Chapter Outline

Experiments and Simulation-to-Real Validation

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

6 Units in This Section

7.8.1 Experiment Entry Points and Advanced Cases
Textbook: Sec. 7.3.1–7.3.2

7.8.2 Motor Fault Validation
Textbook: Sec. 7.2.3, 7.3.1

7.8.3 Automated Testing and Event Logs
Textbook: Sec. 7.1.5.2, 7.3.1

7.8.4 Data Processing and Diagnostic Model Training
Textbook: Sec. 7.1.5.4, 7.3.1

7.8.5 Safety Scoring Experiments
Textbook: Sec. 7.1.5.3, 7.3.1

7.8.6 Simulation-to-Real Validation and Policy Comparison
Textbook: Sec. 7.1.5.1, 7.3.2

7.8 Experiments and Validation

7.8.1 Experiment Entry Points and Advanced Cases: Entry Points

Extend from Offline Records to Closed-Loop Evidence

Level	Actual entry point	Suggested task
Offline introduction	0.ApiExps/e10_data_process	Check CSV timestamps and units
Single-fault interface	1.BasicExps/e4_SensorFault	Compare before, during, and after injection
Batch cases	0.ApiExps/e14_DBExp, e15_CmdExp	Separate cases from execution results
Research extension	PaperRes/6.IoT-DT-Diagnosis, etc.	Check methods and their conditions

Rows 1–3: [installation directory]/RflySimAPIs/7.RflySimPHM/; PaperRes: RflySimAPIs/.
CSV = comma-separated values. 3.CustExps: Full Edition or above only.

7.8.1 Experiment Entry Points and Advanced Cases: Offline Practice

Minimal Experiment: Check One Claim against Existing Records

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/0.ApiExps/e10_data_process/
Log CSV files are in data/.

Run: open Python38Run.bat and enter python dataProcess.py.
It overwrites processed_data.csv; save a separate copy first.

Check: 712 rows × 49 columns, no header; flag=1 after the timestamp threshold.

Verify and submit: compare acceleration standard deviation before/after flag in matched conditions. Provide labeled axes, units, and label provenance.

Troubleshooting: Readme says data_process.py; the actual file is dataProcess.py. Import errors: install pandas and numpy. “No file found”: check all four CSV types in data/.

7.8.1 Experiment Entry Points and Advanced Cases: Reporting Results

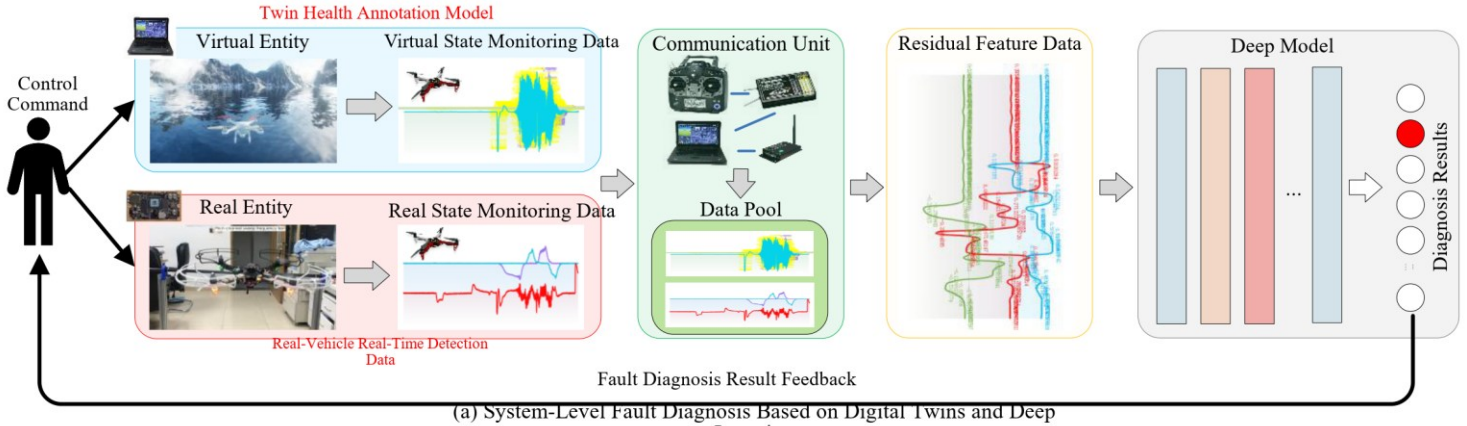
Report Enough Information for Others to Reproduce the Result

Report part	Minimum information
Question and configuration	Mission, fault, conditions, versions, parameters, and evaluation rules
Evidence and reproduction	Raw data, timed events, processing script, and result plots with units
Comparisons and statistics	Healthy baseline, repeat conditions, failed runs, and uncertainty
Conclusions and next steps	What is supported, what is outside coverage, and how to add evidence

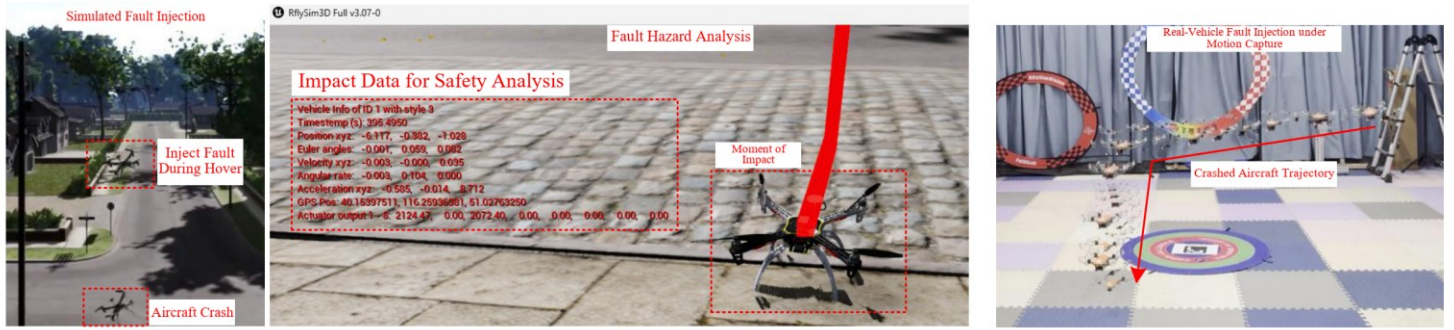
7.8 Experiments and Validation

7.8.1 Experiment Entry Points and Advanced Cases: Advanced Cases

Two Research Paths Link Diagnosis with Simulation-to-Real Validation



(a) System-Level Fault Diagnosis Based on Digital Twins and Deep Learning



(b) Real-Vehicle and Simulation Nodes Running in Parallel

Fig. 7.15 Advanced cases: diagnostic evidence and simulation-to-real comparison evidence

