Chapter Outline

Exercises and Classroom Discussion

Chapter 6
External Control and Trajectory Planning

6.1 External Control Fundamentals
6.2 Control Principles and Modes
6.3 Trajectory Models and Generation
6.4 Planning and Obstacle Avoidance
6.5 Tracking and Aggressive Flight Control
6.6 Frontier Control Technologies
6.7 Platform and Communication Interfaces
6.8 Examples and Experimental Validation
6.9 Exercises and Classroom Discussion

4 Units in This Section

6.9.1 Concepts and Distinctions
Textbook: Sec. 6.4, Exs. 1–3, 5–6

6.9.2 Trajectory Calculations
Textbook: Sec. 6.4, Exs. 8–10

6.9.3 Programs and Experiments
Textbook: Sec. 6.4, Exs. 4, 7–10

6.9.4 Integrated Tasks and Review
Textbook: Secs. 6.3.1, 6.4

6.9.1 Concepts and Distinctions: Textbook Ex. 1

How Three Control Layers Complete an Inspection

Question: describe the three-layer hierarchy of external control and its main functions.

Layer	Main Output	Inspection Example
Swarm level	Task allocation and coordination constraints	Assign inspection regions; coordinate passage times through narrow corridors
High level	Paths, trajectories, and task actions	Choose approach routes, speeds, and imaging positions
Low level	Motion control and actuator action	Estimate state; track position, velocity, and attitude setpoints

Key point: function determines the layer. Ground or onboard deployment is an implementation choice.

6.9 Exercises and Classroom Discussion

6.9.1 Concepts and Distinctions: Textbook Ex. 2

Offboard Control Requires a Continuous Valid Setpoint Stream and Feedback

Question: describe Offboard control characteristics and typical applications.



Feature: continuously update motion setpoints to connect vision, planning, or optimization to the feedback loop.

Examples: visual following, circle tracking, or velocity updates in response to new obstacles.

Conditions: valid state estimation, a continuous setpoint stream (>2 Hz, also a liveness signal), and defined takeover/exit behavior.

6.9 Exercises and Classroom Discussion

6.9.1 Concepts and Distinctions: Textbook Ex. 3

Path, Trajectory, Feasibility, and Optimality Answer Different Questions

Question: distinguish path from trajectory and feasibility from optimality; explain their typical roles in planning.

Concept	Question Answered	Example
Path	Which positions are traversed?	A circle of radius 5 m with no arrival times
Trajectory	Where is the vehicle at each time?	Circular position as a function of time
Feasibility	Are all constraints satisfied?	Meeting the speed limit still requires a turning-acceleration check
Optimality	Which criterion defines better?	Compare time, energy, or smoothness among feasible trajectories

Process: find a path → add timing → check feasibility → optimize within the feasible set.

State constraints and objectives before judging which motion plan is better.

6.9.1 Concepts and Distinctions: Textbook Ex. 5

Position, Velocity, and Acceleration Must Describe One Reference

Question: explain the meaning and advantages of the position–velocity–acceleration (PVA) interface.

$$\mathbf{v}_r = \dot{\mathbf{p}}_r, \quad \mathbf{a}_r = \ddot{\mathbf{p}}_r$$

Subscript r denotes reference; dots denote time derivatives.

Consistency requirements:
One time base
One coordinate frame
Units: position m, velocity m/s, acceleration m/s²

Role: position sets the arrival target; velocity and acceleration describe how the reference changes.

Advantage: feedforward reduces reliance on position error alone to catch a changing reference.

Check: consistent derivatives, matching message fields and enable settings, and retained state feedback.

6.9.1 Concepts and Distinctions: Textbook Ex. 6

Choose a Communication Interface by How the System Organizes Data

Question: compare typical differences and applications of MAVLink and uXRCE-DDS.

Aspect	MAVLink	uXRCE-DDS (PX4 v1.14 or later)
Organization	Messages/fields via a software development kit (SDK)	Client–agent access to the Data Distribution Service (DDS) space
Typical Use	Ground stations, Python, MAVSDK, etc.	PX4 / Robot Operating System 2 (ROS 2) message integration
Configuration	System IDs, ports, message rates, and fields	Message versions, agent, and quality of service (QoS)
Choose For	Reusing the existing MAVLink toolchain	ROS 2 data flow and node coordination

Key point: choose for the link, load, and configuration; measure latency. For DDS and QoS, see Section 6.7, “Onboard Software Communication” and “The uXRCE-DDS Route.”

