Chapter Outline

Trajectory Tracking Principles and Aggressive Flight Control

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

3 Units in This Section

6.5.1 Tracking and Error Compensation
Textbook: Sec. 6.1.3.5 (Steps 4–5)

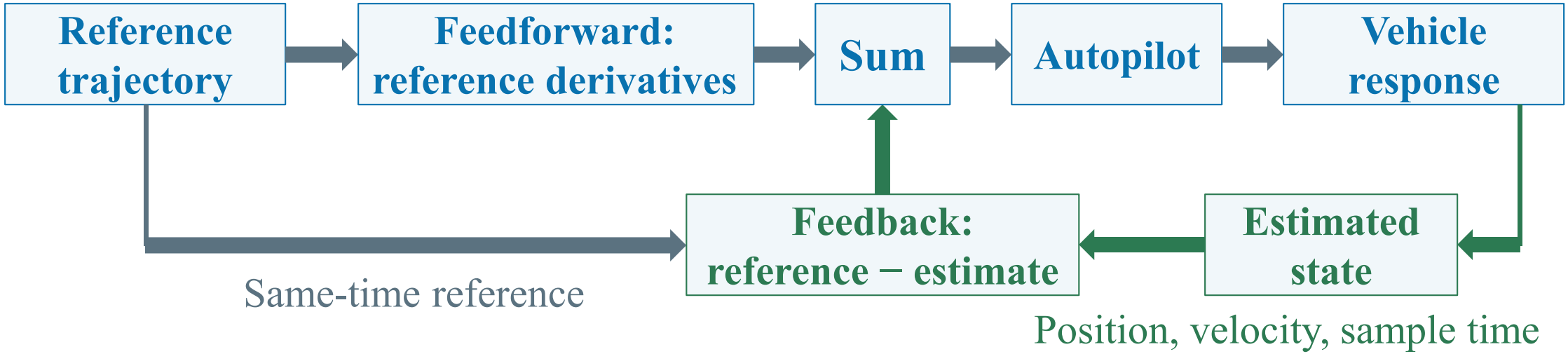
6.5.2 PVA Commands and Trajectory Consistency
Textbook: Sec. 6.1.3.6, Part 1
PVA = position, velocity, acceleration

6.5.3 Attitude, Thrust, and Aggressive Control
Textbook: Sec. 6.1.3.6, Parts 2–5

6.5.1 Tracking and Error Compensation: Feedback and Feedforward

Motion Demands and State Errors Follow Separate Branches

Reference trajectories specify motion; feedback reports actual deviations

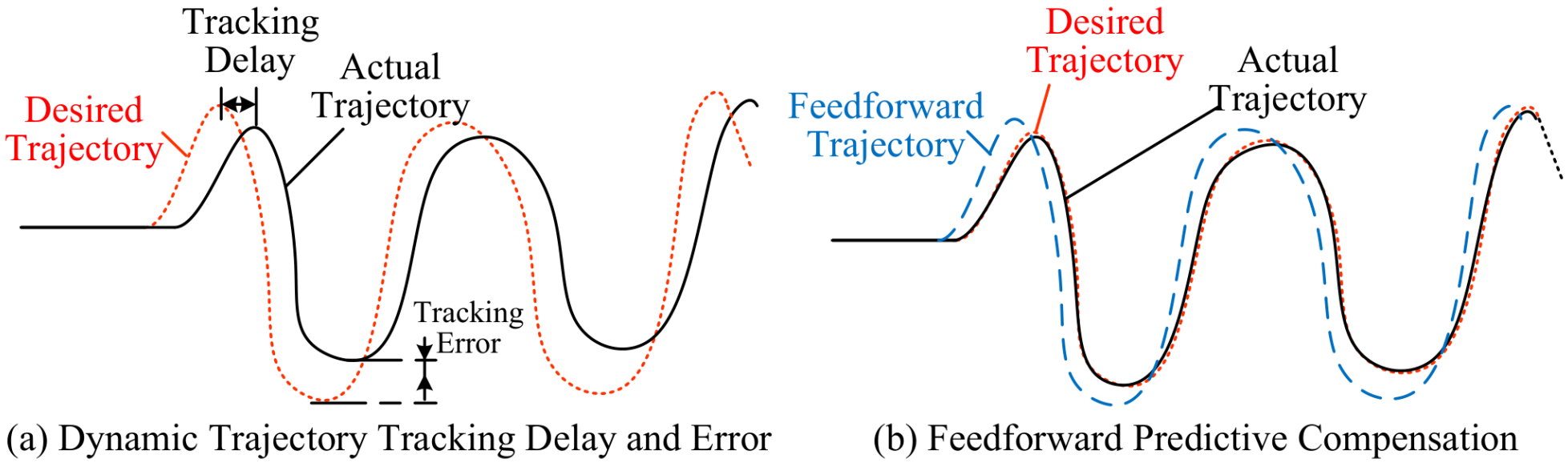


Feedforward: reference-position derivatives anticipate motion demands, mainly compensating for known dynamics.

Feedback: state errors drive corrections for initial offsets and external disturbances.

6.5.1 Tracking and Error Compensation: Tracking Delay

Trajectory Error Includes Timing and Amplitude Differences



(a) Dynamic Trajectory Tracking Delay and Error (b) Feedforward Predictive Compensation

Figure 6.6 Feedforward Correction for Trajectory Tracking Delay and Error

Red dashed: desired; black solid: response; blue dashed: compensated reference. Identified corrections advance the reference timing and adjust its amplitude.

