SysLoopBench: Evaluating Long-Horizon System Evolution by Autonomous Agents

Anonymous authors
Paper under double-blind review

Abstract

Autonomous agents hold promise for accelerating the development of scientific and engineering systems. However, existing benchmarks provide limited insight into whether agents can independently recreate the capabilities accumulated in expert-developed systems and advance beyond them. To address this gap, we introduce SysLoopBench, a benchmark that challenges agents to recreate and surpass expert-developed capabilities through long-horizon evolution of basic starting systems, retaining the original objectives and interfaces while independently developing their own algorithms and designs. SysLoopBench comprises 30 carefully designed tasks across mathematics, physics, biology, chemistry, engineering, and computer science. We construct each task from an expert-developed real-world system, extracting target capabilities, preparing a basic starting system and development environments, and establishing executable references for evaluation. Agents iteratively test and revise their systems using execution feedback, and we evaluate the resulting systems against expert-developed references on held-out workloads. Our evaluation reveals both the potential of long-horizon evolution to advance systems beyond expert references and the challenges of turning iterative improvements into lasting, generalizable gains across performance and efficiency.

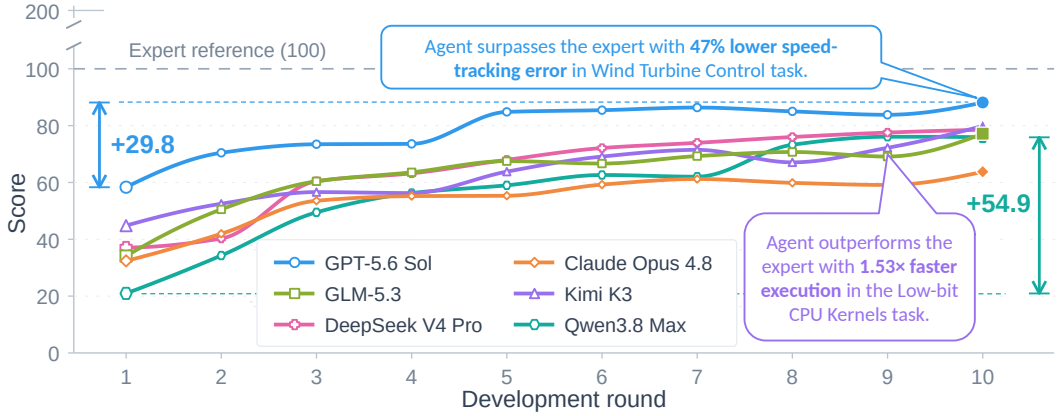

Figure 1: **Long-horizon system evolution on SysLoopBench.** Long-horizon evaluation reveals uneven progress, plateaus, and regressions, with sustained iteration enabling advances unseen in early results. Agents even produce systems with substantial practical advantages over expert references.

1 Introduction

Autonomous agents show promise in scientific discovery and engineering (Gottweis et al., 2026; Lu et al., 2026; Mitchener et al., 2025; Schmidgall et al., 2025) by developing and refining solutions through execution feedback (Romera-Paredes et al., 2024; Novikov et al., 2025). However, building high-performing systems requires sustained, interdependent decisions informed by prior progress and emerging limitations (Swanson et al., 2025; Ghareeb et al., 2026). This motivates our central question: *Can autonomous agents evolve scientific and engineering systems from basic starting code over long horizons to match or surpass expert-developed systems?*

Existing benchmarks primarily assess task completion and iterative optimization (Xie et al., 2024; Jimenez et al., 2024; Chan et al., 2025; Yang et al., 2026; Press et al., 2025; Garikaparthi et al., 2026; Xu et al., 2026c), but provide limited insight into how agents accumulate system-level capabilities

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

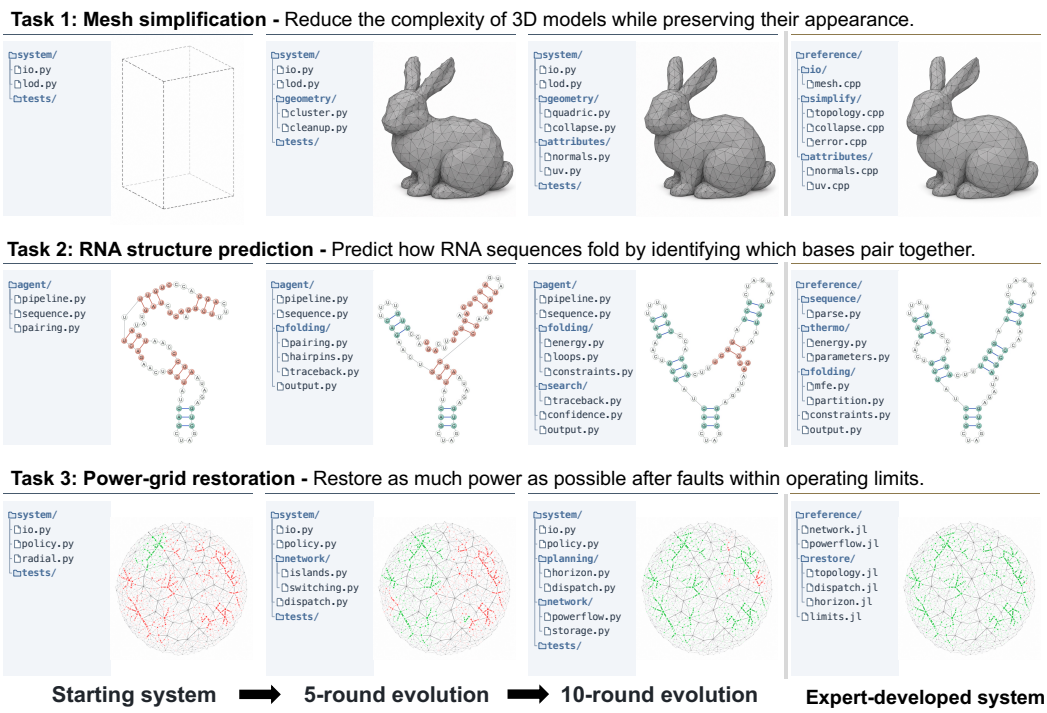

Figure 2: **Representative system evolution across SysLoopBench tasks.** Each row shows a development sequence from the starting system through 5 and 10 rounds of evolution and the expert-developed system. Every stage pairs the system's file and module structure with its task output.

over extended development. The central challenge is to determine whether successive revisions build on prior progress to approach or surpass expert-developed systems. Assessing this requires tracking the evolution of systems while allowing agents to reshape their algorithms and designs.

To address this gap, we introduce SysLoopBench, a benchmark for evaluating whether autonomous agents can evolve basic scientific and engineering systems to match or surpass expert-developed systems (Figure 2). Agents develop their systems over successive rounds in a persistent workspace, using execution feedback to guide further revisions. Task objectives and interfaces remain fixed, while agents can replace algorithms, develop new components, and reorganize system structure.

SysLoopBench comprises 30 tasks across mathematics, physics, biology, chemistry, engineering, and computer science, spanning numerical solvers, scientific prediction pipelines, retrieval systems, and inference components. Each task is derived from an expert-developed system and provides a starting codebase, development tools, and an executable evaluator. We evaluate successive system versions against expert references to track capability growth, regressions, and recovery throughout development, then assess the final system on held-out workloads.

Our evaluation of six frontier agents shows that autonomous system development extends beyond producing valid solutions: it requires sustaining progress, consolidating improvements, and meeting coupled engineering demands. SysLoopBench reveals four key insights:

- **System-building ability and evolutionary potential are distinct.** Initial advantages can narrow through continued development, while temporary plateaus can precede substantial advances. Development trajectories reveal capabilities that neither early performance nor final scores characterize.
- **Extended interaction does not automatically accumulate capability.** More tool use need not yield more revisions, and nearly one-third of positive score increments recover earlier losses. Effective development requires lasting improvements that generalize to unseen workloads.
- **Expert-level capability requires coordinated system quality.** Efficiency advantages can disappear when performance requirements tighten, while aggregate trade-offs conceal systems that can improve on both axes. These results expose the limitations of optimizing metrics independently.
- **Recreating expert capability and advancing beyond it coexist.** Agents remain below expert references overall, yet develop specialized kernels and adaptive controllers with substantial practical advantages. Evaluation against experts reveals capability gaps and engineering opportunities.

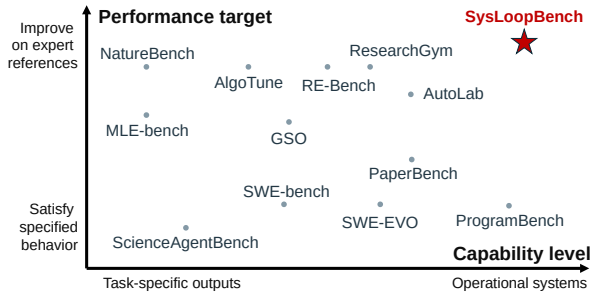

Figure 3: **Benchmark positioning.** Qualitative comparison of benchmarks: the capability level of the system being developed and the performance target used for evaluation.

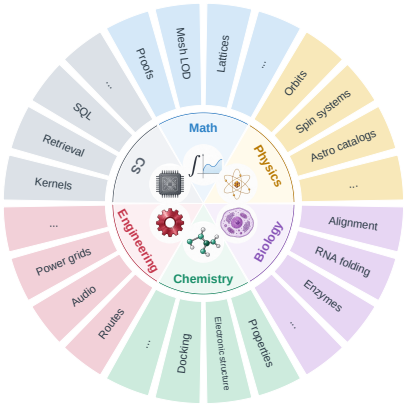

Figure 4: **Task coverage.** Overview of SysLoopBench organized across six scientific and engineering domains.

2 Related Work

System development benchmarks. Agent evaluation has expanded from resolving individual repository issues to coordinated feature and release-level changes (Jimenez et al., 2024; Le et al., 2025; Zhou et al., 2026a), to constructing complete libraries and programs from specifications or reference behavior (Zhao et al., 2025; Ding et al., 2025; Yang et al., 2026), and to recovering published results through code, data, and experiments (Siegel et al., 2024; Starace et al., 2025). These settings broaden the development scope but still judge systems by functional correctness and faithful reproduction. SysLoopBench instead asks agents to keep evolving a basic starting system over a long horizon, toward the quality and efficiency of the expert-developed system it was derived from.

Optimization and research benchmarks. A second line evaluates agents against measurable performance targets rather than a known-correct outcome. Optimization benchmarks target kernels, repositories, and heuristic algorithms (Ouyang et al., 2025; Shetty et al., 2025; Press et al., 2025; Imajuku et al., 2025), while research benchmarks target experiments, model training, and broader scientific and engineering objectives (Huang et al., 2023; Chan et al., 2025; Wijk et al., 2024; Nathani et al., 2025; Garikaparthi et al., 2026; Wang et al., 2026b; Xu et al., 2026c). SysLoopBench evaluates sustained development across scientific and engineering tasks, following how agents revise basic starting systems over successive rounds, approach expert-reference performance, and perform on held-out workloads (Figure 3). Section F provides extended comparisons with more related work.

3 The SysLoopBench

3.1 Task Definition

We formulate each SysLoopBench task as a system evolution problem in which an agent evolves a starting system derived from an expert-developed system. Formally, a task T is defined as

$$T = (G, X_0, E, V, B),$$

- **Development objective** G**.** The intended functionality and performance goals, together with the target interface and operating constraints, grounded in the expert-developed system.
- **Starting system** X_0**.** The initial system supplied to the agent, including its code and associated artifacts, with a defined scope for modification.
- **Development environment** E**.** The persistent workspace, permitted tools, dependencies, and execution facilities available to the agent for developing and testing its system.
- **Evaluator** V**.** The procedure for checking the agent's system for functional validity and measuring its performance against the expert-developed system reference.
- **Development budget** B**.** The limits governing development, including the number of rounds N, elapsed time, and computational resources.

As illustrated in Figure 5, an agent $\mathcal{A}$ starts from X_0 and works toward objective G over N rounds within budget B. It uses the resources in E to develop and test its system, drawing on feedback from V and its development history to guide subsequent changes. We denote this process by Run, which

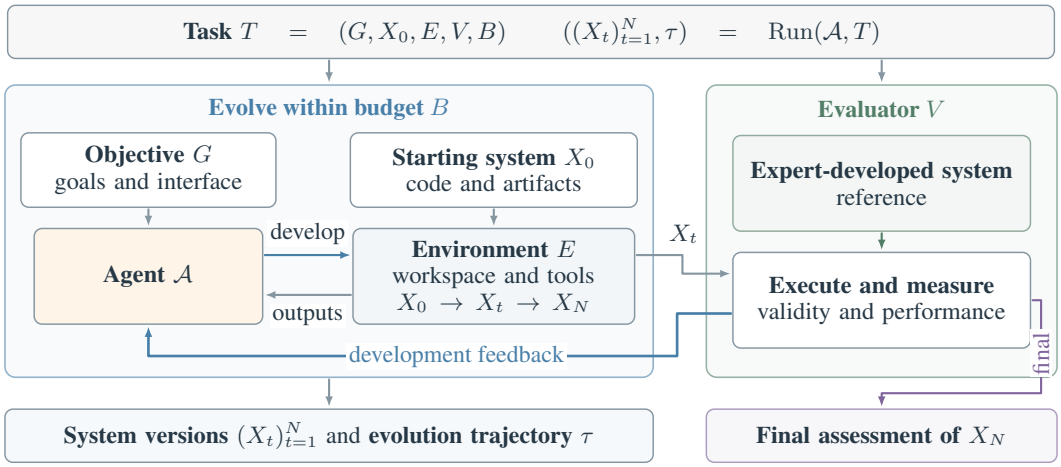

Figure 5: **Evaluation pipeline of SysLoopBench.** An agent evolves the starting system X_0 toward the objective G through repeated interaction with environment E and evaluator V. Under budget B, the process spans N rounds and produces system versions $(X_t)_{t=1}^N$ and development trajectory τ.

produces the system versions $(X_t)_{t=1}^N$ and a development trajectory τ:

$$\big((X_t)_{t=1}^N, \tau\big) = \mathrm{Run}(\mathcal{A}, T).$$

Here, X_t is the system produced at the end of round t, with X_N being the final system. The trajectory τ records the agent's actions, execution feedback, and resource use throughout development.

3.2 Task Collection and Coverage

Task selection. We select tasks that connect agent development to the demands of scientific research and engineering practice. And we choose expert-developed systems that provide both concrete problems worth solving and reference performance worth pursuing. Within these systems, we seek tasks that leave meaningful decisions to the agent, so that progress requires developing and refining solutions rather than following a prescribed procedure. We also require measurable outcomes, allowing different solutions to be compared against the expert-developed system.