7.8 Experiments and Validation

7.8.1 Experiment Entry Points and Advanced Cases: Advanced Diagnosis

Path 1: Twin References and Real Observations for Diagnosis

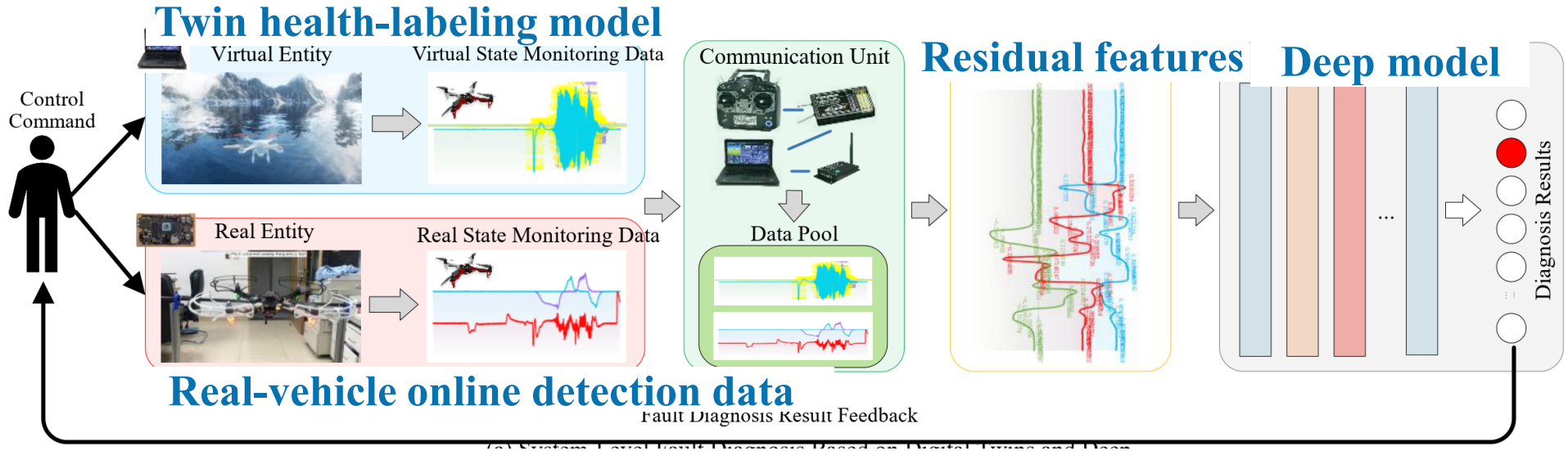


Fig. 7.15(a) System-level fault diagnosis with digital twins and deep learning

Same control input → align virtual/real data → residuals/features → diagnostic output.

Supervised learning suffices; assess diagnosis results with data splits and hard cases.

Details: Section 7.5, "Paper Source and Symbols."

7.8 Experiments and Validation

7.8.1 Experiment Entry Points and Advanced Cases: Advanced Validation

Path 2: Match Fault and Measurement Conditions as Well as Trajectories

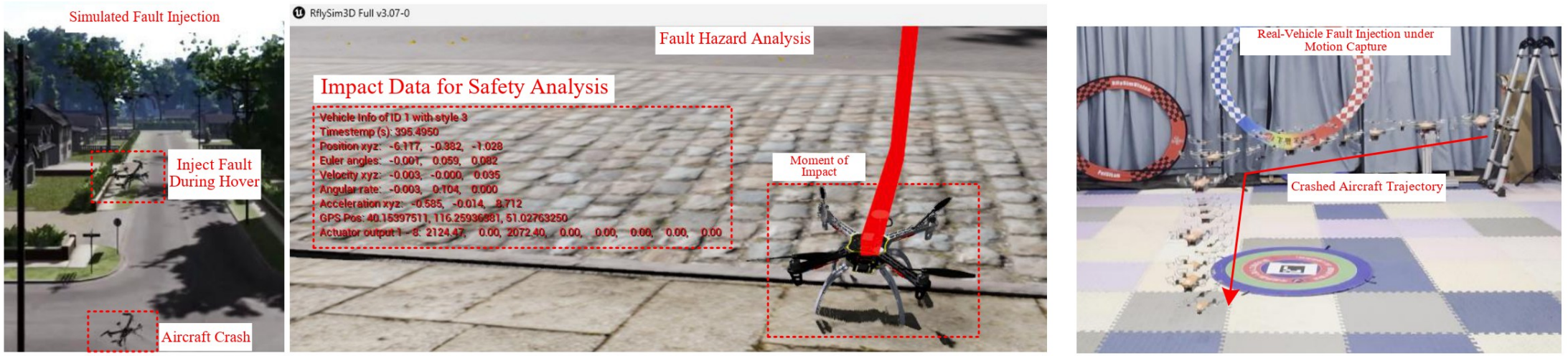


Fig. 7.15(b) Real-vehicle and simulation nodes running in parallel

Align: fault type/severity, injection time, initial state, mission, and sample times.

Compare trajectory, attitude, velocity, consequences, error bounds, and untested conditions.

Full alignment checklist: Section 7.3.1, "Comparability."

Chapter Outline

Experiments and Simulation-to-Real Validation

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

6 Units in This Section

7.8.1 Experiment Entry Points and Advanced Cases
Textbook: Sec. 7.3.1–7.3.2

7.8.2 Motor Fault Validation
Textbook: Sec. 7.2.3, 7.3.1

7.8.3 Automated Testing and Event Logs
Textbook: Sec. 7.1.5.2, 7.3.1

7.8.4 Data Processing and Diagnostic Model Training
Textbook: Sec. 7.1.5.4, 7.3.1

7.8.5 Safety Scoring Experiments
Textbook: Sec. 7.1.5.3, 7.3.1

7.8.6 Simulation-to-Real Validation and Policy Comparison
Textbook: Sec. 7.1.5.1, 7.3.2

7.8.2 Motor Fault Validation: Example Steps

Two Motor Examples: Fault Blocks, then Software-in-the-Loop

API = application programming interface. Software-in-the-loop (SITL; SIL in the textbook).
[installation directory]/RflySimAPIs/7.RflySimPHM/0.ApiExps/<example directory>/
Requires MATLAB 2022B or later and RflySim; see each Readme.pdf. DLL = dynamic-link library.

e6_UseFaultLib: Study the Fault Block

1 Run Init.m to initialize parameters
2 Open MotorFault.slx; return to the top level
3 Set the fault switch to 1 (0 means no injection)
4 Adjust fault parameters and simulate

e8_BaseMotorFault: Minimal SITL

1 Run Init.m
2 Open MulticopterModel.slx; Build Model
3 Run GenerateModelDLLFile.p for the DLL
4 Run MulticopterModelSITL.bat
(Readme calls it SITLRun.bat)

Run effects: administrator rights; copy MulticopterModel.dll into the CopterSim model directory; restart existing PX4 SITL/CopterSim.
Any key in PX4 ends the run, closing CopterSim, QGroundControl (QGC), RflySim3D.