Chapter Outline

Exercises and Classroom Discussion

Chapter 6
External Control and Trajectory Planning

6.1 External Control Fundamentals
6.2 Control Principles and Modes
6.3 Trajectory Models and Generation
6.4 Planning and Obstacle Avoidance
6.5 Tracking and Aggressive Flight Control
6.6 Frontier Control Technologies
6.7 Platform and Communication Interfaces
6.8 Examples and Experimental Validation
6.9 Exercises and Classroom Discussion

4 Units in This Section

6.9.1 Concepts and Distinctions
Textbook: Sec. 6.4, Exs. 1–3, 5–6

6.9.2 Trajectory Calculations
Textbook: Sec. 6.4, Exs. 8–10

6.9.3 Programs and Experiments
Textbook: Sec. 6.4, Exs. 4, 7–10

6.9.4 Integrated Tasks and Review
Textbook: Secs. 6.3.1, 6.4

6.9.2 Trajectory Calculations: Textbook Ex. 8 — Constraints

How Fast Should a 5 m Radius Circle Be Flown?

Question: design a circle of radius 5 m and explain linear-speed and acceleration constraints.

Given for this example:
Planar speed limit: 3 m/s
Planar acceleration limit: 1 m/s²

$$v = R|\omega|$$

$$a_n = R\omega^2 = \frac{v^2}{R}$$

R: radius; v: linear speed
ω: angular velocity; a_n: centripetal acceleration

Find: maximum angular speed, minimum period, and a choice with some margin.

Conditions: fixed altitude and circle centered at the local origin. Calculate the steady circular segment first; design entry and exit separately.

6.9.2 Trajectory Calculations: Textbook Ex. 8 — Calculation

Acceleration Is the Tighter Constraint in This Example

$$|\omega| \le \min\left(\frac{v_{\max}}{R}, \sqrt{\frac{a_{\max}}{R}}\right) = \min(0.6,\ 0.447) \approx 0.447\ \text{rad/s}$$

$$T \ge \frac{2\pi}{0.447} \approx 14.05\ \text{s}$$

Choose 0.4 rad/s
to retain some motion margin.

Chosen Angular Speed	Linear Speed	Centripetal Acceleration	Period
0.4 rad/s	2 m/s	0.8 m/s²	15.71 s

Key point: calculate the bound from each constraint, then choose the stricter one.

6.9.2 Trajectory Calculations: Textbook Ex. 8 — PVA Check

Use a Quarter Revolution to Check Position and Its Derivatives

Use north–east–down (NED) coordinates; angular speed 0.4 rad/s and fixed altitude 5 m.

$$\mathbf{p}_r = (5\cos 0.4\,t,\ 5\sin 0.4\,t,\ -5)^{\mathsf{T}}$$

$$\mathbf{v}_r = (-2\sin 0.4\,t,\ 2\cos 0.4\,t,\ 0)^{\mathsf{T}}$$

$$\mathbf{a}_r = (-0.8\cos 0.4\,t,\ -0.8\sin 0.4\,t,\ 0)^{\mathsf{T}}$$

Quarter revolution: t = (π/2)/0.4 ≈ 3.927 s

$$\mathbf{p}_r = (0,5,-5)^{\mathsf{T}}\ \mathrm{m}$$

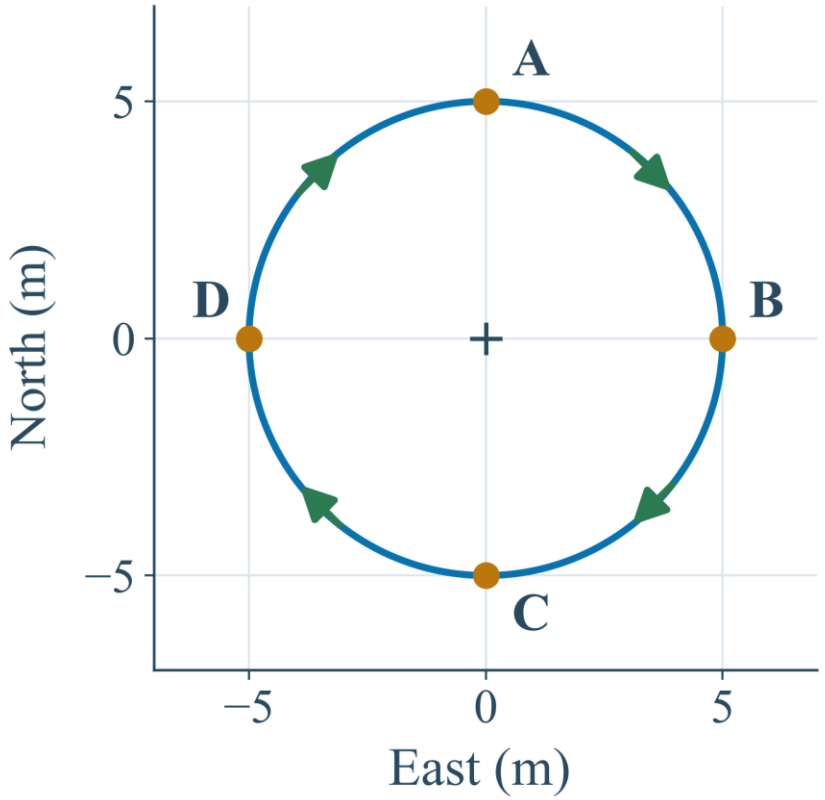
$$\mathbf{v}_r = (-2,0,0)^{\mathsf{T}}\ \mathrm{m/s}$$