Task scope. The resulting collection comprises 30 tasks across six scientific and engineering domains, with five tasks per domain (Figure 4). This breadth exposes agents to different sources of difficulty: numerical methods must remain stable, predictive models must generalize to unseen data, and computational systems must balance solution quality with resource use. These demands often interact. A faster solver may sacrifice accuracy, while a more accurate retrieval system may require greater memory or computation. Developing effective systems therefore requires agents to identify which limitations matter and choose improvements that respect the task's constraints.

3.3 Task Construction

Task specification. Each task starts from an expert-developed system that solves a concrete real-world problem. We describe the intended development objective in terms of the externally observable functionality, interface, and operating constraints that the evolved system must satisfy. We then derive a starting system by retaining a meaningful subset of the expert system and marking the components that the agent may change. This preserves a usable foundation while leaving design decisions open.

Development package. We package the starting system with the dependencies, tools, workspace, and execution commands required for development. The package also records the resource limits and the task-specific data or input generators used by the system. Components that define the task interface or setting remain fixed across runs, while the editable scope determines where the agent can make changes. Versioned package contents make each task initialization explicit and reproducible.

Task validation. Before including a task in the benchmark, we run the expert system through the task interface and verify the complete execution path, from input preparation to measured outputs. We test that the task runs successfully under its declared limits, that invalid outputs are detected, and that repeated evaluations produce consistent measurements. We also separate workloads used for development feedback from those reserved for final assessment, using application-appropriate splits to limit information leakage. These checks ensure that each task presents a well-defined evolution problem and supports a reliable comparison between the evolved system and the expert reference.

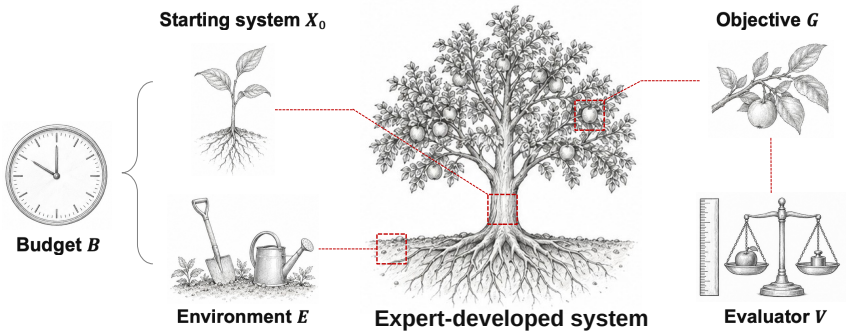

Figure 6: **Task construction of SysLoopBench.** The expert-developed system supplies the target functionality and reference level. We turn it into an evolution problem by defining the objective G, selecting a starting system X_0, and specifying the environment E, evaluator V, and budget B .

3.4 Evaluation Protocol

The *Loop* execution procedure. Each run follows a loop of system revision, execution, and feedback within a persistent workspace. Each round builds on the current system and development history, with evaluation feedback guiding subsequent changes to algorithms, components, and system structure. The task objective, interface, and constraints remain fixed throughout the loop.

Round-based evaluation. Within the development budget, the agent completes N rounds. At the end of round t, the evaluator executes an immutable snapshot of the resulting system X_t and returns feedback before the next round begins. Because each evaluation uses a frozen snapshot, subsequent evolution cannot alter the system or inputs associated with an earlier result. The protocol records each system version, its evaluation results, and the resources consumed during the loop.

Final assessment. After the final round, X_N is frozen for final assessment. The evaluator runs it on workloads reserved from the loop and compares the results with the expert-developed system under the same task conditions. These held-out evaluations are unavailable to the agent during the loop, separating feedback that guides evolution from the measurements used to assess the final system.

3.5 Scoring Framework

Validity and scoring axes. Let $c \in \{0, 1\}$ indicate whether the system satisfies all required functional conditions and hard resource limits. If $c = 0$, the task score is zero. If $c = 1$, the task metrics are divided between the performance and efficiency axes $\mathcal{Q} = \{\text{perf}, \text{eff}\}$. The performance axis measures attainment of the task's primary objective, such as accuracy, or numerical solution quality. The efficiency axis measures the associated resource costs. Metrics are assigned to the two axes according to each task's objective. Let $\mathcal{K}_q$ denote the metrics assigned to axis $q \in \mathcal{Q}$.

Reference-relative normalization. For each metric $k \in \mathcal{K}_q$, let m_k be its value for the system being scored and let $m_k^\star$ be the corresponding value for the expert-developed reference under the same workload. Let $d_k \in \{+1, -1\}$ encode the metric direction, with $d_k = +1$ when larger values are preferable and $d_k = -1$ when smaller values are preferable. We define the direction-aligned ratio $r_k = (m_k/m_k^\star)^{d_k}$. Thus, $r_k = 1$ indicates parity with the reference and $r_k > 1$ indicates better performance. We map this ratio to a bounded metric score, $s_k = 200/(1 + r_k^{-\kappa})$, where $\kappa > 0$ is fixed by the task. Expert parity corresponds to 100, with higher scores indicating better performance.

Axis and task aggregation. We use geometric means at both levels: aggregating normalized metrics within each axis and combining performance and efficiency into an overall task score. This preserves expert parity at 100 and penalizes weaknesses in individual metrics or axes.

4 Benchmarking Autonomous Agents on SysLoopBench

4.1 Experimental Setup

Agent configuration. We evaluate six frontier models: GPT-5.6 Sol (OpenAI, 2026), Claude Opus 4.8 (Anthropic, 2026), GLM-5.3 (GLM-5-Team et al., 2026), Kimi K3 (Team et al., 2026), DeepSeek V4 Pro (Xu et al., 2026a), and Qwen3.8 Max (Qwen Team, 2026). All models use the OpenCode harness[1], with subagent delegation disabled. We configure a model output limit of 128,000 tokens in OpenCode for all six models and retain the default sampling and reasoning settings.

[1] https://opencode.ai/docs/

Table 1: **Overall evaluation results on SysLoopBench.** Coverage counts valid systems and those with overall scores above 100. Bold and underlined values mark the best and second-best.

			Domain score ↑						Coverage ↑	
Model	**Stage**	**Overall ↑**	**Math.**	**Phys.**	**Bio.**	**Chem.**	**Eng.**	**CS**	**Valid**	**> 100**
GPT-5.6 Sol	1-round	58.32	47.79	96.85	38.75	77.70	54.28	34.53	25/30	8/30
	5-round	84.82	87.57	109.05	59.02	104.08	84.70	64.53	29/30	13/30
	10-round	88.14	93.75	110.14	63.42	106.27	86.29	68.95	29/30	15/30
	Final	**73.15**	**59.19**	**110.36**	60.41	71.63	71.94	65.37	28/30	11/30
Claude Opus 4.8	1-round	32.39	25.39	66.65	24.02	24.80	33.13	20.37	24/30	3/30
	5-round	55.32	30.58	91.33	26.69	91.38	57.95	34.00	25/30	9/30
	10-round	63.76	28.77	94.93	46.88	97.85	73.81	40.31	28/30	9/30
	Final	51.49	13.76	94.53	42.15	74.60	47.17	36.74	28/30	7/30
GLM-5.3	1-round	34.08	9.87	60.12	65.76	6.42	31.20	31.11	22/30	2/30
	5-round	67.49	43.20	96.88	72.50	38.39	96.62	57.32	26/30	8/30
	10-round	77.14	52.33	88.29	68.50	68.68	97.99	87.02	30/30	10/30
	Final	61.80	38.28	75.37	57.35	58.98	67.69	73.14	**29/30**	5/30
Kimi K3	1-round	44.80	39.66	71.91	38.14	54.64	37.15	27.26	24/30	3/30
	5-round	63.79	40.82	96.54	62.75	63.70	59.10	59.84	28/30	7/30
	10-round	79.88	46.74	108.26	81.02	100.12	69.22	73.90	29/30	12/30
	Final	72.82	30.42	108.67	**80.73**	73.90	66.43	**76.75**	28/30	11/30
DeepSeek V4 Pro	1-round	37.10	18.03	74.77	15.17	73.43	24.50	16.67	22/30	2/30
	5-round	67.95	46.97	99.53	39.16	94.07	69.64	58.32	27/30	7/30
	10-round	78.54	49.07	102.27	63.56	102.29	91.81	62.22	29/30	9/30
	Final	70.69	33.58	103.82	57.54	**81.29**	**85.50**	62.41	**29/30**	8/30
Qwen3.8 Max	1-round	20.94	12.11	18.59	38.00	5.94	33.74	17.26	23/30	0/30
	5-round	58.98	47.99	85.18	49.96	77.69	41.10	51.93	27/30	10/30
	10-round	75.89	60.42	109.47	66.66	87.40	70.08	61.29	29/30	13/30
	Final	67.92	51.55	109.67	62.55	63.36	63.82	56.58	**29/30**	**12/30**

Execution budgets. We run one ten-round episode for each model–task pair, yielding 180 episodes across 30 tasks. Each evolution round allows up to 45 minutes of agent execution. When this limit is reached, the current system is submitted for evaluation. Experiments run in task-specific Docker environments, with evaluation timeouts specified separately for each task and shared across models.

4.2 Overall System Capability

Finding 4.1: High validity leaves a substantial gap to expert-reference capability. As shown in Table 1, all six models produce valid final systems on 28 or 29 tasks, yet their mean overall scores range from only 51.49 to 73.15. Similar validity rates mask large capability differences. Even after excluding invalid submissions, the highest mean reaches just 78.38. Moreover, only 54 of the 171 valid systems exceed the expert reference. The remaining gap therefore lies largely in the capabilities of valid systems, rather than simply in failures to produce valid submissions. Passing the task requirements is widely achieved; matching expert capability remains substantially harder.

Finding 4.2: Similar overall capability emerges from different domain strengths. GPT-5.6 Sol and Kimi K3 achieve closely matched overall scores of 73.15 and 72.82, but their domain profiles differ substantially. GPT-5.6 Sol leads by 28.77 points in mathematics, whereas Kimi K3 leads by 20.31 points in biology and 11.38 points in computer science. DeepSeek V4 Pro, with an overall score of 70.69, achieves the highest domain scores in both chemistry and engineering. These results reveal multiple routes to strong overall system capability: models reach comparable aggregate scores through distinct combinations of domain strengths. The domain breakdown explains how each model's overall capability is distributed across the benchmark.

Finding 4.3: More above-reference systems can coexist with a larger aggregate shortfall. Qwen3.8 Max exceeds the reference on 12 tasks, compared with 11 for GPT-5.6 Sol, but its overall score is lower, at 67.92 versus 73.15. Averaged across all tasks, Qwen3.8 Max has a larger above-reference surplus, 4.78 versus 3.52 points, but also a larger below-reference shortfall, 36.86 versus 30.37. Its additional successes do not compensate for its deeper aggregate deficits. Coverage captures how often a model crosses the reference; overall score also captures how far its remaining systems fall short. Broad system capability consequently requires reducing weaknesses alongside producing individual strong solutions. The exact score decomposition is given in Section D.2.

Finding 4.4: Above-reference successes cluster on shared tasks across models. Across the six models, 14 of the 30 tasks have at least one final system exceeding the expert reference, and 12 of these have above-reference results from multiple models (Table 7). Despite their different domain strengths, models frequently surpass the reference on the same tasks. On the remaining 16 tasks, none of the agents exceeds it. The overlap reveals a shared boundary in current above-reference coverage: broader capability requires succeeding where no model surpasses the reference.

Table 2: **System revision dynamics on SysLoopBench.** We report the average numbers of distinct system versions and tool calls, the average rounds to reach 50% and 90% of development gains (R_{50} and R_{90}), and the fraction of episodes with renewed progress after a plateau. Details are in Section E.

Model	Distinct versions	Tool calls	$R_{50} \rightarrow R_{90}$	Post-plateau progress
GPT-5.6 Sol	9.47	189.8	$3.19 \rightarrow 5.77$	12/27
Claude Opus 4.8	5.97	296.6	$3.80 \rightarrow 4.84$	7/25
GLM-5.3	7.87	433.9	$4.00 \rightarrow 6.56$	14/26
Kimi K3	7.47	428.2	$4.85 \rightarrow 6.27$	19/28
DeepSeek V4 Pro	7.93	333.6	$3.80 \rightarrow 5.28$	12/27
Qwen3.8 Max	8.17	282.8	$3.68 \rightarrow 5.96$	14/27

Table 3: **Recovery and retention of evolution gains on SysLoopBench.** Recovery share measures gains spent recovering losses; peak-to-final drops measure unretained gains. Held-out success is the fraction of new development breakthroughs above the expert reference on held-out workloads.

Model	Recovery share (%)	Drop from peak (points)	Tasks with drop > 1	Held-out success
GPT-5.6 Sol	38.6	0.33	3/30	5/7
Claude Opus 4.8	26.2	2.60	7/30	4/6
GLM-5.3	30.0	5.57	9/30	4/11
Kimi K3	40.3	5.34	12/30	9/10
DeepSeek V4 Pro	33.6	0.65	4/30	5/7
Qwen3.8 Max	21.6	3.94	7/30	12/14

4.3 Evolution Dynamics

Finding 4.5: Weaker initial systems can achieve larger gains through evolution. The development scores in Table 1 distinguish initial system capability from improvement across ten rounds. Qwen3.8 Max starts at 20.94, compared with 58.32 for GPT-5.6 Sol, but gains 54.95 points to reach 75.89, whereas GPT-5.6 Sol gains 29.82 points to reach 88.14. Their gap consequently narrows from 37.38 to 12.25 points. Kimi K3 and DeepSeek V4 Pro also gain 35.08 and 41.44 points, respectively, reaching 79.88 and 78.54 from lower starting scores. Initial system quality and improvement through revision therefore capture distinct aspects of agent capability. Evaluating sustained evolution reveals how models compensate for weaker systems and narrow gaps that appear larger after a single round.

Finding 4.6: More interaction does not translate into more system revisions. The distinct-version and tool-call counts in Table 2 reveal different patterns of system revision under the same ten-round budget. GPT-5.6 Sol submits 9.47 distinct system versions per episode with 189.8 tool calls on average, whereas Claude Opus 4.8 makes 296.6 calls but submits only 5.97 versions. Kimi K3 makes 428.2 calls for 7.47 versions. GPT-5.6 Sol therefore presents a different system version in nearly every round, while Claude Opus 4.8 revisits submitted versions more often despite substantially more tool calls. The same horizon accommodates markedly different revision patterns, with frequent system updates in some models and more interaction around fewer submitted versions in others.