7.8.2 Motor Fault Validation: Criteria and Troubleshooting

e8_BaseMotorFault: Fault Injection Causes Severe Takeoff Shaking

Check initialization/connection first; response checks follow both Readme.pdf files.

e8_BaseMotorFault: Expected Response

Start: one QGC, CopterSim, and RflySim3D
Initialization: CopterSim prints
GPS 3D fixed & EKF initialization finished
Connection: after Enter Auto Loiter Mode!, take off manually in QGC
Fault effect: takeoff fails, with severe shaking

Common Failures and Checks

Build: example working directory; MATLAB 2022B or later
DLL: paths, MATLAB rights, antivirus
e6 block not visible: return to model top level
No effect: switch, parameters, duration, injection wiring. Replace an incompatible e8 block from the fault library.

e6 check: switch 0 disables injection; 1 enables it. Compare outputs.
GPS = Global Positioning System; EKF = extended Kalman filter.

7.8.2 Motor Fault Validation: Example Entry

Generate the Model, Start Simulation, then Run the Script

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/1.BasicExps/e3_MotorFault/

Entry (see Readme.pdf)	Action and criterion
Init.m → Build → GenerateModelDLLFile.p	Rebuild the DLL after model edits; the supplied MulticopterModel.dll can be used directly
MulticopterModelSITL.bat	Starts QGC, CopterSim, RflySim3D; stops existing CopterSim and PX4
Initialization criterion	CopterSim log: GPS 3D fixed & EKF initialization finished
Python38Run.bat	Run python FaultInjectAPITest.py; exits at about 50 s. Any key in the launcher closes simulation.

7.8.2 Motor Fault Validation: Example Troubleshooting

Check Block Versions, Connections, and Parameters First

Build failure: if MotorFault differs from the software development kit (SDK), replace it from RflySimLib. Missing compiler: run mex -setup in MATLAB.

Connection: administrator batch; wait for initialization; check Python firewall rules and User Datagram Protocol (UDP) port conflicts.

Unexpected response: FaultID=123450; coefficients 0–1; 8 integers and 20 floats with correct types. Arm before injection.

Partial response: match coefficients to motors; four for a quadrotor.

Hardware-in-the-loop (HITL; HIL in the textbook): MulticopterModelHITL.bat. Connect the flight controller via Universal Serial Bus (USB); automatic serial detection; other steps match SITL.

Response comparison: crash and single-motor screenshots are on “Expected Response.”

7.8.2 Motor Fault Validation: Original Call

The Default Script Sets Three Motor Outputs to Zero

```
silInt = np.zeros(8).astype(int).tolist()
silFloat = np.zeros(20).astype(float).tolist()
silInt[0:2] = [123450, 123450]
silFloat[0:4] = [0.6, 0, 0, 0]
mav1.sendSILIntFloat(silInt, silFloat)
```

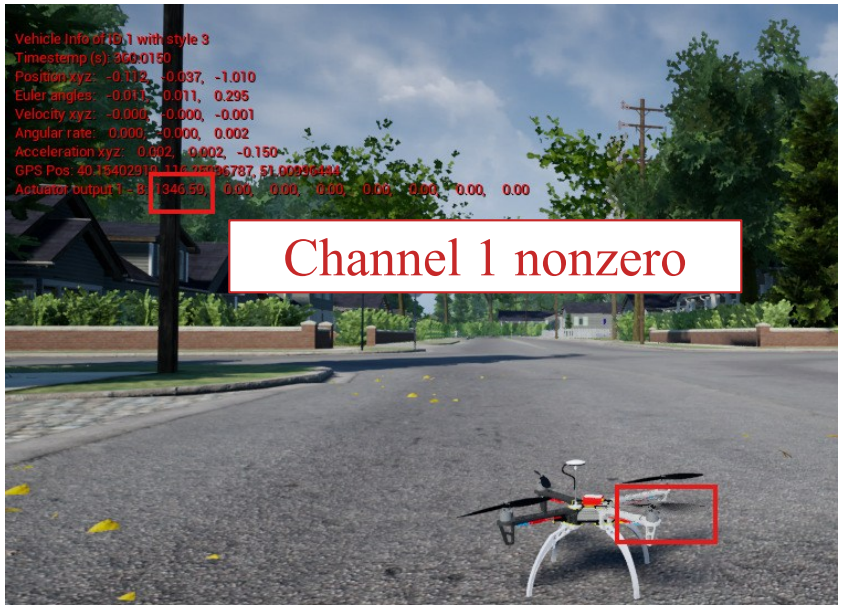
Source timing: arm and command 10 m at 5 s; inject once at 20 s; exit after 50 s.
Expected: the vehicle crashes; a new takeoff command leaves only one motor rotating.
Array meaning: 0 sets that motor's control input to zero, stopping motors 2–4.

7.8.2 Motor Fault Validation: Expected Response

Default Injection Crashes; a New Takeoff Leaves One Output



Crash after injection (Readme p6)



One motor on takeoff (Readme p7)

Crash: [0.6, 0, 0, 0] sets channels 2–4 to zero; the vehicle loses lift.

New takeoff command: only actuator channel 1 is nonzero (original red box).

Chapter Outline

Experiments and Simulation-to-Real Validation

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

6 Units in This Section

7.8.1 Experiment Entry Points and Advanced Cases
Textbook: Sec. 7.3.1–7.3.2

7.8.2 Motor Fault Validation
Textbook: Sec. 7.2.3, 7.3.1

7.8.3 Automated Testing and Event Logs
Textbook: Sec. 7.1.5.2, 7.3.1

7.8.4 Data Processing and Diagnostic Model Training
Textbook: Sec. 7.1.5.4, 7.3.1

7.8.5 Safety Scoring Experiments
Textbook: Sec. 7.1.5.3, 7.3.1

7.8.6 Simulation-to-Real Validation and Policy Comparison
Textbook: Sec. 7.1.5.1, 7.3.2

7.8.3 Automated Testing and Event Logs: Example Workflow

Four Examples Connect Case Entry, Execution, Collection, and Download

e14: Store cases → **e15: Control sequence** → **e9: Collect data** → **e16: Download logs**

e14_DBExp: AutoCase.py generates JavaScript Object Notation (JSON) cases and imports them into Quadcopter.db, table TEST_CASE.

e15_CmdExp: CmdCtrl.py parses db.json control sequences for arming, flight, and injection.

e9_data_collect: data_collect.py executes cases and collects data, writing results to fault.db.

e16_QGCLoadExp: QGCLogDown.py calls ReqQgcLog to download the latest flight log.

Related examples under 0.ApiExps/: e18 writes events; e10 processes offline data.

7.8.3 Automated Testing and Event Logs: Case Database

Count Database Cases Separately from Completed Tests

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/0.ApiExps/e14_DBExp/

Table	Rows (V5.01)	What the count shows
TEST_CASE	175	Designed cases from AutoCase.py parameter scans
TEST_RESULT	0	Empty result table; no execution records yet

Planned coverage: AutoCase.py generates and stores cases from parameter start, end, and step.

Actual coverage: count completed runs with full evidence; report repeats separately.

