Instructions:

- Answer all questions part 1 to 3.
- All relevant materials available in this drive link: https://drive.google.com/drive/folders/1BztDoOqLjzDDqXWUG7xaylAwZIhR3bB3?usp=drive_link
- Use MATLAB, and please share all relevant scripts/code and outputs. (as a drive link if necessary)

Modern Robotics: Analysis and Control

Project

Part 1 In this part, we will use the kinematic car model presented in [1, Section 4.1.1] (references are listed on the last page of this document). You will first create a Simulink diagram, then implement two closed-loop control laws.

(a) The dynamics of the system are governed by

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{\ell}\tan\gamma \end{aligned} \tag{1}$$

Set up a Simulink subsystem which inputs rear wheel velocity v and steering angle γ, and outputs the planar position (x, y) and heading angle θ. Take $\ell = 1$ m. Provide a screenshot of your Simulink design, including the contents of any subsystems you used.

(b) Create a closed-loop control system to move to a desired point. This system inputs a desired planar position $(x_{\text{ref}}, y_{\text{ref}})$, computes the reference heading

$$\theta_{\text{ref}} = \text{atan2}\left(y_{\text{ref}} - y, x_{\text{ref}} - x\right)$$

then employs the control

$$\begin{aligned} v &= k_v\sqrt{(x_{\text{ref}} - x)^2 + (y_{\text{ref}} - y)^2} \\ \gamma &= k_\gamma(\theta_{\text{ref}} \ominus \theta) \end{aligned}$$

where $\theta_{\text{ref}} \ominus \theta$ is the shortest path from θ_{ref} to θ on the unit circle — this is discussed in the course notes. Using an initial state of $(x_0, y_0, \theta_0) = (0, 0, \pi/4)$, tune the control gains k_v and k_γ to drive the car to the reference (desired) position $(x_{\text{ref}}, y_{\text{ref}}) = (10, -8)$ m. Provide a plot of the states (x, y, θ) and inputs (v, γ) as well as the values of the control gains you used.

(c) Using the same initial and desired state as (b), tune the gains k_v and k_γ to make the vehicle drive continuously around the goal without reaching it — this is known as a limit cycle. Provide the values of the gains used along with a plot of x versus y (i.e. overhead position of the car) for your limit cycle.

(d) We will now implement a controller proposed by [2] allowing the kinematic car to move to a desired 2D pose $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$. Introduce the new input ω through

$$\gamma = \text{atan2}(\omega\ell, v) \tag{2}$$

which transforms (1) into

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{3}$$

Define the set of variables

$$\begin{aligned}
\rho &= \sqrt{(x_{\text{ref}} - x)^2 + (y_{\text{ref}} - y)^2} \\
\phi &= \operatorname{atan2}(y_{\text{ref}} - y, x_{\text{ref}} - x) \\
\alpha &= \phi - \theta \\
\beta &= \phi - \theta_{\text{ref}}
\end{aligned}$$

Using the nonlinear feedback control law

$$\begin{aligned}
v &= k_v \rho \cos\alpha \\
\omega &= k_\omega \alpha + k_v \frac{\sin\alpha\cos\alpha}{\alpha}(\alpha + \beta)
\end{aligned}$$

with $k_v > 0$, $k_\omega > 0$ can be shown (see course notes) to make (3) converge asymptotically to $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$. Note because our actual system (1) employs input γ instead of ω, we employ the above result in (2) to obtain γ. Implement the above-described design in Simulink (you may wish to use the `MATLAB Function` block found in Library Browser under `Simulink -> User-Defined Functions` for the control law). Provide a screenshot of your design, plus a print-out of your MATLAB function if you used it.

(e) Using the design in (d) and initial pose $(x_0, y_0, \theta_0) = (0, 0, 0)$, use your design to drive to the following final configurations, tuning gains k_v, k_ω to get smooth performance:

(i) $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}}) = (5, 8, 0)$

(ii) $(x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}}) = (-10, -4, -\pi/2)$

For each run, provide plots of the vehicle states (x, y, θ) and inputs (v, γ) plus the values of control gains you used for each run.

(f) During your gain tuning for this second design, did the system perform a limit cycle?