6.5.1 Tracking and Error Compensation: One-Step Calculation

Velocity Feedforward Supplies Motion Even at Zero Error

$$\mathbf{e} = \mathbf{p}_d - \hat{\mathbf{p}}, \qquad \mathbf{v}_c = \mathbf{v}_d + K_p\mathbf{e}$$

Subscript d: reference; hat: estimate; c: command. Position: m; velocity: m/s.
North/east; ideal velocity execution. R: radius; ω: angular speed; K_p: proportional gain.

$$R = 5\,\text{m}, \quad \omega = 0.4\,\text{rad/s}, \quad K_p = 1\,\text{s}^{-1}$$

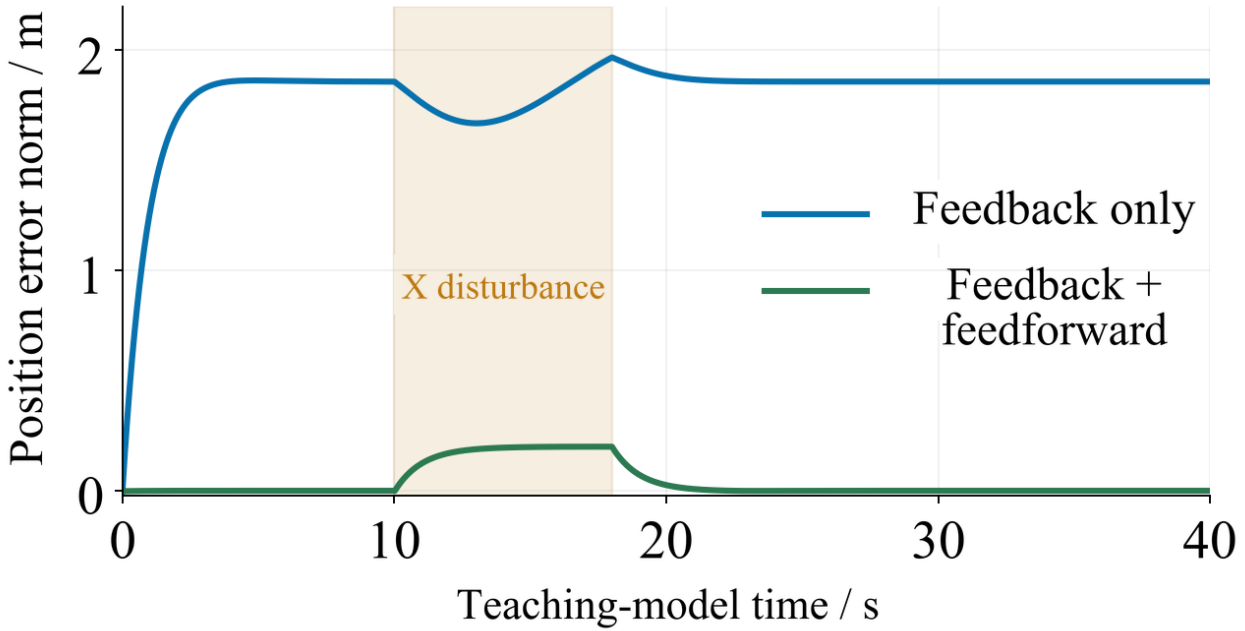
Reference at (5, 0) m; tangent points east: $\mathbf{v}_d = (0,\ R\omega)^{\mathrm{T}} = (0,\ 2)^{\mathrm{T}}$ m/s.

$$\mathbf{p}_d = \begin{bmatrix}5\\0\end{bmatrix}, \quad \hat{\mathbf{p}} = \begin{bmatrix}4.8\\-0.1\end{bmatrix}, \quad \mathbf{v}_d = \begin{bmatrix}0\\2\end{bmatrix} \quad \mathbf{v}_c = \begin{bmatrix}0\\2\end{bmatrix} + 1\begin{bmatrix}0.2\\0.1\end{bmatrix} = \begin{bmatrix}0.2\\2.1\end{bmatrix}\ \text{m/s}$$

Feedforward sustains reference motion; feedback corrects actual deviations.