$$\mathbf{a}_r = (0,-0.8,0)^{\mathsf{T}}\ \mathrm{m/s^2}$$

Check: at a quarter turn, position is 5 m east, velocity points south (negative north), and acceleration points to the center (negative east).

6.9.2 Trajectory Calculations: Textbook Ex. 8 — Full Reference

After Generating One Revolution, Check the Connecting Segments



Top-view circle; arrows show motion (teaching example)

Samples: A → B → C → D correspond to t = 0, T/4, T/2, 3T/4; all have radius 5 m.

Direction: north up, east right. Positive angular velocity turns from north to east, clockwise in top view.

Connections: position and velocity must be continuous from takeoff to circle and from steady motion to hover.

Experiment: see "Textbook Exs. 8 and 10 — Running the Examples" for the installed circle.

6.9.2 Trajectory Calculations: Textbook Ex. 10 — Analytical Design

Spiral Ascent: Add Climb to the Circular Reference

Question: design a spiral-ascent reference of radius 5 m centered on the origin.

$$\mathbf{p}_r = (5\cos 0.4\,t,\ 5\sin 0.4\,t,\ -2-0.5t)^{\mathsf{T}}\ \mathrm{m}$$

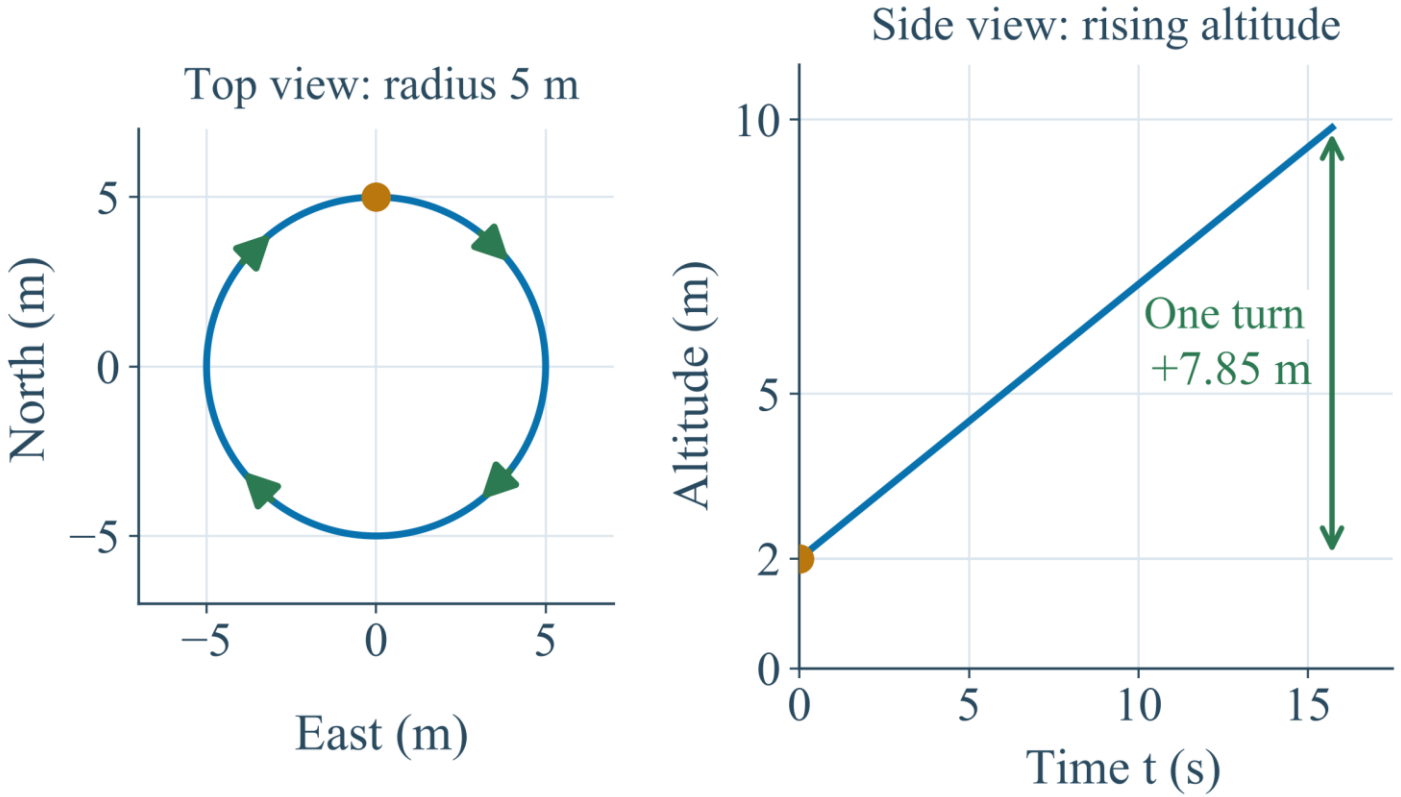
$$v_z = -0.5\ \mathrm{m/s}, \quad a_z = 0 \qquad \|\mathbf{v}_r\| = \sqrt{2^2+0.5^2} \approx 2.06\ \mathrm{m/s}$$

