Chapter Outline

Exercises and Classroom Discussion

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.9.1 Core Concepts and Method Comparisons

Textbook: Sec. 10.4, Questions 1–5

10.9.2 Formation Design and Experiment Deployment

Textbook: Sec. 10.4, Questions 6–7

10.9.3 Learning Policies and Scaling

Textbook: Sec. 10.4, Questions 8–9

10.9.4 Engineering Strategies and Chapter Review

Textbook: Sec. 10.4, Question 10; chapter review

Exercises and Discussion

Q1: What Makes Several Agents a Cooperative System?

What are the core properties and research priorities of multi-agent and unmanned swarm systems?

Core Property	Manifestation	Research Question
Autonomous agents	Each acquires information and acts	Heterogeneous capabilities, control and adaptation
Local interaction and coupling	Neighbor information affects local decisions	Communication topology and coordination rules
An overall task	Task division, formation or shared coverage	Task allocation, fault tolerance and swarm performance

Cooperation: allocate tasks, maintain formations, share information and reconfigure after faults.

Game theory: model decision-makers, their responses and interacting policies.

Exercises and Discussion

Q2: Three Ways to Organize Control Computation

Compare centralized, distributed and hybrid control and give typical use cases.

Architecture	Computation and Information	Use Case and Cost
Centralized	A center collects states and computes commands for all vehicles	Formation in a managed area; center workload and link dependence
Distributed	Each vehicle uses its own and its neighbors' information	Distributed coverage; coordination with local information
Hybrid	Higher levels assign tasks; local groups coordinate autonomously	Inspection by several groups; resolve conflicts between levels

Reference: Slides 10.1; textbook Sec. 10.1.2.1.

Exercises and Discussion

Q3: Which Variables Must Reach Consensus?

How can a consensus protocol support distributed formation control? State its mathematical basis.

$$u_i = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i), \qquad \dot{x} = -Lx$$

Assumptions: fixed, undirected, connected graph; symmetric nonnegative weights; first-order integrator dynamics.

Convergence: the zero mode represents a common state; other modes decay. The limit is the initial average.

For formation control, seek consensus in $z_i = p_i - \delta_i$ to retain the desired offsets.

Reference Points

Q3 Example: From Neighbor Differences to One Update



$$u = (2 - 0,\ (0 - 2) + (4 - 2),\ 2 - 4)^T = (2,0,-2)^T$$

$$x^+ = x + 0.1u = (0.2,2,3.8)^T \text{ m}$$

Check: the endpoints approach the middle, the average remains 2 m, and the update signs are correct.

Exercises and Discussion

Q4: MADDPG During Training and Execution

Which learning architecture does multi-agent deep deterministic policy gradient (MADDPG) use?

Phase	Available Information	Computation
Centralized training	Joint observations or state and joint actions	Each critic evaluates actions to update its actor
Decentralized execution	Local observation and permitted neighbor messages	Each actor produces its own action

CTDE = centralized training with decentralized execution. A critic estimates value; an actor generates actions.

Exercises and Discussion

Q5: A Q-Table Versus a Deep Q-Network

How does Q-learning differ from a deep Q-network (DQN)? What problems do DQN's key mechanisms address?

Comparison	Q-Learning	DQN
Q representation	Table of state–action values	A network estimates each discrete action's Q-value
Typical inputs	Small, enumerable state spaces	High-dimensional states that are difficult to enumerate
Experience replay	Basic updates use individual transitions	Sample past transitions to reduce correlation
Target estimate	Uses current tabular Q estimates	A target network slows target changes

Chapter Outline

Exercises and Classroom Discussion

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.9.1 Core Concepts and Method Comparisons

