PACE-Bench: Benchmarking Physics Adaptation via Code Evolution in Dynamic Environments

Yuhao Zhan[1,2]*, Bingxiang He[1]*, Zecong Tang[2], Chaojun Xiao[1]†
[1] Tsinghua University
[2]Zhejiang University
yuhao.zhan@zju.edu.cn

Abstract

Self-evolving agents improve future behavior from interaction experience, yet existing evaluations typically optimize under fixed execution conditions and do not test recovery after those conditions change. To address this gap, we introduce PACE-BENCH (**P**hysics **A**daptation via **C**ode **E**volution), a simulator-grounded benchmark of 144 source-to-target adaptation pairs across six physics domains. Each pair links a source environment to a mutated target environment with the same goal and interface. A code-driven design that succeeds in the source fails in the target, where agents must iteratively adapt it into a working target design using diagnostic sandbox feedback within a limited attempt budget. We compare ten self-evolving methods from four paradigms. The benchmark remains far from saturated: Reflexion + Qwen3-14B succeeds on only 35.9% of full-benchmark pairs, while GPT-5.5 solves 66.7% of the Statics subset under the full budget. Together, these results show that simulator-grounded reflection is more reliable than unverified self-revision, while memory anchors agents to early designs and broad tree search explores without converging. Even revealing exact physical changes does not raise the performance ceiling, pointing to mechanism redesign rather than parameter inference as the central bottleneck. Data and code are available at https://github.com/thunlp/PACE-Bench.

1 Introduction

Self-evolving agents improve future behavior by updating their parameters, context, memory, or tools from interaction experience (Gao et al., 2025). By turning interaction feedback into autonomous improvement, this paradigm reduces repeated human redesign and supports scalable, lifelong agent systems (Fang et al., 2025).

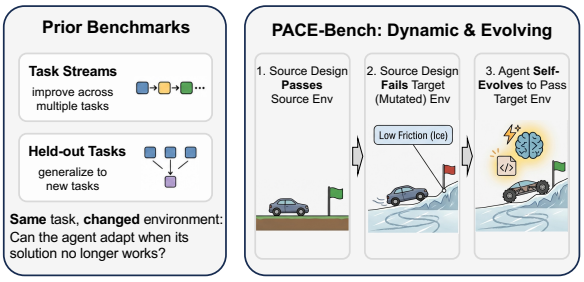

Figure 1: Existing self-evolving benchmarks test transfer or generalization, whereas PACE-BENCH tests adaptation of an executable design after physical environment shift.

Despite this promise, standard evaluations optimize an answer or policy under fixed execution rules. Recent benchmarks measure experience accumulation over task streams (Wu et al., 2024; Zheng et al., 2025a), including long-term performance, efficiency, and stability (Jiang et al., 2026), or transfer learned knowledge and skills to related tasks (Yuan et al., 2026; Zhong et al., 2026; Gao et al., 2026). Others test whether prompt, code, or harness updates generalize to held-out tasks and environments (Zheng et al., 2026; Zhang et al., 2025a). However, none evaluates whether an agent can adapt a previously successful design after a change in the execution environment causes it to fail. Such shifts are fundamental in deployment, where surroundings and operating conditions evolve frequently. Agents must identify obsolete assumptions, preserve useful components, and decide whether to revise or replace the design.

Physical simulation offers a controlled testbed for this adaptation, while code provides an executable interface for revising and verifying designs. Changes in friction, material strength, or dynamics can invalidate a working design without changing its goal. Figure 1 illustrates this with a vehicle that works on high-friction terrain but fails on ice. Existing physical benchmarks, however, either study simulator-based design or executable

*indicates equal contribution.
†indicates corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 35707–35753
October 24-29, 2026 ©2026 Association for Computational Linguistics

Benchmark	Feedback-Driven Improvement	Physical Simulation	Executable Design	Self-Evolution Evaluation	Adaptation After Env. Change
Self-evolving agent benchmarks					
StreamBench (Wu et al., 2024)	✓	×	×	✓	×
LifelongAgentBench (Zheng et al., 2025a)	✓	×	×	✓	×
SE-Bench (Yuan et al., 2026)	×	×	×	✓	×
SEA-Eval (Jiang et al., 2026)	✓	×	×	✓	×
SkillLearnBench (Zhong et al., 2026)	✓	×	×	✓	×
SEAGym (Zheng et al., 2026)	✓	×	×	✓	×
AutoEnv (Zhang et al., 2025a)	✓	×	×	✓	×
EvoAgentBench (Gao et al., 2026)	✓	×	×	✓	×
Physical and engineering benchmarks					
ENGDESIGN (Guo et al., 2026)	×	✓	✓	×	×
Frontier-Eng (Chi et al., 2026)	✓	✓	✓	✓	×
CausalWorld (Ahmed et al., 2020)	✓	✓	×	×	×
NewtonBench (Zheng et al., 2025b)	✓	✓	×	×	×
PACE-Bench	✓	✓	✓	✓	✓

Table 1: Comparison of representative benchmarks for self-evolution, engineering design, and physical environments. The highlighted final column is the core capability evaluated by PACE-BENCH; the other columns describe its evaluation setting.

feedback under fixed settings (Guo et al., 2026; Chi et al., 2026), or vary physical environments without self-evolving redesign (Ahmed et al., 2020; Zheng et al., 2025b). Adaptation of an existing design after an environment change therefore remains untested (Table 1).

To address this gap, we introduce PACE-BENCH, a benchmark of self-evolving adaptation after controlled environment changes using executable physical design. Following prior work on code-based agents (Wang et al., 2024) and executable physical simulation (Xie et al., 2026), each task starts with a code-driven design that succeeds in a source environment. A *mutation* then systematically changes physical parameters such as friction, material strength, or dynamics to create a target environment where the source design fails but a verified reference design succeeds. With the goal and interface fixed, the agent has 20 attempts to iteratively revise its design using diagnostic sandbox feedback. Across 36 base tasks and six physics domains, four targets per task yield 144 source-to-target pairs ($\tau_0 \rightarrow \tau_k$).

We compare ten self-evolving methods from four paradigms across Qwen3 model sizes, then evaluate larger open and frontier LLMs on selected subsets. The benchmark remains far from saturated: Reflexion with Qwen3-14B succeeds on only 35.9% of full-benchmark pairs, while GPT-5.5 solves 66.7% of the Statics subset under the full budget. Simulator-grounded reflection outperforms unverified self-revision, memory can anchor agents to early designs, and broad tree search often struggles to convert exploration into convergence. Code-similarity and nine-category error analyses trace these behaviors to Design Fixation versus Stagnation or undirected Exploration. Finally, revealing exact physical changes still does not raise the performance ceiling, indicating that mechanism redesign ("*know how*") is the genuine bottleneck compared to parameter inference ("*know what*"). Together, these findings position PACE-BENCH as a reproducible testbed for diagnosing and improving self-evolving agents under changing environments.

Contributions. (i) We introduce PACE-BENCH, a systematic benchmark for self-evolving adaptation after environment shifts using code-driven physical design, with a unified codebase and reproducible evaluation suite. (ii) We evaluate ten methods across four paradigms, showing that the benchmark remains unsaturated, simulator-grounded reflection is reliable, tree search is efficient, and memory can constrain stronger models. (iii) We analyze failure trajectories and parameter disclosure, revealing the tension between exploration and exploitation and the difficulty of mechanism redesign even when exact physical changes are known.

2 Related Work

Self-Evolving Benchmarks and Physical Design. Existing self-evolving benchmarks extend evaluation beyond isolated episodes. Some measure whether feedback improves performance over task streams (Wu et al., 2024; Zheng et al., 2025a), including long-term performance, efficiency, and sta-

bility (Jiang et al., 2026), or whether learned knowledge and skills transfer to related tasks (Yuan et al., 2026; Zhong et al., 2026; Gao et al., 2026). Other benchmarks modify prompts, code, or agent harnesses to test generalization to held-out tasks and heterogeneous environments (Zheng et al., 2026; Zhang et al., 2025a). None tests whether an agent can adapt its design when the environment changes while the goal and interface stay fixed. Physical adaptation offers a controlled testbed for this case. Existing work studies simulator-based design (Guo et al., 2026), executable feedback (Chi et al., 2026), causal transfer (Ahmed et al., 2020), or physical-law discovery (Zheng et al., 2025b). Yet none combines a failure after an environment change, code-driven mechanism redesign, and systematic self-evolution evaluation with a verified solvable target. PACE-BENCH bridges these settings by comparing self-evolving methods on executable designs after controlled physics shifts (Table 1).

Self-Evolving Agents. A self-evolving agent modifies its parameters, contextual state, or tools based on its own interaction trajectories to improve future performance (Gao et al., 2025). Recent work spans four paradigms: **context-based** revision through iterative refinement and self-critique (Shinn et al., 2023; Madaan et al., 2023); **memory-augmented** approaches that maintain structured experience across attempts (Zhao et al., 2024; Ouyang et al., 2025; Zhang et al., 2025b); **inference-time search** over solution candidates via evolutionary algorithms (Novikov et al., 2025; Assumpção et al., 2025); and **parameter-based** training through test-time RL (Wang et al., 2025b,a; Yuksekgonul et al., 2026), SFT (Zweiger et al., 2026), genetic search (Zhang et al., 2025c), or self-generated data (Zhao et al., 2026). These methods are driven by textual feedback, scalar rewards, or population-based selection. While prior benchmarks rarely evaluate self-evolving methods systematically, PACE-BENCH compares them across four paradigms under physical adaptation tasks.

3 The PACE-BENCH Benchmark

3.1 Task Formulation

A *base task* in PACE-BENCH is defined by a fixed context template $\mathcal{C}$ and five environments $\{\mathcal{E}_0, \ldots, \mathcal{E}_4\}$. The template specifies the natural-language description, constraints, success criteria, and permitted primitive APIs (*e.g.*, `sandbox.add_beam(x, y, width, height)`). Its environment-conditioned is:

$$\tau_i = \mathcal{C}(\mathcal{E}_i), \tag{1}$$

where $\mathcal{E}_0$ is the source environment and $\mathcal{E}_1, \ldots, \mathcal{E}_4$ (Stages 1–4) are mutated environments with systematically altered physical parameters. Thus, the five environment-conditioned instances share the task goal and interface but differ in physical conditions. We suppress the base-task index below.

Each environment $\mathcal{E}_i = (\theta_i, \phi_i)$ is parameterized by terrain θ_i and physics ϕ_i. A *solution* x is a Python program defining `build_agent()` (structure assembly) and `agent_action()` (control logic). The evaluator executes x in the Box2D[1] sandbox and returns structured feedback:

$$E_i(x) = \big(v(x),\ s(x),\ d(x)\big), \tag{2}$$

where $v(x) = K/N$ is the fraction of N hard constraints satisfied (*e.g.*, mass budget, no collapse), and $d(x)$ is a diagnostic report (*e.g.*, peak joint force, failure timestamp). The task score $s(x)$ is:

$$s(x) = \begin{cases} (v(x) - 1) \times 100, & \text{if } v(x) < 1, \\ \text{task progress}, & \text{if } v(x) = 1. \end{cases} \tag{3}$$

When $v(x) < 1$, the score penalizes violated constraints. When $v(x) = 1$, it measures progress toward the task goal. Moreover, $s(x) = 100$ if and only if $x \in \mathcal{X}$, where $\mathcal{X}$ denotes the set of reference solutions that pass the task.

3.2 Self-Evolving Evaluation Protocol

The core evaluation unit is a *source-to-target environment pair* $(\mathcal{E}_0 \rightarrow \mathcal{E}_k)$ within one base task, evaluated through the corresponding task instances $\tau_0 = \mathcal{C}(\mathcal{E}_0)$ and $\tau_k = \mathcal{C}(\mathcal{E}_k)$, where $k \in \{1, 2, 3, 4\}$. Each pair satisfies:

$$\mathcal{X}_k \neq \emptyset \quad \text{and} \quad x_0 \in \mathcal{X}_0 \setminus \mathcal{X}_k, \tag{4}$$

where $\mathcal{X}_k \neq \emptyset$ ensures that τ_k is solvable, and $x_0 \in \mathcal{X}_0 \setminus \mathcal{X}_k$ rules out trivial reuse of the source solution. The agent must adapt x_0 to τ_k without being told which physics changed, using feedback to infer the change and redesign accordingly.

The agent operates under an interaction budget $B = 20$. Each *attempt* submits a candidate x^t and returns feedback $(v^t, s^t, d^t) = E_k(x^t)$. At step

[1] Box2D is widely used in platforms such as OpenAI Gym (Brockman et al., 2016) and CARL (Benjamins et al., 2021).

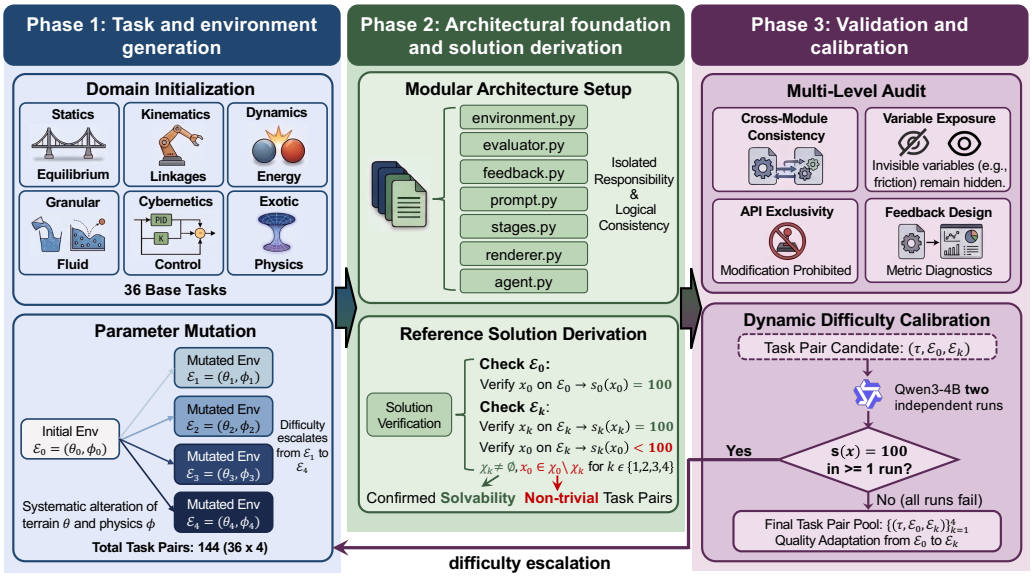

Figure 2: Overview of the three-phase dataset construction pipeline.

t, the agent $\mathcal{A}$, an LLM augmented with a self-evolving method, conditions on the task context and prior history:

$$x^t = \mathcal{A}\big(\mathcal{C}(\mathcal{E}_k) \oplus \mathcal{U},\ H^{t-1}\big), \tag{5}$$

$$H^{t-1} = \{(x^i, v^i, s^i, d^i)\}_{i=0}^{t-1}, \tag{6}$$

where $\mathcal{U}$ is a Uniform Suffix (see §A.1) listing variables that *might* differ across stages without identifying the changed variables. The final task score is:

$$s = \max_{0 \le t < B} s^t. \tag{7}$$

This protocol measures two core capabilities: *physical inference* (*i.e.*, "*know what*": diagnosing hidden parameter changes from feedback) and *mechanism redesign* (*i.e.*, "*know how*": revising a structure to function under altered physics). The 20-attempt budget tests whether agents converge through causal reasoning rather than unproductive trial and error. Each base task yields four such pairs ($\tau_0 \rightarrow \tau_1, \ldots, \tau_0 \rightarrow \tau_4$), providing a fine-grained measure across escalating distribution shifts.

3.3 Dataset Construction

We construct each base task by parameterizing $\mathcal{C}$ over $\{\mathcal{E}_0, \ldots, \mathcal{E}_4\}$ in a three-phase pipeline (Figure 2).

Phase 1: Task and environment generation. We define six physics domains with 6 base tasks per domain. Each base task contributes four source-to-target pairs, yielding $36 \times 4 = 144$ pairs with difficulty increasing from Stage 1 to Stage 4.

Phase 2: Architectural foundation and solution derivation. Each task uses a modular architecture of isolated single-responsibility components (Table 7), so parameter changes propagate across all layers. We derive reference solutions for all five environments through iterative refinement with Claude Code[2], yielding $\mathcal{X}_k \neq \emptyset$ for all k. In sandbox evaluation, the source solution satisfies $x_0 \in \mathcal{X}_0 \setminus \mathcal{X}_k$ at every mutated stage, matching the protocol conditions.