Selected here: initial altitude 2 m, climb 0.5 m/s, ω = 0.4 rad/s.
One revolution: 15.71 s and 7.85 m climb; the third coordinate falls from −2 m to about −9.85 m.

Check: planar speed 2 m/s ≤3 m/s; planar acceleration 0.8 m/s² ≤1 m/s². Total speed is about 2.06 m/s. Check vertical speed and climb entry/exit transitions separately.

6.9.2 Trajectory Calculations: Textbook Ex. 10 — 3D Check

A Circle in Top View Must Also Show Sustained Ascent from the Side



Spiral: top view and altitude vs. time (teaching example)

Top view: north/east coordinates form a 5 m radius circle; orange marks the start.

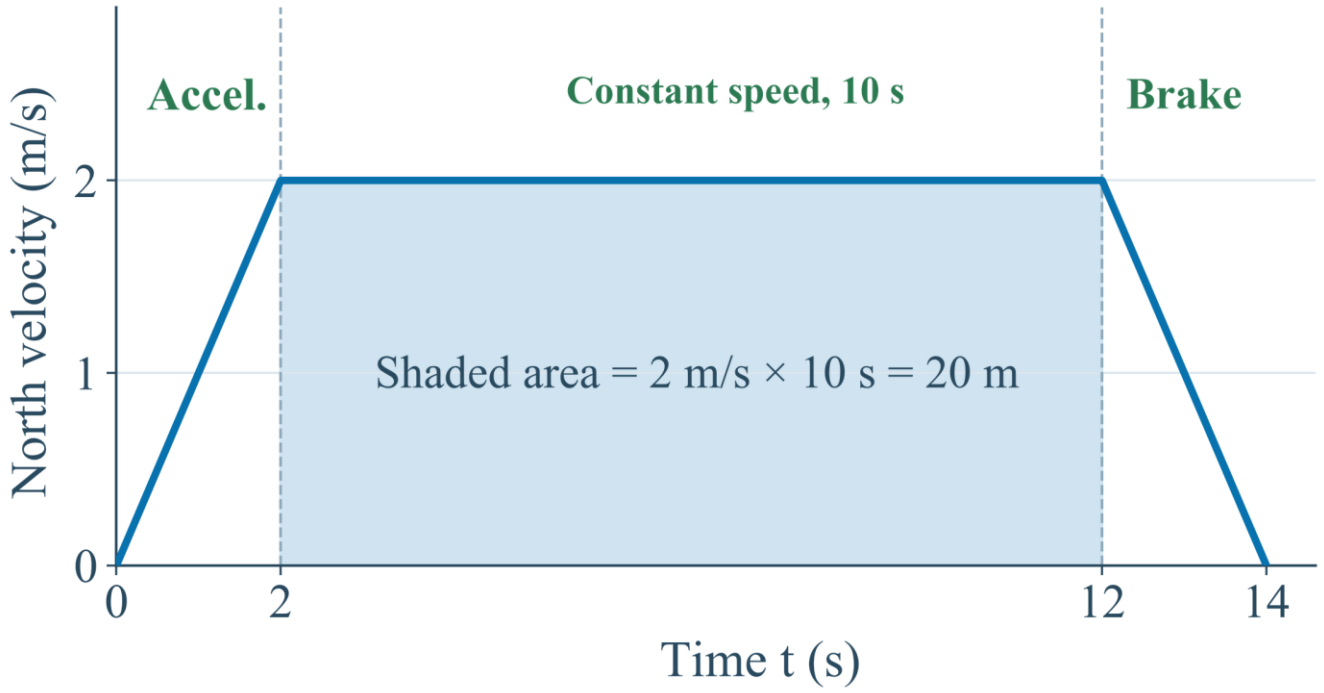
Altitude: about 7.85 m gained per revolution; plotted altitude is the negative of NED coordinate 3.