6.5.1 Tracking and Error Compensation: Complete Teaching Calculation

Feedforward Greatly Reduces Dynamic Error on the Same Circle



Statistics: 5–40 s

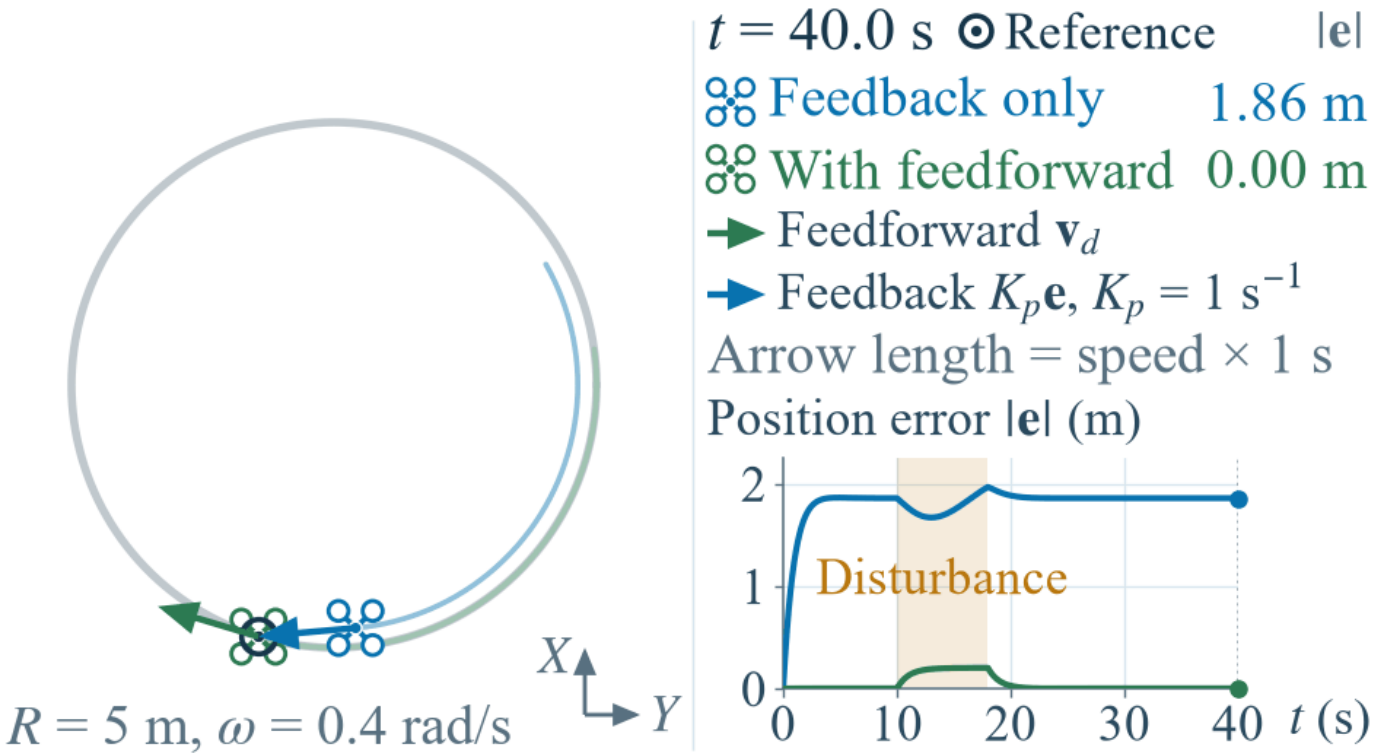
Feedback only
Root mean square error:
1.843 m

With velocity feedforward
Root mean square error:
0.090 m

Teaching calculation: radius 5 m; angular speed 0.4 rad/s; initial position (5, 0) m.
Ideal velocity execution; integration step 0.005 s; X-direction disturbance 0.2 m/s.

6.5.1 Tracking and Error Compensation: Tracking Comparison

Feedback Alone Needs Sustained Lag; Feedforward Removes It



Circular tracking with/without feedforward
(teaching calculation, not measured)

Feedback only: command = $K_p\mathbf{e}$. Zero error gives zero command; about 1.86 m of sustained lag supplies circular velocity.

Feedback + feedforward: tangential velocity $\mathbf{v}_d$ remains at zero error; error is near 0 before the disturbance.

X disturbance at 10–18 s: at about 0.2 m deviation, $K_p\mathbf{e}$ balances the disturbance. Error returns to zero after removal.

6.5.1 Tracking and Error Compensation: Error Prediction

Future References Can Approximately Compensate Known Delay

$$\mathbf{p}_c(t) = \mathbf{p}_d(t+\hat{\tau}) \approx \mathbf{p}_d(t) + \hat{\tau}\mathbf{v}_d(t) + \frac{1}{2}\hat{\tau}^2\mathbf{a}_d(t)$$

Under a pure-delay approximation, estimated delay sets the advance. Velocity corrects first-order displacement; acceleration corrects curvature.

$$e_{\text{delay}} = 2R\left|\sin\frac{\omega\tau}{2}\right| \approx R\omega\tau$$

$$R = 5\,\text{m},\ \omega = 0.4\,\text{rad/s},\ \tau = 0.2\,\text{s} \quad \Rightarrow \quad e_{\text{delay}} \approx 0.400\,\text{m}$$

To first order, velocity feedforward advances reference by $1/K_p$; Fig. 6.6 adjusts amplitude.

Advance compensation needs a delay estimate; feedback corrects residual error.

6.5.1 Tracking and Error Compensation: Error Sources

Compensation Depends on Dynamics and Available Control Margin

Observed behavior	Main cause	Remedy
Response arrives late	Sampling, computation, execution lag	Align clocks; estimate dynamic delay
Amplitude too large/small	Model gain differs from actual gain	Identify frequency response before correction
Command jitter	State noise; unsmooth derivatives	Filter/smooth derivatives; recheck phase
Tracking degrades in turns	Tilt, thrust, or speed saturation	Reduce demand; retain control margin

Hold references and constraints fixed; lower compensated error must still come with executable commands.