Phase 3: Validation. All tasks undergo automated audits for cross-module consistency, variable exposure, and API exclusivity. Prompts disclose explicit constraints and observable variables, omit numeric hidden-physics values such as gravity and friction, and restrict reference solutions to documented primitives (Appendix A.1). We calibrate difficulty by adjusting mutation parameters (*e.g.*, wider gaps or stricter force limits) until Qwen3-4B fails two independent runs while the task remains solvable (Appendix A.2). We then analyze its logs and add diagnostic measurements to `feedback.py`, such as force or constraint margins with timestamps and locations. These diagnostics report *what failed* and by *what margin* without recommending a fix. If enriched feedback enables Qwen3-4B to pass, we escalate the difficulty again (Appendix A.3). Finally, two authors with graduate-level physics or engineering training audit all 36 tasks over three passes. Moderate-to-critical issues fall from approximately 86% of tasks in the first pass to 15–20% in the second and none in the third (Appendix A.4).

[2]https://code.claude.com/docs/en/overview

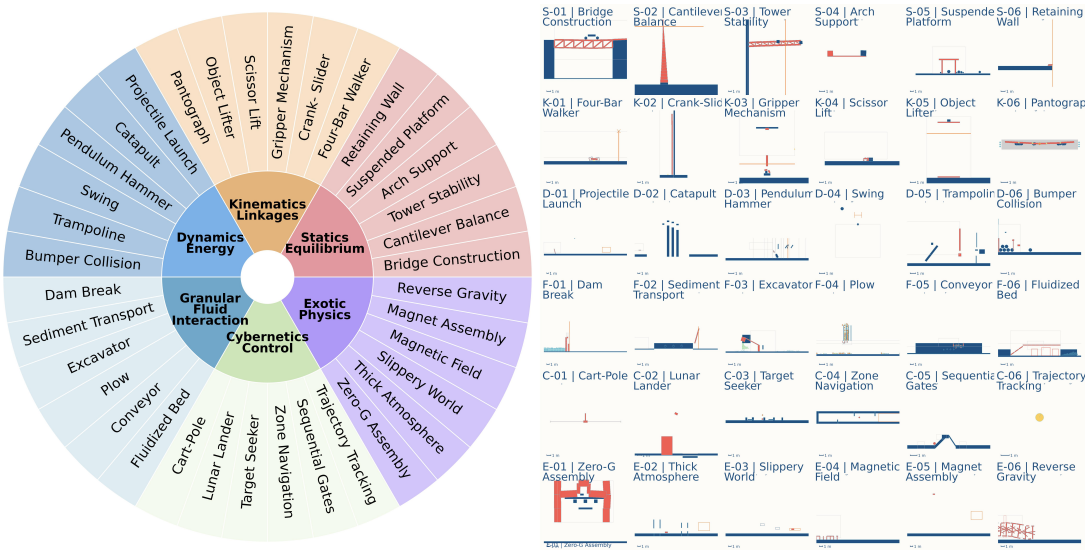

Figure 3: Overview of PACE-BENCH. Left: the 36 tasks organized across six physics categories. Right: rendered snapshots of reference solutions succeeding on the source environments.

3.4 Dataset Overview

Figure 3 presents PACE-BENCH. Detailed statistics appear in Table 9 in the Appendix. Averaged across 6 categories, a task prompt contains 1,051 tokens, 7 hard constraints, and 8 primitive APIs, with mutated parameters escalating from 2 at Stage 1 to 10 at Stage 4. In total, the benchmark spans 180 evaluation environments (36 tasks $\times$ 5 environments), 37,860 prompt tokens, 273 hard constraints, 292 primitive APIs, and 945 stage-level parameter mutations. Each pair tests whether an agent can infer hidden physical changes from feedback and revise its design, making PACE-BENCH the first benchmark to combine environment variation and physics-grounded self-evolving evaluation.

4 Experiments

4.1 Experimental Setup

4.1.1 Baseline

Models. Because sweeping larger models across all tasks and methods is prohibitively expensive (cost details in Appendix D.2), we use two complementary scopes. The full benchmark evaluates Vanilla and all ten self-evolving methods with Qwen3-4B, Qwen3-8B, and Qwen3-14B (Yang et al., 2025). On the Statics subset (24 environment pairs), a 20-attempt Vanilla study adds Qwen3-32B, DeepSeek-V4-Pro, and GPT-5.5 to test whether larger models close the performance gap under the full budget. A separate five-attempt Vanilla study broadens frontier-model coverage at lower cost with GPT-5.5 (OpenAI, 2025), Gemini-3.1-Pro (DeepMind, 2025), Claude-Opus-4.7 (Anthropic, 2 025), DeepSeek-V4-Pro (DeepSeek-AI, 2025), Kimi-K2.6 (AI, 2026), and MiniMax-M2.7 (MiniMax, 2026). All models use `thinking` mode.

Methods. Under the protocol in §3.2, Vanilla iteratively submits candidate solutions and revises them from diagnostic feedback. Ten self-evolving methods extend this protocol across four paradigms: **Context-based**: Reflexion (Shinn et al., 2023), Self-Refine (Madaan et al., 2023); **Memory-augmented**: ACE (Zhang et al., 2025b), ExpeL (Zhao et al., 2024), ReasoningBank (Ouyang et al., 2025); **Inference-time Search**: Tree-of-Thoughts (ToT) (Yao et al., 2023), CodeEvolve (Assumpção et al., 2025); and **Parameter-based**: SEAL (Zweiger et al., 2026), RAGEN (Wang et al., 2025b), TTT-Discover (Yuksekgonul et al., 2026). Detailed adaptation notes are in Appendix C.

By default, agents see only the Uniform Suffix $\mathcal{U}$ (§3.2), which lists variables that *might* differ across stages without identifying the actual changes. Appendix D provides implementation details, including history management, infrastructure, and costs.

4.1.2 Evaluation Metrics

We report two primary metrics over two independent runs per environment pair. **Pass@2** is the fraction of environment pairs where at least one run succeeds. **Score@2** is the mean of the two run-best scores, with per-attempt scores in $[-100, 100]$. Runs lacking a complete executable solution due to output truncation, parse failure, sandbox crashes, or context overflow are excluded; Appendix D.3

Paradigm	Method	Qwen3-4B		Qwen3-8B		Qwen3-14B	
		Pass@2	Score@2	Pass@2	Score@2	Pass@2	Score@2
Context	Vanilla	11.5 (↑0.0)	10.7 (↑0.0)	15.6 (↑0.0)	12.0 (↑0.0)	32.0 (↑0.0)	25.5 (↑0.0)
	Reflexion	19.5 (↑8.0)	14.6 (↑3.9)	26.6 (↑11.0)	18.1 (↑6.1)	35.9 (↑3.9)	28.0 (↑2.5)
	Self-Refine	6.2 (↓5.3)	5.5 (↓5.2)	3.1 (↓12.5)	3.7 (↓8.3)	7.1 (↓24.9)	8.0 (↓17.5)
Memory	ACE	15.5 (↑4.0)	13.3 (↑2.6)	19.5 (↑3.9)	14.9 (↑2.9)	25.0 (↓7.0)	19.6 (↓5.9)
	ExpeL	11.5 (↑0.0)	10.5 (↓0.2)	13.2 (↓2.4)	10.1 (↓1.9)	15.6 (↓16.4)	13.0 (↓12.5)
	ReasoningBank	10.9 (↓0.6)	9.2 (↓1.5)	18.8 (↑3.2)	14.0 (↑2.0)	24.2 (↓7.8)	19.5 (↓6.0)
Search	ToT	23.4 (↑11.9)	17.9 (↑7.2)	17.1 (↑1.5)	14.5 (↑2.5)	20.3 (↓11.7)	16.5 (↓9.0)
	CodeEvolve	10.7 (↓0.8)	5.8 (↓4.9)	5.3 (↓10.3)	3.1 (↓8.9)	5.3 (↓26.7)	3.3 (↓22.2)
Parameter	TTT-Discover	6.5 (↓5.0)	11.3 (↑0.6)	7.6 (↓8.0)	12.3 (↑0.3)	14.3 (↓17.7)	17.3 (↓8.2)
	RAGEN	7.7 (↓3.8)	12.2 (↑1.5)	7.0 (↓8.6)	15.4 (↑3.4)	14.5 (↓17.5)	26.4 (↑0.9)
	SEAL	10.4 (↓1.1)	17.0 (↑6.3)	15.4 (↓0.2)	17.9 (↑5.9)	20.0 (↓12.0)	23.5 (↓2.0)

Table 2: Main Results: Pass@2 (%) and Score@2 by Method and Model. Values in parentheses show Δ vs. Vanilla (↑ = improvement, ↓ = degradation).

Model	Pass@2 (%)	Score@2
Qwen3-4B	8.3	12.0
Qwen3-8B	33.3	21.8
Qwen3-14B	37.5	27.3
Qwen3-32B	37.5	28.4
DeepSeek-V4-Pro	45.8	48.7
GPT-5.5	**66.7**	**78.1**

Table 3: Vanilla performance on Statics subset under the full 20-attempt budget.

details these exclusions. Appendix D.4 also reports reliability statistics and the cost rationale for the two-run protocol.

4.2 Results and Analysis

We organize the experiments around three progressive questions: (RQ1) how well current self-evolving methods perform under dynamic physics; (RQ2) what makes these tasks hard and how agents fail; and (RQ3) which design choices and interventions improve performance.

4.2.1 RQ1: How well do current self-evolving methods perform under dynamic physics?

Takeaway 1: Reflexion leads in Pass@2 and ToT in wall-clock efficiency. Memory-augmented methods anchor search to early designs, while Self-Refine's unverified inner loop compounds errors and adds latency.

Overall Performance. Table 2 reports Pass@2 and Score@2 across methods and models. The strongest configuration, Reflexion + Qwen3-14B, reaches only 35.9% Pass@2. Gains are larger from 8B to 14B than from 4B to 8B (Figure 6 in Appendix). To test whether larger models close this gap, we evaluate Vanilla on 24 environment pairs from the Statics subset under the full 20-attempt budget, adding Qwen3-32B, DeepSeek-V4-Pro, and GPT-5.5. Table 3 suggests a within-family plateau: Qwen3-32B does not improve Pass@2 over Qwen3-14B, whereas DeepSeek-V4-Pro and GPT-5.5 perform better. Even GPT-5.5 fails one third of the pairs, leaving the benchmark unsaturated. A complementary five-attempt comparison broadens coverage to six frontier LLMs in Appendix D.5.

From-Scratch Difficulty. To test whether the tasks remain non-trivial when agents must construct solutions from scratch, we run Vanilla on all five environments for each task. Each agent receives the task description and primitive APIs, then revises its solution with simulator feedback for up to 20 attempts. Table 15 reports low Pass@2 across model scales (11.3–18.3%), showing that direct solution construction remains non-trivial. Table 16 shows a much higher Pass@2 on the source environment (32.3%) than on the target environments (6.4–13.4%). Thus, our mutations create substantially harder targets, motivating source-to-target adaptation via re-design.

Analysis by Method Paradigm. Table 2 reveals clear paradigm-level patterns. **Context-based. Reflexion** leads overall (35.9% Pass@2 at 14B, +4–11 points over Vanilla), while **Self-Refine**, which revises up to five times before sandbox evaluation, never exceeds 7.1%. This gap suggests that unverified revision compounds errors rather than correcting them. **Memory-augmented. ACE** and **ReasoningBank** outperform Vanilla at 8B but underperform at 14B (25.0%, 24.2% vs. 32.0%), suggesting retrieved experiences constrain stronger mod-

Model	Metric	Vanilla	Reflexion	Self-Refine	ACE	ExpeL	Reasoning Bank	ToT	Code Evolve	SEAL	RAGEN	TTT-Discover
Qwen3-4B	Score	24.8	23.7	11.9	23.9	21.5	21.8	**27.4**	13.4	23.3	13.5	14.2
	Time	0.9h	0.8h	6.0h	1.3h	1.4h	0.8h	**0.6h**	1.8h	2.0h	5.5h	5.4h
	S/Hr	27.5	29.6	2.0	18.4	15.4	27.2	**45.7**	7.5	11.6	2.5	2.6
Qwen3-8B	Score	25.5	22.1	15.0	25.7	22.6	25.0	23.2	8.9	**28.0**	14.2	14.7
	Time	1.1h	1.5h	5.1h	1.3h	1.4h	1.0h	**0.5h**	2.2h	2.5h	5.8h	5.7h
	S/Hr	23.2	14.7	2.9	19.8	16.1	25.0	**46.4**	4.1	11.2	2.4	2.6
Qwen3-14B	Score	25.2	**38.1**	18.0	26.3	24.8	25.5	28.9	8.9	35.0	37.1	16.4
	Time	0.9h	1.5h	5.0h	1.5h	1.6h	1.0h	**0.5h**	2.4h	2.8h	5.7h	5.6h
	S/Hr	28.0	25.4	3.6	17.5	15.5	25.5	**57.8**	3.7	12.5	6.5	2.9

Table 4: Cost-normalized results on the Kinematics subset under 20-attempt budget. Scores are fitted to the nearest $1/24$ increment and shown to one decimal. S/Hr is computed from the fitted scores before display rounding.

els. **ExpeL** also trails Vanilla at both scales, possibly because its frozen source-environment memory transfers poorly. **Inference-time Search. ToT** excels at 4B (23.4%, 2.0× Vanilla), but its advantage erodes at larger scales, while **CodeEvolve** scales negatively (10.7%→5.3%). **Parameter-based.** All three methods underperform Vanilla at 14B. **SEAL** is strongest (10.4–20.0%), **RAGEN**'s gap widens with scale ($-3.8 \rightarrow -8.6 \rightarrow -17.5$), and **TTT-Discover** improves at 14B (14.3%) but remains below Vanilla. Detailed failure analysis via error taxonomy and code similarity is in RQ2 (§4.2.2).

Cost-Normalized Comparison. To account for unequal compute overhead, we compare all eleven methods on the Kinematics subset with Qwen3-4B/8B/14B under the 20-attempt budget. Table 4 reports runtime in hours and Score/Hr, defined as average score divided by runtime. At the paradigm level, inference-time search splits sharply. **ToT** is the most efficient method at every scale (45.7–57.8 S/Hr), whereas **CodeEvolve** remains low (3.7–7.5 S/Hr). Context-based methods also diverge. **Reflexion** combines low overhead with strong efficiency, while **Self-Refine**'s unverified inner loop is slow and yields only 2.0–3.6 S/Hr. Memory-augmented methods achieve moderate efficiency, and parameter-based methods incur high runtime without a consistent efficiency gain. Thus, **ToT** remains the efficiency leader even though the highest absolute score varies by scale, suggesting that additional self-evolution compute does not reliably improve cost-normalized performance.

4.2.2 RQ2: What makes these tasks hard, and how do agents fail?

Takeaway 2: Self-evolving agents either fixate on early designs or explore without converging. Model scaling appears to improve reasoning more than combinatorial search.

Analysis Methods. We use two complementary analyses. **Code similarity analysis.** We compute pairwise Jaccard similarity among generated solutions within each run and report global similarity (mean pairwise similarity), convergence trend (the change from early to late attempts), and radicality ($1 -$ global similarity). Positive convergence trends indicate increasing self-similarity. Table 5 compares these metrics across methods and model scales, revealing whether agents lock onto a narrow design family or continue making large changes.[3]

Error taxonomy. Every failed run is assigned one of nine mutually exclusive types. Hard failures detected from simulation diagnostics include Catastrophic Collapse, Constraint Violation, Structural Failure, and Numerical Instability. Trajectory-level patterns identified from score progression and code similarity include Design Fixation, Stagnation, Exploration, Late Convergence, and Budget Exhaustion. A priority-ordered pipeline checks hard failures first, then trajectory-level patterns, with the first match determining the type. Details are in Appendix B.

Analysis by Method. Figure 4 and Table 5 indicate a split between exploration and exploitation behind the RQ1 performance patterns: conservative methods become trapped in self-similar designs, whereas highly radical methods often change code without making directed progress.

Context-based. Vanilla serves as the baseline: 41.0% Design Fixation, global similarity 0.777, positive trend (+0.117). The base model converges early to a narrow design region and rarely escapes. **Self-Refine**'s unverified inner-loop revisions produce chaotic, undirected changes (highest radicality, Stagnation 57.1%), which may explain why it ranks lowest.