7.8.3 Automated Testing and Event Logs: Case JSON and Parameter Scans

AutoCase.py Generates Cases from Start, End, and Step

Example: CaseID 346 in 123542.json (accelerometer noise, parameter 0)


```
"CaseID": 346,
"Subsystem": "Sensor",
"Component": "Accelerometer",
"FaultID": "123542",
"FaultType": "Accelerometer noise",
"FaultMode": "… interference",
"FaultParams": "0",
"ControlSequence": "…;2,6,123542,0;1,1,10",
"TestStatus": "Finished"
```


Scan: increase from start to end by the step.
Four sensor types: 0–30, step 1; 31 each.
GPS: 0–250, step 5; 51 cases.
Total: 175 cases.

Sequence: 2,1 arms; 1,1,5 waits 5 s; 2,3 sets position; 2,4 sets velocity; 2,6,123542,0 injects fault 123542, parameter 0.

TestStatus=Finished is assigned at generation; check execution in the result table.

7.8.3 Automated Testing and Event Logs: Advanced e4

Advanced e4 Runs a Configured Single-Vehicle SITL Test

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/

2.AdvExps/e4_SingFramSingleInsExp/AutoTest.py

```
conf = [['Quadcopter', 'SITL', 1]]
PX4MavCtrler(20100)
```

Configuration: Quadcopter vehicle; SITL; 1 instance; communication port 20100.

Entry: AutoTest.py. Instructions: Readme.pdf. Environment: Python38Run.bat.

Run: start model/Quadcopter/QuadModelSITL.bat, wait for initialization, then open Python38Run.bat and enter python AutoTest.py.

7.8.3 Automated Testing and Event Logs: Advanced e4 Checks

Wait for Initialization; Check Reports and Every Run's Logs

Initialization: QuadModelSITL.bat starts QGC, CopterSim, and RflySim3D. Wait for "GPS 3D fixed & EKF initialization finished" in CopterSim.

Results: TestInfo/TestResult in data/; open Hypertext Markup Language (HTML) files in a browser.

Run logs (.ulg/CSV): data/single/sInstance/Quadcopter/TestCase_N/log/.

Incomplete initialization: if GPS is not fixed, restart QuadModelSITL.bat and wait at least 30 s before AutoTest.py.

Other failures: include the current directory in the Python path. If ulog2csv fails, convert .ulg files under log/ to CSV manually.

7.8.3 Automated Testing and Event Logs: e18 Event Writing

e18 Writes an Event Command to the Flight Log at Injection

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/

0.ApiExps/e18_FaultLogTimestamp/FaultInjectAPITest.py

```
mav1.sendSILIntFloat(silInt, silFloat)
mav1.SendFaultEvent(123450)
```

Run: MulticopterModelSITL.bat → wait for initialization → Python38Run.bat → python FaultInjectAPITest.py. See Readme.pdf.

Effect: at 20 s, inject motor effectiveness fault 123450 and write an event; flight may stop or crash. Disarm after 50 s.

Logs: ReqQgcLog asks QGC to download the latest .ulg into log/; ulog2csv converts it to topic CSV files.

7.8.3 Automated Testing and Event Logs: e18 Fields and Checks

Find command=31000 in vehicle_command_0.csv

Criterion: a row in vehicle_command_0.csv with command=31000 and param1=123450.

Log column	Written by SendFaultEvent	Event field
param1	FaultID, e.g., 123450	fault_id
param2	Vehicle ID	Vehicle number
param3	action: 1 inject, 2 continue fragments, 0 clear	event_type
param4	seq + fragment decimal, e.g., 5.01	Fragments of one event
param5–7	Fault parameters; up to 3 per fragment	parameters
timestamp	Flight controller receive/processing time (μs)	timestamp (flight controller)

Controller reception/processing time; check model/controller response for effective time.

No record: check download/conversion. For 123450, lower coefficient = greater loss.

7.8.3 Automated Testing and Event Logs: Log Organization

One Log Directory per Run, with Summary Reports Above

Advanced e4 data/ (schematic): summary reports above; one log directory for each run below.

data/	Summary: reports and index
TestInfo.csv, TestInfo.html	Cases, parameters, and DataPath
TestResultInfo.csv, TestResult.html	Per-run results/responses
single/sInstance/Quadcopter/	Single-vehicle, single-instance runs
TestCase_1/log/, TestCase_2/log/	Each run: raw .ulg + decoded CSV
e18_FaultLogTimestamp/log/	Events: vehicle_command_0.csv

ULog is PX4's structured binary log format; CSV is a text table with defined columns.

7.8.3 Automated Testing and Event Logs: Historical Reports

Historical e4 Reports Show Structure; Four Gaps Limit Traceability

Field / observation	Installation V5.01 content	Evidence still needed
Two CSV files	6 rows each; repeated CaseID 1 or 2	Unique run_id and a result join key
Case 1 parameters	[1.0, 1.0, 1.0, 1.0]	Model parameter meaning and evidence of effective injection
Case 1 responses	Smooth flight / Violent rolling, etc.	Each run's time series and initial state
DataPath	Referenced raw log directories are absent from the package	Logs, events, configuration, and hashes

Historical fields need run IDs, parameter meaning, and raw logs to verify fault effects.

Chapter Outline

Experiments and Simulation-to-Real Validation

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

6 Units in This Section

7.8.1 Experiment Entry Points and Advanced Cases
Textbook: Sec. 7.3.1–7.3.2

7.8.2 Motor Fault Validation
Textbook: Sec. 7.2.3, 7.3.1

7.8.3 Automated Testing and Event Logs
Textbook: Sec. 7.1.5.2, 7.3.1

7.8.4 Data Processing and Diagnostic Model Training
Textbook: Sec. 7.1.5.4, 7.3.1

7.8.5 Safety Scoring Experiments
Textbook: Sec. 7.1.5.3, 7.3.1

7.8.6 Simulation-to-Real Validation and Policy Comparison
Textbook: Sec. 7.1.5.1, 7.3.2

7.8.4 Data Processing and Diagnostic Model Training: Offline Practice

Existing Files Suffice for a Complete Data Check

1. Locate: use pandas in Python38Run.bat to read e10's four CSV files; save results separately.

2. Check: filter timestamp to 32–36 s; verify counts, units, frames, and event provenance.

3. Reproduce: timestamp union → interpolation → 10-row mean/standard deviation → 34 s label. Check: 712×49; flag 0/1 counts 343/369; first 1 at 34.004 s.

4. Submit: curves, column order, source hashes, label explanation, grouping plan, and exclusion rules.

Explain the data before comparing algorithms; offline reproduction checks the processing chain.

7.8.4 Data Processing and Diagnostic Model Training: Learning Entries

Read the Entry Code, then Check Inference Resources

Directory: [installation directory]/RflySimAPIs/7.RflySimPHM/0.ApiExps/

Directory and entry	What to identify
e12_health_basic/ health_basic.py	Fault: adjacent height gap >0.25 m; full assessment is a placeholder.
e13_model_train/ Model-training.ipynb	Historical output: check dimensions, denominators; no cross-condition score.

Paper: [installation directory]/RflySimAPIs/PaperRes/6.IoT-DT-Diagnosis/
Paper, 5 minimal cases in cases/, Readme.pdf; obtain notebook/data/weights as directed.
Sources / DOI (digital object identifier): 7.5, “Paper Source and Symbols.”