Chapter Outline

Trajectory Tracking Principles and Aggressive Flight Control

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

3 Units in This Section

6.5.1 Tracking and Error Compensation
Textbook: Sec. 6.1.3.5 (Steps 4–5)

6.5.2 PVA Commands and Trajectory Consistency
Textbook: Sec. 6.1.3.6, Part 1
PVA = position, velocity, acceleration

6.5.3 Attitude, Thrust, and Aggressive Control
Textbook: Sec. 6.1.3.6, Parts 2–5

6.5.2 PVA Commands and Trajectory Consistency: Interface Inputs

PVA Comes from Successive Derivatives of One Trajectory

$$\mathbf{v}_d(t) = \dot{\mathbf{p}}_d(t), \qquad \mathbf{a}_d(t) = \ddot{\mathbf{p}}_d(t) \qquad (6.9)$$

$$\mathbf{u}_{\text{ext}}(t) = \begin{bmatrix} \mathbf{p}_d(t) \\ \mathbf{v}_d(t) \\ \mathbf{a}_d(t) \end{bmatrix} \qquad (6.10)$$

P: position, in m.

V: velocity, in m/s.

A: acceleration, in m/s².

North-east-down (NED) frame: X north, Y east, Z down.
Select yaw angle or yaw rate separately, in rad or rad/s.

Position, velocity, acceleration (PVA) share a time, frame, and derivative relationship.

6.5.2 PVA Commands and Trajectory Consistency: Circular Trajectory

A Signed Phase Keeps Circular Position and Derivatives Consistent

$$\theta(t) = s\omega t, \quad s \in \{-1,1\}, \quad \omega > 0$$

$$\mathbf{p}_d = \begin{bmatrix} x_c + R\cos\theta \\ y_c + R\sin\theta \\ z_0 \end{bmatrix}, \quad \mathbf{v}_d = \begin{bmatrix} -sR\omega\sin\theta \\ sR\omega\cos\theta \\ 0 \end{bmatrix}$$

$$\mathbf{a}_d = \begin{bmatrix} -R\omega^2\cos\theta \\ -R\omega^2\sin\theta \\ 0 \end{bmatrix}, \quad \|\mathbf{v}_d\| = R\omega, \quad \|\mathbf{a}_d\| = R\omega^2$$

Given center, radius R, and constant altitude. Positive: north to east (clockwise from above). Extra direction factor in installed X velocity reverses its position derivative; see code reading.

6.5.2 PVA Commands and Trajectory Consistency: PVA Calculation

Check Position and Motion Demands at the Same Sample Time

Given: center (0, 0) m; radius 5 m; angular speed 0.4 rad/s; Z = −10 m; positive rotation.

$$t = \frac{\pi}{4\omega} = 1.9635\,\text{s}, \qquad \theta = \frac{\pi}{4}$$

$$\mathbf{p}_d = \begin{bmatrix} 3.5355 \\ 3.5355 \\ -10 \end{bmatrix}\text{m}, \quad \mathbf{v}_d = \begin{bmatrix} -1.4142 \\ 1.4142 \\ 0 \end{bmatrix}\text{m/s}, \quad \mathbf{a}_d = \begin{bmatrix} -0.5657 \\ -0.5657 \\ 0 \end{bmatrix}\text{m/s}^2$$

$$v = 2.0\ \text{m/s}, \quad a = 0.8\ \text{m/s}^2; \qquad \omega \times 2 \Rightarrow v \times 2,\ a \times 4$$

Faster trajectories are first limited by acceleration and actuator margin.

6.5.2 PVA Commands and Trajectory Consistency: Installed SDK

Explicitly Select the Mask and Feedforward Inputs for PVA

```
mask = PX4MavCtrl.PosTypeMask(ignore_all=True)
mask.enPos_x = mask.enPos_y = mask.enPos_z = True
mask.enVel_x = mask.enVel_y = mask.enVel_z = True
mask.enAcc_x = mask.enAcc_y = mask.enAcc_z = True
mav.SendOffAll(type_mask=mask, coordinate_frame=1,
        pos=p_d, vel=v_d, acc=a_d)
```

Assumes an active position-target stream; frame 1 is LOCAL_NED.
P controls position; V and A provide feedforward. Yaw unused; acceleration is not force.
MAVLink selects channels with type_mask; ROS 2 topics mark unused states with NaN.

6.5.2 PVA Commands and Trajectory Consistency: Reading the Example

The Circular Example's Velocity Conflicts with Its Printed Reference

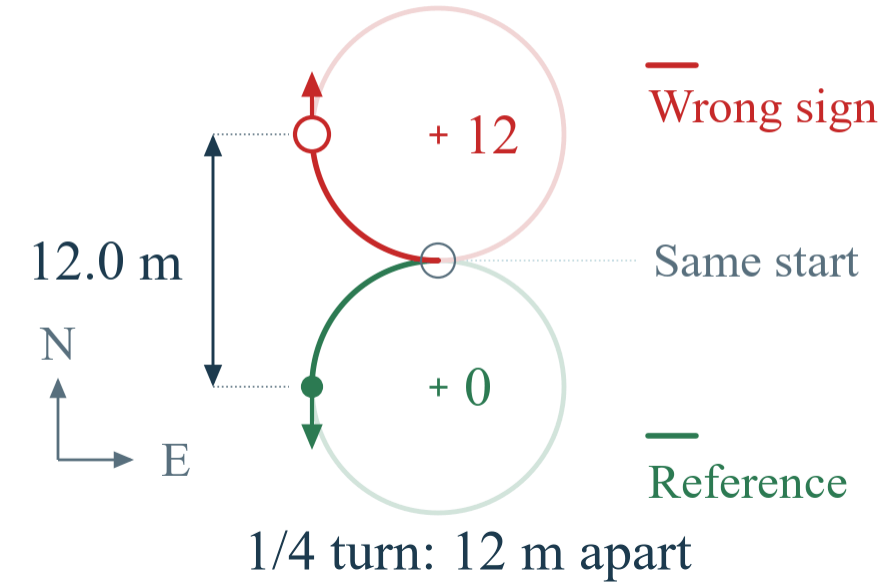
Entry: [installation directory]/RflySimAPIs/6.RflySimExtCtrl/
2.AdvExps/e4.TrajPlanExps/3.CirularTrajMotion/UAVCtrl.py