Memory-augmented. ACE cuts Stagnation to

[3]Methods with < 10 total iterations (multiple attempts per iteration) are excluded for meaningful trend analysis.

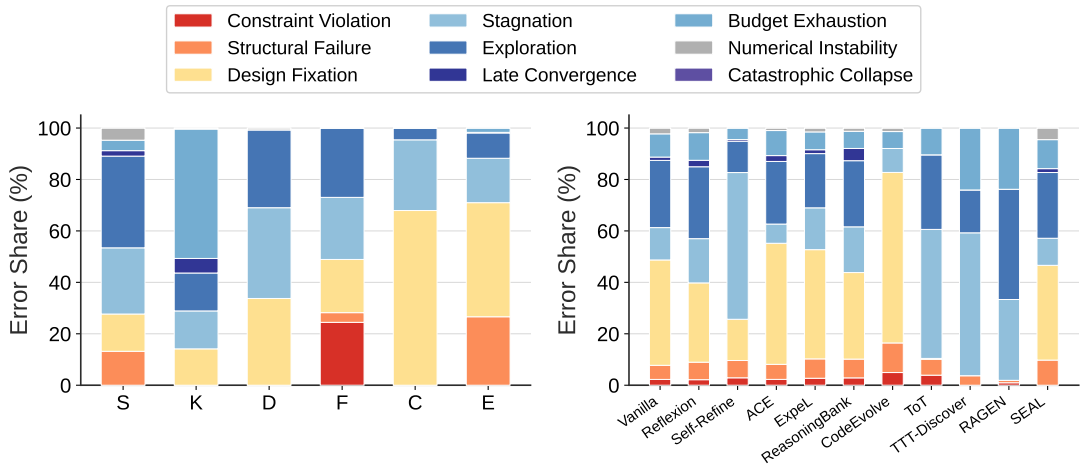

Figure 4: Error type distribution by category (left) and method (right).

Method	Qwen3-4B			Qwen3-8B			Qwen3-14B		
	Global Sim	Trend (L−E)	Radicality	Global Sim	Trend (L−E)	Radicality	Global Sim	Trend (L−E)	Radicality
Vanilla	0.792	+0.112	0.208	0.780	+0.126	0.220	0.760	+0.114	0.240
Reflexion	0.772	+0.111	0.228	0.725	+0.117	0.275	0.720	+0.110	0.280
Self-Refine	0.672	+0.120	**0.328**	0.598	+0.122	**0.402**	0.597	+0.119	**0.403**
ACE	0.821	+0.145	0.179	0.795	+0.164	0.205	**0.807**	+0.163	0.193
ExpeL	**0.840**	+0.132	0.160	**0.807**	+0.152	0.193	0.794	+0.139	0.206
ReasoningBank	0.760	**+0.207**	0.240	0.738	**+0.198**	0.262	0.723	**+0.233**	0.277
SEAL	0.755	+0.097	0.245	0.805	+0.117	0.195	0.726	+0.096	0.274

Table 5: Code Similarity by Method and Model (iterations $\geq$ 10). Trend (L−E) = mean similarity difference between late and early attempts; positive = increasing self-similarity. Bold denotes the highest value in each column.

7.5% (vs. Vanilla 12.6%) but drives Fixation to 47.1%: score-ranked retrieval repeatedly returns early moderate successes, which are reinserted into later prompts and create a positive-feedback anchoring loop. **ReasoningBank** has the steepest convergence trend (+0.233) of any method, suggesting progressive anchoring. At 14B, ACE's global similarity peaks at 0.807, indicating that memory may constrain models which reason better from scratch. **ExpeL**'s Fixation (42.5%) exceeds Vanilla's while Exploration (21.1%) is lower, suggesting that frozen source-environment memory narrows rather than guides search. To probe score-retrieval anchoring over time, we include a case study comparing adjacent-attempt similarity for Vanilla and ACE over one 20-attempt trajectory in Appendix E.2.

Inference-time Search. ToT has near-zero Fixation (0.3%) but 50.2% Stagnation: breadth-first exploration cannot converge within 20 attempts. **CodeEvolve** is the inverse, with 66.3% Fixation (highest), 0.0% Exploration, 0.835 global similarity, and a flat trend. Its LLM-generated crossover and mutation variants remain too close to selected parent templates, so the population collapses prematurely.

Parameter-based. SEAL's error profile mirrors Vanilla's (Fixation 36.8% vs. 41.0%, Stagnation 10.5% vs. 12.6%), because supervised fine-tuning on its own successes preserves rather than reshapes the base generation distribution. **RAGEN** has zero Fixation (weight perturbation promotes generation diversity) yet 42.9% Exploration and 31.4% Stagnation: the continuous reward is informative but 20 rollouts are too few for RL convergence. **TTT-Discover** has the highest Stagnation (55.6%) with zero Fixation because group-based advantage estimation degenerates when most rollouts score identically.

Analysis by Task Category. Figure 4 reports the error taxonomy by category. Failure modes vary sharply across categories, with Design Fixation reaching 58.2% in **Control**. Figure 7 in Appendix further shows that **Exotic Physics** is the most solvable category while **Dynamics** is the hardest across model scales. Task category predicts difficulty far more strongly than model scale: the 4B→14B gain ranges from +16.3 points in **Exotic Physics** to +1.9 points in **Kinematics**. Scaling may improve reasoning more than the combinatorial search required

	Qwen3-4B			Qwen3-8B			Qwen3-14B		
Method	**CH**	**CE**	**Δ**	**CH**	**CE**	**Δ**	**CH**	**CE**	**Δ**
Vanilla	1.6	10.0	**+8.4**	13.3	13.3	+0.0	17.0	9.8	−7.2
Reflexion	9.4	12.0	**+2.6**	**17.9**	13.2	−4.7	17.1	**14.6**	−2.5
ACE	6.0	12.0	**+6.0**	12.5	8.8	−3.7	7.3	10.4	+3.1

Table 6: Change-Hidden (CH) vs. Change-Exposed (CE) Pass@2 (%) pooled over Statics and Kinematics (approximately 48 environment pairs). CE discloses the changed variables and their exact new values.

by the hardest categories. Detailed per-category error analysis and cross-category comparisons are provided in Appendix E.3.

4.2.3 RQ3: What design choices and interventions improve performance?

Takeaway 3: More information is not uniformly helpful: exact parameters do not appear to improve the strongest hidden-setting result, while video feedback tends to help memory-based methods but can hinder context-based methods.

Information Asymmetry. To distinguish "know what" from "know how", we compare the default *Change-Hidden* (CH) setting with *Change-Exposed* (CE), which reveals the changed variables and their exact values. Table 6 reports Vanilla, Reflexion, and ACE at all three model scales across Statics and Kinematics. CE improves all 4B configurations (+2.6 to +8.4 points), but at 8B/14B five of six changes remain within ± 5 points, while Vanilla-14B drops by 7.2 points. The error taxonomy in Table 13 shows that CE reduces Design Fixation in five of six ACE/ExpeL settings, suggesting disclosed variables can complement retrieved experiences. Yet ACE still loses 3.7 points at 8B, so this behavioral gain does not consistently translate into Pass@2. Critically, the best CE result (Reflexion-14B, 14.6%) remains below the best CH result (Reflexion-8B, 17.9%). Thus, revealing what changed does not raise the performance ceiling, suggesting that mechanism redesign ("*know how*") is harder than parameter inference ("*know what*").

VLM Video Feedback. To test whether visual evidence resolves failures missed by scalar diagnostics, we augment textual feedback with natural-language failure descriptions generated from videos of candidate executions in the Box2D simulator. An intermediate VLM, Gemma4-26B-A4B (Team et al., 2026), produces these descriptions. We apply this intervention to representative methods on the Statics and Kinematics subsets while keeping the backbone text LLM fixed. Table 11 in Appendix compares text-only and VLM-augmented feedback. At 4B, additional video feedback helps most methods. At larger scales, it polarizes by paradigm. Context-based methods degrade (Vanilla 14B: $40.9\% \rightarrow 18.2\%$, Reflexion: $40.9\% \rightarrow 27.3\%$), while memory-augmented methods improve (ACE: $13.6\% \rightarrow 22.7\%$, ExpeL: $13.6\% \rightarrow 36.4\%$). The error taxonomy across these subsets in Table 12 suggests possible mechanisms. For context-based methods, video feedback may scatter focused reasoning, with Exploration rising by over 20 points in several settings. For memory-based methods, fresh visual evidence may counter stale retrieved experiences, with Fixation reductions exceeding 25 points in some settings. Search-based ToT is harmed at 4B and 14B, possibly because video feedback adds redundant variation to already-broad exploration.

5 Conclusion

We introduced PACE-BENCH, a benchmark of 144 solvable but non-trivial source-to-target environment pairs across six physics domains with systematic hidden environment mutations and simulator-grounded feedback. Within 20 attempts, agents must infer what changed and revise executable code-driven designs to function under target physics. Frontier LLMs remain unsaturated, and most self-evolving methods deliver limited gains over Vanilla or underperform it. Reflexion's explicit simulator-grounded failure analysis improves performance, whereas Self-Refine's unverified inner-loop revisions compound errors. More broadly, agents either fixate on early designs (often in memory-augmented methods) or explore without converging (as in ToT-style inference-time search). Revealing the changed parameters does not improve the performance ceiling, indicating that mechanism redesign ("*know how*") rather than parameter inference ("*know what*") is the central bottleneck. Together, these results position PACE-BENCH as a testbed for developing this capability.

Acknowledgements

This work is supported by the China National Postdoctoral Program for Innovative Talents (grant no. BX20250388), Tsinghua University (Department of Computer Science and Technology)-Siemens Ltd., China Joint Research Center for Industrial Intelligence and Internet of Things (JCIIOT), Institute Guo Qiang at Tsinghua University.

Limitations

PACE-BENCH evaluates adaptation in controlled 2D Box2D simulations, so its findings may not transfer to 3D environments or real robots with noisy sensing and actuation. The fixed 20-attempt budget limits the interaction horizon, leaving sustained adaptation untested. Due to computational cost, the frontier-model, parameter-disclosure, and visual-feedback evaluations cover only selected domains and methods. Future work should extend the benchmark to richer simulators and real hardware, longer adaptation horizons, and broader models and methods.

Ethical Considerations

PACE-BENCH is a research benchmark for evaluating physical intelligence in LLM agents. It does not involve human subjects, personally identifiable information, or deployment in safety-critical systems. All tasks are synthetic and run in simulation with no real-world physical consequences. Because automated mechanical design carries potential dual-use implications, we encourage responsible development. We release the benchmark publicly to promote transparency and reproducibility.

LLM Usage

During the preparation of this manuscript, we used large language models (LLMs) to assist with language polishing and improving the clarity and readability of the paper. The LLMs were not used to generate research hypotheses, design the methodology, conduct experiments, analyze results, or draw conclusions. All LLM-assisted edits were carefully reviewed and revised by the authors, who take full responsibility for the final content of the manuscript.

References

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. 2020. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*.

Moonshot AI. 2026. Kimi k2.6.

Anthropic. 2 025. Claude opus 4.7.

Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. 2025. Codeevolve: An open source evolutionary coding agent for algorithm discovery and optimization. *arXiv preprint arXiv:2510.14150*.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. 2021. Carl: A benchmark for contextual and adaptive reinforcement learning. *Preprint*, arXiv:2110.02102.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *Preprint*, arXiv:1606.01540.

Yizhe Chi, Deyao Hong, Dapeng Jiang, Tianwei Luo, Kaisen Yang, Boshi Zhang, Zhe Cao, Xiaoyan Fan, Bingxiang He, Han Hao, and 1 others. 2026. Frontier-eng: Benchmarking self-evolving agents on real-world engineering tasks with generative optimization. *arXiv preprint arXiv:2604.12290*.

Google DeepMind. 2025. Gemini 3.1 pro: Our most capable model for agents and coding.

DeepSeek-AI. 2025. Deepseek-v4 technical report.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *Preprint*, arXiv:2508.07407.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1.

Xingze Gao, Chuanrui Hu, Hongda Chen, Pengfei Yao, Zhao Wang, Yi Bai, Zhengwei Wu, Yunyun Han, Xiaofeng Cong, Jie Gui, Yafeng Deng, and Teng Li. 2026. Evoagentbench: Benchmarking agent self-evolution via ability transfer. *Preprint*, arXiv:2607.05202.

Xingang Guo, Yaxin Li, Xiangyi Kong, Yilan Jiang, Xiayu Zhao, Zhihua Gong, Yufan Zhang, Daixuan Li, Tianle Sang, Beixiao Zhu, and 1 others. 2026. Toward engineering agi: Benchmarking the engineering design capabilities of llms. *Advances in Neural Information Processing Systems*, (17).

Sihang Jiang, Lipeng Ma, Zhonghua Hong, Keyi Wang, Zhiyu Lu, Tengfei Wang, Shisong Chen, Jinghao Zhang, Tianjun Pan, Weijia Li, Jiaqing Liang, and Yanghua Xiao. 2026. Sea-eval: A benchmark for evaluating self-evolving agents beyond episodic assessment. *Preprint*, arXiv:2604.08988.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

MiniMax. 2026. Minimax m2.7.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

OpenAI. 2025. Introducing gpt-5.5.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, and 1 others. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, and 304 others. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. *Preprint*, arXiv:2402.01030.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, and 1 others. 2025a. Thetaevolve: Test-time learning on open problems. *arXiv preprint arXiv:2511.23473*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025b. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung yi Lee. 2024. Streambench: Towards benchmarking continuous improvement of language agents. *Preprint*, arXiv:2406.08747.

Tianyidan Xie, Peiyu Wang, Yuyi Qian, Yuxuan Wang, Rui Ma, Ying Tai, Song Wu, Qian Wang, Lanjun Wang, and Zili Yi. 2026. Physcodebench: Benchmarking physics-aware symbolic simulation of 3d scenes via self-corrective multi-agent refinement. *Preprint*, arXiv:2604.23580.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Jiarui Yuan, Tailin Jin, Weize Chen, and Zeyuan Liu. 2026. Se-bench: Benchmarking self-evolution with knowledge internalization. *Preprint*, arXiv:2602.04811.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and 1 others. 2026. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*.

Jiayi Zhang, Yiran Peng, Fanqi Kong, Cheng Yang, Yifan Wu, Zhaoyang Yu, Jinyu Xiang, Jianhao Ruan, Jinlin Wang, Maojia Song, HongZhang Liu, Xiangru Tang, Bang Liu, Chenglin Wu, and Yuyu Luo. 2025a. Autoenv: Automated environments for measuring cross-environment agent learning. *Preprint*, arXiv:2511.19304.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, and 1 others. 2025b. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Yiqun Zhang, Peng Ye, Xiaocui Yang, Shi Feng, Shufei Zhang, Lei Bai, Wanli Ouyang, and Shuyue Hu. 2025c. Nature-inspired population-based evolution of large language models. *arXiv preprint arXiv:2503.01155*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 17, pages 19632–19642.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2026. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, (17):105816–105879.

Congjie Zheng, Chuanyi Xue, Bin Liang, Jun Yang, and Changshui Zhang. 2026. Seagym: An evaluation environment for self-evolving llm agents. *Preprint*, arXiv:2606.17546.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. 2025a. Lifelongagentbench: Evaluating llm agents as lifelong learners. *Preprint*, arXiv:2505.11942.

Tianshi Zheng, Kelvin Kiu-Wai Tam, Newt Hue-Nam K Nguyen, Baixuan Xu, Zhaowei Wang, Jiayang Cheng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, and 1 others. 2025b. Newtonbench: Benchmarking generalizable scientific law discovery in llm agents. *arXiv preprint arXiv:2510.07172*.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. 2026. Skilllearnbench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *Preprint*, arXiv:2604.20087.

Adam Zweiger, Jyo Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. 2026. Self-adapting language models. *Advances in Neural Information Processing Systems*, (17):74084–74115.

Module	Role
`environment.py`	Physics and terrain configuration
`evaluator.py`	Success criteria and scoring logic
`feedback.py`	Diagnostic metrics
`prompt.py`	Task context $\mathcal{C}$
`stages.py`	Mutation specs for $\mathcal{E}_{1-4}$
`renderer.py`	Simulation and GIF generation
`agent.py`	Reference solutions for $\mathcal{E}_{0-4}$

Table 7: Modules of each task in PACE-BENCH.