Finding 4.7: Short horizons can understate the gains from sustained evolution. The R_{50} and R_{90} values in Table 2 show that development gains emerge at different stages across models. Claude Opus 4.8 reaches 90% of its observed gain at round 4.84 on average, while Kimi K3 reaches only 50% at round 4.85. GLM-5.3 reaches 90% at round 6.56. These timing differences accompany substantial differences in later gains: from rounds 5 to 10, Kimi K3 and Qwen3.8 Max improve their mean development scores by 16.09 and 16.91 points, respectively, compared with 3.32 points for GPT-5.6 Sol (Table 1). A five-round horizon therefore captures different portions of the gains these models achieve over ten rounds. Longer evaluation horizons expose both the delayed arrival and the substantial magnitude of improvements from sustained iteration.

Finding 4.8: Temporary plateaus often precede further breakthroughs. The post-plateau progress results in Table 2 show that improvement frequently resumes after apparent stagnation. We identify a plateau when the best development score increases by at most one point in total over two consecutive rounds and the system submitted at the end of this window is valid. Of 160 eligible episodes, 78 subsequently improve on the best score attained by the end of the plateau by more than one point. This includes 19 of 28 episodes for Kimi K3 and 7 of 25 for Claude Opus 4.8. In 51 episodes, the subsequent gain exceeds five points. Progress also resumes in 48 of 134 eligible episodes when the plateau window extends to three rounds (Section E). These trajectories show that long-horizon evolution can advance through alternating periods of stagnation and improvement. Stopping at a short plateau would leave substantial gains unrealized in many episodes.

Finding 4.9: Nearly one-third of score gains recover previously attained levels. Recovery share in Table 3 shows how much of the observed score improvement restores earlier losses. Figure 7(b) decomposes these changes round by round. Across all models, 31.5% of positive score increments recover previously attained levels, with model-level shares ranging from 21.6% for Qwen3.8 Max to

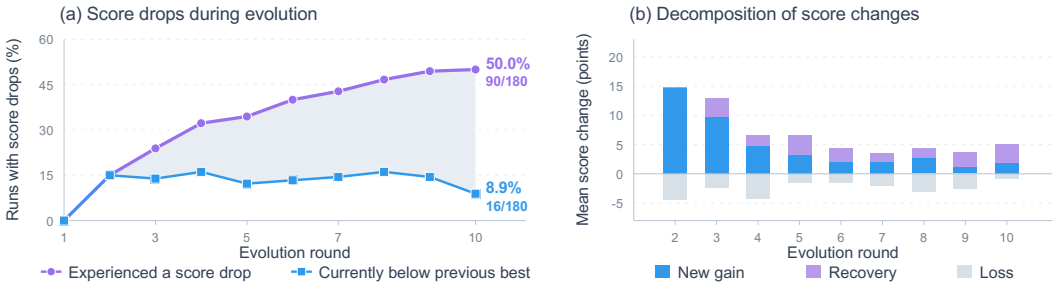

Figure 7: **Score regressions and recovery on SysLoopBench.** (a) Runs that have ever fallen more than five points below the previous best (purple) or remain so at each round (blue). (b) Mean per-round score changes, including new gains beyond the previous best, recovery, and losses.

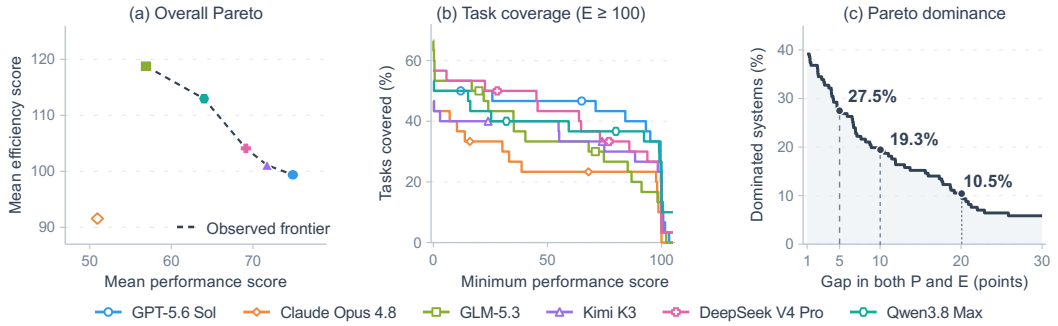

Figure 8: **Performance and efficiency of final systems on SysLoopBench.** (a) Mean P and E scores, with the observed Pareto frontier dashed. (b) Task coverage with $P \geq \tau$ and $E \geq 100$. (c) Systems outscored by at least the indicated margin by another system on the same task and reference.

40.3% for Kimi K3. Validity follows a similar pattern: of 73 invalid-to-valid transitions after round 1, 36 establish validity for the first time and 37 restore it. Upward movements therefore include repeated recovery of the same score range. Separating recovery from new gains reveals how much of the apparent progress extends the best score attained so far and how much retraces earlier losses.

Finding 4.10: Similar final scores conceal different degrees of gain retention. The Drop from peak and Tasks with drop > 1 columns in Table 3 measure the score losses that remain at the end of evolution. Kimi K3 finishes more than one point below its development peak on 12 of 30 tasks, with an average drop of 5.34 points across all tasks. GPT-5.6 Sol shows such drops on only 3 of 30 tasks, averaging 0.33 points. Yet their final held-out overall scores are closely matched at 72.82 and 73.15, respectively (Table 1). Across all models, 42 of 180 episodes end more than one point below their development peak. At the larger five-point threshold, Figure 7(a) shows that 90 of 180 runs experience a drop during evolution, while 16 remain more than five points below their best at round 10. Many regressions are therefore temporary, but some persist through the final revision. Similar final capability thus conceals different amounts of previously attained score left behind at delivery. Reaching a peak and retaining it are distinct challenges. Converting intermediate achievements into strong final systems requires retaining gains through the last revision.

Finding 4.11: Development breakthroughs translate unevenly into held-out success. The Held-out success column in Table 3 tracks whether episodes that first exceed the expert reference after round 1 in development also finish above it on held-out workloads. This occurs in 9 of 10 episodes for Kimi K3 and 12 of 14 for Qwen3.8 Max, compared with 4 of 11 for GLM-5.3. Across all models, 39 of 55 episodes deliver final systems that exceed the reference on held-out evaluation. Most newly attained development breakthroughs therefore extend to unseen workloads, but their conversion into final held-out success varies substantially across models. Long-horizon evolution must turn improvements under development feedback into systems that retain their advantage on new workloads. Held-out success captures this final step from progress to delivered system capability.

4.4 Balancing Performance and Efficiency

Finding 4.12: Performance requirements reverse the ordering of efficiency coverage. The normalized scores in Figure 8(a) show that GPT-5.6 Sol achieves the highest mean performance score ($P = 74.88$), whereas GLM-5.3 achieves the highest mean efficiency score ($E = 118.77$). However, their ordering by the number of tasks meeting the efficiency reference changes when

Table 4: Results on Low-bit CPU Kernels .

Metric	Expert	Kimi K3
Latency (ms) ↓	1.213	**0.794**
Decode: bandwidth	0.356	**0.254**
Decode: batch	0.927	**0.582**
Prefill: compute	3.080	**1.804**
Prefill: tail	2.133	**1.490**
Memory (MiB) ↓	**1.688**	1.879
Correct cases	64/64	64/64
Overall score ↑	100.00	**120.50**

Table 5: Results on Wind Turbine Control .

Metric	Expert	GPT-5.6 Sol
Speed error (%) ↓	4.360	**2.305**
Energy capture (%) ↑	88.98	**93.79**
Blade DEL ↓	0.2232	**0.2231**
Tower DEL ↓	0.2174	**0.1930**
Peak load ↓	**0.8083**	0.8892
Platform RMS (rad) ↓	**0.01704**	0.02068
Safety utilization ↓	0.8941	**0.8830**
Step p99 (μs) ↓	**28.16**	45.92
Overall score ↑	**100.00**	96.72

performance requirements are introduced. Without a minimum performance requirement, GLM-5.3 achieves $E \geq 100$ on 20 of 30 tasks, compared with 15 for GPT-5.6 Sol. Requiring $P \geq 95$ reverses this ordering: only 5 GLM-5.3 systems meet both thresholds, compared with 11 for GPT-5.6 Sol (Figure 8(b)). At $P \geq 100$, the corresponding counts fall to 2 and 7. GLM-5.3's broader efficiency coverage thus gives way to GPT-5.6 Sol's stronger joint coverage as the required performance approaches the expert reference. Evaluating efficiency at a specified performance level reveals which models deliver resource-efficient systems that also meet the intended performance target.

Finding 4.13: Models on the aggregate frontier still leave substantial room for joint improvement. Five agents lie on the observed aggregate Pareto frontier in Figure 8(a), yet many of their individual systems remain dominated on both axes. For each valid final system, we identify whether another agent's system on the same task and reference exceeds both P and E by at least a common margin. As shown in Figure 8(c), such a peer exists for 47/171 systems (27.5%) at a five-point margin, 33/171 (19.3%) at ten points, and 18/171 (10.5%) at twenty points. The five-point gaps span 22 tasks and all six agents; 36 of the 47 systems come from agents on the aggregate frontier. These evaluated alternatives demonstrate that substantial improvements in both objectives are jointly attainable under the task constraints. The trade-off across model averages therefore coexists with widespread room to improve individual systems without sacrificing either axis. Detailed rules are specified in Section D.2.

4.5 Beyond-Reference Improvements in Practice

Finding 4.14: Long-horizon evolution delivers concrete engineering gains beyond expert references. The cases in Figure 1 connect beyond-reference improvements to concrete changes in system implementation. Kimi K3 accelerates low-bit computation through specialized CPU kernels, while GPT-5.6 Sol improves turbine regulation and energy capture through an adaptive control design. Their final implementations first appear in rounds 9 and 10, respectively, and deliver gains across the held-out workloads evaluated for each task. These cases illustrate how extended iteration produces system-specific solutions with measurable engineering benefits and explicit resource trade-offs.

Faster low-bit computation. On Low-bit CPU Kernels, Kimi K3 achieves a $1.53\times$ geometric-mean speedup in amortized latency over the native reference while passing all 64 held-out numerical checks (Table 4). The acceleration spans all four decode and prefill regimes. The implementation retains native compressed weights, decodes weight blocks on demand with AVX-512 VNNI kernels, and reuses them across activation rows. Specialized small-batch matrix–vector paths use loop unrolling and independent accumulators to expose instruction-level parallelism. The system achieves these gains with 11.3% more working memory and an overall score of 120.50.

Improved turbine regulation and energy capture. On Wind Turbine Control, GPT-5.6 Sol reduces mean RMS speed-tracking error by 47.1% and increases mean net energy capture by 5.4% relative to the ROSCO (Abbas et al., 2022) robust reference (Table 5). Both metrics improve in all 12 held-out OpenFAST (OpenFAST Developers) scenarios. The controller combines gain-scheduled proportional-integral control, phase-lead speed feedback, and overspeed damping based on available torque margin. Pitch gains incorporate turbine-specific physical parameters. These changes deliver tighter regulation and greater energy capture alongside higher peak loads, greater platform motion in floating scenarios, and increased latency. The overall score of 96.72 reflects these trade-offs.

5 Conclusion

We introduced SysLoopBench to evaluate whether autonomous agents can evolve scientific and engineering systems toward and beyond expert-developed capabilities. Its 30 tasks across six domains connect development trajectories with final system quality. Our evaluation reveals practical advances beyond expert references alongside persistent challenges in preserving gains, generalizing improvements, and balancing performance with efficiency. These findings establish SysLoopBench as a testbed for understanding and advancing long-horizon autonomous system development.

AI USE STATEMENT

We used AI tools to assist with manuscript polishing, code optimization, and the cross-checking of experimental results. For manuscript preparation, these tools helped refine wording, improve clarity and readability, and maintain consistent terminology. For code optimization, they supported the review and refinement of implementation details. We also used AI assistance to check consistency among reported results, tables, figures, and their accompanying descriptions. These uses supported the preparation and review of the work; they do not constitute independent validation of its findings. The authors retain responsibility for the research methodology, interpretation of results, and all content in the final manuscript.

ETHICS STATEMENT

SYSLOOPBENCH evaluates autonomous system evolution for scientific and engineering systems. Improvements on benchmark workloads do not establish that agent-generated systems are safe or reliable for deployment, particularly in physical control or other consequential applications. Such deployment requires independent validation and appropriate human oversight. Use and redistribution of the third-party software and datasets underlying the tasks must respect their applicable licenses and access conditions.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the benchmark design, evaluation protocol, and experimental setup to support reproducibility. Section 4.1 specifies the agent configuration and execution budgets. Sections A and B document task construction, workloads, interfaces, evaluation criteria, and expert references. Section C describes submission handling, measurement boundaries, and score computation. Complete task-level results and definitions of the trajectory analyses are provided in Sections D and E, enabling readers to examine the reported comparisons and their underlying evaluation procedures.

REFERENCES

Nikhar J Abbas, Daniel S Zalkind, Lucy Pao, and Alan Wright. A reference open-source controller for fixed and floating offshore wind turbines. *Wind Energy Science*, 7(1):53–73, 2022.

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. Llm-coordination: evaluating and analyzing multi-agent coordination abilities in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 8053–8072, 2025.

Anthropic. Introducing Claude Opus 4.8, 2026. URL https://www.anthropic.com/news/claude-opus-4-8.

Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. Super: Evaluating agents on setting up and executing tasks from research repositories. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12622–12645, 2024.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2025.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025.

Jialong Chen, Xander Xu, Hu Wei, Chuan Chen, and Bing Zhao. Swe-ci: Evaluating agent capabilities in maintaining codebases via continuous integration. *arXiv preprint arXiv:2603.03823*, 2026a.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, et al. Skillcraft: Can llm agents learn to use tools skillfully? *arXiv preprint arXiv:2603.00718*, 2026b.

Zhi Chen, Zhensu Sun, Yuling Shi, David Lo, and Lingxiao Jiang. Are performance-optimization benchmarks reliably measuring coding agents? *arXiv preprint arXiv:2607.01211*, 2026c.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Jingzhe Ding, Shengda Long, Changxin Pu, Huan Zhou, Hongwan Gao, Xiang Gao, Chao He, Yue Hou, Fei Hu, Zhaojian Li, et al. Nl2repo-bench: Towards long-horizon repository generation evaluation of coding agents. *arXiv preprint arXiv:2512.12730*, 2025.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Nicholas Edwards, Yukyung Lee, Yujun Audrey Mao, Yulu Qin, Sebastian Schuster, and Najoung Kim. Rexbench: Can coding agents autonomously implement ai research extensions? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16380–16417, 2026.