Part 2 Load the satellite Simulink diagram provided to you on google drive link This system inputs torques τ_1, τ_2, τ_3 about its three body-fixed axes, and outputs attitude $R \in SO(3)$ and body-frame angular velocity $\omega^b \in \mathbb{R}^3$. Note the output attitude is a matrix signal which gets converted to Euler angles (ϕ, θ, ψ) using a custom `MATLAB Function` block named `R2e`.

(a) First run the system in open-loop mode, inputting a step input of magnitude 0.1 N m along an individual axis, then scoping the output angular velocity vector ω^b. You should observe a "cross-coupling" effect between the output axes. What is the cause of this? How could the satellite be structurally re-designed to avoid this effect?

(b) Now implement the linear proportional-derivative control law

$$\begin{aligned}
\tau_1 &= k_1^P(\phi_{\text{ref}} - \phi) + k_1^D(-\dot{\phi}) \\
\tau_2 &= k_2^P(\theta_{\text{ref}} - \theta) + k_2^D(-\dot{\theta}) \\
\tau_3 &= k_3^P(\psi_{\text{ref}} - \psi) + k_3^D(-\dot{\psi})
\end{aligned}$$

where the Euler angle time derivatives $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ can be obtained from

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \omega_1^b \\ \omega_2^b \\ \omega_3^b \end{bmatrix}$$

Provide a screenshot of your design. What is the reason for including a derivative feedback action?

(c) Starting from initial conditions $R(0) = I \implies (\phi(0), \theta(0), \psi(0)) = (0, 0, 0)$, tune the linear control gains in order for the system to stabilize an attitude of $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (\pi/6, \pi/6, \pi/6)$. Provide plots of (i) Euler angles versus time and (ii) torque inputs versus time during this maneuver.

(d) Using the same control gains as (c), try stabilizing $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (2\pi/3, 2\pi/3, 2\pi/3)$. Explain what you see happening. Can performance be improved by re-tuning the controller gains?

(e) Now replace the linear closed-loop control with the nonlinear controller described in [3],

$$\tau = -K_v \omega^b - K_p \sum_{i=1}^{3} \alpha_i e_i \times (R_{\text{ref}}^T R e_i)$$

where $K_v, K_p \in \mathbb{R}^{3\times3}$ are positive-definite matrices (for instance diagonal matrices with positive, non-zero entries), $\omega^b \in \mathbb{R}^3$ is the angular velocity vector as before, $\alpha_k \in \mathbb{R}^+$ are positive, non-zero and *distinct* scalars, $e_1 = [1 \quad 0 \quad 0]^T$ etc, and $R_{\text{ref}}, R \in SO(3)$ are the desired and actual attitude, respectively. Suggestion: use the `MATLAB Function` block to implement your control law. Provide a screenshot of your design and a print-out of your MATLAB code if you used it.

(f) Tune the nonlinear controller to stabilize R_{ref} corresponding to $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (\pi/6, \pi/6, \pi/6)$. Provide plots of Euler angles versus time during this maneuver. How does performance compare with the linear control from (c)?

(g) Now run the nonlinear controller using R_{ref} corresponding to $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}}) = (2\pi/3, 2\pi/3, 2\pi/3)$. Provide nine plots (you can use MATLAB's `subplot` command) of the individual entries of R and R_{ref} versus time. Based on these plots, does your closed-loop system work properly?

Part 3 Load the quadcopter Simulink diagram provided on google drive link. This system is a 6 DoF rigid-body with inputs f_t (thrust along b_3 vector) and τ_r, τ_p, τ_y (torques about roll, pitch and yaw axes, respectively), and outputs $R \in SO(3)$ (attitude), ω (angular velocity vector in the body-fixed frame), and p, v (position and velocity with respect to the ground-fixed North-East-Down frame). Remark $f_t < 0$ corresponds to an upward thrust due to b_3 pointing down, and that R is converted to roll-pitch-yaw Euler angles (ϕ, θ, ψ) in the diagram. You will need to manually rotate the Simscape visualization perspective to orient the third reference axis into the down position.