A Dataset Construction Details

A.1 Module Auditing

An automated audit script (`auto_audit.sh`) systematically checks each task directory against seven rules:

- *Cross-module consistency*: physical parameters, constraints, and success criteria are checked for coherence across `environment.py`, `evaluator.py`, `feedback.py`, `prompt.py`, `stages.py`, and `renderer.py`. See Table 7 for the role and responsibility of each module.
- *Constraint completeness*: every hardcoded constraint value in `environment.py` (*e.g.*, mass budgets, force limits) is checked against its numeric value in `prompt.py`.
- *Visible variable synchronization*: every visually observable variable (*e.g.*, gap widths, target positions) is checked against its numeric value in `prompt.py` and the current environment configuration.
- *Invisible variable sweep*: any numeric value of invisible physics variables (*e.g.*, gravity, friction, damping) found in `prompt.py` is deleted, leaving only qualitative descriptions.
- *Mutation synchronization*: when a constraint or visible variable changes across stages in `stages.py`, the prompt is checked for the new value in the format "`new_value (originally old_value)`".
- *Uniform Suffix $\mathcal{U}$ tone*: the suffix appended to mutated task descriptions is checked for generic warnings without specific numeric values or directions of change.
- *Sandbox execution*: every target-environment reference solution must score 100, while the source solution must fail on all four target environments.

Violations are reported with file paths and line numbers for immediate correction. The full audit prompt appears in Appendix F.

These rules collectively enforce the information asymmetry central to PACE-BENCH: constraint and visible variables are always disclosed with precise values, while invisible variables are systematically scrubbed from the prompt and the Uniform Suffix $\mathcal{U}$ provides only generic awareness that shifts may exist, forcing agents to discover the specific physics of each environment through interaction.

Audit Outcomes. Table 8 summarizes the three severity levels used in the human audit and provides a representative example for each. Under this rubric, the first pass identified Lv 2–3 issues in approximately 86% of tasks. After correction, the second pass found residual issues in approximately 15–20%. A third pass reran the checklist and found no remaining issues across the 36 tasks. Claude Code assisted construction, reference-solution derivation, feedback design, and difficulty tuning, but it does not judge submitted solutions: hard-coded evaluator logic and Box2D execution determine all scores. Figure 5 shows that Claude-Opus-4.7 ranks third among the six proprietary models in the constrained-budget study, consistent with the absence of a Claude-specific scoring advantage.

A.2 Difficulty Escalation

The difficulty escalation pipeline uses Qwen3-4B only as a difficulty check. For each adaptation $\tau_0 \rightarrow \tau_k$, the model is evaluated for two independent runs. If it achieves $s(x) = 100$ on any run, the script adjusts the mutation parameters in `stages.py` to create a harder configuration (*e.g.*, wider gaps, higher gravity, or stricter force limits). After each adjustment, we rerun the target-environment reference solution and revise it if necessary until it scores 100. Only then do we re-evaluate Qwen3-4B. This loop continues until the model fails both runs. Qwen3-4B does not construct the task or determine success criteria, and the target environment is shared unchanged by all evaluated models. The escalation focuses on invisible variables so that difficulty increases cannot be trivially detected from prompt changes alone.

A.3 Feedback Design

Each task includes a dedicated `feedback.py` module that translates raw simulation metrics into structured diagnostics. The feedback follows four design principles:

- *Physics-grounded*: reports numerical physical quantities (*e.g.*, peak joint forces and torques, collision impulses, structural deformation, constraint margins) with timestamps and locations,

Severity	Definition and example
Lv 3: Critical	Task unsolvable or evaluator broken (e.g., prompt constraints contradict the evaluator).
Lv 2: Moderate	Task solvable but prompt misleading (e.g., a stale visible-variable value).
Lv 1: Minor	Cosmetic issue (e.g., a renderer color mismatch or unused import).

Table 8: Severity levels used during human verification.

instead of binary pass/fail.

- *Dynamic thresholds*: all limit values are read from the current environment configuration rather than hardcoded to maintain accuracy across target environments.
- *Diagnostic, not prescriptive*: observations identify *what* failed and *by what margin* (*e.g.*, "Joint at anchor A broke at step 42 because torque exceeded the limit by 35%"), without prescribing engineering fixes.
- *Phase-aware*: for tasks with distinct simulation phases (*e.g.*, static load, dynamic traversal), metrics are reported per phase to help the agent isolate when and where failures occur.

A.4 Human Verification

Personnel and Coverage. To verify benchmark quality independently of LLM-assisted construction, two authors with graduate-level physics or engineering training divided the six categories, each auditing three categories (18 tasks). All 36 tasks and all LLM-generated modules were manually cross-checked against executable task behavior rather than model judgment.

Protocol. For every task, the auditor executed the reference solution on the source and target environments and recorded $s = 100$, then ran the source solution and observed failure on all four target environments. The auditor also checked exact numeric consistency across `environment.py`, `evaluator.py`, `feedback.py`, and `prompt.py`. The same audit scrubbed numeric values of invisible physics variables from prompts, checked the propagation of constraint and visible-variable mutations, and inspected the Uniform Suffix for generic variable awareness only. Together, these sandbox outcomes support the intended solvability and non-triviality conditions without relying on LLM preference.

Feedback optimization pipeline. We construct feedback modules through a two-phase automated pipeline (`auto_feedback.sh`), driven by Claude Code.

- Phase 1: Forensic failure analysis. For each task, Qwen3-4B is evaluated on all four target environments. Since Qwen3-4B consistently fails, its execution logs (saved as JSON) provide rich failure samples. An LLM analyzes these logs against six diagnostic dimensions: (1) temporal event chronology, (2) spatial margins to limits, (3) load and stress distribution ranked by severity, (4) energy flow and loss mechanisms (dynamics tasks), (5) constraint satisfaction profile with PASS/FAIL margins for *all* constraints, and (6) numerical health (NaN, Inf, extreme velocities). For each dimension, the analysis identifies missing diagnostic signals, specifically metrics that could have surfaced the root cause. The output is a prioritized list of top 3–5 missing diagnostics, each annotated with the reasoning step it would enable and how it can be computed from existing sandbox state.
- Phase 2: Feedback implementation and debloating. Guided by the forensic analysis, Claude Code rewrites `feedback.py` to produce diagnostic reports covering all applicable dimensions for the task type (*e.g.*, temporal chronology and spatial margins for kinematics, and energy chain and loss breakdown for dynamics). The implementation rules are strict: every reported value must be tracked in the metrics dictionary, every threshold must come from `metrics.get()` rather than a hardcoded literal, and the output must never prescribe engineering fixes. If key simulation quantities are missing from the metrics dict, new keys, tracking variables, or getter methods may be added to `evaluator.py` and `environment.py`, but existing defaults, pass/fail logic, and function signatures must remain intact. A subsequent debloating pass trims redundancy: cross-section duplication is eliminated, low-information lists are collapsed to one-line summaries, speculative math is removed, and unchanged sections in multi-moment reports are delta-compressed. The target is a concise, high-signal report where every line carries actionable diagnostic value. The full prompts used for both phases appear in Appendix F.

B Error Taxonomy Judging Criteria

Each failed run is assigned a single error type via a priority-ordered classifier (first matching condition wins). The priority order is: hard fail-

Category	#Prompt Tokens	#Constraints	#Prim. APIs	Stage 1	Stage 2	Stage 3	Stage 4
Statics & Equilibrium	935	8	3	4	4	11	11
Kinematics & Linkages	835	7	9	3	4	7	9
Dynamics & Energy	894	12	9	1	2	5	8
Granular & Fluid Interaction	1,136	9	10	2	2	13	12
Cybernetics & Control	1,730	5	9	2	4	12	12
Exotic Physics	778	1	6	1	4	6	8
Avg.	**1,051**	**7**	**8**	**2**	**3**	**9**	**10**
Total	**37,860**	**273**	**292**	**94**	**139**	**340**	**372**

Table 9: Dataset statistics by category (averaged over 6 tasks each). Stages 1–4 report average mutated physical parameters per stage, indicating progressive escalation in difficulty.

ures first (Catastrophic Collapse, Constraint Violation, Structural Failure, Numerical Instability), then trajectory-level patterns (Design Fixation, Late Convergence, Stagnation, Exploration), with Budget Exhaustion as fallback for runs that make progress but exhaust all attempts.

- **Catastrophic Collapse**: best score ≤ -60.
- **Constraint Violation**: any hard constraint flag set in the best attempt.
- **Structural Failure**: joint or beam breakage detected.
- **Numerical Instability**: NaN, Inf, or solver divergence with best score < 0.
- **Design Fixation**: avg code similarity > 0.85 over last 3 iterations, score trend < 3, best score < 50.
- **Late Convergence**: best in second half exceeds best in first half by > 2 points, and $0 <$ best score < 100.
- **Stagnation**: never achieves positive score, flat trajectory, $\leq$1 unique failure type.
- **Exploration**: never achieves positive score, $\geq$3 unique failure types.
- **Budget Exhaustion**: $0 <$ best score < 100, all 20 attempts used, no other type applies.

The main-text analysis in Table 5 reports the cross-model code-similarity evidence used to characterize these trajectory-level failure patterns.

C Method Adaptation Details

Each self-evolving method was originally designed for text, code, or discrete action domains. We adapt them to the physics simulation setting of PACEBENCH. All methods share the same base protocol: receive Box2D diagnostic feedback after each attempt, and produce a revised solution (Python program) within a 20-attempt budget. All auxiliary LLM calls (for reflection, memory induction, or rule extraction) use the same backbone solver model, preserving full autonomy.

Baseline. The Vanilla iterative refinement loop generates code via the LLM, executes it in the Box2D sandbox, appends structured feedback to the revision prompt, and repeats. No extra memory, agents, or weight updates.

Reflexion (Shinn et al., 2023). After each failed attempt, the backbone model produces a 3–8 sentence diagnosis with a high-level fix plan. Up to 3 reflections are accumulated (FIFO) and injected before the task description in subsequent revision prompts.

Self-Refine (Madaan et al., 2023). The model self-critiques and self-corrects in an inner loop (max 5 steps) without external verification. The outer verifier runs once after the inner loop. Contradiction detection prevents premature stopping when the model declares correctness but simultaneously outputs new code.

ACE (Zhang et al., 2025b). Maintains a structured *playbook* (strategies, code snippets, common mistakes). A Reflector analyzes each attempt, while a Curator merges and prunes the playbook. Both use the backbone model.

ExpeL (Zhao et al., 2024). Extracts distilled *rules* from rollout trajectories via LLM critique, retrieved by embedding similarity (Sup-SimCSE). Operates in pair-based mode: rules are learned from the source-environment rollout and injected into the prompt (never raw code, since environment parameters differ across mutations).

ReasoningBank (Ouyang et al., 2025). Induces structured memory items from each attempt via the backbone model. Supports parallel *MaTTS*: k candidates are generated and evaluated in the sandbox, then a contrast-and-distill call extracts what distinguished successes from failures. Success is judged by verifier score ≥ 99. Memory persists for cross-mutation transfer. Retrieval uses instruction-aware embedding similarity ($k = 5$).

Tree-of-Thoughts (Yao et al., 2023). Beam search

over code revisions: retain top-b states by verifier score, generate n candidates from each, evaluate all, keep top b. Default $b = 3$, $n = 2$. A budget formula matches the attempt budget: $(b \times n + 1) \times$ rounds $\leq$ max_iterations.

CodeEvolve (Assumpção et al., 2025). This population-based method uses the LLM as crossover and mutation operator to create variant physics programs, while the Box2D verifier score is fitness for selection. The population improves over generations, with population size 8 and at most 20 total LLM calls. The official CodeEvolve CLI runs as a subprocess with one island per GPU worker.

SEAL (Zweiger et al., 2026). This supervised test-time method resets LoRA at every iteration, retrains from scratch on all accumulated (prompt, code) pairs with positive Box2D scores, and then generates the next solution. It uses 20 iterations. Training reuses prior verifications and does not add evaluation attempts.

RAGEN (Wang et al., 2025b). This online RL method treats code revision as a multi-turn episode (generate $\rightarrow$ execute $\rightarrow$ revise), computes GRPO advantages per turn, and updates LoRA via PPO-clip, thereby learning the refinement trajectory rather than only final outputs. Five iterations $\times$ two episodes $\times$ two turns match the 20-attempt budget. LoRA rank is 64, with learning rate 10^{-5} and two PPO epochs.

TTT-Discover (Yuksekgonul et al., 2026). This test-time RL method generates independent candidates, computes leave-one-out entropic advantages, and updates LoRA with an importance-sampling loss. Identical candidate scores trigger feedback-driven expansion to break the deadlock. Five iterations $\times$ four rollouts match the 20-attempt budget. After iteration 1, revision prompts condition on prior feedback. The group size is 4, with 50 training epochs, learning rate 4×10^{-5}, and LoRA rank 32.

D Supplementary Experiment Results

D.1 Implementation

Hardware. Local-model inference runs on 25 NVIDIA A100-80GB GPUs. Qwen3-4B, -8B, -14B, and -32B are served through vLLM with tensor parallelism across 2–4 GPUs per instance. GPT-5.5, Gemini-3.1-Pro, Claude-Opus-4.7, DeepSeek-V4-Pro, Kimi-K2.6, and MiniMax-M2.7 are accessed through their respective cloud APIs.

Hyperparameters. All methods use temperature

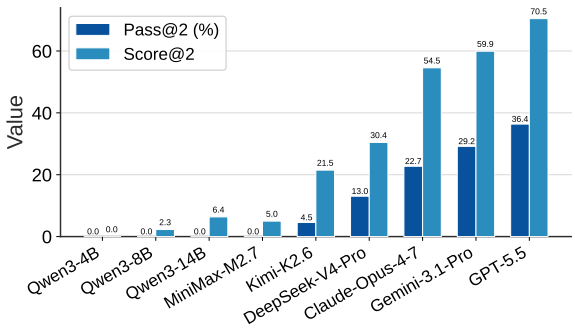

Figure 5: Pass@2 and Score@2 by model on Statics, Vanilla method, five-attempt budget.

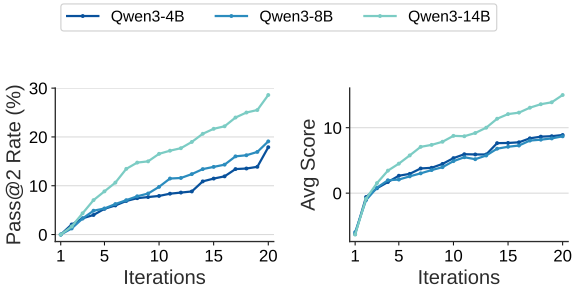

Figure 6: Pass@2@k and Score@2@k by model (all methods, all categories).

$T = 0.7$, top-$p = 0.95$, and a maximum of 65,536 output tokens. The Box2D sandbox runs at 60 FPS (PPM = 40 pixels/meter) with up to 10,000 simulation steps per attempt. Each environment pair ($\tau_0 \rightarrow \tau_k$) is evaluated with two independent runs. Pass@2 records whether either run passes, while Score@2 averages the best score obtained within each run.

History Management. Under the 20-attempt budget, naive retention of the full interaction history $(x^i, s^i, d^i)_{i=0}^{t-1}$ rapidly exceeds LLM context limits (especially for verbose diagnostic feedback and long code outputs). We adopt a *Previous-One + Best* truncation strategy: at each revision step, the agent receives the most recent attempt's full details plus the highest-scoring attempt across the entire history, while intermediate attempts are excluded. This retains both local momentum (the latest result) and global signal (the best result) without context overflow. Preliminary experiments suggest that this scheme preserves over 95% of the final pass rate compared to the full-history oracle.

Infrastructure. The evaluation pipeline supports task selection, local/vLLM/API backends, and parallel assignment across methods and tasks. The verifier then enforces API exclusivity, executes candidates in an isolated Box2D sandbox, and logs

Full-benchmark statistic	Value
Pass/fail agreement between runs	86.6%
Median absolute score difference	0.0
Environment pairs with score difference ≤ 5	82.0%
Spearman ρ, run 1 vs. run 2 (4B / 8B / 14B)	0.667 / 0.881 / 0.976
Kinematics: two-run vs. three-run statistic	**Value**
Pair-level pass agreement	94.9%
Pearson r, pair-level mean scores	0.940
Mean absolute error, pair-level mean scores	5.44
Spearman ρ, method ranks (4B / 8B / 14B)	1.000 / 0.700 / 0.900

Table 10: Run-reliability analysis. The full-benchmark panel compares the two independent runs; the Kinematics panel compares statistics computed from two runs with those computed after adding a third run for five representative methods.