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

Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. Researchgym: Evaluating language model agents on real-world ai research. *arXiv preprint arXiv:2602.15112*, 2026.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Dmytro Shved, Gavin J Gyimesi, Jon M Laurent, Samantha M Wright, Muhammed T Razzak, et al. A multi-agent system for automating scientific discovery. *Nature*, 2026.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with co-scientist. *Nature*, 2026.

Xinyi He, Qian Liu, Mingzhe Du, Lin Yan, Zhijie Fan, Yiming Huang, Yin Zheng, Zejian Yuan, and Zejun Ma. Swe-perf: Can language models optimize code performance on real-world repositories? *arXiv preprint arXiv:2507.12415*, 2025.

Deyao Hong, Yizhe Chi, Wenyi Li, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. Swe refactor bench: Can coding agents complete a long-horizon, whole-repository stack migration? *arXiv preprint arXiv:2608.23564*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, zili wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf.

Lisheng Huang, Chen Yang, Hao Zhou, Huatong Song, Zongchao Chen, Ran Le, Yang Song, Wayne Xin Zhao, and Tao Zhang. Evo-bench: Can language models improve agent harness? *arXiv preprint arXiv:2608.09096*, 2026.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. Ale-bench: A benchmark for long-horizon objective-driven algorithm engineering. *Advances in Neural Information Processing Systems*, 38, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long software tasks. *Advances in Neural Information Processing Systems*, 38:92213–92266, 2025.

Tue Le, Minh VT Thai, Dung Nguyen Manh, Huy Phan Nhat, and Nghi DQ Bui. Swe-evo: Benchmarking coding agents in long-horizon software evolution scenarios. *arXiv preprint arXiv:2512.18470*, 2025.

Han Li, Zhemin Fang, Rili Feng, Yingqi Zhao, Jiaheng Liu, Pengfei Gao, He Ye, Dayi Lin, Qingwei Lin, Saravan Rajmohan, et al. Loopsbench: From harness engineering to loop engineering in benchmarking coding agent. *arXiv preprint arXiv:2608.00267*, 2026a.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026b.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, et al. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *arXiv preprint arXiv:2607.08964*, 2026c.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, et al. Airs-bench: a suite of tasks for frontier ai research science agents. *arXiv preprint arXiv:2602.06855*, 2026.

Jeffrey Jian Ma, Milad Hashemi, Amir Yazdanbakhsh, Kevin Swersky, Ofir Press, Enhui Li, Vijay Janapa Reddi, and Parthasarathy Ranganathan. Swe-fficiency: Can language models optimize real-world repositories on real workloads? *arXiv preprint arXiv:2511.06090*, 2025.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C Landsness, Daniel L Barabasi, Siddharth Narayanan, Nicky Evans, et al. Kosmos: An ai scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*, 2025.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

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

OpenAI. GPT-5.6 Sol. OpenAI model documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol.

OpenFAST Developers. OpenFAST. GitHub repository. URL https://github.com/OpenFAST/openfast.

Anne Ouyang, Simon Guo, Simran Arora, Alex L Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. Kernelbench: Can llms write efficient gpu kernels? *arXiv preprint arXiv:2502.10517*, 2025.

Ori Press, Brandon Amos, Haoyu Zhao, Yikai Wu, Samuel Ainsworth, Dominik Krupke, Patrick Kidger, Touqir Sajed, Bartolomeo Stellato, Jisun Park, et al. Algotune: Can language models speed up general-purpose numerical programs? *Advances in Neural Information Processing Systems*, 38, 2025.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Qwen Team. Qwen3.8-Max. QwenCloud model documentation, 2026. URL https://www.qwencloud.com/models/qwen3.8-max.

David Rein, Joel Becker, Amy Deng, Seraphina Nix, Chris Canal, Daniel O'Connel, Pip Arnott, Ryan Bloom, Thomas Broadley, Katharyn Garcia, et al. Hcast: Human-calibrated autonomy software tasks. *arXiv preprint arXiv:2503.17354*, 2025.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Jiaqi Shao, Hanck Chen, Wei Zhang, Maxm Pan, and Bing Luo. Do agent benchmarks measure capability? protocol validity in the age of agentic ai. *arXiv preprint arXiv:2607.22368*, 2026.

Haiyang Shen, Xuanzhong Chen, Wendong Xu, Yun Ma, Liang Chen, and Kuan Li. Evocode-bench: Evaluating coding agents in multi-turn iterative interactions. *arXiv preprint arXiv:2605.24110*, 2026.

Manish Shetty, Naman Jain, Jinjian Liu, Vijay Kethanaboyina, Koushik Sen, and Ion Stoica. Gso: Challenging software optimization tasks for evaluating swe-agents. *Advances in Neural Information Processing Systems*, 38, 2025.

Zachary S Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. Corebench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *arXiv preprint arXiv:2409.11363*, 2024.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

Haochen Sun, Shuwen Zhang, Lujie Niu, Lei Ren, Hao Xu, Hao Fu, Fangkun Zhao, Caixia Yuan, and Xiaojie Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4922–4951, 2025.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Varun Ursekar, Apaar Shanker, Yash Maurya, Shehab Yasser, Vijay S Kalmath, Veronica Chatrath, and Yuan Xue. Harnessopt-bench: Evaluating llms at harness optimization. *arXiv preprint arXiv:2608.06301*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Yike Wang, Huaisheng Zhu, Zhengyu Hu, Yige Yuan, Zhengyu Chen, Shakti Senthil, Hannaneh Hajishirzi, Yulia Tsvetkov, Pradeep Dasigi, and Teng Xiao. Rethinking the evaluation of harness evolution for agents. In *COLM 2026 The 2nd Workshop on Lifelong Agents: Learning, Aligning, and Evolving*, 2026a.

Yuru Wang, Lejun Cheng, Yuxin Zuo, Sihang Zeng, Bingxiang He, Che Jiang, Junlin Yang, Yuchong Wang, Kaikai Zhao, Weifeng Huang, et al. Naturebench: Can coding agents match the published sota of nature-family papers? *arXiv preprint arXiv:2606.24530*, 2026b.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, et al. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026a.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-0315. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0d744742f6fac4d1134c019b7cef3c8a-Paper-Datasets_and_Benchmarks_Track.pdf.

Xinbo Xu, Ruihan Yang, Haiyang Shen, Wendong Xu, Bofei Gao, Ruoyu Wu, Kean Shi, Weichu Xie, Xuanzhong Chen, Ming Wu, et al. Roadmapbench: Evaluating long-horizon agentic software development across version upgrades. *arXiv preprint arXiv:2605.15846*, 2026b.

Zhangchen Xu, Junda Chen, Yue Huang, Dongfu Jiang, Jiefeng Chen, Hang Hua, Zijian Wu, Zheyuan Liu, Zexue He, Lichi Li, et al. Autolab: Can frontier models solve long-horizon auto research and engineering tasks? *arXiv preprint arXiv:2606.05080*, 2026c.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, et al. Programbench: Can language models rebuild programs from scratch? *arXiv preprint arXiv:2605.03546*, 2026.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, et al. Harness-bench: Measuring harness effects across models in realistic agent workflows. *arXiv preprint arXiv:2605.27922*, 2026.

Yunxiang Zhang, Muhammad Khalifa, Shitanshu Bhushan, Grant Murphy, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Mlrc-bench: Can language agents solve machine learning research challenges? *Advances in Neural Information Processing Systems*, 38, 2025.

Yunxiang Zhang, Ping Yu, Jianyu Wang, Julian Reed, Azalia Mirhoseini, Will Su, et al. Kernelbench-verified: Do llm-generated kernels actually beat pytorch? *arXiv preprint arXiv:2607.16241*, 2026a.

Zhaoxi Zhang, Yiming Xu, Jiahui Liang, Weikang Li, Xiaoshuai Chen, Liwei Qian, Xin Pei, Jizhou Huang, Run Sun, and Yunfang Wu. Repozero: Can llms generate a code repository from scratch? *arXiv preprint arXiv:2605.07122*, 2026b.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, et al. Skillflow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*, 2026c.

Bingchen Zhao, Dhruv Srikanth, Yuxiang Wu, and Zhengyao Jiang. Specbench: Measuring reward hacking in long-horizon coding agents. *arXiv preprint arXiv:2605.21384*, 2026.

Wenting Zhao, Nan Jiang, Celine Lee, Justin Chiu, Claire Cardie, Matthias Gallé, and Alexander Rush. Commit0: Library generation from scratch. In *International Conference on Learning Representations*, volume 2025, pp. 12061–12076, 2025.

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

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

Qixing Zhou, Jiacheng Zhang, Haiyang Wang, Rui Hao, Jiahe Wang, Minghao Han, Yuxue Yang, Shuzhe Wu, Feiyang Pan, Lue Fan, et al. Featurebench: Benchmarking agentic coding for complex feature development. *arXiv preprint arXiv:2602.10975*, 2026a.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Zhanke Zhou, Zongze Li, Weikai Huang, Xuan Li, Chentao Cao, Xiao Feng, Xiangyu Lu, Jinbo Hu, Menghan Lu, Yi Xie, et al. Reasoning is more than the model: Harness-aware evaluation of agents on verifiable reasoning tasks. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026b.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench : Evaluating the collaboration and competition of LLM agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.421. URL https://aclanthology.org/2025.acl-long.421/.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, et al. Establishing best practices in building rigorous agentic benchmarks. *Advances in Neural Information Processing Systems*, 38, 2025b.

APPENDIX CONTENTS

A Task Design Details **18**
A.1 Mathematics . . . 18
A.2 Physics . . . 18
A.3 Biology . . . 19
A.4 Chemistry . . . 19
A.5 Engineering . . . 19
A.6 Computer Science . . . 20

B Task Cards **21**
B.1 Mathematics . . . 22
B.2 Physics . . . 23
B.3 Biology . . . 24
B.4 Chemistry . . . 26
B.5 Engineering . . . 27
B.6 Computer Science . . . 29

C Construction and Evaluation Implementation **31**
C.1 Construction implementation . . . 31
C.2 Submission and feedback . . . 31
C.3 Measurement boundaries . . . 31
C.4 Task metrics and score computation . . . 32
C.5 Experiment records . . . 32

D Full Task-Level Agent Results **33**
D.1 Performance and efficiency by task . . . 34
D.2 Final-system analysis . . . 36

E Evolution Trajectories and Analysis **37**
E.1 Evolution statistics . . . 37
E.2 Round-level results . . . 37

F Extended Related Work **44**
F.1 Interactive task and workflow completion . . . 44
F.2 Executable artifact development . . . 44
F.3 Skills, harnesses, and multi-agent organization . . . 47
F.4 Scale, validity, and reproducibility . . . 48
F.5 Landscape summary . . . 49

A Task Design Details

This appendix details the discipline-specific choices behind the task interfaces, workload composition, and evaluation measures introduced in Section 3.3. The focus is on how each design exposes the scientific or engineering decisions an agent must make. Individual objectives, starting systems, and deliverables are listed in Section B.

A.1 Mathematics

Preserving structure under approximation. Mesh simplification requires outputs at 50%, 20%, and 5% of the original triangle count. These levels expose how an algorithm's local decisions affect progressively coarser approximations. The workload combines smooth surfaces with open shells, disconnected components, texture seams, and faceted CAD geometry. Evaluation samples the triangle surfaces by area, measures geometric distance relative to object size, interpolates normals and texture coordinates at corresponding surface points, and compares rasterized silhouettes from multiple views. These complementary measurements distinguish preserving appearance from merely reducing the number of triangles.

Lattice reduction couples numerical search with an exact structural invariant. The workload spans q-ary, integer-relation, and triangular bases in dimensions 80–160. Input and output bases are compared through their canonical Hermite normal forms, establishing that the integer lattice is preserved. Quality measures include both the shortest vector and the shortest quarter of the basis, rewarding improvement across several directions. Lean proof search similarly separates search decisions from certification: the agent composes a strategy over structural and rule-application primitives, while the Lean kernel checks completed proofs. Competing rules, failed branches, and witness choices make search order consequential.

Coupled sequential decisions. Inventory workloads include a six-stage backlog network, allocation across heterogeneous stores, and single- and multi-warehouse settings with real demand. The supplied transitions make stock movement and cost accumulation explicit, leaving the agent to learn a causal policy. Costs are compared within network families before aggregation. Portfolio construction uses stress scenarios at two asset scales and evaluates realized holdings after transaction costs, liquidity constraints, and rounding. Both designs expose the effect of current decisions on future states; the measured outcome depends on the complete sequence of feasible actions.

A.2 Physics

Mechanism-based workload variation. N-body dynamics uses ten cases in each split across eight physical regimes, including eccentric orbits, hierarchical triples, strong few-body interactions, encounters, and collapse. Initial conditions and spatial rotations vary between splits. Position error is normalized by the initial system radius, and velocity error by the corresponding gravitational velocity scale, allowing different systems to contribute comparable accuracy measurements. Quantum spin workloads span nine mechanisms, including critical and disordered chains, frustrated interactions, different spin values, and ladder geometries. Energy and both near- and far-pair correlations characterize the returned state.

Observation-driven tasks vary the information available to the estimator. Astronomical association combines positional uncertainty, missing counterparts, clutter, and coupled assignment patterns in 26 generated families. Orbit determination varies orbital regimes, observation noise, and outliers, while supplying propagation and measurement primitives. The remaining development problem concerns association or estimation, including how evidence is combined and how uncertainty affects the output.

Numerical reference checks. The N-body reference trajectory is checked using a tighter integration tolerance. Quantum-spin reference calculations are checked with a larger bond dimension and alternative initial states, while the evaluator independently contracts each submitted state. Radiative-transfer workloads vary optical thickness, absorption, scattering, illumination angle, and layer structure; the 64-stream reference fluxes are checked at 96 streams. These checks connect the accuracy scale used for comparison to the numerical resolution of the reference calculation.

A.3 BIOLOGY

Workloads that separate biological mechanisms. The multiple-sequence-alignment evaluation contains 24 full-length cases, with four cases in each of six BAliBASE families: divergent sequences, orphan sequences, divergent subgroups, terminal extensions, internal insertions, and repeats. The development cohort covers the first five mechanisms, and overlap screening removes duplicate sequences and substantial shared sequence fragments. Alignment quality is measured on author-annotated structural core blocks, using the corresponding residue-pair and column criteria for each family. Equal family weighting retains the contribution of each alignment mechanism.

RNA workloads contain circular sequences across four composition and structural families at lengths 64, 96, 144, and 192. Visible per-base and pairwise soft energies enter both the minimum-free-energy calculation and the partition function. Targets are computed from the resulting thermodynamic model. Evaluation therefore examines a predicted structure together with its pairing uncertainty: each nucleotide's probability distribution includes alternative partners and the unpaired state, and the comparison covers that complete support.