Textbook: Sec. 10.4, Questions 1–5

10.9.2 Formation Design and Experiment Deployment

Textbook: Sec. 10.4, Questions 6–7

10.9.3 Learning Policies and Scaling

Textbook: Sec. 10.4, Questions 8–9

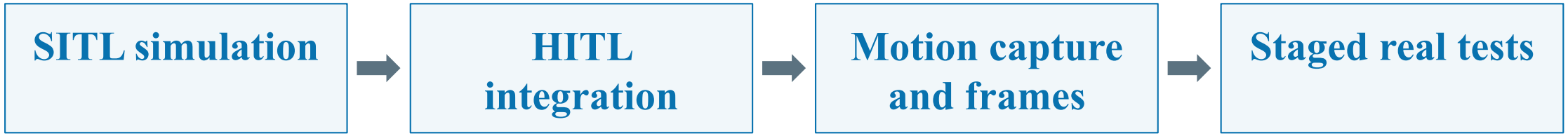
10.9.4 Engineering Strategies and Chapter Review

Textbook: Sec. 10.4, Question 10; chapter review

Exercises and Discussion

Q6: Validation Stages for Six Indoor Vehicles

Design validation stages for a centralized six-unmanned-aerial-vehicle (UAV) formation using indoor motion capture.



SITL / HITL: test control and fault logic, then verify real hardware interfaces and timing.

Motion capture / flight: establish a common localization frame; expand from one to two to six vehicles.

Reference Points

Q6 Reference: Validation at Each Stage

Stage	Main Checks	Risk Reduced
Software-in-the-loop (SITL)	Formation error, minimum separation, link-loss response	Control logic and role-assignment errors
Hardware-in-the-loop (HITL)	Update period, limits, mode changes and logs	Interface and timing mismatches
Motion-capture setup	Camera calibration, rigid-body IDs Frame transforms and time synchronization	Position direction and object-association errors
One to six vehicles	Fence, spacing, emergency stop and takeover drills	Risks from simultaneous multi-vehicle operation

Exercises and Discussion

Q7: Planar Control for One Leader and Four Followers

Design a control law for one leader and four followers. Explain how its parameters affect stability.

$$e_i = p_i - p_0 - \delta_i, \qquad \delta_i \in \{(-2,0), (0,-2), (2,0), (0,2)\}\ \mathrm{m}$$

$$u_i = v_0 - k_p e_i - k_c \sum_{j \in \mathcal{N}_i} a_{ij}(e_i - e_j)$$

Model and information: planar first-order velocity dynamics; every follower knows the leader state; unit-weight follower chain.

Positive gains set leader tracking and neighbor coordination. Velocity commands are in m/s.

Reference Points

Q7 Calculation: A Target to the Right Requires Rightward Motion

Given $p_0 = (10,5)$, $p_1 = (7,5)$, $\delta_1 = (-2,0)$, $e_2 = (0,0)$, with positions in m.

Parameters: $k_p = 0.6$, $k_c = 0.2$, $\Delta t = 0.1$ s, $v_0 = (0.5,0)$ m/s

$$e_1 = (7,5) - (10,5) - (-2,0) = (-1,0)\ \text{m}$$

$$u_1 = (0.5,0) - 0.6(-1,0) - 0.2[(-1,0) - (0,0)] = (1.3,0)\ \text{m/s}$$

$$e_1^+ = (-1,0) + 0.1[(1.3,0) - (0.5,0)] = (-0.92,0)\ \text{m}$$

Full-run initial errors: $e_1 = (-1,0)$, $e_2 = (0,0)$, $e_3 = (0,1)$, $e_4 = (0.6,-0.4)$ m.