Experiment: use state feedback to check actual climb, planar radius, and mode.

6.9.2 Trajectory Calculations: Textbook Ex. 9 — Baseline Calculation

2 m/s for 10 s Gives a 20 m Constant-Speed Segment

Question: use RflySim Simulink modules to command forward flight at 2 m/s for 10 s.



Trapezoidal speed: teaching example, not platform data

$$\Delta x = \int_0^{10} 2\,\mathrm{d}t = 20\ \mathrm{m}$$

Direction: forward is north here; hold yaw fixed.

Stages: time takeoff, acceleration, constant speed, braking, and landing separately.

Acceptance: compare feedback displacement with the velocity integral to locate the stage where error arises.

Chapter Outline

Exercises and Classroom Discussion

Chapter 6
External Control and Trajectory Planning

6.1 External Control Fundamentals
6.2 Control Principles and Modes
6.3 Trajectory Models and Generation
6.4 Planning and Obstacle Avoidance
6.5 Tracking and Aggressive Flight Control
6.6 Frontier Control Technologies
6.7 Platform and Communication Interfaces
6.8 Examples and Experimental Validation
6.9 Exercises and Classroom Discussion

4 Units in This Section

6.9.1 Concepts and Distinctions
Textbook: Sec. 6.4, Exs. 1–3, 5–6

6.9.2 Trajectory Calculations
Textbook: Sec. 6.4, Exs. 8–10

6.9.3 Programs and Experiments
Textbook: Sec. 6.4, Exs. 4, 7–10

6.9.4 Integrated Tasks and Review
Textbook: Secs. 6.3.1, 6.4

6.9.3 Programs and Experiments: Textbook Ex. 4 — State Acquisition

Read State First, Then Use It to Judge the Task

Question: acquire flight state in real time through Python; first verify reading while disarmed.

```
import time
import PX4MavCtrlV4 as PX4MavCtrl
mav = PX4MavCtrl.PX4MavCtrler(1)
mav.InitMavLoop()
while True:
    print(mav.uavPosNED, mav.uavVelNED)
    time.sleep(0.1)
```

What is read: latest cached local NED position and velocity.

Who updates it: the SDK receiver thread. Print period is not the telemetry update period.

When usable: only after valid, fresh state arrives should it enter control.

Entry: [Installation directory]/RflySimAPIs/6.RflySimExtCtrl/0.ApiExps/e1_PX4MavCtrlAPITest/Readme.pdf
Run PX4MavCtrlAPITest.bat for software-in-the-loop (SITL; SIL in the textbook); wait for CopterSim initialization.
Run this script in Python38Run.bat; Ctrl+C stops it. Press Enter in the bat window to close simulation.

6.9.3 Programs and Experiments: Textbook Ex. 4 — Logging and Acceptance

Printing Numbers Does Not Establish Feedback Fit for Control

Record	Check	Common Misinterpretation
Position-message validity	The corresponding state was actually received	Treating initial cache values as valid localization
Sample and receive times	Data freshness and time alignment	Using print period to prove telemetry rate
Position, velocity, mode	Motion agrees with control mode	Saving targets without feedback
Packet loss, timeouts, final state	Task interruptions and recovery	Keeping only one 3D screenshot

Key point: use the local-position message's own time. Caches from different messages may represent different instants.

6.9.3 Programs and Experiments: Textbook Ex. 7 — Task Design

Define Task Stages for Hovering at (10, 0, −5) and Returning

Question: fly from the origin to hover at (10, 0, −5), then return; use local NED in m.

Stage	Position or Action	Transition Criterion
Takeoff	Reach (0, 0, −5)	Altitude established and speed reduced
Outbound / Hover	Reach (10, 0, −5) and dwell	Position error and speed meet thresholds
Return	Return to (0, 0, −5)	Stable above the starting point
End	Land at the origin	Confirm touchdown and disarming, then clean up connections

Key point: coordinate arrival, stable hover, overhead return and completed landing are distinct states.