Original position	Consistent derivative	Original velocity issue
$x_d = x_c + R\cos(\omega t)$	$v_x = -R\omega\sin(\omega t)$	Extra direction factor in X
$y_d = y_c + dR\sin(\omega t)$	$v_y = dR\omega\cos(\omega t)$	Y agrees with position

d = direction. clockwise=True gives d = s = −1 (counterclockwise from above).
current_x/current_y are references; only velocity is sent, with no position-error feedback.

6.5.2 PVA Commands and Trajectory Consistency: Code Counterexample

A Sign Error Misses the Target Circle Even with Ideal Integration



Parameters
Radius: 6 m
Angular speed: 0.4 rad/s
Initial position: (6, 0) m

Quarter circle
Reference: (0, −6) m
Original: (12, −6) m
Deviation: 12 m

Code-consistency counterexample: analytic integration, ideal velocity execution, offline data.

6.5.2 PVA Commands and Trajectory Consistency: Helical Trajectory

Climbing Along a Helix Requires Negative Vertical Velocity in NED

Entry: 6.RflySimExtCtrl/2.AdvExps/e4.TrajPlanExps/4.SpiralTrajMotion/UAVCtrl.py
Source parameters: radius 5 m; angular speed 0.6 rad/s; duration 40 s.

$$z_d(t) = z_0 + \frac{z_f - z_0}{T}t, \quad v_z = \frac{z_f - z_0}{T}, \quad a_z = 0$$

$$v_z = \frac{-25 - (-10)}{40} = -0.375\ \mathrm{m/s}, \qquad z_d(20) = -17.5\,\mathrm{m}$$

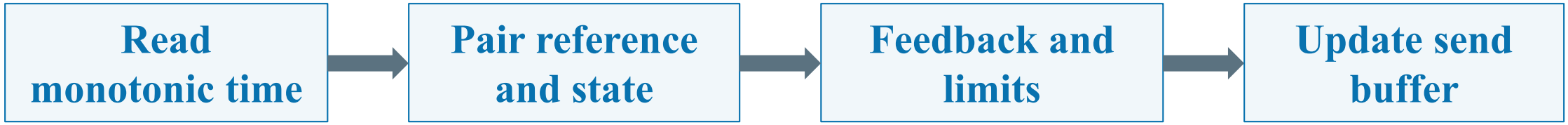
$$v_{xy} = R\omega = 3.0\ \mathrm{m/s}, \quad a_{xy} = R\omega^2 = 1.8\ \mathrm{m/s^2}$$

NED Z is positive downward; negative vertical velocity means climbing.

6.5 Tracking and Aggressive Flight Control

6.5.2 PVA Commands and Trajectory Consistency: Loop Implementation

Reference Updates, Background Sending, and Sampling Have Separate Clocks



Timing quantity	Meaning in this example/code	How to record
Reference update period	1/30 s loop sleep + computation	Host monotonic time (never runs backward)
SDK background sending	Full: ~100 Hz; Simple: ~30 Hz. Repeats cache until next reference.	Actual packet times and sequence numbers
Position source timestamp	Timestamp within the position message	Retain message time_boot_ms

6.5.2 PVA Commands and Trajectory Consistency: Terminal Reference

Construct a Continuous Deceleration Segment Before Trajectory End

$$\sigma = \frac{t - t_b}{T_b} \in [0,1], \quad \alpha(\sigma) = 1 - 3\sigma^2 + 2\sigma^3$$

$$\mathbf{v}_d(t) = \alpha(\sigma)\mathbf{v}_b, \qquad \mathbf{p}_d(t) = \mathbf{p}_b + \int_{t_b}^{t} \mathbf{v}_d(\xi)\, d\xi$$

Segment entry: t_b is start time; $T_b > 0$ is deceleration duration (s); $\mathbf{p}_b, \mathbf{v}_b$ are entry reference position and velocity.

Motion constraints: include stopping distance; check acceleration transitions, motion limits, and obstacle clearance.

Segment exit: keep sending zero velocity or a hover target; judge arrival from actual speed and error.

Chapter Outline

Trajectory Tracking Principles and Aggressive Flight Control

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

3 Units in This Section

6.5.1 Tracking and Error Compensation
Textbook: Sec. 6.1.3.5 (Steps 4–5)

6.5.2 PVA Commands and Trajectory Consistency
Textbook: Sec. 6.1.3.6, Part 1
PVA = position, velocity, acceleration

6.5.3 Attitude, Thrust, and Aggressive Control
Textbook: Sec. 6.1.3.6, Parts 2–5

6.5.3 Attitude, Thrust, and Aggressive Control: Responsibilities

External Code Computes Attitude and Thrust; the Autopilot Closes Inner Loops

Stage	Computation	Output (responsible component)
Error and feedforward	Position/velocity error + reference acceleration	Desired resultant force (external program)
Construct attitude	Force direction + desired yaw	Target attitude and thrust (external program)
Attitude inner loop	Attitude and angular-rate error feedback	Torque demand (autopilot)
Control allocation	Allocate torque and thrust to motors	Motor outputs (autopilot)

Inputs: Slides 6.2, "Position, Velocity, Attitude, and Force". Here: attitude + thrust.
Combinations depend on firmware/vehicle; see Slides 5.5, "Control Outputs Across Versions".