7.8.4 Data Processing and Diagnostic Model Training: Running and Checks

Know the Steps, Expected Output, and Effects on Other Software

e12_health_basic/ (Starts Simulation)	e13_model_train/ (Offline)
Run: Python38Run.bat → python health_basic.py. **Mode:** line 11, Off_ctrl_connect_copter.py; select SITL/HITL bat. It starts simulation. **Output at 5 m:** “健康” (healthy); “故障” (fault) exits the loop. **At 30 s:** “退出测试” (exit test); closes CopterSim, QGC, RflySim3D. **Check:** GPS 3D fixed & EKF initialization finished in CopterSim.	**Entry:** Visual Studio Code (VS Code), Model-training.ipynb. **Steps:** run cells; read local processed_data.csv; last cell saves mymodel.model. **Model:** 48 inputs; hidden layers 56 and 32, rectified linear unit (ReLU); 2 outputs. Labels 0/1 come from a time threshold. **Checks:** install keras, numpy, pandas with pip; reduce batch size if memory is insufficient. **Evaluation:** check dimensions/denominators; no cross-condition performance claim.

Chapter Outline

Experiments and Simulation-to-Real Validation

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

6 Units in This Section

7.8.1 Experiment Entry Points and Advanced Cases
Textbook: Sec. 7.3.1–7.3.2

7.8.2 Motor Fault Validation
Textbook: Sec. 7.2.3, 7.3.1

7.8.3 Automated Testing and Event Logs
Textbook: Sec. 7.1.5.2, 7.3.1

7.8.4 Data Processing and Diagnostic Model Training
Textbook: Sec. 7.1.5.4, 7.3.1

7.8.5 Safety Scoring Experiments
Textbook: Sec. 7.1.5.3, 7.3.1

7.8.6 Simulation-to-Real Validation and Policy Comparison
Textbook: Sec. 7.1.5.1, 7.3.2

7.8.5 Safety Scoring Experiments: Entry and Interpretation

Advanced e8 Scores the Second Velocity Component

[installation directory]/RflySimAPIs/7.RflySimPHM/2.AdvExps/e8_SafetyAssExp/
Entry: Readme.pdf, Python38Run.bat, SafeAss.py (calls ProfustSA.py; reads truedata_vel.csv)

Setting or step	Purpose	Output and criterion
Python38Run.bat **python SafeAss.py**	Open integrated Python; run assessment	“安全得分为” (score) “安全等级为” (level)
EvalDim=[1] **VelCmd=[0,0,2]**	Evaluate component 2; target 0	Indices [7000,9500): 2500 records; average each pair
Common failures	Import errors; Python environment fails to start	Check local ProfustSA.py and installed pandas/numpy

FallIndex=0: no impact; five output levels, Unaffected to Catastrophic.

Exercises and Discussion

Assess the Same Fault at Three Levels

- **Scenario:** one motor retains 70% effectiveness; inspection moves near power lines. How do exposure and measurements change? With warnings, how do severity (S), occurrence (O), and detection (D) ratings change in failure mode and effects analysis (FMEA)?
- Reliability: exponential fault intervals, $\lambda = 0.004\,\mathrm{h}^{-1}$. Find the probability of at least one fault in 45 min; how many times the Section 7.4.3 value is λt?
- **Parameters:** set EvalDim=[2], keep VelCmd=[0,0,2]. Which component and target are selected? What level is output for positive FallIndex?

State separately: scenario/consequences, model/units, and parameters/branches.

Reference Points

Trace Risk Changes to Exposure or Effective Safeguards

- **Revise:** exposure becomes lines/equipment; consequences include line damage and secondary outages. Monitor clearance, wind, and position error.
- **Conditions for protection:** detect the fault in time; retain capability to avoid facilities or land under control; act before clearance is lost.
- **Ratings:** warnings improve detectability, lowering D. Motor-fault occurrence O stays unchanged. S changes only if consequences are actually limited.
- **Priority:** address severe harm to people first, then compare empirical scores on the same basis. Update verified residual risk.

Reference Points

Mission Probability and Scoring Parameter Substitutions

$$t = 45/60 = 0.75\,\mathrm{h}, \quad \lambda t = 0.004 \times 0.75 = 0.003$$

$$F(t) = 1 - e^{-0.003} = 0.0029955 \approx 0.2996\%$$

- Comparison: $\lambda t = 0.003$ is 3 times the 30 min example; when $\lambda t \ll 1$, F is also about 3 times as large.
- **Component/target:** EvalDim=[2] selects component 3 (zero-based); its target in VelCmd is 2.
- **Impact branch:** FallIndex>0 gives Catastrophic; with FallEnergy=0, the score is 0.1.