6.9.3 Programs and Experiments: Textbook Ex. 7 — Arrival Criteria

Switch Legs by Feedback Conditions, Not a Fixed Wait

Teaching thresholds: position error ≤0.3 m and speed ≤0.2 m/s, maintained continuously for 2 s.

$$\|\mathbf{p} - \mathbf{p}_{\text{goal}}\| \le 0.3\ \text{m}, \quad \|\mathbf{v}\| \le 0.2\ \text{m/s}$$

Set entered=None for each leg. now is monotonic time (s); start timing when all conditions first hold.

```
if state_ok and pos_ok and vel_ok:
    if entered is None: entered = now
    if now - entered >= 2.0:
        advance_to_next_stage()
else:
    entered = None
```

pos_ok, vel_ok: the inequalities above.
state_ok: valid, timely, continuous state.
Interruption or timeout: reset the timer, pause progression, and enter the configured safety branch.

6.9.3 Programs and Experiments: Textbook Ex. 7 — Exit Sequence

Confirm the Vehicle Has Stopped Before Program Exit

Entry: initOffboard() → SendPosNED(x, y, z, yaw) → read back mode and armed state

Request Landing Observe Mode	→	Confirm Touchdown Confirm Disarming	→	Close Communication Save Logs

endOffboard(): stops Offboard setpoint transmission; it does not establish landing.

stopRun(): forces disarming if armed. Call only after confirmed touchdown.

initOffboard(): includes automatic arming and parameter changes; see Section 6.7, "SDK Side Effects."

Interface baseline: 6.RflySimExtCtrl/0.ApiExps/e1_PX4MavCtrlAPITest/PX4MavCtrlAPITest.py

Exercise 7: copy the script; replace timed takeoff/landing with "Task Design" and "Arrival Criteria."

6.9.3 Programs and Experiments: Textbook Exs. 4 and 7 — Checks

The Original Script Tests Connectivity; Exercise 7 Requires a Round Trip

Prerequisite: PX4MavCtrlAPITest.bat opens QGroundControl (QGC), CopterSim and two RflySim3D windows. Await GPS 3D fixed & EKF initialization finished (extended Kalman filter).

Exercise 4 Reading Script
About one printed line per 0.1 s; position and velocity each have north, east, and down components.

Original SDK Connectivity Baseline
Take off to about 1.7 m; after 10 s, descend at 0.2 m/s for about 10 s.
Exercise 7: separate round-trip adaptation.

Exercise 7: hover at (10, 0, −5), return to (0, 0, −5), then land at the origin.
Switch by arrival criteria; log feedback for 5 m altitude, the 10 m leg, hover, touchdown, and disarming.

Troubleshooting: missing module—check Python38Run.bat's environment; no arming/takeoff—check initialization and QGC connection.
All-zero position: verify reception, validity flags and timestamps before assuming the origin.

6.9.3 Programs and Experiments: Textbook Ex. 9 — Experiment Steps

Implement a 10 s Constant-Speed Segment in Simulink

Question: use RflySim Simulink modules to fly forward at 2 m/s for 10 s.
Entry: 6.RflySimExtCtrl/1.BasicExps/e2_VelCtrl/Velocity.slx

Step	Action	Acceptance
Prepare	Follow Readme.pdf to establish simulation and state estimation	Define mode, frame, and speed units
Establish Altitude	Take off and stabilize; hold yaw fixed	Start the 10 s timer after constant speed stabilizes
Constant Speed	Accelerate north to 2 m/s; once stable, time 10 s, then brake smoothly	Record reference and feedback velocity
Check and End	Check integral versus displacement; then land under control	About 20 m at constant speed; evaluate transitions separately

Forward means north here. For body-frame velocity, also specify yaw and the coordinate transform.

No arming: await initialization. No response: check Offboard and reasonable setpoints.

6.9.3 Programs and Experiments: Textbook Ex. 10 — Program Check

Check Spiral Derivatives Before Connecting the Control Interface

Check the reference offline first. After connecting, execute and end as in Section 6.7, "Termination and Exceptions."

```
theta = omega * t
x = 5 * math.cos(theta)
y = 5 * math.sin(theta)
z = -2 - 0.5 * t
vx = -5 * omega * math.sin(theta)
vy = 5 * omega * math.cos(theta)
ax, ay = -omega**2 * x, -omega**2 * y
vz, az = -0.5, 0.0
```

Direction: signed omega, positive or negative, must enter both phase and derivatives.

Self-check: compare central-difference velocity estimates with analytical velocity.

Data: save reference PVA, state feedback, and task stages separately.

Entry: 6.RflySimExtCtrl/2.AdvExps/e4.TrajPlanExps/4.SpiralTrajMotion/UAVCtrl.py
First correct X-velocity direction; see "Textbook Exs. 8 and 10 — Running the Examples."