(a) Implement the linear closed-loop control described in [1, Section 4.2]. This control consists of three proportional-derivative controllers working in tandem: The **lateral and longitudinal** control

$$\tau_r = k_r^P(\phi_{\text{ref}} - \phi) - k_r^D \dot{\phi}$$
$$\tau_p = k_p^P(\theta_{\text{ref}} - \theta) - k_p^D \dot{\theta}$$

where $\dot{\phi}, \dot{\theta}$ are obtained from ω^b as in the satellite controller, and $(\phi_{\text{ref}}, \theta_{\text{ref}})$ are computed by

$$\begin{aligned}\phi_{\text{ref}} &= k_\phi^P(p_{\text{lat,ref}} - p_{\text{lat}}) - k_\phi^D v_{\text{lat}}\\ \theta_{\text{ref}} &= -k_\theta^P(p_{\text{lon,ref}} - p_{\text{lon}}) + k_\theta^D v_{\text{lon}}\end{aligned}$$

where

$$\begin{bmatrix} p_{\text{lat}} \\ p_{\text{lon}} \end{bmatrix} = \underbrace{\begin{bmatrix} -\sin\psi & \cos\psi \\ \cos\psi & \sin\psi \end{bmatrix}}_{R_\psi} \begin{bmatrix} p_N \\ p_E \end{bmatrix}$$

and

$$\begin{bmatrix} p_{\text{lat,ref}} \\ p_{\text{lon,ref}} \end{bmatrix} = R_\psi \begin{bmatrix} p_{N,\text{ref}} \\ p_{E,\text{ref}} \end{bmatrix}, \quad \begin{bmatrix} v_{\text{lat}} \\ v_{\text{lon}} \end{bmatrix} = R_\psi \begin{bmatrix} v_N \\ v_E \end{bmatrix}$$

with $[p_N \quad p_E]^T$, $[v_N \quad v_E]^T$ the vehicle position and velocity in the NED frame, and $[p_{N,\text{ref}} \quad p_{E,\text{ref}}]^T$ the reference trajectory in the NED frame; the **yaw** control

$$\tau_y = k_y^P(\psi_{\text{ref}} \ominus \psi) - k_y^D \dot{\psi}$$

where ψ_{ref} is the reference yaw angle of the vehicle and $\ominus$ is the same operator as in Part 1b; and the **altitude** control

$$f_t = k_t^P(p_{D,\text{ref}} - p_D) - k_t^D v_D - mg$$

where $f_t < 0$ represents an upwards thrust, p_D and v_D are the vehicle's position and velocity along the down axis, $m = 1.216$ kg is the mass of the vehicle, and $g = 9.81\ \text{m/s}^2$. Provide a screenshot of your design.

(b) Using your control from (a) and zero initial conditions (representing a take-off configuration with the drone pointing north), tune your gains to stabilize $[p_{N,\text{ref}} \quad p_{E,\text{ref}} \quad p_{D,\text{ref}} \quad \psi_{\text{ref}}] = [1 \quad 1 \quad -2 \quad \pi/2]$ i.e. a hover 2 m above the ground facing E. Your gains should be tuned to avoid any significant overshoots. Provide plots of position (as p_N, p_E and p_D) and attitude (as ϕ, θ and ψ) versus time during this maneuver.

(c) Starting from an initial hover at $[p_N(0) \quad p_E(0) \quad p_D(0)] = [0 \quad 0 \quad -2]$ with attitude $[\phi(0) \quad \theta(0) \quad \psi(0)] = [0 \quad 0 \quad 0]$, tune the gains to track a figure-8 trajectory in space described by the parametric curves

$$\begin{aligned}p_{N,\text{ref}}(t) &= l_M \sin(2\pi t/t_c)\\ p_{E,\text{ref}}(t) &= (l_m/2)\sin(4\pi t/t_c)\\ p_{D,\text{ref}}(t) &= (-h/2)\sin(\pi t/t_c) - 2\end{aligned}$$

where $l_M = 2$ m, $l_m = 1$ m are the major and minor diameters of each lobe, $h = 2$ m is the total vertical height of the trajectory, and $t_c = 10$ s is the time required to complete one full circuit. Use the reference yaw angle

$$\psi_{\text{ref}}(t) = \text{atan2}\left[(d/dt)p_{E,\text{ref}}(t), (d/dt)p_{N,\text{ref}}(t)\right]$$

which points the vehicle in the direction of travel, and where the time derivatives can be obtained by analytically differentiating the parametric curves. The initial condition on p_D can be specified by going inside the quadcopter subsystem, double-clicking the `6-DOF Joint`, expanding `Z Prismatic Primitive (Pz)` $\rightarrow$ `State Targets` $\rightarrow$ `Specify Position Target` (this should be enabled), and then entering the desired $p_D(0)$ into the `Value` field.