Linking outputs to biological evidence. Peptide-spectrum matching combines measured spectra with sequence and modification-location labels, allowing sequence identification and modification localization to be assessed separately within the same search problem. Phylogenetic evaluation holds the substitution model fixed and fits branch lengths for the submitted topology before comparing likelihoods. This makes topology search the agent's central decision. Enzyme prediction supplies frozen protein embeddings and separate fitting and calibration examples, focusing development on multilabel prediction and confidence estimation. Its deployment measurements include loading and prediction latency, inference memory, and the saved model representation.

A.4 CHEMISTRY

Separating physical primitives from solver decisions. Hartree–Fock search gives candidate and reference solvers a common initial density and the same integral oracle. The agent chooses the iterative method, mixing, and stability-search strategy. The evaluation cohort contains 26 cases covering eight overlapping slices: stretched bonds, near-degeneracy, multicentre systems, transition metals, open shells, charge, diffuse functions, and larger bases. From each returned density, the evaluator checks electron count, Hermiticity, and idempotency, then recomputes energy, convergence residuals, and internal orbital stability. Integral-build counts measure the solver's computational work across these regimes.

The coupled-cluster tasks use frozen canonical molecular-orbital integrals to isolate amplitude solving and the perturbative triples calculation. Their separate development and evaluation cohorts each contain eight cases spanning atoms, diatomic and polyatomic molecules, stretched bonds, and basis sets. CCSD evaluation combines correlation-energy accuracy with amplitude residuals; the triples task measures the correction directly. The fixed inputs make contraction design and numerical solution the main sources of variation.

Molecular search and prediction. Macrocycle docking uses prepared complexes with 12–29-membered rings and varying torsional freedom. Initial conformers are generated from molecular connectivity, while the candidate and native search share the same force field and energy-gradient primitive. Receptor identities separate the development and evaluation cohorts. The scored objective includes intermolecular, intramolecular, and ring-closure contributions, so search must jointly resolve pose placement and molecular conformation. ADMET prediction instead uses molecular-scaffold splits and endpoint-specific classification or regression measures, connecting representation choices to transfer across chemical structures.

A.5 ENGINEERING

Evaluating actions through system dynamics. Grid restoration preserves phase connections, impedances, device limits, fault availability, and the forecast supplied to the policy. Submitted switching and load decisions are fixed before an unbalanced AC calculation determines feasible dispatch and restored service. A policy may emit updated plans during computation; the evaluator

records the quality of the latest plan over time. This captures both eventual restoration quality and how soon useful decisions become available.

Wind control connects candidate commands directly to OpenFAST through a bridge that transfers sensor readings and actuator commands. Candidate and reference controllers receive the same turbine summary and rotor-performance surface. Controller state advances once per control instant, and saved state is tested through a fresh-process restore. Power, speed regulation, structural-load measures, and floating-platform motion are derived from simulator outputs, linking the controller's local decisions to complete closed-loop behavior.

Complete input-to-output lifecycles. Image compression evaluates 24 images per split, with four images in each of six source families, at five rates from 0.125 to 2 bits per pixel. Every rate point uses an actual bitstream and a fresh-process decoder. Encoding cost includes rate-search trials, and efficiency sums encoding and decoding time over the measured image-rate pairs. This couples rate control, representation, and decoding in the same evaluation.

Road matching evaluates complete route geometry and gives equal source weight to recorded Hof journeys and Utrecht routes generated on road geometry. Observation profiles use separate serving state, while graph indexes can be reused within a profile. Audio separation balances two- and three-speaker mixtures with clean and noisy conditions, and measures inference from mixture release to complete waveform return. These lifecycle choices measure the usable artifact produced by the agent.

A.6 Computer Science

Crossing workload dimensions. Low-bit computation spans four weight formats and four computational regimes, with separate matrix shapes, padding, tails, and numerical distributions. Reference arithmetic is checked against an independent higher-precision format decoder. Working-memory measurements include packed or expanded weights, inputs, outputs, and intermediate storage, exposing the cost of alternative representations. Flash Attention combines ragged batches, grouped and multi-query attention, asymmetric context lengths, local windows, and several head dimensions. Fresh input variants change segment and cache lengths, so kernel specialization must account for both layout and operating regime.

Approximate nearest-neighbor search increases the collection from 100,000 vectors in development to one million in evaluation. Independent queries and exact-neighbor truth make the effects of compression and candidate selection observable at the larger scale. SQL planning uses 40 evaluation queries, including held-out templates and join structures with up to 17 relations. Cardinality estimates are compared on the candidate's own intermediate relation sets, and its join trees are executed to measure the consequences of those estimates.

State and measurement semantics. Order-to-Cash workloads vary event versions, identity merges, exchange-rate revisions, retrospective matching rules, and source retractions. The system must maintain ledger balances and the evidence supporting them across checkpoints. Integer monetary identities, clean replay, and withdrawal of upstream events test the consistency of the complete result. For computational tasks, timing follows the operation under study: resident index queries and attention execution exclude initialization, while planning cost is measured separately from query execution. These boundaries tie resource measurements to the algorithmic decisions being evaluated.

B TASK CARDS

The cards describe the 30 tasks by primary discipline. Each presents the system to be developed, the supplied starting point and algorithmic challenges, followed by its workload and evaluation against the expert reference. Table 6 links each task to its deliverable. The construction choices underlying these tasks are detailed in Section A.

Table 6: The 30 tasks grouped by primary discipline. Card numbers link to the corresponding descriptions.

Card	Task	Deliverable
Mathematics		
1	Lean proof search	Reusable proof-search strategy
2	Mesh simplification	Attribute-aware mesh simplifier
3	Inventory replenishment	Causal replenishment and allocation policy
4	Portfolio construction	Transaction-aware portfolio policy
5	Lattice reduction	Short-basis reduction algorithm
Physics		
6	Astronomical catalog association	Uncertainty-aware catalog matcher
7	Orbit determination	Persistent orbit estimator
8	Quantum spin ground-state search	Matrix-product-state solver
9	N-body dynamics	Newtonian trajectory integrator
10	Radiative transfer	Layered-medium flux solver
Biology		
11	Enzyme function prediction	Multilabel enzyme predictor
12	Peptide-spectrum matching	Spectrum database-search engine
13	RNA secondary structure	RNA folding and probability predictor
14	Multiple sequence alignment	Protein alignment algorithm
15	Phylogenetic tree inference	Maximum-likelihood tree search
Chemistry		
16	Hartree–Fock self-consistent field	Self-consistent-field solver
17	Molecular ADMET prediction	Molecular property predictor
18	Flexible macrocycle docking	Molecular pose-search algorithm
19	CCSD amplitude solver	Coupled-cluster amplitude solver
20	Perturbative triples correction	Triples-energy correction algorithm
Engineering		
21	Distribution-grid restoration	Anytime restoration policy
22	Wind turbine control	Persistent closed-loop controller
23	Learned image compression	Trainable image codec
24	Road map matching	GPS-to-road route matcher
25	Blind audio separation	Trainable speech separator
Computer Science		
26	Low-bit CPU kernels	Quantized matrix-computation kernels
27	Approximate nearest neighbors	Persistent vector index
28	Order-to-Cash reconciliation	Incremental ledger and audit pipeline
29	SQL cardinality and planning	Cardinality estimator and join planner
30	Flash Attention	Prefill and cache-attention kernels

B.1 MATHEMATICS

Task 1. Lean proof search

Develop a reusable proof-search strategy that navigates competing rules, witness choices, and expensive failed branches while producing Lean-certified proofs. The shallow starting strategy provides basic structural steps, leaving the agent to coordinate search ordering, backtracking, and termination. The central challenge is to resolve more theorems without allowing unproductive branches to dominate computation.

Workload.
Development problems expose the search primitives. Separate evaluation problems introduce competing rules and costly failed branches, so a strategy must transfer beyond individual theorem statements.

Evaluation and reference.
The Lean kernel certifies each completed proof. Solve rate and theorem-checking time are compared with Aesop, jointly measuring search coverage and computational cost.

Task 2. Mesh simplification

Reduce a mesh to several levels of detail while preserving geometry, surface normals, texture coordinates, and silhouette. The starting system provides mesh input and output but no simplification algorithm. The agent must reconcile aggressive triangle reduction with appearance preservation: a locally inexpensive removal can damage a sharp feature or introduce distortion that becomes pronounced at coarser resolutions.

Workload.
Development and evaluation use separate mesh cohorts, including smooth surfaces and flat-shaded CAD geometry. Each input must produce the requested levels of detail.

Evaluation and reference.
The evaluator checks mesh validity and triangle budgets, then measures surface, normal, and silhouette distortion against attribute-aware meshoptimizer. Computation, peak memory, and output size capture the cost of preserving appearance at reduced resolution.

Task 3. Inventory replenishment

Develop a causal replenishment policy for networks in which stores compete for limited upstream inventory and decisions affect future shortages and backlogs. The supplied transitions establish the dynamics but leave policy design and learning to the agent. Effective decisions must account for uncertain demand, heterogeneous costs, and coupled allocation constraints rather than optimize each location independently.

Workload.
Workloads cover serial backlogs, heterogeneous stores, and single- and multi-warehouse demand settings, using synthetic trajectories and real demand histories. Decisions may use observed state but not future demand.

Evaluation and reference.
Inventory feasibility is checked at each decision. Operating costs and policy inference cost are compared with the author-provided HDPO reference within each network family; training is measured separately.

Task 4. Portfolio construction

Develop a portfolio policy that earns utility above cash while accounting for transaction costs, liquidity limits, and allocation constraints. The supplied interface leaves forecasting and sequential allocation to the agent. Frequent rebalancing may exploit a changing signal yet lose its benefit to trading costs, while conservative decisions can miss opportunities; the policy must manage this trade-off across distinct market stress regimes.

Workload.
Synthetic stress scenarios vary market behavior and portfolio size. The policy receives only causal observations, and a common simulator applies trading costs, liquidity constraints, and rounding.

Evaluation and reference.
Feasible portfolio decisions are evaluated by excess log utility above cash and decision cost. Native multi-period optimization in cvxportfolio provides the reference, with balanced comparisons across stress scenarios.

Task 5. Lattice reduction

Find substantially shorter vectors in high-dimensional integer lattices while preserving the exact lattice. The agent must coordinate reduction, orthogonalization, block search, and stopping rules without a complete reduction package. Larger search neighborhoods can improve basis quality but increase computation, while numerical errors in intermediate calculations must not compromise the exact integer output.

Workload.
Separate development and evaluation matrices span q-ary, integer-relation, and triangular lattice families at different dimensions. The returned basis must preserve the exact integer row lattice.

Evaluation and reference.
Exact lattice equality is verified independently. Shortest-vector length, the lengths of the shortest quarter of basis vectors, and reduction time are compared with native fplll BKZ.

B.2 PHYSICS

Task 6. Astronomical catalog association

Resolve source-to-catalog correspondences when positional uncertainty, clutter, missing counterparts, and competing assignments make proximity alone insufficient. Starting from a spherical nearest-neighbor matcher, the agent must decide when to associate, reject, or reconsider a match in light of alternatives. The task couples uncertainty-aware assignment with the time and memory demands of larger catalogs.

Workload.
Separate generated cases vary spherical geometry, uncertainty, clutter, coupled assignment patterns, and catalog size. Ground-truth correspondences are determined by latent source identities.

Evaluation and reference.
Independent source correspondences determine association quality. The native LSST association routine supplies the reference for quality, computation time, and peak memory.

Task 7. Orbit determination

Develop a persistent orbit estimator that separates orbital state from nuisance parameters using noisy sequential observations that may contain outliers. The damped least-squares starter leaves robust updates, uncertainty estimation, and convergence management to the agent. A small observation residual alone is insufficient: the recovered state, covariance, and future trajectory must remain mutually consistent across satellites and orbital regimes.

Workload.
Development and evaluation separate satellites and orbital regimes. Workloads contain noisy observations and outliers, with controlled trajectories providing known states; estimates must survive saving and reloading.

Evaluation and reference.
The evaluator checks finite orbital states, covariance validity, and consistency of propagated trajectories. State accuracy, prediction accuracy, uncertainty behavior, and computational cost are compared with the Orekit-based reference.

Task 8. Quantum spin ground-state search

Optimize a matrix-product state that reproduces both low energy and spin correlations across critical, disordered, and frustrated systems. The product-state starter contains no nontrivial inter-site correlations. The agent must choose and coordinate state updates that capture increasingly complex interactions within a bounded representation, balancing approximation quality, numerical stability, and the cost of optimization.

Workload.
Independently generated systems cover critical, disordered, frustrated, and ladder geometries, including different spin values. These cases require both local and longer-range correlations.

Evaluation and reference.
Independent contraction of the submitted state yields its energy and correlations. These quantities and solving time are compared with a fixed TeNPy density-matrix renormalization-group reference.

Task 9. N-body dynamics

Integrate interacting gravitational trajectories across regimes with very different dynamical timescales, including eccentric motion and close encounters. The free-motion starter leaves force evaluation, integration, and step control to the agent. Resolving rapid interactions can require costly small steps, while larger steps risk accumulating position and velocity error over the requested trajectory; one integration strategy must cope with both behaviors.

Workload.
Separate synthetic systems cover eccentric orbits, strong few-body interactions, hierarchical systems, planetary configurations, encounters, and collapse. The force law is direct Newtonian gravity without added softening or collision removal.

Evaluation and reference.
Dimensionless position and velocity errors measure trajectory accuracy against REBOUND's IAS15 integrator. Solving cost includes integration and trajectory output, with comparisons balanced across physical regimes.

Task 10. Radiative transfer

Solve coupled absorption and multiple scattering through layered media, including reflective boundaries and grazing illumination. The supplied interface leaves angular treatment, propagation between layers, and numerical solution to the agent. Direct-beam or low-order approximations can miss repeated scattering and boundary interactions, while increasing resolution raises cost; accurate upward and downward fluxes must be recovered across contrasting optical regimes.

Workload.
Layered inputs specify optical depth, scattering properties, incident radiation, and boundary reflection. Cases vary absorption, scattering strength, illumination angle, and layer structure.

Evaluation and reference.
Flux accuracy and solving cost are compared with PythonicDISORT's discrete-ordinates solution. Higher-resolution reference calculations establish the numerical comparison point for the scattering workloads.

B.3 BIOLOGY

Task 11. Enzyme function prediction

Develop a calibrated multilabel enzyme predictor from frozen protein embeddings when labels differ substantially in frequency and proteins may support several functions. The agent must choose the classifier and confidence calibration without changing the embedding representation. Recovering rare functions while controlling spurious assignments, and approaching a mature pretrained reference within a compact deployable model, creates coupled prediction and deployment demands.