6.9.3 Programs and Experiments: Textbook Ex. 8 — Parameter Adaptation

Adapt the 6 m Interface Baseline to the Textbook's 5 m Circle

Original SDK baseline: main() passes radius 6 m, altitude 10 m, clockwise=True.
The Exercise 8 copy uses radius 5 m and altitude 5 m, matching "Trajectory Calculations."

Exercise 8: edit the call in main(), not only the function defaults.
radius=5.0, height=-5.0, angular_velocity=0.4,
duration=30.0, clockwise=False, center_x=0.0, center_y=0.0

Start: the parameters give (5, 0, −5). Confirm arrival from feedback, then enter the circle smoothly.
clockwise=False gives direction=+1, matching the positive-angular-velocity theory example.

PVA check: remove the extra direction factor in X velocity; use the same elapsed for position and derivatives.
Use the three formula sets in "Textbook Ex. 8 — PVA Check"; verify a quarter revolution.

Interface: SendVelNEDNoYaw sends V only; P and A support reference checking and logging.
Printed position is still a reference. Add state feedback, smooth starts/stops, and controlled landing.

6.9.3 Programs and Experiments: Textbook Exs. 8 and 10 — Running the Examples

Circle and Spiral: Correct Copies, Then Run in Log Order

Directory: [Installation directory]/RflySimAPIs/6.RflySimExtCtrl/2.AdvExps/e4.TrajPlanExps/
Exercise 8: 3.CirularTrajMotion/; Exercise 10: 4.SpiralTrajMotion/; entry: UAVCtrl.py

Adapt: copy UAVCtrl.py. For Exercise 8, apply "Parameter Adaptation"; correct the extra direction in X velocity in both examples.
Add feedback logging, smooth starts/stops, and controlled landing. Derivatives: Section 6.5, "Circular Trajectory."

Start: run SITLRun.bat as administrator. It closes existing CopterSim, then opens QGC, CopterSim, and RflySim3D.

Wait: CopterSim reports GPS 3D fixed & EKF initialization finished.

Run: enter python UAVCtrl.py in Python38Run.bat. Press D, then T in RflySim3D for state and path.

Close: press any key in SITLRun.bat to close CopterSim, QGC, and RflySim3D.

6.9 Exercises and Classroom Discussion

6.9.3 Programs and Experiments: Textbook Exs. 8 and 10 — Expectations and Troubleshooting

Check Center, Radius, and Altitude from Feedback, Then the Final State

Exercise 8: Adapted 5 m Circle
Start (5, 0, −5), altitude 5 m
0.4 rad/s for 30 s, about 1.9 revolutions
Speed 2 m/s, acceleration 0.8 m/s²
Decelerate smoothly; land under control

Spiral: 4.SpiralTrajMotion
Arm and reach (5, 0, −10)
Radius 5 m, 0.6 rad/s, 40 s
Climb from altitude 10 m to 25 m
Decelerate, then call endOffboard()

After correction: Ex. 8 clockwise=False → clockwise; Ex. 10 default → counterclockwise.

Acceptance: Exercise 8 feedback has center at the origin, radius 5 m, altitude 5 m; compare with the 2 m/s and 0.8 m/s² reference.
Check limits of 3 m/s and 1 m/s². Both exercises require landing-mode, touchdown, and disarming confirmation.

Troubleshooting: no initialization log—check environment, startup logs, processes, and ports; connection failure—inspect firewall logs.
Only after confirmed false blocking, allow the trusted program specifically and retry SITLRun.bat.

Chapter Outline

Exercises and Classroom Discussion

Chapter 6
External Control and Trajectory Planning

6.1 External Control Fundamentals
6.2 Control Principles and Modes
6.3 Trajectory Models and Generation
6.4 Planning and Obstacle Avoidance
6.5 Tracking and Aggressive Flight Control
6.6 Frontier Control Technologies
6.7 Platform and Communication Interfaces
6.8 Examples and Experimental Validation
6.9 Exercises and Classroom Discussion

4 Units in This Section

6.9.1 Concepts and Distinctions
Textbook: Sec. 6.4, Exs. 1–3, 5–6

6.9.2 Trajectory Calculations
Textbook: Sec. 6.4, Exs. 8–10

6.9.3 Programs and Experiments
Textbook: Sec. 6.4, Exs. 4, 7–10

6.9.4 Integrated Tasks and Review
Textbook: Secs. 6.3.1, 6.4

6.9.4 Integrated Tasks and Review: Chapter Experiment List

Choose Experiment Entries by Learning Objective

Category	Entries Used Here	Deliverable Focus
Interface Learning	0.ApiExps/e1_PX4MavCtrlAPITest/	State, setpoints, modes, and times
Basic Usage	1.BasicExps/e1_PosCtrl/ 1.BasicExps/e2_VelCtrl/	Position/speed baselines and explainable motion results
Trajectory Tracking	1.BasicExps/ e9_TrajectoireFollowing_Segment/	Reference/response comparison Model/parameter comparison
Advanced Development	2.AdvExps/e4.TrajPlanExps/ 3.CirularTrajMotion/, 4.SpiralTrajMotion/	Reference generation, derivative consistency, and interface adaptation

Root: [Installation directory]/RflySimAPIs/6.RflySimExtCtrl/; full list: Index.pdf.