10.9.2 Formation Design and Experiment Deployment: Full Numerical Update

Q7 Results: Gains, Time Step, and Error Decay

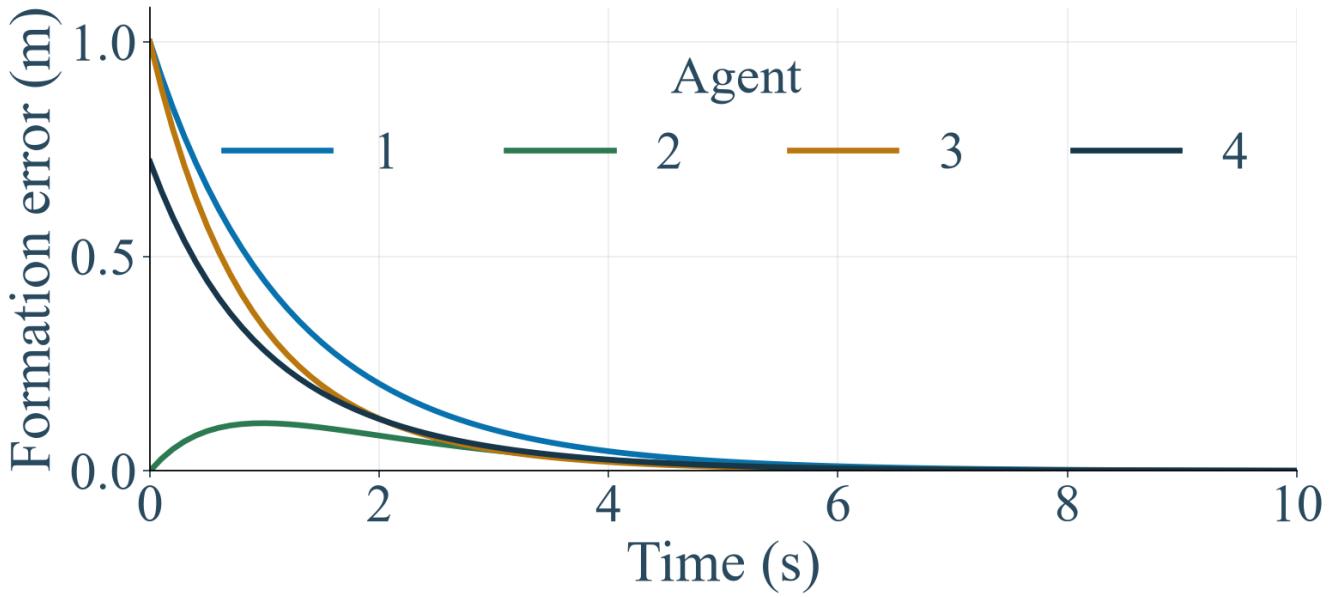


Figure: numerical formation-error norms for four followers

Each curve is one follower's error norm. Chain neighbors; constant leader velocity.

$$\dot{e} = -(k_p I + k_c L)e$$

$$0 < \Delta t \lambda_{\max}(k_p I + k_c L) < 2$$

Condition for this ideal discrete model; delay and saturation are excluded.

Chapter Outline

Exercises and Classroom Discussion

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.9.1 Core Concepts and Method Comparisons

Textbook: Sec. 10.4, Questions 1–5

10.9.2 Formation Design and Experiment Deployment

Textbook: Sec. 10.4, Questions 6–7

10.9.3 Learning Policies and Scaling

Textbook: Sec. 10.4, Questions 8–9

10.9.4 Engineering Strategies and Chapter Review

Textbook: Sec. 10.4, Question 10; chapter review

Exercises and Discussion

Q8: A High-Level Learning Design for a Ten-on-Ten Task

Design a ten-on-ten reinforcement-learning loop: define observations, actions, rewards and validation.

Design Item	Definition for an Abstract Resource Contest
State and observation	The global state stores all positions and resources; each agent receives only locally available information
Action	Motion targets or discrete moves subject to speed and boundary limits
Reward	Resource score and team progress, with penalties for collisions, boundary violations and excess energy
Termination and assessment	Time limit or task completion; record scores, safety events and completion time

Reference Points

Q8 Reference: Training and Independent Evaluation



Start small: verify interfaces and rewards with simple scenes and rule-based baselines; then enlarge teams and randomize conditions.

Separate evaluation: hold out maps, initial states and opponent policies; compare task scores and safety events.

Tool roles: RflySim supplies the environment and interfaces; an external learner samples, updates, saves and loads policies.

Exercises and Discussion

Q9: Shared Policies with a Changing Neighbor Set

Design parameter sharing with a graph neural network (GNN) for a large swarm. When is it useful?

$$m_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \phi(o_j, p_j - p_i), \qquad a_i = \pi_\theta(o_i, m_i, c_i)$$

Encode and aggregate valid neighbor messages, then combine them with local observations and role/type features.