Workload.
CLEAN-derived fitting and calibration examples support development, while evaluation retains the full specified test cohort, including rare labels. The reference uses the published CLEAN weights and original inference procedure.

Evaluation and reference.
Predictive quality is measured against annotated enzyme functions, with the published CLEAN model as the reference. Deployment latency, inference memory, and saved model size measure the cost of the resulting predictor.

Task 12. Peptide-spectrum matching

Search a peptide database for both sequence identity and modification location from tandem mass spectra. The starter supplies constants and output conventions but no search algorithm. The agent must manage the candidate expansion induced by digestion and modification choices, distinguish competing fragment explanations, and enforce precursor-mass constraints while keeping indexing and spectrum search computationally practical.

Workload.
Measured synthetic-peptide spectra are paired with a public sequence database and declared digestion, mass-tolerance, and modification rules. Protected labels identify the peptide and modification positions.

Evaluation and reference.
The evaluator checks peptide admissibility and measures identification accuracy, modification localization, and search cost. Native Comet supplies the reference under the same search rules.

Task 13. RNA secondary structure

Recover a thermodynamically appropriate RNA fold and its base-pair probabilities under circular topology and sequence-specific energy adjustments. The pair-counting starter does not resolve the energetic competition among alternative folds. The agent must combine structural constraints, branching and repeated pairing registers, and uncertainty over competing structures, rather than optimize a single pairing count in isolation.

Workload.
Development and evaluation use independent generated circular RNA sequences with varied composition, repeated pairing registers, branching patterns, and lengths. Per-base and pairwise energy adjustments affect both structures and probabilities.

Evaluation and reference.
Valid structures and pairing probabilities are compared with ViennaRNA thermodynamic outputs. Folding quality, probability accuracy, confidence calibration, runtime, and memory characterize the resulting prediction system.

Task 14. Multiple sequence alignment

Infer a joint residue correspondence across divergent proteins with extensions, insertions, repeats, and uneven sequence similarity. Starting from padding alone, the agent must construct and refine a coherent multiple alignment. Pairwise similarities need not yield a consistent global alignment, and early gap or grouping decisions can propagate errors across many sequences; refinement must improve structural correspondence without excessive search cost.

Workload.
BAliBASE workloads cover divergent sequences, orphan sequences, divergent subgroups, extensions, insertions, and repeats. Development is screened for sequence overlap; the repeat family is reserved for evaluation.

Evaluation and reference.
Author-provided structural-core scores measure recovered residue correspondences and alignment columns. Quality and solving time are compared with fixed MAFFT L-INS-i, with equal weighting across workload families.

Task 15. Phylogenetic tree inference

Search for an evolutionary topology that explains a nucleotide or protein alignment under a fixed substitution model. The neighbor-joining starter supplies a valid tree but leaves likelihood-oriented topology search to the agent. The number of possible trees grows rapidly with the number of taxa, requiring selective exploration of alternative relationships while maintaining valid taxon coverage and controlling computation.

Workload.
Separate nucleotide and protein alignments define development and evaluation workloads. The model is fixed for each case, so comparisons focus on the tree inferred from the supplied alignment.

Evaluation and reference.
Branch lengths are fitted under the specified model before computing likelihood improvement over a star-tree baseline. Native IQ-TREE3 search supplies the reference; efficiency measures candidate topology-search time.

B.4 CHEMISTRY

Task 16. Hartree–Fock self-consistent field

Reach accurate, internally stable Hartree–Fock solutions for molecular systems where simple damped iteration may oscillate, converge slowly, or settle on an unfavorable stationary state. The integral oracle supplies basic operations but leaves occupation selection, mixing, convergence acceleration, and stability search to the agent. The solver must coordinate these decisions across difficult electronic regimes under a finite integral-build budget.

Workload.
Separate molecular geometries and compositions cover stretched bonds, near-degeneracy, multicentre systems, transition metals, open shells, charge, diffuse functions, and larger basis sets.

Evaluation and reference.
Energy, convergence, and stability are recomputed from the submitted densities and compared with native PySCF using convergence and stability fallbacks. Normalized integral-oracle calls measure the work needed to obtain those solutions.

Task 17. Molecular ADMET prediction

Develop molecular predictors that transfer across chemical scaffolds and heterogeneous ADMET endpoints, including both classification and regression. The weak starter leaves representation, fitting, and model selection to the agent. Features and learning choices useful for one property need not transfer to another, so broad predictive quality must be obtained while controlling inference cost and memory across endpoints.

Workload.
Therapeutics Data Commons endpoints include classification and regression problems. Frozen molecular-scaffold splits separate training and evaluation and test transfer across chemical structures.

Evaluation and reference.
Endpoint-appropriate classification and regression measures are compared with the full author-provided MapLight-TDC reference. Batch prediction cost and inference memory measure deployment efficiency; training is measured separately.

Task 18. Flexible macrocycle docking

Search a coupled translational, rotational, and torsional landscape for low-energy macrocycle poses while satisfying ring-closure energetics. The starter retains an unbound conformation, and the supplied energy primitive provides no search strategy. The agent must coordinate local refinement and broader exploration across competing basins, allocating restarts and computation without access to the bound starting pose.

Workload.
Prepared complexes vary receptor identity, ring size, and torsional freedom. Development and evaluation separate receptor accessions; bound coordinates are not supplied as candidate starting conformations.

Evaluation and reference.
The full force-field search objective, including intramolecular and ring-closure terms, measures pose quality. Native AutoDock Vina supplies the reference for that objective and solving time.

Task 19. CCSD amplitude solver

Solve coupled nonlinear singles and doubles amplitude equations to recover electron correlation from molecular-orbital integrals. Starting from zero amplitudes, the agent must implement and organize tensor contractions, iterative updates, and convergence acceleration. Accurate energy alone does not establish that the amplitude equations are solved; residual reduction and numerical stability must be maintained as molecular structure and basis size change.

Workload.
Separate development and evaluation cohorts contain atoms, diatomic and polyatomic molecules, stretched bonds, and different basis sets. Fixed input integrals isolate the amplitude-solving problem from orbital preparation.

Evaluation and reference.
Correlation-energy accuracy and amplitude residuals measure the numerical solution against native PySCF RCCSD. Reference-relative solving time measures efficiency, with balanced comparisons across molecular families.

Task 20. Perturbative triples correction

Evaluate the perturbative triples energy correction through high-order contractions over occupied and virtual orbitals. The zero-correction starter leaves the full numerical procedure to the agent. Correct handling of permutations and energy denominators must be combined with an efficient contraction order: straightforward intermediate construction can become costly as the orbital space grows, even though the final output is a single energy correction.

Workload.
Development and evaluation cover atoms, small molecules, stretched bonds, and different basis sets using frozen canonical molecular-orbital inputs. The supplied molecular quantities define the triples-correction calculation.

Evaluation and reference.
Absolute error in the triples correction and solving time are compared with native PySCF RCCSD(T). Molecular families receive balanced weight in the aggregated result.

B.5 Engineering

Task 21. Distribution-grid restoration

Develop an anytime restoration policy that coordinates switching and load delivery across multiple periods under unbalanced electrical constraints. The conservative radial starter leaves improved search and allocation to the agent. A promising topological plan may fail AC feasibility, and an expensive search may delay useful restoration; the policy must produce feasible intermediate plans while improving their eventual quality.

Workload.
Multi-period workloads provide network information and forecasts, while protected realizations determine the actual operating conditions. Separate network topologies test whether the policy transfers beyond development feeders.

Evaluation and reference.
Unbalanced AC calculations establish the feasibility and service quality of emitted plans. Peak

restoration quality and the area under the quality-over-time curve are compared with the executable PowerModelsONM reference.

Task 22. Wind turbine control

Develop a power-producing feedback controller from a minimal parking policy, coordinating torque, pitch, and yaw within actuator limits. The agent must tune persistent control behavior across changing operating conditions and different turbine configurations. Energy capture competes with load and safety objectives, while delayed or poorly damped responses can degrade closed-loop behavior, making isolated command accuracy insufficient.

Workload.
Development and evaluation use different turbines and scenarios, including land-based and floating configurations. Candidate commands drive the same OpenFAST simulation loop used for the reference controller.

Evaluation and reference.
OpenFAST outputs determine power production, operating loads, and safety-related behavior relative to ROSCO. Controller-response latency measures the computational cost of the closed-loop policy.

Task 23. Learned image compression

Develop a complete image codec that jointly manages representation learning, quantization, entropy coding, and independent decoding across several rate targets. The supplied interface leaves these coupled components to the agent. Better distortion at one rate may require a more costly representation or decoder, and apparent compression gains must survive actual bitstream generation and a fresh-process round trip across different image families.

Workload.
A source-disjoint corpus spans six image families and multiple rate targets. Candidate and reference share numerical initialization conditions, and decoding is repeated in a fresh process to verify the complete round trip.

Evaluation and reference.
Mean squared error and structural-similarity distortion at the specified rates are compared with CompressAI's variable-rate hyperprior reference. Total encoding and decoding time measures the cost of the complete codec.

Task 24. Road map matching

Infer a coherent road route from noisy GPS observations when locally plausible road choices can imply incompatible transitions. The empty-route starter leaves spatial candidate selection and sequence-level inference to the agent. The matcher must reconcile observation fit with network connectivity and direction, while efficient indexing and transition computation keep repeated route requests practical on complete road graphs.

Workload.
Workloads combine recorded Hof journeys and independent Utrecht routes generated on real road geometry. Development and evaluation contain distinct journeys within shared cities and road networks.

Evaluation and reference.
Direction-aware correct-route fraction measures agreement with known route geometry. Native MapMatching2 Viterbi matching supplies the reference, and persistent-request latency measures inference cost after loading and warmup.

Task 25. Blind audio separation

Develop a speech separation system that recovers individual speakers from overlapping two- and three-speaker mixtures, including noisy recordings. The spectral-mask starter leaves representation

and separation methods to the agent. The system must resolve overlapping speech and background noise while maintaining practical inference cost.

Workload.
LibriMix workloads include clean and noisy mixtures with two and three speakers. The reference uses the author-released ConvTasNet model and its inference procedure.

Evaluation and reference.
Scale-invariant signal-to-distortion improvement measures separation quality across mixtures and difficult groups. Inference real-time factors measure efficiency relative to ConvTasNet through the same request path.

B.6 Computer Science

Task 26. Low-bit CPU kernels

Optimize quantized matrix computations across several packed weight formats and matrix regimes while preserving their exact arithmetic semantics. The correct scalar starter leaves vectorization, data reuse, and parallel decomposition to the agent. Format decoding, padding, and irregular tails complicate fast execution, and an optimization that benefits one shape or degree of parallelism may impose overhead on another.

Workload.
Separate cases vary weight formats, computational regimes, matrix shapes, padding, tails, numerical distributions, and available parallelism. Inputs include both ordinary and boundary layouts.

Evaluation and reference.
Every output is checked against independently validated quantized arithmetic. Computation cost and working memory are compared with the native exact low-bit kernels in ik_llama.cpp.

Task 27. Approximate nearest neighbors

Develop an approximate vector index that sustains retrieval quality under a memory budget as the collection grows. The minimal starter leaves partitioning, compression, and candidate search to the agent. More aggressive compression can discard distinctions needed to recover true neighbors, while exploring more candidates raises query latency; the index must balance these competing effects across independent queries.

Workload.
Real SIFT vectors define separate development and evaluation workloads, with a larger evaluation collection and independent queries. The distance measure is fixed so retrieval accuracy can be checked against exact neighbors.

Evaluation and reference.
Exact neighbors establish retrieval truth, while native Faiss IVF-PQ supplies the approximate-search reference. Recall, completed-query latency, and resource measurements characterize the index's quality–cost trade-off.

Task 28. Order-to-Cash reconciliation

Maintain a correct and auditable ledger across interacting order, invoice, payment, fulfillment, refund, and governance events. The starter handles basic first-version cases, leaving revision-aware reconciliation to the agent. Late or withdrawn evidence can invalidate earlier conclusions and their dependents, so persistent state, ledger values, and provenance must be updated together without leaving stale relationships or inconsistent replay results.

Workload.
WideWorldImporters-derived workloads deliver successive checkpoints with revisions and late evidence. Candidate state persists between checkpoints, and the system must update both ledger entries and their supporting relationships.

Evaluation and reference.
The expert-developed Order-to-Cash reconciliation system provides the reference for ledger and audit outcomes. Quality and computational cost are evaluated across checkpoints, with replay and evidence-withdrawal tests checking consistency of dependent outputs.

Task 29. SQL cardinality and planning

Choose efficient join plans from incomplete statistical summaries when estimation errors can change which intermediate results appear economical. The independence-and-greedy starter leaves improved estimation and combinatorial plan search to the agent. The planner must reason about coupled predicates and alternative join structures without probing the protected database, while ensuring that additional planning effort produces worthwhile execution gains.

Workload.
Workloads include held-out query templates and join structures from JOB and TPC-H, plus a separate TPC-DS database. Candidates use the provided statistics rather than probing protected databases.

Evaluation and reference.
Submitted join plans are executed and their query results verified. Cardinality error on the candidate's intermediate relation sets, query execution time, and planning cost are compared with native DuckDB.

Task 30. Flash Attention

Develop attention kernels that remain fast across packed prefill and cache-attention workloads with substantially different shapes and reuse patterns. The eager starter establishes semantics but leaves memory movement, tiling, and execution strategy to the agent. Ragged lengths, grouped heads, local windows, and different numerical formats complicate specialization; gains must extend across workload families while preserving numerical correctness.

Workload.
Separate workloads vary ragged batches, grouped and multi-query attention, asymmetric context lengths, local windows, numerical formats, and head dimensions. Fresh variants discourage reliance on a single layout.

Evaluation and reference.
Numerical checks verify the specified attention operation. Packed-prefill throughput and cache-attention latency are compared with unmodified FlashAttention-2 across balanced workload families, after compilation and warmup.

C Construction and Evaluation Implementation

The evaluation runtime connects agent development, task execution, and scoring through AlphaDiana (Zhou et al., 2026b). This appendix describes the executable task configuration and the submission and measurement procedures underlying Section 3.4. Task-specific design choices and interfaces appear in Sections A and B.

C.1 Construction implementation

Executable task contract. Each task specifies the starting implementation, editable components, allowed dependencies, input and output schemas, and required lifecycle operations. The evaluator implements these operations for both the submitted system and the expert reference. The contract also fixes the workload splits, validity checks, resource limits, and score configuration. This connects the scientific objective to an executable development and evaluation interface.