Method	Qwen3-4B		Qwen3-8B		Qwen3-14B	
	Pass@2 w/o VLM	Pass@2 w/ VLM	Pass@2 w/o VLM	Pass@2 w/ VLM	Pass@2 w/o VLM	Pass@2 w/ VLM
Vanilla	0.0	18.2 (+18.2)	24.4	17.1 (-7.3)	40.9	18.2 (-22.7)
Reflexion	9.1	22.7 (+13.6)	42.9	23.8 (-19.1)	40.9	27.3 (-13.6)
ACE	4.5	13.6 (+9.1)	18.2	31.8 (+13.6)	13.6	22.7 (+9.1)
ExpeL	13.6	9.1 (-4.5)	14.3	42.9 (+28.6)	13.6	36.4 (+22.8)
ToT	13.6	9.1 (-4.5)	9.1	18.2 (+9.1)	4.5	0.0 (-4.5)

Table 11: VLM Video Feedback Impact on Statics and Kinematics.

structured JSON diagnostics.

D.2 Cost

The full local-model sweep evaluates Qwen3-4B, -8B, and -14B on 36 tasks $\times$ 4 environment pairs $\times$ 11 methods $\times$ 3 model sizes $\times$ 2 runs, totaling approximately 9,500 trajectories and up to 190K LLM calls over 4,200 A100 GPU-hours. One additional run at the same scope would require approximately 2,100 A100 GPU-hours. The constrained API study evaluates six proprietary models with Vanilla on 6 tasks $\times$ 4 environment pairs $\times$ 2 runs $\times$ 5 attempts, totaling 1,440 calls and approximately \$2,000. An extra run would add approximately \$1,000. At the observed average API rate, extending these models to all 36 tasks, 4 environment pairs, 11 methods, and 20 attempts would cost approximately \$264,000 per run.

D.3 Result Validity and Denominators

To explain why Pass@2 values need not be exact multiples of $1/144$, we audit valid denominators and exclusion causes. Table 2 nominally covers 144 environment pairs per cell. Across its method and model entries, the valid denominators average 139.20 (standard deviation 4.67) because runs with no complete, executable solution are excluded. Four model-intrinsic failure types are tracked: **output truncation**, when the response reaches the token limit before a complete solution; **parse failure**, when no solution can be extracted from the required code-block format; **sandbox runtime crash**, when candidate execution terminates without a valid score; and **context overflow**, when accumulated prompt content exceeds the model context window. For example, Qwen3-4B + Reflexion has denominator 140: two output truncations, one parse failure, zero sandbox runtime crashes, and one context overflow. These exclusions are recorded before aggregation and do not change the reported method ordering.

D.4 Run Reliability and Cost Tradeoff

Table 10 evaluates whether two stochastic runs support reproducible conclusions. On the full benchmark, 86.6% pass/fail agreement, a zero median score difference, and stronger rank agreement at larger scales suggest that most outcomes are not run-specific. Adding a third Kinematics run changes only 5.1% of pair-level pass decisions and leaves mean scores closely aligned, indicating limited marginal value from another repeat. The lower 8B rank correlation reflects a near tie among ACE, Reflexion, and ToT, which are separated by at most 1.1 score points, rather than broad instability. Since a full additional run costs approximately 2,100 A100 GPU-hours, two runs provide a practical balance between reliability and cost.

Model	Method	Catastrophic Collapse	Structural Failure	Constraint Violation	Numerical Instability	Design Fixation	Stagnation	Exploration	Late Convergence	Budget Exhaustion
Qwen3-4B	Vanilla	0.0	+3.0	0.0	+3.0	+5.1	-8.1	+6.1	-4.5	-4.5
	Reflexion	0.0	+1.8	0.0	-9.1	-4.1	-4.1	+24.7	+5.9	-15.0
	ACE	0.0	+1.5	0.0	+11.0	-12.3	+15.8	-11.3	0.0	-4.8
	ExpeL	0.0	+4.2	0.0	+9.7	-26.6	+4.2	+3.4	+5.0	0.0
	ToT	0.0	+4.5	0.0	0.0	0.0	+18.9	-23.2	0.0	0.0
Qwen3-8B	Vanilla	0.0	+2.4	0.0	-0.6	-8.2	+2.7	+3.0	+2.9	-2.3
	Reflexion	0.0	-10.4	0.0	+2.1	0.0	-8.3	+25.0	0.0	-8.3
	ACE	0.0	+8.9	0.0	0.0	-31.1	-5.6	+33.3	0.0	-5.6
	ExpeL	0.0	+5.6	0.0	+2.8	-2.8	-8.3	+5.6	+2.8	-5.6
	ToT	0.0	+6.7	0.0	0.0	+5.6	-11.7	-6.7	0.0	+6.1
Qwen3-14B	Vanilla	0.0	+1.3	0.0	+1.3	-2.1	-15.4	+22.6	-7.7	0.0
	Reflexion	0.0	-10.6	0.0	0.0	+11.1	-7.7	+10.1	-1.4	-1.4
	ACE	0.0	+6.5	0.0	-10.5	+13.6	-3.4	-6.8	-5.3	+5.9
	ExpeL	0.0	-1.5	0.0	+3.8	-5.3	+1.9	+4.5	-5.3	+1.9
	ToT	0.0	-5.4	0.0	0.0	0.0	-6.3	+7.4	0.0	+4.3

Table 12: VLM Effect on Error Taxonomy: Δ error share when VLM video feedback is added. Positive = VLM increases this error type; negative = VLM reduces it.

Model	Method	Catastrophic Collapse	Structural Failure	Constraint Violation	Numerical Instability	Design Fixation	Stagnation	Exploration	Late Convergence	Budget Exhaustion
Qwen3-4B	Vanilla	0.0	+3.0	0.0	-2.5	-11.6	-2.5	+22.7	-4.5	-4.5
	Reflexion	0.0	+10.0	0.0	+5.0	+3.3	+3.3	-13.3	0.0	-8.3
	ACE	0.0	+2.4	0.0	+6.3	-11.1	+5.6	-4.0	+5.6	-4.8
	ExpeL	0.0	-0.8	0.0	+4.7	-6.6	-0.8	+3.4	0.0	0.0
	ToT	0.0	+5.3	0.0	+5.3	0.0	+5.3	-15.8	0.0	0.0
Qwen3-8B	Vanilla	0.0	-1.0	0.0	-14.3	-1.4	+6.7	+17.1	0.0	-7.1
	Reflexion	0.0	-4.9	0.0	-15.4	+15.8	-7.7	+14.6	+5.3	-7.7
	ACE	0.0	0.0	0.0	0.0	-16.7	-5.6	+16.7	0.0	+5.6
	ExpeL	0.0	+3.0	0.0	+7.2	-10.5	-3.3	+1.6	+7.2	-5.3
	ToT	0.0	+1.1	0.0	0.0	0.0	-0.6	-1.1	0.0	+0.6
Qwen3-14B	Vanilla	0.0	+2.3	0.0	+2.3	+4.1	-9.5	+8.6	-7.7	0.0
	Reflexion	0.0	0.0	0.0	+23.1	0.0	0.0	-7.7	-7.7	-7.7
	ACE	0.0	+6.5	0.0	-4.6	-4.0	-15.2	+22.6	-5.3	0.0
	ExpeL	0.0	0.0	0.0	0.0	+5.3	+5.3	-15.8	0.0	+5.3
	ToT	0.0	-4.0	0.0	0.0	0.0	-8.1	+6.9	0.0	+5.2

Table 13: CE Effect on Error Taxonomy: Δ error share when mutated variables are disclosed (CE minus CH). Positive = CE increases this error type; negative = CE reduces it. Statics only.

Method	1	2	3	4	5	6	7	8	9	10
Vanilla	0.255	0.468	0.536	0.843	0.518	0.493	0.778	0.793	0.553	0.943
ACE	0.185	0.870	0.842	0.762	0.772	0.895	0.814	**1.000**	0.670	0.828
Method	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Vanilla	0.524	**0.963**	0.887	0.767	0.564	0.895	0.770	0.500	0.886	0.723
ACE	0.497	0.734	0.375	0.635	0.965	0.965	0.908	0.877	0.847	**1.000**

Table 14: Adjacent-attempt Jaccard code similarity for one Qwen3-14B Stage 4 case study. Each entry compares solutions on consecutive transitions.

Model	Pass@2 (%)	Score@2
Qwen3-4B	11.3	17.2
Qwen3-8B	12.8	16.8
Qwen3-14B	18.3	21.7

Table 15: Vanilla from-scratch results by model, computed over eligible task–environment instances.

Target environment	Pass@2 (%)	Score@2
Source	32.3	30.8
Mutated-1	13.4	18.5
Mutated-2	10.9	16.6
Mutated-3	11.2	15.3
Mutated-4	6.4	11.6

Table 16: Vanilla from-scratch results by target environment, computed over eligible task–environment instances.

D.5 Larger-Model Results

To broaden frontier-model coverage at manageable cost, we compare six frontier LLMs with Qwen3-4B/8B/14B on the 24-pair Statics subset using Vanilla and a five-attempt budget. Because this budget differs from the main protocol, we use the study only for within-setting model comparison.

Figure 5 reveals a sharp capability gap under the five-attempt budget. Qwen3-4B/8B/14B and MiniMax-M2.7 achieve 0% Pass@2, although Qwen3's Score@2 rises from 0.0 to 6.4 with scale. Larger Qwen3 models therefore make more partial progress but still fail to complete a design within the short interaction horizon. Among models with nonzero success, Score@2 and Pass@2 yield the same ranking: GPT-5.5 > Gemini-3.1-Pro > Claude-Opus-4.7 > DeepSeek-V4-Pro > Kimi-K2.6. GPT-5.5 leads with 36.4% Pass@2 and 70.5 Score@2, yet still fails 63.6% of pairs. Stronger base models thus improve both progress and success probability, but do not make short-budget adaptation reliable.

E Supplementary Analysis

E.1 Convergence Analysis

To test how interaction budget affects model scaling, we evaluate Pass@2@k and Score@2@k after truncating each trajectory to k attempts. Figure 6 reports these metrics aggregated across all methods. No model succeeds with only one attempt ($k = 1$). Performance then rises monotonically with budget, and the gap between model scales widens with more attempts, suggesting that larger models may extract more value from additional interaction.

E.2 Per-Attempt Similarity

To probe score-retrieval anchoring over time, we compare adjacent-attempt Jaccard similarity for Vanilla and ACE in one Qwen3-14B Stage 4 case study. Table 14 shows that mean similarity rises from the first five to the last five transitions for both Vanilla (0.52→0.75) and ACE (0.69→0.92), with ACE remaining higher overall (global mean 0.77 vs. 0.68). ACE submits identical consecutive code at transitions 8 and 20. Vanilla instead alternates between near-fixation (0.963 at transition 12) and renewed exploration (0.500 at transition 18). This trace suggests progressive convergence in both methods and stronger score-retrieval anchoring in ACE.

E.3 Category-Level Analysis

Figure 7 reports Pass@2 aggregated across methods by model and category. A stable difficulty hierarchy emerges: **Exotic Physics** is consistently the most solvable category, while **Dynamics** remains the hardest across all model scales. Model scaling gains vary sharply by category: the 4B→14B improvement is largest in **Exotic Physics** (+16.3) and **Fluid** (+12.2), and smallest in **Kinematics** (+1.9).

Error patterns are strongly category-specific. **Category 1 (Statics, S)** tests equilibrium reason-

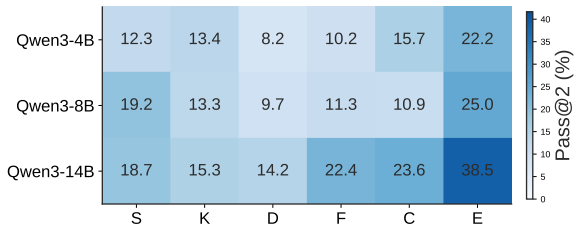

Figure 7: Model × category Pass@2 heatmap (all methods aggregated).

ing. Agents explore diversely (radicality 0.311) but show 34.1% Exploration without convergence. Because "almost balanced" structures still collapse, iterative refinement lacks a gradient. **Category 2 (Kinematics, K)** tests linkage topology design. The smallest scaling gain (+1.9) and 44.5% Budget Exhaustion suggest that combinatorial search, rather than reasoning depth, may be the main bottleneck. **Category 3 (Dynamics, D)** tests coupled energy, momentum, and timing. All three trajectory failures co-occur (30.6% Fixation, 30.1% Stagnation, 28.2% Exploration) because improving one constraint often breaks another. **Category 4 (Granular/Fluid, F)** tests many-body interaction reasoning. The unique 20.3% Constraint Violation rate may reflect failure to anticipate emergent particle behavior from code alone. **Category 5 (Control, C)** tests closed-loop dynamics. Extreme Fixation (58.2%, radicality 0.153) may reflect the difficulty of isolating which control parameter failed from aggregate feedback. **Category 6 (Exotic Physics, E)** tests survival under unfamiliar regimes. Agents either fail early (19.3% Structural Failure) or converge quickly (Budget Exhaustion 1.1%). Because survival is coarser than precision, the budget is rarely the bottleneck.

F Prompts

This section documents the key prompts used during benchmark construction, corresponding to the auditing, feedback design, and difficulty escalation pipelines described in §3.3 and Appendix §A.

F.1 Module Auditing Prompt

The following prompt is issued to an LLM for automated auditing of each task directory (see §A.1):

Module Auditing Prompt.

```
## Objective
Conduct a strict, exhaustive audit of the
current task directory.
All file paths must be relative to ..

## Anti-Laziness Rule
Do NOT stop after finding 1 or 2 errors.
Provide an EXHAUSTIVE,
line-by-line enumeration of EVERY SINGLE
violation.

## Variable Classification

1. CONSTRAINT: MUST have numeric value in
prompt.py
Variables defining absolute maxima,
minima, or failure thresholds
required to solve the task. Examples: max
structure mass,
max joint torque, gate positions, target
coordinates.
Invisible constraints STILL need numeric
values in prompt.py.

2. INVISIBLE NON-CONSTRAINT: NEVER numeric
in prompt.py
Background physics that cannot be visually
observed. If ANY
numeric value appears in prompt.py, DELETE
it. Affected:
gravity, linear_damping, angular_damping,
wind_amplitude,
drain_velocity_factor,
slip_backward_force.
Qualitative descriptions allowed, but
numeric values prohibited.

3. VISIBLE VARIABLE: MUST have numeric
value in prompt.py
Observable physical properties: gate
positions, target zone,
initial craft position.

## Audit Steps
Step 1: Cross-Module Consistency: check
physics across all
modules (env, eval, feedback, prompt,
stages, renderer).
Step 2: Invisible Non-Constraint Sweep:
delete any numeric
value of invisible variables from
prompt.py.
Step 3: Constraint Completeness: every
hardcoded constraint
in environment.py must appear in
prompt.py.
Step 4: Mutation Sync: check "(originally
OLD_VALUE)"
format for constraint/visible changes in
stages.py.
Step 5: UNIFORM_SUFFIX Tone: suffix must
list only generic
warnings, never specific values or
directions.
Step 6: Runtime Pipeline Check: run full
cross-mutation
pipeline with mock model across all 4
stages.
```

F.2 Feedback Optimization Prompts

The feedback pipeline (see §A.3) uses a two-phase LLM workflow.

Phase 1: Forensic Analysis.

Phase 1: Forensic Analysis.
You are a physics simulation forensic analyst. Analyze the execution logs to determine why an LLM agent failed.

Six Diagnostic Dimensions:
- Temporal Chronology: ordered failure timeline with step numbers and positions.
- Spatial Margins: every measurement paired with limit and margin.
- Load Distribution: components ranked by stress percentage.
- Energy Flow: stored → delivered → losses (dynamics tasks).
- Constraint Profile: ALL constraints with PASS/FAIL + margins.
- Numerical Health: flag NaN, Inf, extreme velocities.

List the TOP 3-5 missing diagnostics that would have unlocked success, and specify how each can be computed from the sandbox.

Phase 2: Feedback Implementation.