6.9.4 Integrated Tasks and Review: Task Design

Establish a Baseline, Then Choose One Extension

Common Baseline
Take Off, Reach a Point, Return

Required Records:
Reference, feedback, mode, final state

Optional Branch	Change Only One Condition
Trajectory and Speed	Same path; compare two completion times
Constraints and Avoidance	Keep motion capability; add known obstacles Directory: e11_Avoidance_Segment
Control Interface	Same reference; compare position and PVA methods

A reproducible baseline lets you attribute performance changes to a specific design.

6.9.4 Integrated Tasks and Review: Result Calculation

Evaluate Error Magnitude and Task Completion Together

Calculation: three position-error magnitudes paired at matching times are 0.1, 0.3, and 0.4 m. Find root-mean-square (RMS) and maximum error.

$$e_{\mathrm{RMS}} = \sqrt{\frac{0.1^2 + 0.3^2 + 0.4^2}{3}} \approx 0.294\ \mathrm{m}$$

$$e_{\max} = 0.4\ \mathrm{m}$$

Also Check:
Are constraints met? Is the task complete?

Alignment: pair reference and feedback by timestamp before calculating each error.

Stages: report takeoff, steady tracking, and landing separately; explain when and why errors occur.

6.9.4 Integrated Tasks and Review: Failure Analysis

Trace Unexpected Results Along the Information Flow

Reference	→	Command	→	Response	→	Final State
Frame, turn, derivatives		Message arrival, mode		Saturation, delay, error		Landing, touchdown and disarming

Position unchanged → command: check message times, reception records and current mode.

Opposite turn → reference: check frame, turn sign and derivatives; verify a quarter turn.

Deviation after acceleration → response: compare reference/feedback, limits, reference changes, and delay.

Still flying after exit → final state: check whether termination began, plus mode, touchdown, and armed state.

Order: reference → command → response → final state. An error upstream prevents explaining downstream behavior.

6.9.4 Integrated Tasks and Review: Integrated Task Deliverables

What Evidence Makes an Experiment Report Reproducible?

Explain	Submit	Self-Check
Task and Conditions	Initial state, path/target, thresholds, interface, version	Can it be compared again under the same conditions?
Input and Response	Reference/feedback paths, error and velocity curves	Do curves share frames and time?
Constraints and Final State	Minimum separation, saturation, mode, touchdown records	Required separation met? Landing complete?
Problems and Improvements	One single-factor comparison and causal analysis	Does the change explain the difference in results?

Classroom Exercise

Three Counterexamples Test Whether the Task Is Complete

Case 1: actual position always lies on the reference circle, but lags by a fixed angle.

Key point: path error may be zero while position error at the same time is nonzero.

Case 2: completion time doubles. Can better tracking be declared immediately?

Key point: first note the lower speed/acceleration demands, then compare results using the same metrics.

Case 3: the external program stops. Has the task ended safely?

Key point: use flight state to confirm mode, touchdown, and disarming.

Chapter Summary

External Control Turns Task Requirements into Verifiable Motion

Tasks and Roles	→	Setpoints/feedback	→	Trajectory limits	→	Tracking / Review
By responsibility		Continuous updates		Feasible → optimal		Error + final state

Concept: external control sets roles and interface boundaries; Offboard is a specific mode.

Methods: match remote, mission or Offboard to the task. Find feasible references by search, sampling or optimization. Artificial potential fields (APF), dynamic window approach (DWA) and model predictive control (MPC) use feedback for local avoidance. Evaluate them and learning methods with same-time feedback.

Boundaries: external targets enter the chosen input; the autopilot closes lower loops. Language-model or vision-language-action (VLA) outputs need constrained interfaces: frames, units, periods and conditions. Revalidate simulation results before real-vehicle use.

Pitfalls: reference ≠ measurement; message rate ≠ latency; collision-free samples ≠ a safe whole segment; stopped sending ≠ completed landing.

Next: stopping transmission cannot confirm safety. Chapter 7: anomaly detection/handling.

Autonomy involves generating actions, observing results, and making continuous corrections.