Provide a screenshot of your design. Plot positions p and reference positions p_{ref} versus time, attitude (ϕ, θ, ψ) and reference attitude $(\phi_{\text{ref}}, \theta_{\text{ref}}, \psi_{\text{ref}})$ versus time, and control inputs $(\tau_r, \tau_p, \tau_y, f_t)$ versus time, for $0 \le t \le t_c$ i.e. one complete circuit. Note you can `unwrap` the ψ data prior to plotting in order to remove jumps of 2π. The controller gains should be tuned to provide accurate tracking and avoid overshoots.

(d) Now replace the linear control from (a) by the geometric tracking control law proposed in [4]. First, calculate the vector

$$z = k_p(p_{\text{ref}} - p) + k_v(\dot{p}_{\text{ref}} - v) + m\ddot{p}_{\text{ref}} - mge_3 \in \mathbb{R}^3$$

where $k_p, k_v \in \mathbb{R}$ are positive constants, $p_{\text{ref}} = [p_{N,\text{ref}} \quad p_{E,\text{ref}} \quad p_{D,\text{ref}}]^T$ and $p = [p_N \quad p_E \quad p_D]^T$ are the desired and actual position vector in the world frame, $\dot{p}_{\text{ref}} = (d/dt)p_{\text{ref}}$ and v are the desired and actual velocity vector in the world frame, $\ddot{p}_{\text{ref}} = (d^2/dt^2)p_{\text{ref}}$ is the desired acceleration vector in the world frame, and m is the mass of the vehicle and g is the acceleration due to gravity as in (a). Note that $\dot{p}_{\text{ref}}$ and $\ddot{p}_{\text{ref}}$ are obtained by analytic differentiation of the parametric curve p_{ref} given in (c). The thrust input f_t is calculated as

$$f_t = z \cdot Re_3$$

where $R \in SO(3)$ is the current attitude of the drone and $e_3 = [0 \quad 0 \quad 1]^T$. As before, $f_t < 0$ indicates an upward thrust. The remaining three inputs τ_r, τ_p, τ_y are employed to point the body-fixed frame vector b_3 in the direction $-z$ as follows. Define the reference (desired) attitude matrix $R_{\text{ref}} \in SO(3)$ as

$$R_{\text{ref}} := \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

The right-most column v_3, which represents the third body-fixed axis of the drone at its reference attitude expressed in ground-frame coordinates, is assigned as

$$v_3 = -\frac{z}{\|z\|}$$

Columns v_1 and v_2, representing the ground-frame coordinates of the first and second body-fixed axes of the drone at its reference attitude, lie in the plane normal to v_3. We will employ the reference yaw angle ψ_{ref} to find their direction using the following series of calculations. First, calculate the vectors

$$v_A = R_3(\psi_{\text{ref}})e_1 = \begin{bmatrix} \cos\psi_{\text{ref}} & -\sin\psi_{\text{ref}} & 0 \\ \sin\psi_{\text{ref}} & \cos\psi_{\text{ref}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\psi_{\text{ref}} \\ \sin\psi_{\text{ref}} \\ 0 \end{bmatrix}$$

$$v_B = R_3(\psi_{\text{ref}})e_2 = \begin{bmatrix} \cos\psi_{\text{ref}} & -\sin\psi_{\text{ref}} & 0 \\ \sin\psi_{\text{ref}} & \cos\psi_{\text{ref}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\sin\psi_{\text{ref}} \\ \cos\psi_{\text{ref}} \\ 0 \end{bmatrix}$$