6.5.3 Attitude, Thrust, and Aggressive Control: Input Representation

Attitude and Thrust Together Define the External Input

$$\mathbf{u}_{\text{ext}} = \begin{bmatrix} \mathbf{q}_d \text{ or } \mathbf{R}_d \\ f_d \text{ or } \mathbf{f}_d \\ {}^{b}\boldsymbol{\omega}_d \text{ (optional)} \end{bmatrix} \quad (6.11)$$

Attitude: unit quaternion or rotation matrix.

Thrust: physical force in N; direction is limited by vehicle configuration.

Body angular velocity: optional, in rad/s.

For a conventional quadrotor, attitude sets thrust direction; a scalar sets magnitude. Physical thrust and normalized thrust are distinct; mapping depends on calibration and firmware.

Position and attitude commands must satisfy the same force constraints.

6.5.3 Attitude, Thrust, and Aggressive Control: Attitude/Thrust Mapping

Target Acceleration Determines Thrust Direction and Magnitude

$$m\ddot{\mathbf{p}} = mg\mathbf{e}_3 - f\mathbf{R}\mathbf{e}_3, \qquad \mathbf{e}_3 = (0,0,1)^T$$

$$\mathbf{a}_c = \mathbf{a}_d + K_p(\mathbf{p}_d - \hat{\mathbf{p}}) + K_v(\mathbf{v}_d - \hat{\mathbf{v}})$$

$$f_d = m\|g\mathbf{e}_3 - \mathbf{a}_c\|, \qquad \mathbf{b}_{3d} = \frac{g\mathbf{e}_3 - \mathbf{a}_c}{\|g\mathbf{e}_3 - \mathbf{a}_c\|}$$

R rotates body to NED; $\mathbf{b}_{3d}$ is the desired body down axis; thrust acts opposite to it. m: mass. g: acceleration due to gravity. No drag; thrust nonzero; down axis not parallel to yaw direction. Given yaw, construct attitude below; convert to a unit quaternion (w,x,y,z):

$$\mathbf{b}_{1c} = (\cos\psi_d, \sin\psi_d, 0)^T,\ \mathbf{b}_{2d} = \frac{\mathbf{b}_{3d} \times \mathbf{b}_{1c}}{\|\mathbf{b}_{3d} \times \mathbf{b}_{1c}\|},\ \mathbf{b}_{1d} = \mathbf{b}_{2d} \times \mathbf{b}_{3d},\ \mathbf{R}_d = [\mathbf{b}_{1d}\ \mathbf{b}_{2d}\ \mathbf{b}_{3d}]$$

Here K_p, K_v are position/velocity feedback gains, in s^{-2} and s^{-1}, respectively.

6.5.3 Attitude, Thrust, and Aggressive Control: Numerical Example

Horizontal Acceleration at Constant Altitude Requires More Total Thrust

Given: mass 1.5 kg; gravity 9.81 m/s²; constant altitude; northward acceleration 2 m/s².
F_N: northward force; F_{up}: upward force; β: thrust tilt from vertical.

$$F_N = ma_N = 3.0\,\text{N}, \qquad F_{\text{up}} = mg = 14.715\,\text{N}$$

$$f_d = \sqrt{F_N^2 + F_{\text{up}}^2} = 15.018\,\text{N}$$

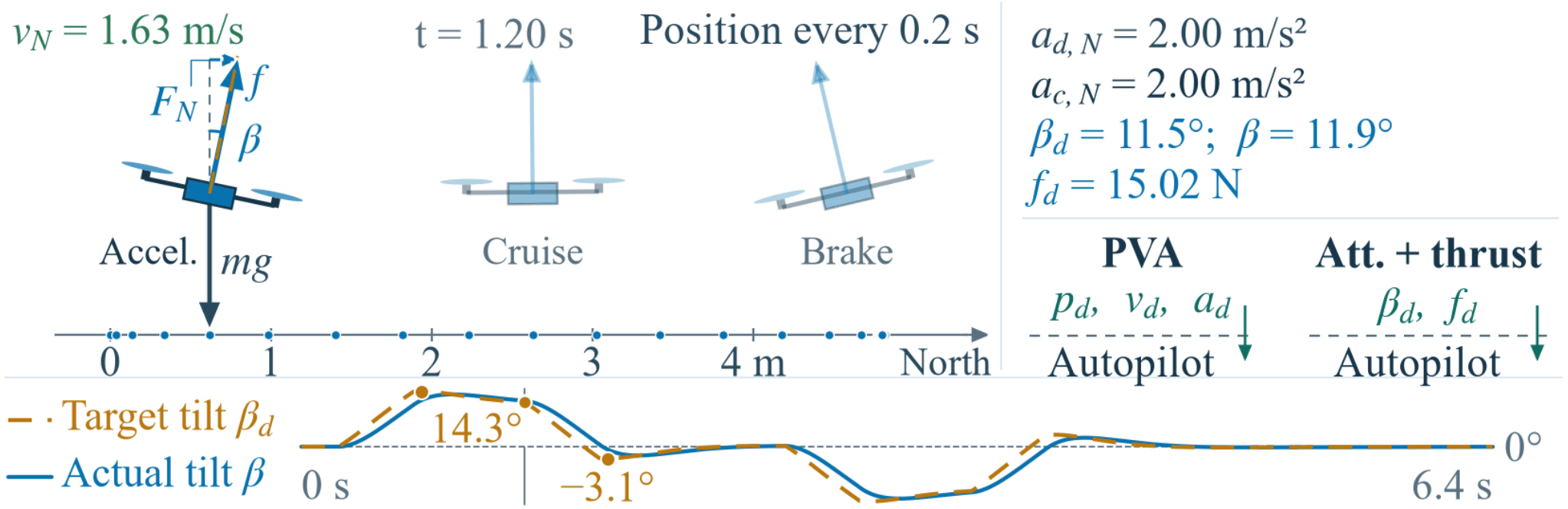
$$\beta = \arctan\frac{F_N}{F_{\text{up}}} = 11.52^\circ$$