Chapter Outline

Experiments and Simulation-to-Real Validation

Chapter 7 Health Management and Safety Assessment

7.1 Health Management Fundamentals

7.2 Unified Fault Modeling

7.3 Safety Reinforcement and Fault Data

7.4 Fault Hazards and Safety Assessment

7.5 Diagnosis and Prognostics

7.6 Safety Decisions and Improvement

7.7 Fault Injection Interfaces

7.8 Experiments and Validation

7.9 Exercises and Classroom Discussion

6 Units in This Section

7.8.1 Experiment Entry Points and Advanced Cases
Textbook: Sec. 7.3.1–7.3.2

7.8.2 Motor Fault Validation
Textbook: Sec. 7.2.3, 7.3.1

7.8.3 Automated Testing and Event Logs
Textbook: Sec. 7.1.5.2, 7.3.1

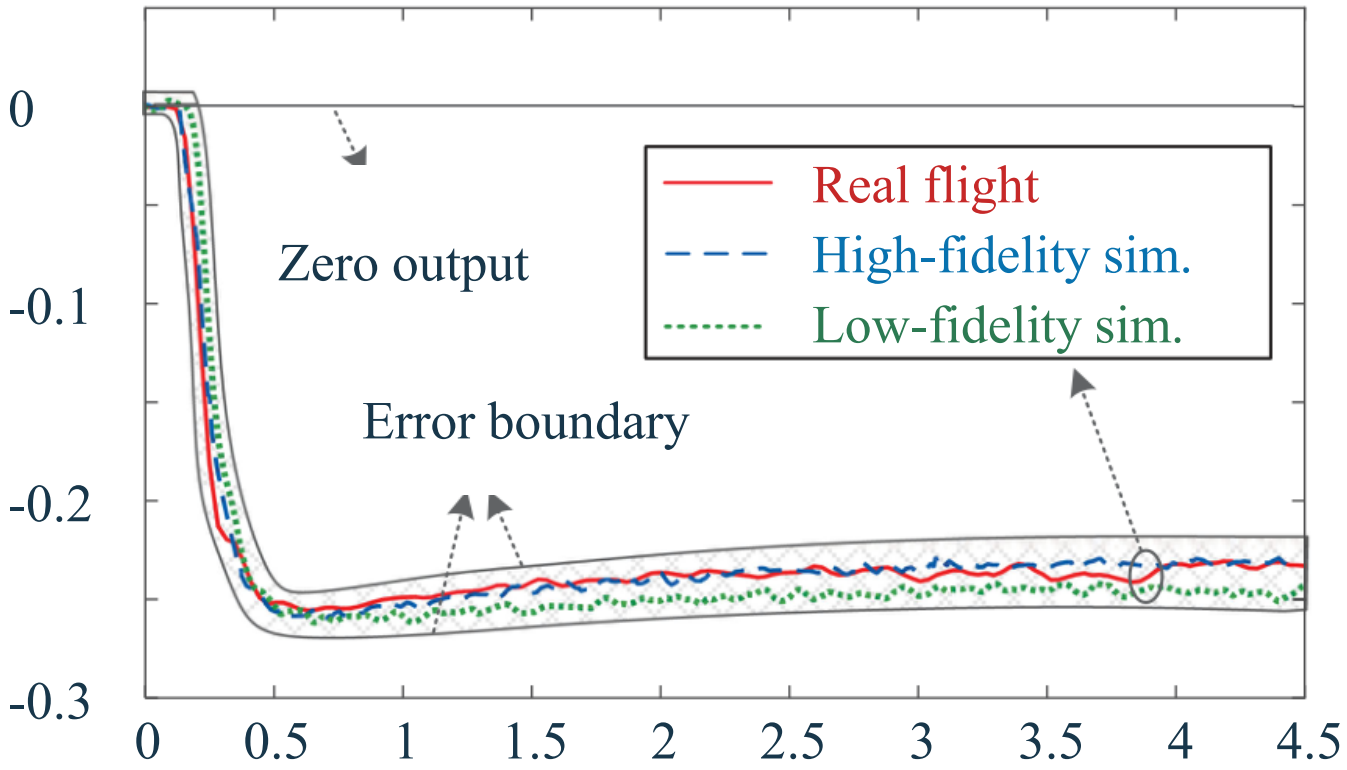
7.8.4 Data Processing and Diagnostic Model Training
Textbook: Sec. 7.1.5.4, 7.3.1

7.8.5 Safety Scoring Experiments
Textbook: Sec. 7.1.5.3, 7.3.1

7.8.6 Simulation-to-Real Validation and Policy Comparison
Textbook: Sec. 7.1.5.1, 7.3.2

7.8.6 Simulation-to-Real Validation and Policy Comparison: Model Evidence

Real, High/Low-Fidelity, and Zero-Output References Give Error Context



Red: real outdoor flight provides this baseline response.

Blue/green: compare high/low-fidelity responses.

Gray: error acceptance limits; zero output mismatches.

DOI: 10.1109/TIE.2020.2982122

[installation directory]/RflySimAPIs/PaperRes/0.RflySimToolChain/

Dai et al., IEEE Transactions on Industrial Electronics (2021), Fig. 11; pitch/rad, time/s

7.8.6 Simulation-to-Real Validation and Policy Comparison: Error Calculation

Root Mean Square Error Measures Response Differences

$$e = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\mathrm{sim},i} - y_{\mathrm{real},i})^2}$$

n is the aligned sample count; $y_{\mathrm{sim},i}$ and $y_{\mathrm{real},i}$ are model/real responses at the same time. Both are pitch in rad; e is root mean square error (RMSE).

Teaching errors: 0, 0.1, 0, −0.1 rad. Positive and negative deviations do not cancel.

$$e = \sqrt{\frac{0^2 + 0.1^2 + 0^2 + (-0.1)^2}{4}} = 0.0707\ \mathrm{rad}$$

7.8 Experiments and Validation

7.8.6 Simulation-to-Real Validation and Policy Comparison: Score Mapping

Tolerance and Acceptance Limits Give the Score Engineering Meaning

$$\eta = \frac{K_e \varepsilon}{\sqrt{(K_e \varepsilon)^2 + e^2}}$$

$$K_e = \frac{\eta_{\text{pass}}}{\sqrt{1 - \eta_{\text{pass}}^2}}$$

$\varepsilon > 0$: same-unit tolerance; $\eta_{\text{pass}} = 0.6$ gives $K_e = 0.75$.

$\varepsilon = 0.1$ rad is a teaching assumption; η is a matching score, not safety probability.

Error relative to tolerance	Mapping score	Interpretation
Error = tolerance	0.600	At the chosen acceptance limit
Error = 0.707 × tolerance	About 0.728	RMSE example: above acceptance
Error = 2 × tolerance	About 0.351	Exceeds this application's tolerance

7.8.6 Simulation-to-Real Validation and Policy Comparison: Paper Results

Paper Scores Distinguish Fidelity within the Stated Scope

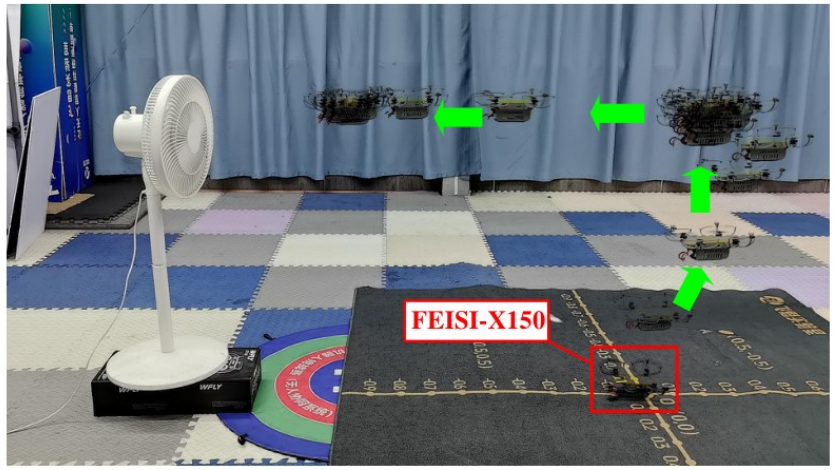
Dai et al. (2021), Table II: the same level-flight response and tolerance mapping; scores shown on a 100-point scale.

Comparison	Time-domain score	Meaning in this evaluation
High-fidelity model	94.4	Closer to the real response
Low-fidelity model	73.8	Still above the chosen 60-point limit
Error boundary	60.0	Acceptance limit for this example
Zero output	1.75	Clearly mismatches the real response

Paper error, tolerance, and scope: https://rfly.buaa.edu.cn/doc/2020DaiSimulationCredibility.pdf

7.8.6 Simulation-to-Real Validation and Policy Comparison: Test Scenario

Compare Four Policies on the Same Vehicle, Task, and Wind



FEISI-X150 wind test (Fig. 14(a))
Dai et al., Unmanned Systems, 2025
DOI: 10.1142/S2301385027500464
[installation directory]/RflySimAPIs/PaperRes/2.UAV-DT-Control-Sim2Real/

PID: proportional–integral–derivative feedback.
LQR: linear quadratic regulator.

Twin frequency sweeps + a few real measurements design LQR2.