Phase 2: Feedback Implementation.
Implement this task's feedback.py to produce forensic-quality diagnostic reports covering all six dimensions.

Rules: Every limit from metrics.get(). Never hardcode. thresholds. Report margins (distance to limit), not raw values. Order events chronologically. Sort stress data by severity. NEVER prescribe engineering fixes: identify WHAT failed and BY WHAT MARGIN, never "you should...". NEVER modify existing defaults or pass/fail logic outside feedback.py.

Task-Type Guidance:
S_ (Statics): joint failure cascade, spatial margins, load.
K_ (Kinematics): distance vs target, joint angle extents.
D_ (Dynamics): energy chain, loss breakdown, trajectory margins.
F_ (Granular/Fluid): leakage vs limit, containment timeline.
C_ (Control): control error per zone, stability margins.
E_ (Exotic): ALL six dimensions, unusual physics.

F.3 Difficulty Escalation Prompt

The following prompt is issued for each environment pair requiring escalation (see §A.2):

Difficulty Escalation Prompt.
Escalate the difficulty of this target environment stage.
Goal: make it as hard as possible while remaining solvable by the stage-specific reference solution.

Critical Constraints:
- Only mutate existing variables, with no new physics.
- Only modify stages.py and agent.py.
- Source reference MUST pass the source environment but FAIL on the target environment.
- UNIFORM_SUFFIX identical across ALL 4 stages, generically.

Verification:
1. Read all task modules.
2. Escalate target stage in stages.py and update agent.py.
3. Update UNIFORM_SUFFIX across all stages if needed.
4. Run test_reference_solutions.py: source passes $\mathcal{E}_0$ and fails ALL 4 target environments.
5. Run mock evaluation pipeline to catch runtime errors.

G Full Task Specifications

Tables 17–22 provide detailed specifications for all 36 tasks, including the source task description and the physical mutations applied in each of the four target environments.

ID	Stage	Mutations
S_01: Bridge for vehicle crossing		
	Stage-1: Brittle Material	• max_structure_mass = 492.0 • joint_max_force = 40.0 • anchor_max_force = 50.0 • joint_max_torque = 0.1 • anchor_max_torque = 0.1
	Stage-2: Paper Joints	• max_structure_mass = 1000.0 • joint_max_force = 5.5 • anchor_max_force = 15.0
	Stage-3: The Vortex Gorge	• gap_width = 20.0 • max_structure_mass = 550.0 • gravity = (0, -20.0) • wind_force = (-20.0, -4.0) • joint_max_torque = 0.2 • anchor_max_torque = 0.2
	Stage-4: Abyssal Crossing	• gap_width = 26.0 • max_structure_mass = 330.0 • gravity = (0, -28.0) • wind_force = (-45.0, -8.0) • joint_max_force = 40.0 • anchor_max_force = 60.0 • joint_max_torque = 80.0 • anchor_max_torque = 120.0
S_02: Tower resisting seismic vibration		
	Stage-1: The Brittle Foundation	• max_joint_torque = 5000.0 • max_joint_force = 7000.0
	Stage-2: The Glass Skeleton	• max_joint_torque = 5000.0
	Stage-3: The Resonant Collapse	• earthquake_amplitude = 0.6 • earthquake_frequency = 1.2 • gravity = (0, -11.5) • max_joint_force = 30000.0 • max_joint_torque = 35000.0
	Stage-4: The Coupled Storm	• earthquake_amplitude = 0.85 • earthquake_frequency = 3.4 • earthquake_amplitude_evolution = 0.012 • wind_force = 250.0 • wind_height_threshold = 5.0 • wind_shear_factor = 0.08 • wind_oscillation_frequency = 4.5 • gravity = (0, -20.0) • max_joint_force = 5000.0 • max_joint_torque = 10000.0
S_03: Cantilever bearing loads		
	Stage-1: The Torque Threshold	• max_anchor_torque = 1500.0
	Stage-2: The Monolithic Mandate	• max_anchor_force = 18000.0 • max_internal_force = 100.0

ID	Stage	Mutations
	Stage-3: The Subterranean Gorge	• target_reach = 35.0 • load_mass = 2500.0 • max_structure_mass = 7000.0 • forbidden_anchor_y = [-6.0, 30.0] • anchor_strength_map = [[-20.0, -6.0, 0.04, 0.04]] • obstacle_active = True • obstacle_rects = [[0.0, 3.0, 30.0, 30.0]] • load_type = dropped • drop_height = 5.0 • max_internal_force = 5000000.0 • max_internal_torque = 5000000.0 • spatial_force.center = (18.0, -8.0) • spatial_force.magnitude = 600000.0 • spatial_force.radius = 32.0 • spatial_force.type = attraction • wind.force = (2000.0, -1500.0) • wind.oscillatory = True • wind.frequency = 0.35 • gravity = (0, -16)
	Stage-4: The Event Horizon	• target_reach = 40.0 • load_mass = 3500.0 • max_structure_mass = 4000.0 • forbidden_anchor_y = [-3.0, 30.0] • anchor_strength_map = [[-20.0, -3.0, 0.015, 0.015]] • load_type = dropped • drop_height = 20.0 • max_internal_force = 1800000.0 • max_internal_torque = 1800000.0 • min_tip_height_limit = -20.0 • spatial_force.center = (34.0, 18.0) • spatial_force.magnitude = 250000.0 • spatial_force.radius = 18.0 • spatial_force.type = repulsion • wind.force = (5000.0, -3500.0) • wind.oscillatory = True • wind.frequency = 0.25 • gravity = (0, -25)
S_04: Balanced structure on pivot		
	Stage-1: The Orthogonal Load Path	• force_pivot_joint = True • fragile_joints = False • load_mass = 200.0 • max_angle_deviation_deg = 10.0 • gravity = (10.0, 0.0) • angular_damping = 2.0

ID	Stage	Mutations
	Stage-2: The Resultant-Line Pendulum	• force_pivot_joint = True • pivot_friction = 0.0 • load_mass = 200.0 • max_angle_deviation_deg = 1.0 • balance_time = 60.0 • gravity = (-24.0, -24.0) • angular_damping = 0.0
	Stage-3: The Hurricane Siege	• force_pivot_joint = True • obstacle_active = True • obstacles = [[-5.0, -2.0, -2.0, 0.0], [1.5, 0.0, 3.5, 1.5]] • wind_active = True • wind_force_multiplier = 12.0 • load_mass = 200.0 • fragile_joints = True • max_joint_torque = 8000.0 • max_angle_deviation_deg = 15.0 • balance_time = 90.0 • gravity = (0, -40.0) • angular_damping = 0.5
	Stage-4: The Dirac Fulcrum	• force_pivot_joint = True • fragile_joints = False • wind_active = True • wind_force_multiplier = 30.0 • load_mass = 200.0 • max_angle_deviation_deg = 0.5 • balance_time = 180.0 • obstacle_active = True • obstacles = [[-4.0, 0.2, -2.0, 1.0], [1.0, 0.6, 3.5, 2.5], [-6.0, -7.0, -4.0, -5.0]] • gravity = (0, -60.0) • angular_damping = 2.0
S_05: Shelter from falling boulders		
	Stage-1: The Fragile Anchor	• max_joint_force = 5000.0 • max_joint_torque = 1000000000000.0
	Stage-2: The Gale-Force Crucible	• core_x = 5.0 • max_core_force = 30.0 • wind_force = -250.0 • max_joint_force = 3500.0 • max_joint_torque = 3500.0 • max_structure_mass = 100.0 • meteor_density = 9.0 • meteor_restitution = 0.8

ID	Stage	Mutations
	Stage-3: The Oblique Crossfire	• max_structure_mass = 1.2 • max_joint_force = 8000.0 • max_core_force = 12.0 • meteor_vx_range = [-14.0, 14.0] • structure_restitution = 0.55 • has_walls = True • seed = 123 • gravity = (0, -16.0)
	Stage-4: The Eccentric Gauntlet	• core_x = 13.5 • max_core_force = 4.0 • max_structure_mass = 0.8 • max_joint_force = 10000.0 • max_joint_torque = 27000.0 • wind_force = 90.0 • has_walls = True • meteor_restitution = 0.9 • meteor_density = 8.0 • meteor_vx_range = [-18.0, 18.0] • structure_restitution = 0.65 • gravity = (0, -22.0)
S_06: Block stacking for overhang		
	Stage-1: The Slippery Gale	• target_overhang = 0.5 • floor_length = 20.0 • spawn_zone = [-10.0, 0.0] • max_total_mass = 35.0 • table_friction = 0.015 • block_friction = 0.25 • oscillate = False • osc_amplitude = 0.0 • osc_frequency = 0.0 • gravity = (0, -10.0) • wind_force = 2.5
	Stage-2: The Wind-Loaded Mass Cascade	• target_overhang = 0.61 • floor_length = 20.0 • spawn_zone = [-10.0, 0.12] • max_total_mass = 30.0 • table_friction = 0.02 • block_friction = 0.35 • oscillate = False • osc_amplitude = 0.0 • osc_frequency = 0.0 • gravity = (0, -10.0) • wind_force = 1.2

ID	Stage	Mutations
	Stage-3: The Harmonic Lift Cantilever	• target_overhang = 1.38 • floor_length = 20.0 • spawn_zone = [-10.0, 0.89] • max_total_mass = 18.0 • table_friction = 0.22 • block_friction = 0.8 • oscillate = False • osc_amplitude = 0.0 • osc_frequency = 0.0 • gravity = (0, -10.0) • wind_force = (-3.0, 9.5)
	Stage-4: The Gravitational Siege	• target_overhang = 0.55 • floor_length = 20.0 • spawn_zone = [-10.0, 0.05] • max_total_mass = 32.0 • table_friction = 0.1 • block_friction = 0.1 • oscillate = True • osc_amplitude = 0.12 • osc_frequency = 3.0 • gravity = (0, -12.0) • wind_force = 1.8

Table 17: Category: Statics & Equilibrium task specifications.

ID	Stage	Mutations
K_01: Motor-driven 2D walker		
	Stage-1: Near-Zero Structure Mass Budget	• max_structure_mass = 0.8
	Stage-2: Near-Zero Ground Traction + Restricted Joints	• ground_friction = 0.02 • default_joint_lower_limit = -0.2617993877991494 • default_joint_upper_limit = 0.2617993877991494
	Stage-3: Dissipative Low-Traction Walking Crown	• ground_friction = 0.015 • max_structure_mass = 1.5 • gravity = (0, -17.0) • max_body_friction = 0.2 • default_joint_lower_limit = -0.2243994752564138 • default_joint_upper_limit = 0.2243994752564138 • linear_damping = 4.0 • angular_damping = 4.0

ID	Stage	Mutations
	Stage-4: Maximum Difficulty — All Variables Beyond Stage-3 Extremes	• ground_friction = 0.01 • max_structure_mass = 1.2 • gravity = (0, -20.0) • max_body_friction = 0.008 • default_joint_lower_limit = -0.20943951023931953 • default_joint_upper_limit = 0.20943951023931953 • linear_damping = 18.0 • angular_damping = 18.0
K_02: Vertical wall climber		
	Stage-1: Fragile Structural Integrity	• build_zone_y_max = 5.0 • max_joint_force = 1.5 • max_joint_torque = 3.0
	Stage-2: Gravitational Flux & Void Zones	• build_zone_y_max = 8.0 • suction_zones = [(0, 16), (19, 35)] • gravity = (0, -12.0) • gravity_evolution = -0.1
	Stage-3: Discontinuous Adhesion Corridor	• build_zone_y_max = 5.0 • min_structure_mass = 41.0 • max_structure_mass = 43.0 • wind_force = -40.0 • suction_zones = [(0, 8), (11, 16), (19, 35)] • max_joint_force = 1650.0 • max_joint_torque = 250.0 • gravity = (0, -26.0) • gravity_evolution = -0.35 • angular_damping = 8.0
	Stage-4: The Resonant Singularity	• build_zone_y_max = 5.0 • min_structure_mass = 25.0 • wind_force = -15.0 • vortex_y = 5.0 • vortex_force_x = 15.0 • vortex_force_y = -5.0 • suction_zones = [(0, 7), (9, 16), (18, 25), (27, 35)] • max_joint_force = 3000.0
K_03: Gantry gripper		
	Stage-1: Slippery Object	• objects.shape = circle • objects.mass = 1.0 • objects.friction = 0.25 • objects.x = 5.0 • objects.y = 2.0 • linear_damping = 0.5 • angular_damping = 0.5

ID	Stage	Mutations
	Stage-2: Crushing Gravity	• objects.shape = box • objects.mass = 10.0 • objects.friction = 0.6 • objects.x = 5.0 • objects.y = 2.0 • gravity = (0, -30.0) • linear_damping = 0.5 • angular_damping = 0.5
	Stage-3: Slippery Object + Heavy World + Damping	• objects.shape = box • objects.mass = 5.0 • objects.friction = 0.2 • objects.x = 5.0 • objects.y = 2.0 • gravity = (0, -20.0) • linear_damping = 0.5 • angular_damping = 0.5
	Stage-4: Circular Object + Slippery + Heavy + Damping	• objects.shape = circle • objects.mass = 5.0 • objects.friction = 0.2 • objects.x = 5.0 • objects.y = 2.0 • gravity = (0, -20.0) • linear_damping = 0.5 • angular_damping = 0.5
K_04: Ground cart pushing objects		
	Stage-1: Tipping Hazard and Mass Budget	• object.center_of_mass_offset = [0.2, 0.25] • max_structure_mass = 26.0
	Stage-2: Payload Mass Variation	• object.mass = 95.0
	Stage-3: Modified Friction, Damping, and Mass Budget	• ground_friction = 0.3 • object.friction = 0.08 • object.linear_damping = 4.0 • max_structure_mass = 26.0
	Stage-4: Heavy Object, Slippery Surfaces, Strong Damping, Tight Budget, and High Gravity	• ground_friction = 0.06 • object.mass = 110.0 • object.friction = 0.03 • object.linear_damping = 12.0 • object.center_of_mass_offset = [0.24, 0.28] • target_distance = 16.0 • max_structure_mass = 22.0 • gravity = (0, -18.0)
K_05: Scissor-lift mechanism		
	Stage-1: Severe Hurricane Wind	• wind_force = (400.0, 0.0)
	Stage-2: Crushing Gravity	• gravity = (0.0, -150.0)

ID	Stage	Mutations
	Stage-3: The Tipping Gauntlet	• target_object_y = 12.0 • max_structure_mass = 40.0 • object.mass = 90.0 • object.friction = 0.05 • object.com_offset = (0.25, 0.15) • wind_force = (200.0, 0.0)
	Stage-4: The Master's Gauntlet	• ceiling_gap.x_min = 3.2 • ceiling_gap.x_max = 4.8 • ceiling_gap.y = 6.0 • target_object_y = 10.0 • object.mass = 40.0 • object.friction = 0.2 • wind_force = (150.0, 0.0) • max_joint_force = 1500.0
K_06: Wiper clearing particles		
	Stage-1: Extreme Motor Torque Starvation	• max_motor_torque = 1.0 • particles.count = 45 • particles.seed = 42 • particles.friction = 0.35 • particles.mass = 0.15
	Stage-2: Severe Torque Starvation with Near-Maximum Particle Adhesion	• max_motor_torque = 10.0 • particles.count = 45 • particles.seed = 42 • particles.friction = 0.88 • particles.mass = 0.15
	Stage-3: Mass Starvation vs Heavy Sticky Particle Overload with Capped Torque	• max_structure_mass = 0.1 • max_motor_torque = 28.0 • particles.count = 78 • particles.seed = 37 • particles.friction = 0.76 • particles.mass = 0.4
	Stage-4: Extreme: Double Gravity, Glass Adhesion & Torque Starvation Paradox	• max_structure_mass = 0.06 • max_motor_torque = 10.0 • glass_friction = 0.55 • particles.count = 65 • particles.seed = 51 • particles.friction = 0.75 • particles.mass = 0.15 • gravity = (0, -22)

Table 18: Category: Kinematics & Linkages task specifications.