Note that v_2 is perpendicular to both v_3 and v_A, while v_1 is perpendicular to both v_3 and v_B. The first relationship uniquely determines v_2 except when $v_3 \parallel v_A$ (singularity type "A"), and the second relationship uniquely determines v_1 except when $v_3 \parallel v_B$ (singularity type "B"); however, these two singularities will never occur at the same time. We thus compute the vector norms $\|v_3 \times v_A\|$ and $\|v_B \times v_3\|$, then perform calculations corresponding to the larger of the two values, namely

$$\text{if} \quad \|v_3 \times v_A\| > \|v_B \times v_3\| \Longrightarrow v_2 = \frac{v_3 \times v_A}{\|v_3 \times v_A\|}, \quad v_1 = v_2 \times v_3$$

or

$$\text{if} \quad \|v_3 \times v_B\| > \|v_A \times v_3\| \Longrightarrow v_1 = \frac{v_B \times v_3}{\|v_B \times v_3\|}, \quad v_2 = v_3 \times v_1$$

The result of the above calculations is the reference attitude matrix $R_{\text{ref}} = [v_1 \quad v_2 \quad v_3]$. Next define the attitude and angular velocity errors

$$\begin{aligned} e_R &= \frac{1}{2} S^{-1} \left(R^T R_{\text{ref}} - R_{\text{ref}}^T R \right) \\ e_\omega &= R^T R_{\text{ref}} \, \omega_{\text{ref}} - \omega \end{aligned}$$

where $\omega \in \mathbb{R}^3$ is the angular velocity vector of the drone expressed in the body-fixed frame, and $\omega_{\text{ref}} = [0 \quad 0 \quad 0]^T$ as discussed in the notes. The torque inputs are then calculated as

$$\begin{bmatrix} \tau_r \\ \tau_p \\ \tau_y \end{bmatrix} = k_R \, e_R + k_\omega e_\omega + \omega \times \mathcal{I}\omega + \mathcal{I} \left[R^T R_{\text{ref}} \, \dot{\omega}_{\text{ref}} - S(\omega) R^T R_{\text{ref}} \, \omega_{\text{ref}} \right]$$

where $k_R, k_\omega \in \mathbb{R}^+$ are controller gains, $\mathcal{I}$ is the mass moment of inertia matrix of the drone, in our case

$$\mathcal{I} = \begin{bmatrix} 0.0202 & 0 & 0.004 \\ 0 & 0.0207 & 0 \\ 0.004 & 0 & 0.0356 \end{bmatrix} \text{ kg m}^2$$

and $\dot{\omega}_{\text{ref}} = [0 \quad 0 \quad 0]^T$ as discussed in the notes.

Implement the above-discussed nonlinear control in Simulink. Provide a print-out of your controller code within the `MATLAB Function` block.

(e) Test your nonlinear control design with the Figure-8 reference trajectory from (c), using the initial conditions $[p_N(0) \quad p_E(0) \quad p_D(0)] = [0 \quad 0 \quad -4]$ and $[\phi(0) \quad \theta(0) \quad \psi(0)] = [0 \quad \pi \quad 0]$ — note this represents an initially upside-down drone. The initial height is specified as in section (c). To specify the initial attitude, go into the drone subsystem, double-click the `6-DOF Joint`, expand `Spherical Primitive (S)`, then set the values as follows:

Field	**Value**
Specify Position Target	[Enabled]
Priority	High (desired)
Value	Rotation Sequence
Rotation About	Base Axes
Sequence	X-Y-Z
Angles	[0 pi 0] rad

Provide plots of positions p and p_{ref} versus time, attitudes (ϕ, θ, ψ) versus time, and control inputs $(f_t, \tau_r, \tau_p, \tau_y)$ versus time for two complete circuits $0 \leq t \leq 2t_c$, as well as the values of controller gains k_p, k_v, k_R, k_ω you used to obtain a stable closed-loop system response. Could the linear control law from (a) be used to perform this experiment? Why or why not?

References

[1] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, volume 118 of *Springer Tracts in Advanced Robotics*. Springer, second edition, 2017.

[2] Michele Aicardi, Giuseppe Casalino, Antonio Bicchi, and Aldo Balestrino. Closed loop steering of unicycle-like vehicles via Lyapunov techniques. *IEEE Robotics & Automation Magazine*, 2(1):27–35, March 1995.

[3] Nalin A. Chaturvedi, Amit K. Sanyal, and N. Harris McClamroch. Rigid-body attitude control: Using rotation matrices for continuous, singularity-free control laws. *IEEE Control Systems Magazine*, 31(3):30–51, June 2011.

[4] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *Proceedings of the 49th IEEE Conference on Decision and Control*, pages 5420–5425, Atlanta, GA, December 2010.