Agent-facing instructions. Task prompts are assembled from structured descriptions of the objective, implementation interface, available resources, and evaluation feedback. These instructions identify the system to be delivered and the operations used to test it. Task-specific components provide domain guidance, while the shared runtime supplies submission and feedback handling.

Workload and reference integration. The task adapter binds workload generation or dataset partitions to the corresponding reference implementation and measurement settings. Candidate and reference executions use matching input semantics, metric units, and aggregation rules. Where tasks contain multiple regimes or families, those groups define the aggregation structure. Concrete workload and reference choices are described in Sections A and B.

C.2 Submission and feedback

Agent configuration. All six models use the OpenCode harness with subagent delegation disabled. The configured model output limit is 128,000 tokens, with no experiment-level override of sampling parameters or reasoning effort. Each round allows 45 minutes of agent execution; the current system is submitted when that budget expires. Task-specific evaluation budgets apply separately.

Episode initialization. The runner resolves the starting system, task instructions, dependencies, workload definitions, reference configuration, and development budget. It initializes the persistent agent workspace and starts the task's development evaluation service. The agent receives the task interface, permitted tools, and a client for requesting evaluation. Task data and reference artifacts are supplied to the evaluation service according to their roles in the protocol.

Evaluation of a submitted system. At each round boundary, the runner copies the submitted implementation and associated artifacts into an immutable snapshot. The task adapter loads that snapshot in a separate execution environment and invokes the required lifecycle, such as fitting and prediction, index construction and search, sequential control, or numerical solving. It collects outputs, applies the task's validity checks, and computes raw quality and cost measurements. Development feedback contains the validity result, task-defined metrics, scores, and execution diagnostics needed for the next revision. The workspace and conversation persist as the agent continues development.

Final evaluation. At termination, the runner submits the final frozen system to the held-out evaluation procedure. This invocation supplies the final workload and the corresponding reference measurements. The resulting record binds the submitted snapshot to its validity decision, raw measurements, and scores. The retained record links the final implementation to its measured outcome.

C.3 Measurement boundaries

Measuring the target operation. Task adapters define the beginning and end of the operation being compared. For a persistent search service, queries are timed after index loading. For attention

kernels, execution is measured after compilation and warmup. For a numerical solver, measurement includes solving and returning the requested state or trajectory. For image compression, encoding and fresh-process decoding contribute to the measured codec cost. The corresponding reference uses the task's specified measurement boundary.

Work, time, and memory. Resource measures follow the development objective. Hartree–Fock uses integral-oracle calls to quantify solver work; grid restoration measures the quality available over the planning interval; other tasks measure execution time, memory, or artifact size. Measurements are obtained by the task's execution wrapper and evaluator. Task-specific repetitions, warmup, and aggregation establish the measurement unit, whether an individual query, a complete workload, or a persistent interaction sequence.

Reference execution. Reference values are obtained from executable expert implementations or from retained measurements bound to the task workload and evaluation configuration. The record identifies the reference implementation and the workload used for comparison. Numerical checks and lifecycle-specific reference measurements are described by discipline in Section A.

C.4 Task metrics and score computation

Preparing domain measurements. The task adapter converts outputs into domain measures before applying the common normalization in Section 3.5. This step can include accuracy utilities, normalization by physical scales, or conversion to the units used for comparison. For example, audio signal-to-distortion improvement is converted from decibels to a linear power ratio, and image distortion is aggregated in its linear error representation. Where specified, the scorer applies the same affine transform and positive numerical floor to candidate and reference measurements. Metric directions, transforms, and weights are part of the task configuration.

Workload aggregation. Measurements are combined according to the task's workload structure. Family-balanced tasks first aggregate within a physical regime, molecular family, or data source and then across those groups. This gives the intended coverage units their declared weight. The task adapter also resolves case-level validity and quality contributions before passing the resulting metric vector to the shared scorer. Cross-task macro-averaging is described with the final-system analysis in Section D.2.

Reference-relative scoring. The scorer compares the submitted system's measurements with the expert-developed reference under the same workload, using the normalization in Section 3.5. For each axis $q \in \mathcal{Q}$, normalized metric scores are combined using task-specific nonnegative weights w_{qk}:

$$S_q = \prod_{k \in \mathcal{K}_q} s_k^{w_{qk}}, \qquad \sum_{k \in \mathcal{K}_q} w_{qk} = 1.$$

The Overall score combines the two axes using nonnegative weights W_{perf} and W_{eff}:

$$S_{\mathrm{overall}} = S_{\mathrm{perf}}^{W_{\mathrm{perf}}} S_{\mathrm{eff}}^{W_{\mathrm{eff}}}, \qquad W_{\mathrm{perf}} + W_{\mathrm{eff}} = 1.$$

Metric weights, axis weights, and the normalization parameter κ are specified by each task's scoring configuration. Invalid systems receive zero, following the validity rule in Section 3.5. The evaluation record retains the reference implementation, raw measurements, and normalized scores.

C.5 Experiment records

The runner retains the resolved configuration, starting and submitted systems, development feedback, execution logs, and final evaluation result. Each measurement record identifies its task, workload, reference, and scoring configuration. Content identifiers connect the evaluated implementation to the saved snapshot, and the score record retains raw measurements and intermediate normalized values. These records support replay of a submitted system, inspection of metric trade-offs, and the trajectory analyses reported in Section E.

D FULL TASK-LEVEL AGENT RESULTS

Table 7 reports final held-out Overall scores for all 30 tasks and six models. Table 8 provides the corresponding performance and efficiency scores. Each model–task pair contributes one final system. Means and medians include all tasks; ranks are computed within each task using unrounded scores, with average ranks for ties.

Table 7: **Final held-out Overall scores on SYSLOOPBENCH.** Bold and underlined values mark the best and second-best results in each row. † denotes an invalid result; other displayed zeros are rounded valid scores.

Task	GPT-5.6 Sol	Claude Opus 4.8	GLM-5.3	Kimi K3	DeepSeek V4 Pro	Qwen3.8 Max
Mathematics						
Lean proof search	0.00	0.00	2.43	2.88	0.50	**42.37**
Mesh simplification	72.27	14.94	47.39	**82.96**	79.13	26.63
Inventory replenishment	98.92	0.00†	97.09	65.24	73.86	**103.03**
Portfolio construction	**98.95**	46.94	44.33	0.00†	0.00†	72.02
Lattice reduction	**25.84**	6.91	0.17	1.02	14.41	13.68
Physics						
Astronomical catalog association	**104.49**	100.81	94.12	93.42	91.24	102.55
Orbit determination	107.04	53.96	2.24	113.11	98.62	**120.71**
Quantum spin ground-state search	**114.76**	114.32	114.42	113.29	114.73	114.39
N-body dynamics	113.80	**114.51**	113.38	113.20	114.20	114.29
Radiative transfer	**111.68**	89.07	52.72	110.34	100.29	96.38
Biology						
Enzyme function prediction	0.00†	51.73	**73.02**	59.86	52.01	64.04
Peptide-spectrum matching	71.01	0.00†	0.00†	**80.23**	42.34	60.35
RNA secondary structure	39.82	27.73	50.54	**68.17**	48.11	40.40
Multiple sequence alignment	80.95	28.56	83.68	**83.86**	58.01	40.36
Phylogenetic tree inference	110.29	102.71	79.50	**111.51**	87.23	107.60
Chemistry						
Hartree–Fock self-consistent field	0.00†	**62.28**	3.14	0.00†	42.51	45.45
Molecular ADMET prediction	103.15	103.56	105.89	105.00	108.73	**110.02**
Flexible macrocycle docking	38.55	40.97	**47.92**	38.74	40.97	23.19
CCSD amplitude solver	113.68	109.80	113.66	**114.60**	110.72	111.39
Perturbative triples correction	102.75	56.39	24.31	**111.18**	103.55	26.75
Engineering						
Distribution-grid restoration	20.94	0.95	0.95	0.95	**74.92**	0.95
Wind turbine control	96.72	66.96	94.03	**99.01**	88.69	53.24
Learned image compression	110.78	36.99	120.72	103.20	138.34	**139.26**
Road map matching	113.12	112.19	86.40	103.29	**113.54**	112.98
Blind audio separation	18.14	18.73	**36.34**	25.71	11.99	12.70
Computer Science						
Low-bit CPU kernels	71.46	77.02	90.20	**120.50**	97.89	102.98
Approximate nearest neighbors	58.28	33.72	81.85	**82.40**	59.86	0.00†
Order-to-Cash reconciliation	60.68	9.47	68.58	**77.65**	69.33	73.80
SQL cardinality and planning	**56.61**	6.80	32.74	6.11	17.38	1.99
Flash Attention	79.84	56.67	92.31	97.10	67.59	**104.12**
Mean	**73.15**	51.49	61.80	72.82	70.69	67.92
Median	80.40	49.34	70.80	**83.41**	74.39	68.03
Mean rank ↓	3.22	4.48	3.65	**3.03**	3.38	3.23
Valid (/30)	28	28	**29**	28	**29**	**29**
Above reference (/30)	11	7	5	11	8	**12**

D.1 PERFORMANCE AND EFFICIENCY BY TASK

The two axes expose the quality and resource trade-offs underlying each Overall score. Scores follow the task-specific metrics and expert references in Sections B and C.4.

Table 8: **Final performance (P) and efficiency (E) scores on SYSLOOPBENCH.** Higher is better on both axes. † denotes an invalid system.

Axis	GPT-5.6 Sol	Claude Opus 4.8	GLM-5.3	Kimi K3	DeepSeek V4 Pro	Qwen3.8 Max
Lean proof search						
P	0.00	0.00	1.98	3.49	0.22	61.54
E	73.14	84.38	5.48	1.32	12.25	9.52
Mesh simplification						
P	99.56	12.09	57.44	103.82	99.99	27.07
E	20.06	34.76	21.95	33.83	31.04	24.93
Inventory replenishment						
P	93.15	0.00†	91.25	55.15	63.82	100.20
E	125.81	0.00†	124.45	127.81	132.50	115.21
Portfolio construction						
P	84.13	33.19	35.11	0.00†	0.00†	68.55
E	189.32	187.85	112.68	0.00†	0.00†	87.80
Lattice reduction						
P	22.36	6.43	0.03	1.46	18.15	9.87
E	46.09	9.21	186.86	0.24	5.71	50.46
Astronomical catalog association						
P	100.75	97.61	97.70	96.77	98.48	99.83
E	113.77	108.70	86.27	86.05	76.34	109.18
Orbit determination						
P	103.33	56.60	0.51	105.71	93.79	105.68
E	119.00	46.74	191.72	138.60	114.66	179.90
Quantum spin ground-state search						
P	99.97	99.96	99.98	99.98	99.95	99.97
E	199.34	195.55	196.21	186.75	199.22	196.08
N-body dynamics						
P	99.60	100.00	100.00	100.00	99.71	99.99
E	194.00	196.91	187.40	185.92	196.50	195.13
Radiative transfer						
P	99.59	100.00	40.34	99.99	100.00	100.00
E	176.59	56.06	153.71	163.62	101.47	83.18
Enzyme function prediction						
P	0.00†	49.34	68.07	54.89	45.69	59.33
E	0.00†	79.24	137.24	130.55	166.97	127.36
Peptide-spectrum matching						
P	99.89	0.00†	0.00†	89.17	90.18	93.96
E	56.55	0.00†	0.00†	74.77	25.57	44.92
RNA secondary structure						
P	50.97	13.80	47.07	97.84	51.54	30.99
E	22.38	141.32	59.66	29.34	40.97	74.95

Continued on next page

Table 8 (continued)

Axis	GPT-5.6 Sol	Claude Opus 4.8	GLM-5.3	Kimi K3	DeepSeek V4 Pro	Qwen3.8 Max
Multiple sequence alignment						
P	71.12	21.62	74.74	74.91	45.18	32.88
E	135.91	86.95	131.52	131.71	157.78	91.64
Phylogenetic tree inference						
P	95.07	98.62	92.06	98.35	73.01	92.33
E	199.75	120.85	44.20	184.33	177.74	198.48
Hartree–Fock self-consistent field						
P	0.00†	38.68	0.61	0.00†	22.63	25.33
E	0.00†	189.17	145.24	0.00†	185.19	177.74
Molecular ADMET prediction						
P	100.10	98.63	98.21	100.82	99.55	98.92
E	116.28	125.90	143.13	123.52	154.71	168.36
Flexible macrocycle docking						
P	25.86	30.26	35.28	56.74	36.06	15.23
E	190.55	137.58	162.98	8.41	68.25	124.63
CCSD amplitude solver						
P	100.00	100.00	100.00	100.00	100.00	100.00
E	189.88	159.61	189.72	197.69	166.41	171.49
Perturbative triples correction						
P	100.00	100.00	16.82	100.00	100.00	16.19
E	114.54	5.70	105.95	169.90	119.03	199.54
Distribution-grid restoration						
P	85.59	0.28	0.28	0.28	64.74	0.28
E	0.08	134.31	134.29	134.28	134.31	134.29
Wind turbine control						
P	106.87	67.43	85.19	102.11	85.89	64.60
E	64.89	65.12	139.62	87.56	100.83	24.56
Learned image compression						
P	100.64	27.08	102.04	121.71	122.04	126.48
E	138.59	76.61	178.71	70.22	185.33	174.31
Road map matching						
P	101.55	98.10	91.06	88.40	99.94	99.07
E	174.17	191.93	70.03	192.59	189.20	191.10
Blind audio separation						
P	13.04	10.37	24.07	17.10	5.77	6.44
E	48.86	110.29	125.01	87.45	107.69	97.09
Low-bit CPU kernels						
P	69.62	87.84	88.35	129.89	102.44	106.21
E	79.32	45.54	97.98	89.25	81.61	91.01
Approximate nearest neighbors						
P	91.76	93.18	82.88	76.99	97.86	0.00†
E	25.08	5.11	79.99	93.46	24.02	0.00†
Order-to-Cash reconciliation						
P	58.05	7.17	66.86	76.53	68.06	72.86
E	90.32	116.01	86.26	88.58	81.85	82.84

Continued on next page

Table 8 (continued)

Axis	GPT-5.6 Sol	Claude Opus 4.8	GLM-5.3	Kimi K3	DeepSeek V4 Pro	Qwen3.8 Max
SQL cardinality and planning						
P	88.32	9.05	22.08	2.92	14.94	0.84
E	9.56	2.17	158.44	117.58	31.80	62.97
Flash Attention						
P	85.62	70.89	86.85	96.65	74.27	105.73
E	67.82	33.60	106.43	98.15	54.23	100.47

D.2 Final-system analysis

Task coverage and joint improvement. Let $P_{i,m}$ and $E_{i,m}$ be final held-out axis scores, and $v_{i,m}$ indicate validity. Figure 8(a) averages each axis over all $N = 30$ tasks, assigning zero to invalid systems. Panel (b) reports

$$C_m(\tau) = \frac{100}{N} \sum_{i=1}^{N} \mathbf{1}\{v_{i,m} = 1,\ P_{i,m} \geq \tau,\ E_{i,m} \geq 100\}.$$

For panel (c), let $\mathcal{J}_{i,m}$ contain valid systems produced by other models on the same task and evaluated against the task's expert reference. The largest joint improvement margin is

$$d_{i,m} = \max\left\{0, \max_{m' \in \mathcal{J}_{i,m}} \min(P_{i,m'} - P_{i,m}, E_{i,m'} - E_{i,m})\right\},$$

with $d_{i,m} = 0$ for an empty peer set. Panel (c) reports the fraction of 171 valid systems with $d_{i,m} \geq \delta$. Margins are measured in normalized score points; each comparison uses both axes from the same peer.