Useful for many similar agents with changing neighbors. If none are valid, use a zero message and an explicit missing-data flag.

Shared parameters can still produce different actions. Heterogeneous vehicles may need type features or separate output heads.

Reference Points

Q9 Reference: How Much Interaction Does a Local Graph Save?

Structure	Relations / Update: 100 Nodes	Trade-Off
Fully connected	$100 \times 99 = 9900$ directed relations	Direct information; greater communication and processing load.
Mean degree of 6	$100 \times 6 = 600$ directed relations	Lower local load; multiple hops for distant information.

Aggregation usually scales with graph edges. Swarm algorithms do not all have exponential complexity.

Compare computation delay, link load, connectivity and task error together.

Chapter Outline

Exercises and Classroom Discussion

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures
10.2 Consensus and Formation Control
10.3 Game Theory and RL Fundamentals
10.4 Single-Agent RL Algorithms
10.5 Multi-Agent RL and Scaling
10.6 Technologies and Challenges
10.7 RflySim Framework and Interfaces
10.8 Experiments and Validation
10.9 Exercises and Classroom Discussion

4 Units in This Section

10.9.1 Core Concepts and Method Comparisons

Textbook: Sec. 10.4, Questions 1–5

10.9.2 Formation Design and Experiment Deployment

Textbook: Sec. 10.4, Questions 6–7

10.9.3 Learning Policies and Scaling

Textbook: Sec. 10.4, Questions 8–9

10.9.4 Engineering Strategies and Chapter Review

Textbook: Sec. 10.4, Question 10; chapter review

Exercises and Discussion

Q10: Combining Localization, Time Synchronization, and Prediction

How should localization fusion, clock synchronization and motion prediction work together? Give two scenarios.

Coordinate and time reference → **Timestamped fused state** → **Predict the current state**

$$\hat{p}(t) = p(t_s) + v(t_s)(t - t_s)$$

Example: age 0.2 s and speed 3 m/s give a constant-velocity extrapolation of 0.6 m.

Turns and acceleration create residual error. Apply the predefined degraded mode when the age limit is exceeded.

Reference Points

Q10 Reference: Two Scenarios

Scenario	Position / Frames	Time and Prediction	Checks
Indoor formation	Motion capture + inertial fusion Common site frame	Sampling timestamps Short-horizon velocity extrapolation	Rigid-body identification rate Error and minimum separation
Outdoor air–ground cooperation	Global navigation satellite system (GNSS) + inertial fusion Shared map origin	Common time reference Predict rover motion	Information age Following error and loss-of-lock response

A shared log timeline helps distinguish localization, communication and control errors.

Exercises and Discussion

Discussion: What Changes When One Link Fails?

How can a formation still fly while losing its ability to complete the original cooperative task?

Draw the changed graph. Identify affected observations, commands and nodes; test whether the remaining topology supports the task.

Separate individual and collective behavior. Local control may maintain flight while formation geometry or task allocation fails.

Record control modes, information age, formation error and task completion together.

Chapter Summary

Information Conditions Set the Limits of a Method

Topic	Problem Addressed	Important Distinction
Control architecture	Where information gathers and commands are computed	Multiple computers can still run centralized control
Consensus and formation	Turn relative relationships into feedback commands	Finite safe geometry needs offsets and distance constraints
Reinforcement learning	Improve an action policy from task returns	A deployed policy uses only available observations
Engineering and platform	Coordinate time, frames and interface constraints	Outcomes depend on the complete closed loop.

Chapter Summary

Compare Methods on the Same Task

Task and information → **Method and parameters** → **Error and task outcome**

Use a common baseline: initial states, task, vehicle limits and metrics. Change the architecture or algorithm separately.

Include abnormal conditions: delay, packet loss, node departures and changing localization quality.

Match methods to conditions; verify reliable cooperation through repeatable tests.