Data label	Paper control policy
PID4	Default PID
PID3	Manually tuned PID
LQR1	Real-flight-data LQR
LQR2	Synthetic-data LQR

7.8.6 Simulation-to-Real Validation and Policy Comparison: Position Records

Position Reveals Initial Offset, Tracking Speed, and Steady Error

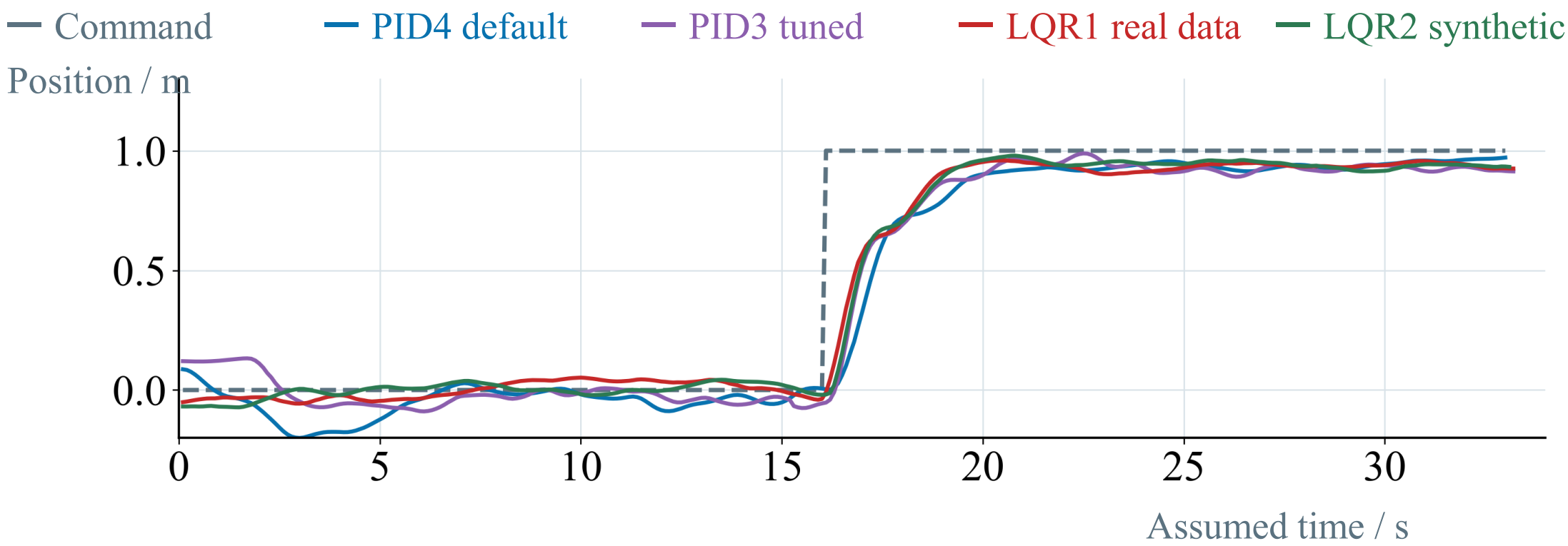


Figure: four x-position records from MATLAB data files (MAT); assumed interval 0.1 s

7.8.6 Simulation-to-Real Validation and Policy Comparison: Acceleration

Raw Acceleration Shows Default and Tuned PID Amplitudes

Raw body-X acceleration, units m/s^2; unfiltered, retaining the recorded sign.

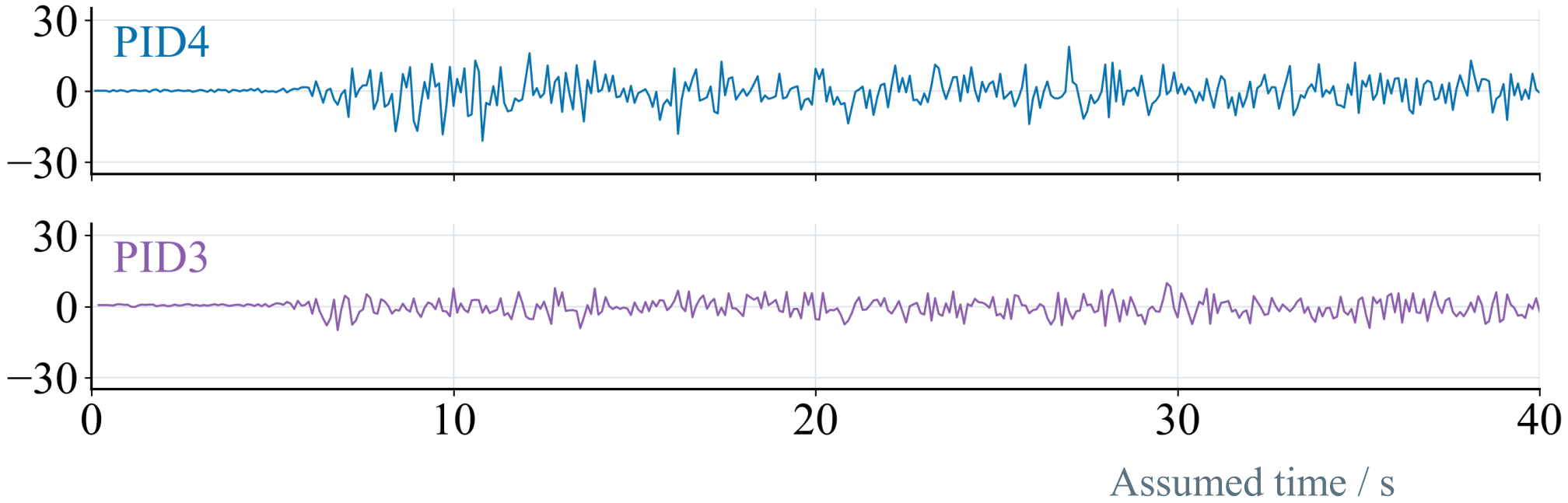


Figure: MAT acceleration records for PID4 (default) and PID3 (tuned). RMS = root mean square.

7.8.6 Simulation-to-Real Validation and Policy Comparison: Acceleration

Check Both Tracking and Acceleration for the Two LQR Policies

LQR1 uses real-vehicle data; LQR2 uses synthetic data. Units: m/s^2.