ID	Stage	Mutations
D_01: Projectile launcher		
	Stage-1: The Dense Atmosphere	• linear_damping = 2.5 • angular_damping = 2.5

ID	Stage	Mutations
	Stage-2: The Distant Target	• target_x_min = 50.0 • target_x_max = 55.0 • target_y_min = 2.0 • target_y_max = 5.0
	Stage-3: Heavy World and Drag	• gravity = (0, -15.0) • linear_damping = 1.5 • angular_damping = 1.5
	Stage-4: Extreme Range and Conditions	• target_x_min = 52.0 • target_x_max = 57.0 • target_y_min = 2.0 • target_y_max = 5.0 • gravity = (0, -18.0) • linear_damping = 1.2 • angular_damping = 1.2
D_02: Jumping mechanism		
	Stage-1: Viscous Void	• linear_damping = 2.0
	Stage-2: Alternating Passages	• slot1_floor = 4.8 • slot1_ceil = 5.7 • slot3_floor = 15.0 • slot3_ceil = 15.9 • slot2_floor = 4.4 • slot2_ceil = 5.3
	Stage-3: Gale-Force Gravity	• gravity = (0, -35.0) • wind = (-20.0, 0)
	Stage-4: The Perfect Storm	• gravity = (0, -30.0) • wind = (-15.0, 0) • linear_damping = 1.0
D_03: Momentum-modulated cart		
	Stage-1: Massive first impulse	• impulse_magnitude = 40.0
	Stage-2: Extreme ambient linear drag	• linear_damping = 5.0
	Stage-3: Extreme resistance cascade	• impulse_magnitude = 30.0 • impulse2_magnitude = 4.0 • decel_damping = 12.0 • mud_damping = 10.0 • linear_damping = 2.5
	Stage-4: Heavy world and strong impulses	• impulse_magnitude = 2.6 • impulse2_magnitude = 0.95 • decel_damping = 4.5 • gravity = (0, -12) • linear_damping = 0.4 • angular_damping = 0.4
D_04: Swing control strategy		
	Stage-1: The Velocity-Gated Dead Zone	• dead_zone = [9.5, 11.0] • dead_zone_min_speed = 14.0
	Stage-2: Quadratic Energy Drain	• quadratic_damping = 0.36

ID	Stage	Mutations
	Stage-3: One-Way Actuator & Gale	• actuator_fault = left_only • wind_strength = 30.0 • wind_period = 0.0
	Stage-4: The Ultimate Crucible	• actuator_fault = right_only • dead_zone = [9.8, 10.2] • quadratic_damping = 0.1 • wind_strength = -25.0 • wind_period = 0.0
D_05: Hammer breaking shells		
	Stage-1: Harder Shell	• shell_break_force = 16000.0
	Stage-2: Shifted Slot Bar Phase	• slot_bar_omega = 0.014
	Stage-3: Harder Shell and Damping	• shell_break_force = 13000.0 • angular_damping = 0.6
	Stage-4: Gravity, Shell, Bar Phase and Damping	• shell_break_force = 11000.0 • slot_bar_omega = 0.013 • gravity = (0, -14.0) • angular_damping = 0.35
D_06: Capture structure		
	Stage-1: Absolute joint annihilation — 1.0 N peak limit	• max_joint_force = 1.0
	Stage-2: Shifted legal corridor and low-drag projectiles	• build_zone_x_max = 13.0 • forbidden_zone_4_x_min = 10.5 • forbidden_zone_4_x_max = 11.0 • ball_linear_damping = 0.3
	Stage-3: Extreme density storm — fragile joints, violent gravity, ultra-compressed launch	• ball_density = 520.0 • max_joint_force = 55.0 • joint_fatigue_threshold = 30.0 • ball_velocity_x = -52.0 • ball2_velocity_x = -54.0 • ball3_velocity_x = -52.0 • ball4_velocity_x = -58.0 • ball5_velocity_x = -53.0 • ball6_velocity_x = -54.0 • ball7_velocity_x = -53.0 • second_ball_launch_time = 0.14 • third_ball_launch_time = 0.34 • fourth_ball_launch_time = 0.54 • fifth_ball_launch_time = 0.74 • sixth_ball_launch_time = 0.94 • seventh_ball_launch_time = 1.14 • gravity_pulse_amplitude = 10.0 • gravity_pulse_period = 0.4

ID	Stage	Mutations
	Stage-4: Stage-3 storm + bouncy balls, lighter self-damping, stronger structural gusts	• ball_density = 232.0 • max_joint_force = 300.0 • joint_fatigue_threshold = 225.0 • ball_velocity_x = -36.0 • ball2_velocity_x = -38.0 • ball3_velocity_x = -36.0 • ball4_velocity_x = -40.0 • ball5_velocity_x = -37.0 • ball6_velocity_x = -38.0 • ball7_velocity_x = -37.0 • second_ball_launch_time = 0.26 • third_ball_launch_time = 0.58 • fourth_ball_launch_time = 0.88 • fifth_ball_launch_time = 1.18 • sixth_ball_launch_time = 1.48 • seventh_ball_launch_time = 1.78 • gravity_pulse_amplitude = 3.5 • gravity_pulse_period = 1.0 • wind_on_structure = True • structure_wind_scale = 0.178 • wind_amplitude = 9.0 • ball_restitution = 0.24 • ball_linear_damping = 0.52

Table 19: Category: Dynamics & Energy task specifications.

ID	Stage	Mutations
F_01: Dam blocking water flow		
	Stage-1: Low weld ceiling (threshold physics)	• joint_break_force = 41000.0
	Stage-2: Elastic reservoir granules	• fluid_particle_restitution = 0.78
	Stage-3: Bounce apocalypse + wide squeeze + fast debris + heavy world	• joint_break_force = 50000.0 • joint_break_consecutive_steps = 2 • downstream_wall_amplitude = 1.0 • downstream_wall_phase_divisor = 30.0 • fluid_particle_restitution = 0.95 • debris_linear_velocity_x = 3.2 • max_leakage_rate = 0.004 • gravity = (0, -10.5)

ID	Stage	Mutations
	Stage-4: Apocalypse: every physics dial at breaking point	• joint_break_force = 50000.0 • joint_break_consecutive_steps = 2 • downstream_wall_amplitude = 1.0 • downstream_wall_phase_divisor = 20.0 • fluid_particle_restitution = 0.97 • fluid_particle_friction = 0.02 • debris_linear_velocity_x = 4.0 • earthquake_impulse_x = 0.5 • upward_surge_impulse_y = 1.5 • max_structure_mass = 130.0 • backward_slosh_impulse_x = -0.9 • max_leakage_rate = 0.005 • surge_impulses = [1.0, 1.2, 1.3, 1.5, 1.6, 1.8, 1.9, 2.1, 2.2] • gravity = (0, -11.5)
F_02: Amphibious vehicle		
	Stage-1: The Sluggish Dawn	• thrust_cooldown_steps = 200
	Stage-2: The Brittle Expanse	• max_joint_force = 1.0
	Stage-3: The Maelstrom	• whirlpool.x = 17.0 • whirlpool.width = 4.0 • whirlpool.force = 200.0 • max_joint_force = 200.0
	Stage-4: The Gauntlet of Contradictions	• emp_zone = [14.0, 15.2] • corrosive_y = 2.7 • whirlpool.x = 17.5 • whirlpool.width = 4.0 • whirlpool.force = 240.0 • current_per_kg = 28.0 • max_joint_force = 30.0 • thrust_cooldown_steps = 18
F_03: Excavator digging sand		
	Stage-1: Mass starvation — paper-thin mandate	• max_structure_mass = 30
	Stage-2: Micro-scoop restriction	• scoop_capacity = 2
	Stage-3: Heavy gravity, ice particles, drift storm, mass starvation, capacity famine, time crunch	• particles.friction = 0.04 • particles.count = 200 • particles.radius = 0.06 • particles.density = 1500.0 • particles.seed = 42 • min_particles_in_hopper = 30 • pit_drift_force = 1.0 • max_time_seconds = 20 • max_structure_mass = 120 • scoop_capacity = 8 • gravity = (0, -20.0) • linear_damping = 0.08 • angular_damping = 0.08

ID	Stage	Mutations
	Stage-4: Cataclysmic excavation nightmare	• particles.friction = 0.02 • particles.count = 200 • particles.radius = 0.06 • particles.density = 1500.0 • particles.seed = 42 • min_particles_in_hopper = 22 • pit_drift_force = 1.2 • max_time_seconds = 17 • max_structure_mass = 100 • scoop_capacity = 6 • gravity = (0, -22.0) • linear_damping = 0.1 • angular_damping = 0.1
F_04: Particle filter		
	Stage-1: Anti-Gravity Siege — Extreme Upward Field	• min_purity = 0.35 • gravity = (0.0, 32.0)
	Stage-2: Diagonal Gravity Hurricane — Extreme Rightward Drift with Violent Upward Pull	• min_purity = 0.35 • gravity = (55.0, 16.0)
	Stage-3: Hypergravity Cascade — Near-Cataclysmic Multi-Field Particle Avalanche	• min_purity = 0.35 • max_beams = 3 • max_structure_mass = 1.6 • baffles.y_bottom = 0.05 • sweeper.speed_scale = 45.0 • wind_amplitude = 6200.0 • wind_period_steps = 13 • gust_amplitude = 2300.0 • gust_period_steps = 7 • beam_friction = 0.0 • feed_y_min = 1.78 • second_wave_step = 140 • third_wave_step = 300 • mix.density = 17500.0 • mix.restitution = 0.998 • mix.friction = 0.0001 • mix.radius_small = 0.08 • mix.radius_medium = 0.099 • mix.radius_large = 0.138 • mix.count_small = 48 • mix.count_medium = 48 • mix.count_large = 48 • mix.count_third_small = 48 • mix.count_third_medium = 48 • mix.count_third_large = 48 • gravity = (93.0, 1.15) • gravity_oscillation_amplitude = 52.0 • gravity_oscillation_period = 11 • linear_damping = 0.992 • angular_damping = 0.992

ID	Stage	Mutations
	Stage-4: Singularity Siege — Cataclysmic Multi-Field Entropy Collapse	• min_purity = 0.35 • max_beams = 3 • max_structure_mass = 1.5 • baffles.y_bottom = 0.01 • sweeper.speed_scale = 50.0 • wind_amplitude = 6500.0 • wind_period_steps = 10 • gust_amplitude = 2500.0 • gust_period_steps = 5 • beam_friction = 0.0 • feed_y_min = 1.76 • second_wave_step = 120 • third_wave_step = 240 • mix.density = 18000.0 • mix.restitution = 1.0 • mix.friction = 0.0 • mix.radius_small = 0.08 • mix.radius_medium = 0.099 • mix.radius_large = 0.14 • mix.count_small = 50 • mix.count_medium = 50 • mix.count_large = 50 • mix.count_third_small = 50 • mix.count_third_medium = 50 • mix.count_third_large = 50 • gravity = (95.0, 1.2) • gravity_oscillation_amplitude = 55.0 • gravity_oscillation_period = 10 • linear_damping = 0.995 • angular_damping = 0.995
F_05: Cargo ship in rough seas		
	Stage-1: Metacentric Deficit	• cargo_water_y = 1.96 • cargo_loss_grace_steps = 180 • hull_roll_impulse_amplitude = 40.0 • hull_roll_impulse_interval_steps = 22
	Stage-2: Raised Loss Plane	• cargo_water_y = 1.98 • cargo_loss_grace_steps = 180 • hull_roll_impulse_amplitude = 50.0 • hull_roll_impulse_interval_steps = 22

ID	Stage	Mutations
	Stage-3: Shoal Lock-In	• cargo_water_y = 2.28 • cargo_loss_grace_steps = 180 • hull_roll_impulse_amplitude = 20.5 • hull_roll_impulse_interval_steps = 73 • boat_y_offset = -0.1 • rocks = [{'x': 13.05, 'y': 2.06, 'r': 0.46}, {'x': 15.0, 'y': 2.02, 'r': 0.52}, {'x': 16.95, 'y': 2.06, 'r': 0.46}, {'x': 15.0, 'y': 1.48, 'r': 0.33}] • build_zone_y_min = 2.58 • joint_max_force = 1750.0 • current_strength = 0.64 • cargo_restitution = 0.36 • cargo.friction = 0.62 • cargo.linear_damping = 0.24 • restoring_coeff = 240.0
	Stage-4: Perfect Storm Assembly Budget	• cargo_water_y = 2.28 • cargo_loss_grace_steps = 180 • hull_roll_impulse_amplitude = 160.0 • hull_roll_impulse_interval_steps = 14 • boat_y_offset = -0.1 • rocks = [{'x': 13.15, 'y': 2.02, 'r': 0.44}, {'x': 15.0, 'y': 1.97, 'r': 0.5}, {'x': 16.85, 'y': 2.02, 'r': 0.44}] • build_zone_y_min = 2.58 • joint_max_force = 99999.0 • current_strength = 2.0 • cargo_restitution = 0.88 • cargo.friction = 0.018 • cargo.linear_damping = 0.003 • restoring_coeff = 30.0 • deck_friction = 0.0 • max_structure_mass = 18.0 • wind_amplitude = 40.0 • wind_frequency = 0.32 • lateral_impulse_amplitude = 340.0 • lateral_impulse_interval_steps = 35 • rogue_amplitude = 65.0 • rogue_interval_steps = 90 • wave_amplitude = 55.0 • wave_frequency = 0.85 • wave2_amplitude = 35.0 • wave2_frequency = 0.65 • gust_amplitude = 35.0 • gust_interval_steps = 16 • gravity = (0, -42.0) • linear_damping = 0.002 • angular_damping = 0.0008
F_06: Pipeline pumping uphill		

ID	Stage	Mutations
	Stage-1: Localized Transport Anomaly	• gravwell_fy = -2500.0 • fluid.viscosity = 20.0 • fluid.count = 20 • min_delivery_ratio = 0.45 • max_steps = 2400 • max_time_seconds = 40.0 • force_budget = 5000.0
	Stage-2: Raised Delivery Target	• target_y_min = 2.5 • target_y_max = 4.0 • fluid.count = 20 • min_delivery_ratio = 0.45 • max_steps = 2400 • max_time_seconds = 40.0 • force_budget = 12000.0
	Stage-3: Transport Anomaly	• fluid.viscosity = 30.0 • fluid.count = 20 • min_delivery_ratio = 0.45 • max_steps = 2400 • max_time_seconds = 40.0 • force_budget = 12000.0
	Stage-4: Raised Delivery Adaptation	• target_y_min = 2.5 • target_y_max = 4.0 • fluid.viscosity = 2.0 • fluid.count = 20 • min_delivery_ratio = 0.45 • gravity = (0, -15.0) • max_steps = 2400 • max_time_seconds = 40.0 • force_budget = 12000.0

Table 20: Category: Granular & Fluid Interaction task specifications.

