you derived to follow a commanded lateral position, z, given by a square wave with amplitude 1.5 m and frequency 0.03 Hz at a constant height of 10 m and zero tilt. **You do not need**

the Simulink files that were sent out with the assignment to answer this question, build your own simulation using the block diagram from Question 6, the transfer functions from Question 5, and controllers you calculated in Questions 8 and 9. Show any plots you need to demonstrate the performance of your controller and a picture of your Simulink block diagram.
Remark: When you implement a derivative controller to track a square wave, the error signal $e_z = z_r - z_\delta$ changes discontinuously every time the square wave oscillates. This results in excessively large $\dot{e}_z$, and as a result, excessively large control inputs. To avoid this, the desired square wave is usually modified to smoothen it out. Or, if you know apriori that the desired signal is a square wave, you can assume that the time derivative of the desired signal is zero, and instead of differentiating e_z in simulink, you can differentiate the output, in this case, z_δ, to get $\dot{z}_\delta$. The PD controller can then be implemented as $u_z(t) = k_p e_z - k_d \dot{z}_\delta$ instead of $u_z(t) = k_p e_z - k_d \dot{e}_z$.

Question 11 (Nonlinear Simulation): You are given a simulink model that implements the full nonlinear dynamics of the bicopter with added disturbances and sensor noise. The model takes the motor voltages v_r and v_l as inputs and produces the lateral position, z, the altitude, h, the tilt, θ, and the motor angular velocities, Ω_r and Ω_l as the outputs. Implement the five controllers (one controller for the longitudinal dynamics, inner and outer loop controllers for lateral dynamics, and two motor controllers) you designed to control the bicopter. Show any plots you need to demonstrate the performance of your controller and a picture of your Simulink block diagram.
Remark: Do not forget that your model is in terms of deviation from (x^*, u^*). You need to track a given z, h, and θ. The model is in terms of z_δ, h_δ, and θ_δ. Do not forget to convert between the two wherever you need.
Remark: When you implement a derivative controller to track a square wave, the error signal $e_z = z_r - z_\delta$ changes discontinuously every time the square wave oscillates. This results in excessively large $\dot{e}_z$, and as a result, excessively large control inputs. To avoid this, the desired square wave is usually modified to smooth it out. Or, if you know apriori that the desired signal is a square wave, you can assume that the time derivative of the desired signal is zero, and instead of differentiating e_z in Simulink, you can differentiate the output, in this case, z_δ, to get $\dot{z}_\delta$. The PD controller can then be implemented as $u_z(t) = k_p e_z - k_d \dot{z}_\delta$ instead of $u_z(t) = k_p e_z - k_d \dot{e}_z$.

Question 12: Compare the performance of the controllers when implemented using linear transfer function models with their performance when implemented on the nonlinear model. This should be a discussion on what differences (if any) were observed and why.

Question 13: Open your nonlinear simulation from Question 10. Double click on the bicopter parameters block, change the value of the `Noise Variance` from 0 to `1e-6`. Save and re-run the nonlinear simulation and comment on its performance as compared with the noise-free simulation.

Controller modification for robustness

Question 14: Open your nonlinear simulation from Question 10. Reset `Noise Variance` to 0. Double click on the bicopter parameters block, change the value of `Parameter Variation Percentage` from 0 to 20. Save and re-run the nonlinear simulation and comment on its performance as compared with the noise-free simulation. Does the bicopter stay at the desired 10 m altitude? If not, **explain why** and modify the longitudinal controller to correct the altitude error. Explain why you made the choices you did and include any plots you need to demonstrate that the modifications corrected the problem.

Question 15: Open your nonlinear simulation from Question 10. Double click on the bicopter parameters block, change the value of `Wind Velocity` from 0 to 1 $\mathrm{m\,s^{-1}}$. Save and re-run the nonlinear simulation and comment on its performance as compared with the noise-free simulation. Does the bicopter still have zero steady-state error when tracking the desired lateral position square wave? If not, explain why and modify the lateral controller to correct the position error. Explain why you made the choices you did and include any plots you need to demonstrate that the modifications corrected the problem.