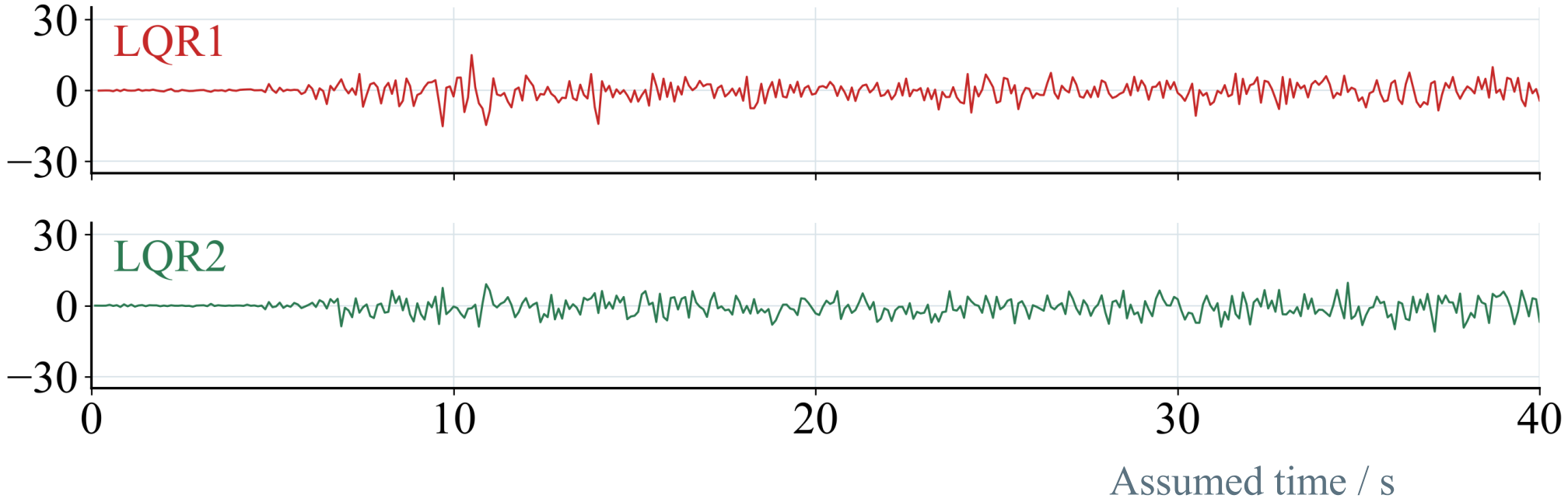


Figure: MAT acceleration records for LQR1 (real data) and LQR2 (synthetic data)

7.8.6 Simulation-to-Real Validation and Policy Comparison: Metrics

Error Integrals and Acceleration RMS Measure Different Tradeoffs

$$\mathrm{IAE} = \sum_{i=1}^{n-1} \frac{|e_i| + |e_{i+1}|}{2} \Delta t$$

Integral of absolute error (IAE) accumulates deviation, in $\mathrm{m} \cdot \mathrm{s}$.

$$\mathrm{ISE} = \sum_{i=1}^{n-1} \frac{e_i^2 + e_{i+1}^2}{2} \Delta t$$

Integral of squared error (ISE) weights large deviations more; units $\mathrm{m}^2 \cdot \mathrm{s}$.

$$a_{\mathrm{RMS}} = \sqrt{\frac{1}{M} \sum_{j=1}^{M} a_j^2}$$

Root mean square (RMS) describes acceleration amplitude; units $\mathrm{m/s}^2$.

y_i: measured position; u_i: commanded position; a_j: raw body-X acceleration.

$e_i = y_i - u_i$; n/M: position/acceleration sample counts; $\Delta t = 0.1\ \mathrm{s}$ assumed by the script.

7.8.6 Simulation-to-Real Validation and Policy Comparison: Common Windows

LQR2 Tracks Best; Tuned PID Has the Lowest Acceleration RMS

Windows: position samples 1–330, acceleration 2–400; 0.1 s is the script's assumption.

Policy / data label	IAE / m · s	ISE / m^2 · s	RMS / m/s^2
Default PID (PID4)	3.4485	1.2158	5.7373
Tuned PID (PID3)	3.1953	1.0149	3.5310
LQR1 (real data)	2.7337	0.8796	3.7532
LQR2 (synthetic data)	2.4438	0.8661	3.6013

Same vehicle and wind: twin-data LQR2 is no worse than LQR1 on all three metrics. Under these conditions, twin data can replace some real-flight data.

7.8.6 Simulation-to-Real Validation and Policy Comparison: Reproduction

Reproduce Common-Window Metrics from Existing MAT Files

[installation directory]/RflySimAPIs/PaperRes/
2.UAV-DT-Control-Sim2Real/3_Sim2RealCompare/
Entries: compare_step_response.m; compare_acceleration.m; data/ (8 MAT files)

Requires MATLAB R2022a or later. Full instructions: parent Readme.pdf, Sec. 4.4.

Steps: ① switch MATLAB to this directory; ② run the PID4 snippet below;
③ change filenames and variable prefixes together for PID3/pid3, LQR1/lqr1, LQR2/lqr2.

```
load('data/PID4_data.mat'); load('data/PID4_acc.mat');
e=x_pid4_y(1:330)-x_pid4_u(1:330); t=(1:330)'*0.1;
IAE=trapz(t,abs(e)); ISE=trapz(t,e.^2);
RMS=sqrt(mean(pid4_acc_x(2:400).^2));
```

7.8.6 Simulation-to-Real Validation and Policy Comparison: Reproduction Checks

Matching the Table Confirms Data and Calculation Conventions

Expected Output and Criteria

Snippet: PID4 IAE≈3.4485, ISE≈1.2158, RMS≈5.7373, as tabulated.
Position script: prints four IAE/ISE pairs and step plots. Full-record integrals are slightly higher for the other three, e.g., PID3 IAE=3.2124.
Acceleration script: prints RMS, variance, peak-to-peak; RMS uses samples 2–400.

Common Failures and Checks

load cannot find files: switch to 3_Sim2RealCompare.
Undefined variable: change the prefix as well as the filename, e.g., x_pid3_y.
Values differ: check position 1–330, acceleration 2–400, and Δt=0.1 s.

Run effects: both scripts begin with clc; clear; close all, clearing the workspace and closing open figures. Save needed variables first.

Exercises and Discussion

Choose a Policy with Two Metrics, then Design Fault Regression Tests

Classroom limits: IAE $\leq 3.3\ \mathrm{m \cdot s}$; acceleration RMS $\leq 3.55\ \mathrm{m/s^2}$.
Use this section's common-window results; these limits apply only to this data comparison.

Model score: acceptance limit 0.6, tolerance $\varepsilon = 0.1\ \mathrm{rad}$, error $e = 0.05\ \mathrm{rad}$. Calculate and interpret the score.

Tradeoff: which policy meets both limits? What if the IAE limit tightens to $3.0\ \mathrm{m \cdot s}$?

Fault validation: after these wind records, add two concrete regression cases for motor-failure response.

Reference Points

Filter Feasible Policies; Preserve Metric and Condition Differences

$$\eta = \frac{0.75 \times 0.1}{\sqrt{(0.75 \times 0.1)^2 + 0.05^2}} = \frac{0.075}{0.0901} \approx 0.832$$

Score: 0.832 exceeds 0.6; error is half the tolerance. It describes response matching, not accident probability.

Both limits: only tuned PID qualifies (IAE=3.1953, RMS=3.5310). At IAE≤3.0, none qualifies.

Add cases: ① same motor-failure parameters, old/new policies; ② motor fault plus degraded positioning or low battery.

Section Summary

Trace Experimental Claims through Inputs, Events, and Results

Entry and configuration: identify model, firmware, interface, and data versions; check running steps and success criteria separately.

Data and diagnosis: retain command/effect times, logs, and label provenance; check processing and training by independent flights.

Scoring and comparison: state components, units, windows, and targets; reproduce metrics consistently before interpreting tradeoffs.

Deliverables: save inputs, code, raw records, and calculations; state supported conclusions and conditions outside coverage.