Tilting generates horizontal acceleration and changes the thrust needed to hold altitude.

6.5.3 Attitude, Thrust, and Aggressive Control: Acceleration and Braking

With an Attitude/Thrust Interface, External Code Computes Both

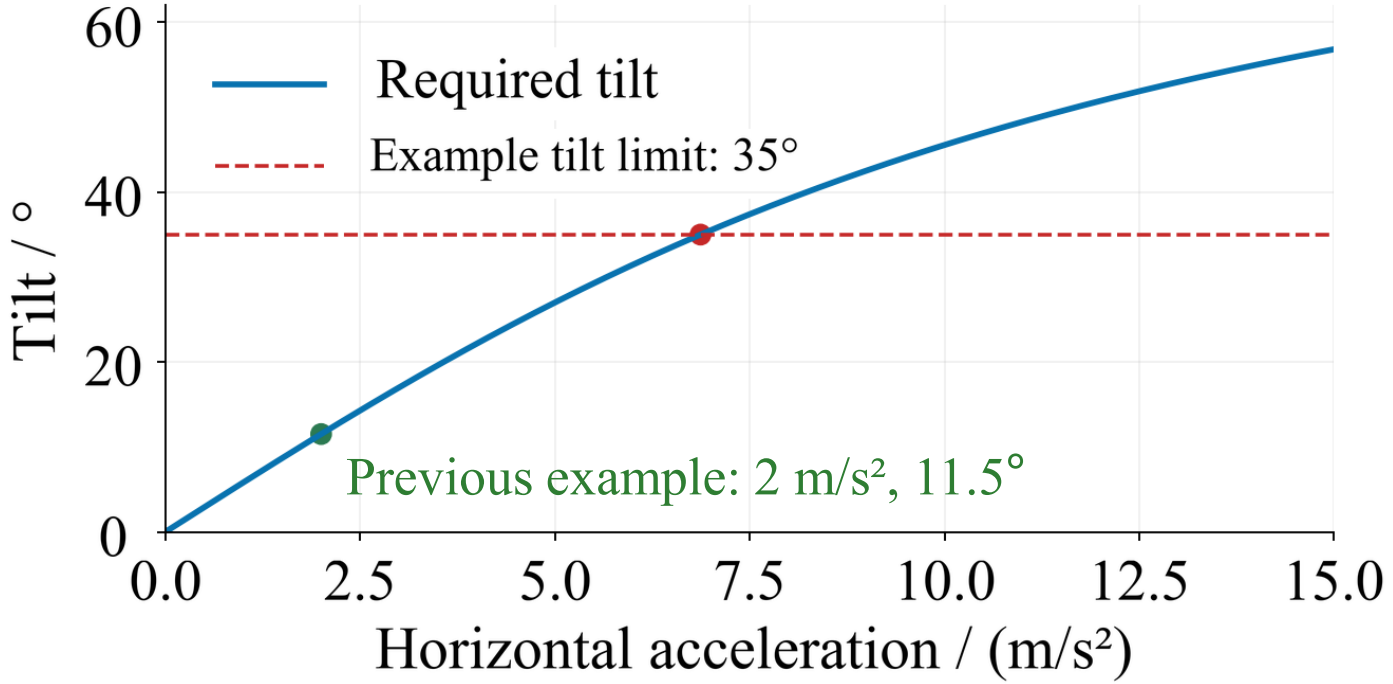
One computation, two control interfaces



PVA reference to tilt and total thrust during acceleration and braking.
Position/velocity feedback briefly sets target tilt to 14.3° for attitude lag or −3.1° for speed overshoot.

6.5.3 Attitude, Thrust, and Aggressive Control: Motion Constraints

A Tilt Limit Sets a Feasible Acceleration Boundary



$$a_h = g \tan \beta$$

Example limit: 35°
Corresponds to 6.869 m/s²
Also check total thrust and transient dynamics.

Teaching parameter sweep: constant altitude, no drag; stay within the vehicle flight envelope.