ID	Stage	Mutations
C_01: Inverted pendulum cart-pole		
	Stage-1: Curriculum stage 1	• track_center_x = 50.0 • pole_start_angle = 0.1658062789394612 • sensor_delay_angle_steps = 10 • sensor_delay_omega_steps = 10
	Stage-2: Curriculum stage 2	• track_center_x = 50.0 • cart_force_limit_newtons = 0.5 • max_steps = 2000 • gravity = 12.0 • sensor_delay_angle_steps = 10 • sensor_delay_omega_steps = 10 • pole_start_angle = 0.003 • safe_half_range = 0.1

ID	Stage	Mutations
	Stage-3: Curriculum stage 3	• track_center_x = 50.0 • gravity = 22.0 • sensor_delay_angle_steps = 3 • sensor_delay_omega_steps = 3 • cart_force_limit_newtons = 4.0 • pole_mass = 5.0 • cart_mass = 3.0 • safe_half_range = 0.2 • pole_length = 0.7 • max_steps = 1000 • pole_start_angle = 0.0004
	Stage-4: Curriculum stage 4	• track_center_x = 50.0 • gravity = 35.0 • sensor_delay_angle_steps = 3 • sensor_delay_omega_steps = 3 • cart_force_limit_newtons = 50.0 • pole_mass = 12.0 • cart_mass = 2.0 • safe_half_range = 0.15 • pole_length = 0.5 • max_steps = 500 • pole_start_angle = 5e-05 • cart_rail_center_y = 6.0
C_02: Lunar lander		
	Stage-1: Fragile Touchdown	• max_safe_vertical_speed = 0.05
	Stage-2: Micro-Corridor	• barrier_y_bottom = 7.0
	Stage-3: The Squeeze	• max_safe_vertical_speed = 1.0 • max_landing_angle = 0.07853981633974483 • barrier_y_bottom = 8.5 • total_fuel_impulse = 5000.0 • max_thrust = 650.0 • min_fuel_remaining_at_landing = 650.0 • wind_amplitude = 38.0 • gust_amplitude = 75.0 • platform_half_width = 1.2 • thrust_delay_steps = 6 • gravity_mutation.at_step = 250 • gravity_mutation.gravity_after = (0, -11.8)

ID	Stage	Mutations
	Stage-4: The Perfect Storm	• max_safe_vertical_speed = 2.6 • max_landing_angle = 0.12217304763960307 • thrust_delay_steps = 12 • total_fuel_impulse = 100000.0 • max_thrust = 1200.0 • min_fuel_remaining_at_landing = 500.0 • wind_amplitude = 15.0 • gust_amplitude = 20.0 • platform_half_width = 1.5 • barrier_y_bottom = 15.5 • gravity_mutation.at_step = 150 • gravity_mutation.gravity_after = (0, -11.5)
C_03: Homing vehicle		
	Stage-1: Adaptive Pursuit I	• impulse_budget = 5000.0 • target_speed = 1.9 • rendezvous_rel_speed = 1.5 • max_thrust_magnitude = 95.0 • cooldown_threshold = 90.0 • cooldown_max_thrust = 20.0 • cooldown_steps = 120 • blind_zone_x_min = 8.0 • blind_zone_x_max = 10.0 • gravity = (0.0, -0.5)
	Stage-2: Adaptive Pursuit II	• ground_friction = 0.0 • impulse_budget = 40000.0 • spawn_x = 14.0 • obstacles = [] • ice_zones = [] • gravity = (-3.5, 0.0)

ID	Stage	Mutations
	Stage-3: Adaptive Pursuit III	• ground_friction = 0.02 • impulse_budget = 20000.0 • rendezvous_rel_speed = 2.0 • rendezvous_heading_tolerance_deg = 120.0 • max_thrust_magnitude = 200.0 • cooldown_threshold = 190.0 • cooldown_max_thrust = 18.0 • cooldown_steps = 100 • blind_zone_x_min = 12.5 • blind_zone_x_max = 17.0 • speed_blind_threshold_mps = 4.0 • track_distance = 10.5 • target_speed = 2.0 • target_change_interval = 0.7 • slots_phase1 = [[3700, 3880], [4200, 4380]] • slots_phase2 = [[6200, 6380], [6700, 6880]] • spawn_x = 11.0 • spawn_y = 1.55 • ice_zones = [] • moving_obstacle = None • moving_obstacle_2 = None • obstacles = [(7.5, 1.3, 0.3, 0.3), (20.5, 1.3, 0.3, 0.3)] • linear_damping = 1.4 • angular_damping = 1.5 • gravity = (-3.5, -9.5)

ID	Stage	Mutations
	Stage-4: Adaptive Pursuit IV	• impulse_budget = 12000.0 • ground_friction = 0.01 • cooldown_threshold = 65.0 • cooldown_max_thrust = 10.0 • cooldown_steps = 250 • rendezvous_distance = 3.5 • rendezvous_rel_speed = 1.0 • rendezvous_heading_tolerance_deg = 30.0 • track_distance = 9.2 • target_speed = 1.0 • target_change_interval = 0.7 • blind_zone_x_min = 15.0 • blind_zone_x_max = 17.5 • speed_blind_threshold_mps = 1.2 • slots_phase1 = [[3650, 3880], [4150, 4410]] • slots_phase2 = [[6150, 6380], [6650, 6910]] • spawn_x = 14.0 • spawn_y = 1.45 • obstacles = [] • ice_zones = [] • moving_obstacle = None • moving_obstacle_2 = None • linear_damping = 2.5 • angular_damping = 3.5 • gravity = (0.0, -10.0)
C_04: Maze escape agent		
	Stage-1: Controller Adaptation I	• control_lag_steps = 25 • structural_impulse_scale_k = 12.0 • magnetic_floor_y_max = 1.6 • magnetic_floor_force = -60.0
	Stage-2: Altered Passage Geometry	• whisker_blind_front_x_lo = 5.0 • whisker_blind_front_x_hi = 13.0 • wall_overrides.5 = (9.0, 0.0, 0.2, 2.0) • wall_overrides.6 = (14.0, 0.0, 0.2, 2.0)
	Stage-3: Altered Passage Adaptation	• wall_overrides.4 = (5.0, 0.0, 0.2, 1.7) • wall_overrides.5 = (9.0, 1.7, 0.2, 0.6) • fluid_drag_x_min = 6.0 • fluid_drag_x_max = 14.0 • fluid_drag_coeff = 0.8 • turbulence_intensity = 80.0 • structural_impulse_scale_k = 50.0
	Stage-4: Controller Adaptation IV	• control_reversal_x_min = 0.0 • control_reversal_x_max = 20.0 • magnetic_floor_y_max = 1.5 • magnetic_floor_force = -80.0 • turbulence_intensity = 150.0 • control_lag_steps = 0 • structural_impulse_scale_k = 50.0

ID	Stage	Mutations
C_05: Sequential switch triggering		
	Stage-1: Extended trigger cooldown	• cooldown_steps = 300
	Stage-2: Long dwell, strong repulsion, strict zone speed	• trigger_stay_steps = 300 • speed_cap_inside = 0.05 • repulsion_mag = 40.0 • recent_a_for_b = 5000 • recent_b_for_c = 5000 • c_high_history = 5000
	Stage-3: Arctic Vortex — Ice Ramp, Extreme Dwell, Low Force Cap	• ramp_friction = 0.04 • ground_friction = 0.1 • speed_cap_inside = 0.03 • repulsion_mag = 55.0 • repulsion_tangential_mag = 38.0 • force_limit_inside = 24.0 • trigger_stay_steps = 150 • recent_a_for_b = 5000 • recent_b_for_c = 5000 • c_high_history = 5000
	Stage-4: Compound friction, barrier delay, repulsion	• ramp_friction = 0.02 • ground_friction = 0.2 • speed_cap_inside = 0.08 • repulsion_mag = 45.0 • repulsion_tangential_mag = 40.0 • force_limit_inside = 60.0 • trigger_stay_steps = 120 • barrier_delay_steps = 350 • recent_a_for_b = 5000 • recent_b_for_c = 5000 • c_high_history = 5000
C_06: Rotor speed matching		
	Stage-1: Curriculum variant 1	• k_drag = 2.7
	Stage-2: Curriculum variant 2	• k_drag = 2.6
	Stage-3: Curriculum variant 3	• step_load_at_step = 500 • torque_limit_at_zero = 2.9 • k_drag = 2.4 • torque_deadzone = 2.6 • measure_delay_steps = 7 • cogging_amplitude = 2.2
	Stage-4: Curriculum variant 4	• measure_delay_steps = 7 • torque_deadzone = 3.0 • torque_limit_at_zero = 3.5 • k_drag = 1.0 • cogging_amplitude = 4.0 • stiction_factor = 2.2

Table 21: Category: Cybernetics & Control task specifications.

ID	Stage	Mutations
E_01: Surviving reversing gravity		
	Stage-1: Altered Field I	• arena_y_max = 20.0 • build_zone_y_max = 18.0 • gravity = gravity_resonant_violent(t) • joint_force_limit = 1.0
	Stage-2: Altered Field II	• arena_y_max = 9.0 • build_zone_y_max = 8.5 • friction = 0.0 • gravity = gravity_cataclysm(t) • beam_density_scale = 200.0 • linear_damping = -6.0 • angular_damping = -4.0 • joint_force_limit = 0.001 • max_beam_count = 1 • max_structure_mass = 0.02
	Stage-3: Altered Field III	• arena_y_max = 9.5 • build_zone_y_max = 9.0 • friction = 0.0 • gravity = gravity_vortex_singularity(t) • beam_density_scale = 150.0 • linear_damping = -6.0 • angular_damping = -4.0 • joint_force_limit = 0.15 • max_beam_count = 2 • max_structure_mass = 0.3
	Stage-4: Altered Field IV	• arena_y_max = 9.0 • build_zone_y_max = 8.5 • friction = 0.0 • gravity = gravity_cataclysm(t) • beam_density_scale = 300.0 • linear_damping = -6.0 • angular_damping = -4.0 • joint_force_limit = 0.01 • max_beam_count = 1 • max_structure_mass = 0.005
E_02: Thruster flyer in fluid		
	Stage-1: Environment Variation 1	• linear_damping = 12.0 • constant_force_x = -180.0 • overheat_limit = 30000.0
	Stage-2: Environment Variation 2	• constant_force_y = 500.0
	Stage-3: Environment Variation 3	• linear_damping = 7.0 • drain_velocity_factor = 0.0 • slip_backward_force = -520.0 • overheat_limit = 9000.0 • constant_force_x = 260.0 • constant_force_y = -450.0 • wind_amplitude = 85.0 • wind_omega = 0.23

ID	Stage	Mutations
	Stage-4: Environment Variation 4	• linear_damping = 4.5 • constant_force_x = -48.0 • slip_backward_force = -75.0 • overheat_limit = 6000.0 • wind_amplitude = 105.0 • wind_omega = 0.48
E_03: Ice sled to goal zones		
	Stage-1: Environmental Adaptation I	• ground_friction = 1.0 • sled_friction = 1.0 • linear_damping = 0.8
	Stage-2: Environmental Adaptation II	• gravity = (0, -15)
	Stage-3: Environmental Adaptation III	• momentum_drain_factor = 0.7 • linear_damping = 0.5
	Stage-4: Environmental Adaptation IV	• ground_friction = 3.0 • sled_friction = 3.0 • gravity = (0, -18) • linear_damping = 5.0 • momentum_drain_factor = 0.3 • thrust_scale_factor = 0.06 • speed_penalty_threshold = 1.2 • speed_penalty_factor = 0.04
E_04: Time-varying beam mass		
	Stage-1: Environment Variation 1	• mass_freq_1 = 0.45 • mass_amp_1 = 0.75 • joint_break_force = 100.0 • joint_break_torque = 0.002 • wind_pressure = 10.0
	Stage-2: Environment Variation 2	• joint_break_force = 30000.0 • joint_break_torque = 1e-10 • wind_pressure = 10000.0 • fatigue_tau_seconds = 400.0
	Stage-3: Environment Variation 3	• joint_break_force = 50000.0 • wind_pressure = 1500.0 • mass_phase_gradient = 15.0 • mass_amp_1 = 0.4 • fatigue_tau_seconds = 400.0 • joint_break_torque = 0.05
	Stage-4: Environment Variation 4	• gravity = (15000.0, -30.0) • joint_break_force = 10000.0 • joint_break_torque = 1e-10 • fatigue_tau_seconds = 60.0
E_05: Invisible force fields		

ID	Stage	Mutations
	Stage-1: Field Variant I	• magnets = [(14.0, 0, -800.0), (14.0, 1, -800.0), (14.0, 2, -800.0), (14.0, 3, -800.0), (14.0, 4, -800.0), (14.0, 5, -800.0), (14.0, 6, -800.0), (14.0, 7, -800.0), (14.0, 8, -800.0), (14.0, 9, -800.0), (14.0, 10, -800.0), (14.0, 11, -800.0), (14.0, 12, -800.0), (14.0, 13, -800.0), (14.0, 14, -800.0)]
	Stage-2: Field Variant II	• magnets = [(13.0, 1.0, -140.0), (13.0, 2.0, -140.0), (13.0, 6.0, -140.0), (13.0, 8.0, -140.0), (18.0, 1.0, -140.0), (18.0, 3.0, -140.0), (18.0, 4.0, -140.0), (18.0, 5.0, -140.0), (18.0, 9.0, -140.0), (23.0, 1.0, -140.0), (23.0, 3.0, -140.0), (23.0, 4.0, -140.0), (23.0, 5.0, -140.0), (23.0, 9.0, -140.0), (4.0, 10.0, -140.0), (6.0, 10.0, -140.0), (8.0, 10.0, -140.0), (10.0, 10.0, -140.0), (12.0, 10.0, -140.0), (14.0, 10.0, -140.0), (16.0, 10.0, -140.0), (18.0, 10.0, -140.0), (20.0, 10.0, -140.0), (22.0, 10.0, -140.0), (24.0, 10.0, -140.0), (26.0, 10.0, -140.0), (28.0, 10.0, -140.0), (30.0, 10.0, -140.0), (32.0, 10.0, -140.0), (34.0, 10.0, -140.0), (36.0, 10.0, -140.0), (38.0, 10.0, -140.0)]
	Stage-3: Field Variant III	• magnets = [(12.0, 4.0, -420.0), (12.0, 5.0, -420.0), (12.0, 6.0, -420.0), (12.0, 7.0, -420.0), (12.0, 8.0, -400.0), (12.0, 8.3, -380.0), (11.0, 9.7, -280.0), (13.0, 9.7, -280.0), (15.0, 9.7, -280.0), (17.0, 9.7, -280.0), (19.0, 9.7, -280.0), (21.0, 9.7, -260.0), (15.0, 9.0, -420.0, 380.0, 0.12), (20.0, 9.0, -480.0, 440.0, 0.15, 3.14159), (19.0, 3.0, 320.0), (21.0, 3.5, 280.0), (24.0, 5.0, -250.0), (24.0, 8.2, -240.0), (24.0, 6.6, -240.0, 200.0, 0.165), (26.0, 5.5, -190.0), (27.0, 9.5, -190.0), (29.5, 7.5, 95.0)] • max_thrust = 420.0 • gravity = (0, -28.0) • linear_damping = 8.0 • angular_damping = 4.0
	Stage-4: Field Variant IV	• magnets = [(20.0, 8.0, -250.0, 240.0, 0.12), (12.0, 7.0, -80.0), (16.0, 7.0, -60.0), (24.0, 7.0, -90.0), (28.0, 7.0, -70.0), (29.5, 7.5, 120.0), (30.0, 8.0, 80.0)] • max_thrust = 580.0 • gravity = (0, -51.0) • linear_damping = 10.0 • angular_damping = 5.0
E_06: Random high-frequency impacts		

ID	Stage	Mutations
	Stage-1: Endurance Variant I	• max_structure_mass = 12.0 • gravity = (0, 16) • angular_damping = 0.0 • joint_break_force = 10800.0 • joint_break_torque = 14400.0 • beam_angvel_thresh = 2.1 • beam_angvel_tolerance_steps = 1
	Stage-2: Endurance Variant II	• max_structure_mass = 1.0 • gravity = (1500, 0) • linear_damping = 0.0 • angular_damping = 8.0 • noise_strength = 0.0 • coherent_pulse_interval = 501 • coherent_pulse_force = 0.0 • joint_break_force = 7200.0 • joint_break_torque = 4320.0 • damage_limit = 12.0 • damage_force_thresh = 3600.0 • damage_torque_thresh = 7200.0 • cascade_shock_damage = 40.0 • beam_angvel_thresh = 6.0 • beam_angvel_tolerance_steps = 3 • phased_storm_mult = 1.0 • phased_storm_start = 501 • phased_storm_end = 501 • burst_prob = 0.0
	Stage-3: Endurance Variant III	• max_structure_mass = 35.0 • gravity = (0, -17) • joint_break_force = 7200.0 • joint_break_torque = 12600.0 • coherent_pulse_interval = 1 • coherent_pulse_force = 130.0 • angular_damping = 0.0 • linear_damping = 0.0 • noise_strength = 14.0 • damage_limit = 1.0 • damage_force_thresh = 1260.0 • damage_torque_thresh = 2880.0 • cascade_shock_damage = 500.0 • beam_angvel_thresh = 1.5 • beam_angvel_tolerance_steps = 2 • phased_storm_mult = 4.5 • phased_storm_start = 15 • burst_prob = 0.15

ID	Stage	Mutations
	Stage-4: Endurance Variant IV	• max_structure_mass = 20.0 • allowed_anchor_x_lo = 5.6 • allowed_anchor_x_hi = 5.9 • gravity = (0, -30) • noise_strength = 2.0 • coherent_pulse_interval = 1 • coherent_pulse_force = 550.0 • angular_damping = 0.2 • linear_damping = 0.0 • joint_break_force = 7200.0 • joint_break_torque = 12600.0 • damage_limit = 0.9 • damage_force_thresh = 792.0 • damage_torque_thresh = 1368.0 • cascade_shock_damage = 5000.0 • beam_angvel_thresh = 1.0 • beam_angvel_tolerance_steps = 2 • phased_storm_mult = 4.5 • burst_prob = 0.4 • phased_storm_start = 0 • phased_storm_end = 500

Table 22: Category: Exotic Physics task specifications.