6.5.3 Attitude, Thrust, and Aggressive Control: Throttle Mapping

Calibrate Physical Thrust Before Converting to a Normalized Command

$$\eta = \mathcal{F}^{-1}(f_d), \qquad 0 \le \eta \le 1$$

$$\eta \approx \eta_h \frac{f_d}{mg} = 0.5 \times \frac{15.018}{14.715} \approx 0.510$$

```
mav.SendAttPX4(att=q_d, thrust=eta, CtrlFlag=2, AltFlg=0)
```

Valid connection required. q_d: unit quaternion (w,x,y,z); CtrlFlag=2 uses it directly.
Mapping: $\mathcal{F}$ maps normalized η to force; AltFlg=0 means thrust=eta is normalized thrust.
Calibration: η_h is the hover command (0.5 here); actual mapping depends on propulsion and firmware.
Range: linear approximation near hover; nonlinear mapping for highly dynamic flight.

6.5.3 Attitude, Thrust, and Aggressive Control: Differential Flatness

Flat Outputs Map Trajectory Derivatives to States and Inputs

$$\boldsymbol{\sigma} = (x, y, z, \psi)^T$$

Position, yaw ψ, and their finite-order derivatives determine all states and control inputs.

Position derivative	Motion quantity	Role in the ideal quadrotor mapping
First	Velocity	Translational state and velocity feedforward
Second	Acceleration	With gravity, determines thrust and attitude
Third	Jerk	Body-angular-velocity feedforward
Fourth	Snap	With yaw derivatives, gives angular acceleration and torque

Conditions: valid model, nonzero thrust, nondegenerate attitude construction, actuator constraints met.

6.5.3 Attitude, Thrust, and Aggressive Control: SE(3) Control

Geometric Control Compares Rotations and Position Errors Directly

Special Euclidean group SE(3): describes 3D position and attitude together.
Rotation matrices or unit quaternions avoid Euler-coordinate singularities.

Translation error + feedforward → **Desired force and attitude** → **Attitude/angular-rate error feedback**

Attitude error: $\mathbf{e}_R = \frac{1}{2}(\mathbf{R}_d^T\mathbf{R} - \mathbf{R}^T\mathbf{R}_d)^\vee$; $\vee$ maps a skew-symmetric matrix to a 3D vector.

Higher-order feedforward: jerk gives attitude-change demands; snap supports torque-demand calculation.

Tradeoff: direct computation still needs physical parameters, state estimates and input limits.

Tracking evaluation: monitor position/attitude errors, saturation fraction and computation time.

6.5.3 Attitude, Thrust, and Aggressive Control: Nonlinear Model Predictive Control (NMPC)

Optimize Tracking Error and Input Constraints over a Short Horizon

$$\min_{\{\mathbf{u}_j\}} \sum_{j=0}^{H-1} \left(\|\mathbf{x}_j - \mathbf{x}_{d,j}\|_Q^2 + \|\mathbf{u}_j\|_{R_u}^2 \right)$$

State $\mathbf{x}$, control $\mathbf{u}$, horizon H; Q and R_u are weighting matrices.
Constraints include discrete dynamics and thrust/angular-rate limits.

Start from estimated state → **Optimize a short input sequence** → **Execute first step**

Recede and repeat: solve again after receiving a new state.
Open-source acados supports real-time solving; compute budget, local convergence, and model error determine usability.

6.5.3 Attitude, Thrust, and Aggressive Control: Deep Reinforcement Learning (DRL)

A Learned Policy Still Needs Explicit Control Outputs



Learning mechanism: rewards optimize policy parameters; the deployed network computes actions.

Swift: a racing system that beat human champions; visual-inertial estimation plus a learned policy outputs thrust and angular velocity.

Practical cost: substantial training/data demands; sensitive to distribution shift, model mismatch, and action constraints.

Evaluation scope: report task conditions, success rate, error, and failure types; avoid a universal performance ranking.

6.5.3 Attitude, Thrust, and Aggressive Control: Performance Analysis

Compare Controllers under the Same Task and Execution Conditions

Method	Main computation	Conditions affecting performance
SE(3) geometric control	Geometric error feedback and higher-order feedforward	Dynamics parameters, inner-loop response, saturation handling
NMPC	Receding-horizon optimization of states and inputs	Model accuracy, constraints, solving successfully before deadline
DRL	Policy inference from observations to actions	Training coverage, deployment distribution, action limits

Comparable results need the same task, constraints, and complete failure records.

6.5.3 Attitude, Thrust, and Aggressive Control: Real-Time Implementation

Sum End-to-End Delay Along the Chain; Report Loop Period Separately

Chain stage	Teaching budget	Main effect
Estimate available after sampling	6 ms	Time from sampling to usable state
Control computation	8 ms	Algorithm and real-time scheduling
Transport and decoding	4 ms	Link, queues, and protocol
Inner loops and actuators	10 ms	Attitude, rotor-speed, and mechanical response

End-to-end delay: state sampling to effective action. See Slides 6.1, "Frequency and Latency".

$$\tau_{\text{total}} = 6 + 8 + 4 + 10 = 28\,\text{ms} \quad \neq \quad T_{\text{loop}} = 20\,\text{ms}$$

Section Summary

Consistent References, Feasible Tracking, Real-Time Computing Set Performance

Consistent references: derive PVA from one trajectory, with common coordinates, time, and units.

Effective feedback loop: feedforward supplies known motion demands; feedback corrects residual error. Retain control margin.

Interface adaptation: attitude and thrust are coupled; calibrate normalized commands. Compare algorithms under stated conditions.

Real-time operation: log loop period and end-to-end delay separately; verify each stage and worst-case computation time.