Reference surplus and shortfall. For final Overall score $h_{i,m}$, define

$$U_m = \frac{1}{N} \sum_i \max(h_{i,m} - 100, 0), \qquad L_m = \frac{1}{N} \sum_i \max(100 - h_{i,m}, 0).$$

The all-task mean is $\bar{h}_m = 100 + U_m - L_m$. Valid-only means average over valid submissions, while task-score medians include all 30 outcomes. The task-wise maximum averages the highest final score among the six evaluated models on each task.

E Evolution Trajectories and Analysis

The trajectory analysis covers 180 episodes, each with ten development rounds. Table 9 summarizes progress across rounds, and Table 11 reports all 1,800 development scores. At each round boundary, the evaluator executes an immutable snapshot of the submitted system and returns development feedback before the next round begins, following Section 3.4. Invalid submissions and evaluations that exhaust their execution budget without a valid result receive zero. The analysis uses the recorded system versions, evaluation results, and interaction logs.

E.1 Evolution statistics

Progress milestones. For episode i, let $s_{i,t}$ be the development score at round t and $b_{i,t} = \max_{u \le t} s_{i,u}$ the best score attained so far. The observed gain is $G_i = b_{i,10} - s_{i,1}$. For episodes with $G_i > 1$,

$$R_{q,i} = \min\{t : s_{i,t} \ge s_{i,1} + qG_i\}, \qquad q \in \{0.5, 0.9\}.$$

Table 2 averages these milestones over the eligible episodes: 26, 25, 25, 26, 25, and 28 in the displayed model order.

System revisions and tool use. Distinct versions count unique content fingerprints of submitted systems within an episode. Tool calls count recorded invocations across all ten rounds, including unsuccessful calls. Both statistics are averaged over the 30 episodes of each model. The identity of a system is determined by its stored implementation and artifacts.

Progress after a plateau. With patience p and score threshold ϵ, the first plateau ends at

$$\tau_i(p, \epsilon) = \min\{t : p + 1 \le t < 10,\ v_{i,t} = 1,\ b_{i,t} - b_{i,t-p} \le \epsilon\},$$

where $v_{i,t}$ denotes validity. An episode has renewed progress when $b_{i,10} - b_{i,\tau_i} > \epsilon$. The main results use $p = 2$ and $\epsilon = 1$. Table 10 reports additional choices of patience and threshold using the same trajectories.

New gains, recovery, and losses. For $t \ge 2$, define

$$g_{i,t} = b_{i,t} - b_{i,t-1}, \qquad u_{i,t} = \max(s_{i,t} - s_{i,t-1}, 0), \qquad r_{i,t} = u_{i,t} - g_{i,t}.$$

Here $g_{i,t}$ is a new gain beyond the previous best and $r_{i,t}$ is recovery of an earlier loss. The recovery share is

$$\rho = 100\, \frac{\sum_i \sum_{t=2}^{10} r_{i,t}}{\sum_i \sum_{t=2}^{10} u_{i,t}}.$$

The negative component is $\min(s_{i,t} - s_{i,t-1}, 0)$. Figure 7(b) averages each component over all 180 episodes at each round. A round interval's contribution to new gains is $\sum_i \sum_{t \in I} g_{i,t} / \sum_i G_i$.

Score drops and gain retention. A score drop at round t is measured by $b_{i,t} - s_{i,t}$. Figure 7(a) reports the fractions of episodes with a gap greater than five points at the current round or at any round up to it. The final drop is $D_i = b_{i,10} - s_{i,10}$. Table 3 reports its mean across all 30 tasks per model and the fraction with $D_i > 1$.

Held-out confirmation. New breakthroughs are episodes whose first valid development score above 100 occurs in rounds 2–10. Held-out success is the fraction of these episodes whose final system also obtains a valid held-out score above 100. The model-wise counts are 5/7, 4/6, 4/11, 9/10, 5/7, and 12/14.

E.2 Round-level results

Table 9: **Development scores and coverage by round.** Each cell reports the mean Overall score, number of valid systems, and number above 100, in that order. Counts are out of 30 tasks.

Round	GPT-5.6 Sol	Claude Opus 4.8	GLM-5.3	Kimi K3	DeepSeek V4 Pro	Qwen3.8 Max
1	58.32 / 25 / 8	32.39 / 24 / 3	34.08 / 22 / 2	44.80 / 24 / 3	37.10 / 22 / 2	20.94 / 23 / 0
2	70.43 / 27 / 10	41.94 / 24 / 2	50.50 / 27 / 4	52.50 / 24 / 7	40.25 / 21 / 3	34.27 / 25 / 3
3	73.44 / 27 / 11	53.57 / 25 / 8	60.39 / 26 / 6	56.59 / 26 / 8	60.43 / 27 / 7	49.46 / 28 / 6
4	73.58 / 27 / 10	55.19 / 25 / 9	63.55 / 26 / 7	56.25 / 27 / 7	63.18 / 27 / 6	56.23 / 27 / 6
5	84.82 / 29 / 13	55.32 / 25 / 9	67.49 / 26 / 8	63.79 / 28 / 7	67.95 / 27 / 7	58.98 / 27 / 10
6	85.43 / 29 / 14	59.24 / 26 / 9	66.64 / 27 / 7	69.10 / 27 / 11	72.10 / 28 / 8	62.60 / 28 / 10
7	86.35 / 29 / 14	61.21 / 28 / 8	69.26 / 27 / 7	71.46 / 28 / 10	73.93 / 29 / 8	62.01 / 28 / 8
8	85.00 / 28 / 14	59.86 / 27 / 9	70.70 / 28 / 8	67.06 / 26 / 10	75.98 / 29 / 8	73.24 / 29 / 12
9	83.81 / 28 / 14	59.12 / 27 / 9	69.10 / 26 / 6	72.18 / 26 / 11	77.54 / 29 / 9	76.06 / 29 / 13
10	88.14 / 29 / 15	63.76 / 28 / 9	77.14 / 30 / 10	79.88 / 29 / 12	78.54 / 29 / 9	75.89 / 29 / 13

Table 10: **Renewed progress after plateaus.** Eligible episodes contain a plateau of length p with gain at most ϵ; renewed progress exceeds ϵ after that plateau.

Patience p	Threshold ϵ	Eligible episodes	Renewed progress
2	0.5	153	81 (52.9%)
2	1	160	78 (48.8%)
2	2	165	74 (44.8%)
2	5	170	63 (37.1%)
3	1	134	48 (35.8%)

Table 11: **Complete development trajectories on SysLoopBench.** Columns give Overall scores at rounds 1–10. Bold marks the highest valid score within each episode; † denotes an invalid result.

Model	1	2	3	4	5	6	7	8	9	10
Mathematics — Lean proof search										
GPT-5.6 Sol	55.22	53.21	43.95	58.75	62.73	62.71	66.95	**68.19**	68.04	68.03
Claude Opus 4.8	42.55	42.87	42.77	42.84	**43.39**	42.92	42.79	42.97	43.25	43.04
GLM-5.3	0.00	0.00	62.25	**63.00**	62.26	56.74	56.64	51.85	52.10	54.10
Kimi K3	**67.08**	66.84	63.73	66.47	66.20	66.62	64.42	63.48	66.81	66.34
DeepSeek V4 Pro	**63.14**	62.37	62.11	61.93	62.81	61.50	62.91	62.61	62.86	62.41
Qwen3.8 Max	42.75	29.75	29.39	29.56	29.26	60.52	60.41	61.68	66.77	**67.02**
Mathematics — Mesh simplification										
GPT-5.6 Sol	36.10	42.86	68.95	73.06	73.04	77.22	77.04	**77.41**	0.00†	77.38
Claude Opus 4.8	0.00†	0.00†	**33.52**	31.40	30.29	31.32	31.36	8.84	8.86	15.39
GLM-5.3	48.68	48.41	48.34	39.26	48.69	**49.29**	42.38	49.22	46.68	49.26
Kimi K3	86.47	74.30	74.11	73.85	87.54	87.51	**88.66**	88.49	84.95	85.92
DeepSeek V4 Pro	0.00†	0.00†	66.42	57.21	76.92	**79.12**	78.85	78.88	78.95	78.66
Qwen3.8 Max	17.79	18.05	19.02	20.49	30.23	31.89	33.49	34.39	**34.47**	33.56
Mathematics — Inventory replenishment										
GPT-5.6 Sol	0.00†	64.76	81.67	79.65	86.82	87.71	87.82	92.77	94.93	**96.00**
Claude Opus 4.8	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†
GLM-5.3	0.67	56.91	53.07	54.21	56.54	56.60	96.52	50.43	96.46	**100.00**
Kimi K3	44.77	46.66	46.91	43.56	50.12	71.95	72.72	**81.20**	77.54	78.45
DeepSeek V4 Pro	0.00†	0.00†	0.00†	6.32	68.30	75.53	73.45	74.30	75.09	**77.21**
Qwen3.8 Max	0.00†	0.00†	100.92	102.63	**104.49**	102.90	102.93	102.56	102.74	104.24
Mathematics — Portfolio construction										
GPT-5.6 Sol	118.68	80.71	117.53	120.73	120.96	121.19	122.33	123.11	124.07	**124.12**
Claude Opus 4.8	56.57	56.45	56.57	69.30	69.24	69.17	**69.57**	69.24	69.24	69.16
GLM-5.3	0.00†	48.31	47.01	48.61	48.53	48.03	55.25	**57.62**	0.00†	57.53
Kimi K3	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†
DeepSeek V4 Pro	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†
Qwen3.8 Max	0.00†	49.85	64.05	70.35	71.06	69.80	**73.11**	72.68	68.52	71.55
Mathematics — Lattice reduction										
GPT-5.6 Sol	28.94	75.58	87.50	95.05	94.32	95.44	95.17	99.71	102.73	**103.25**
Claude Opus 4.8	27.84	25.88	**28.54**	8.45	9.98	19.27	22.84	21.40	0.78	16.28
GLM-5.3	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	**0.78**
Kimi K3	0.00†	0.00†	0.20	0.21	0.21	0.20	0.20	0.15	**3.52**	2.99
DeepSeek V4 Pro	26.99	26.65	26.97	26.51	26.80	27.19	26.86	**27.23**	26.98	27.05
Qwen3.8 Max	0.00†	0.77	16.63	16.61	4.94	4.47	6.93	25.72	17.16	**25.72**
Physics — Astronomical catalog association										
GPT-5.6 Sol	103.81	101.71	104.27	**104.71**	104.41	104.63	104.70	104.38	76.12	104.58
Claude Opus 4.8	0.00†	97.15	101.27	**101.52**	100.86	101.31	101.34	101.31	101.31	101.24
GLM-5.3	**97.97**	95.69	75.45	91.27	95.25	95.82	95.80	95.12	95.90	95.97
Kimi K3	88.79	82.09	90.05	90.09	89.46	93.67	93.60	93.61	**93.70**	93.68
DeepSeek V4 Pro	83.62	87.13	86.74	71.90	82.87	87.31	87.47	91.20	91.16	**91.28**
Qwen3.8 Max	87.39	82.53	101.62	101.40	101.54	102.45	93.79	102.22	102.26	**102.53**
Physics — Orbit determination										
GPT-5.6 Sol	52.46	73.34	87.35	86.93	101.10	102.37	104.37	104.29	104.16	**104.87**
Claude Opus 4.8	36.81	36.69	36.73	36.74	36.72	41.54	41.47	**53.91**	53.76	52.96
GLM-5.3	0.00	0.00	69.88	70.04	70.05	69.91	**93.27**	57.58	6.46	6.46
Kimi K3	5.51	55.67	69.72	0.04	68.18	100.27	100.09	102.86	102.52	**109.44**
DeepSeek V4 Pro	81.23	0.01	**95.71**	88.11	88.36	88.61	88.47	88.86	87.66	88.45
Qwen3.8 Max	5.54	0.00	77.15	78.25	113.22	115.68	115.72	115.64	**115.98**	115.74

Continued on next page

Table 11 (continued)

Model	1	2	3	4	5	6	7	8	9	10
Physics — Quantum spin ground-state search										
GPT-5.6 Sol	109.18	114.49	114.63	114.70	114.15	114.71	114.72	114.71	114.72	**114.73**
Claude Opus 4.8	105.24	**114.43**	114.10	114.09	114.10	113.44	114.11	114.10	114.11	114.08
GLM-5.3	99.68	109.49	109.28	109.36	111.45	96.63	105.03	113.09	114.36	**114.39**
Kimi K3	79.42	113.22	113.96	**114.63**	114.08	112.61	112.58	113.21	112.95	113.09
DeepSeek V4 Pro	0.00†	110.42	113.17	113.36	113.25	113.21	114.54	114.53	114.66	**114.69**
Qwen3.8 Max	0.01	0.01	0.01	0.00	1.46	1.40	3.95	114.35	**114.35**	114.34
Physics — N-body dynamics										
GPT-5.6 Sol	114.02	113.25	113.37	113.74	113.98	113.62	113.96	113.96	114.23	**114.35**
Claude Opus 4.8	98.73	96.81	114.31	114.30	114.46	114.41	114.40	114.47	114.47	**114.52**
GLM-5.3	102.85	102.81	99.64	102.39	102.88	101.73	107.07	106.67	108.05	**113.48**
Kimi K3	101.66	101.67	102.20	101.85	102.14	105.63	107.77	109.35	111.41	**113.56**
DeepSeek V4 Pro	114.21	114.35	114.33	114.16	114.32	114.34	114.33	**114.35**	114.35	114.34
Qwen3.8 Max	0.00	100.51	108.39	108.81	108.97	109.31	112.42	114.17	**114.37**	114.23
Physics — Radiative transfer										
GPT-5.6 Sol	104.80	108.72	107.91	111.17	111.62	111.86	**112.28**	111.73	109.99	112.19
Claude Opus 4.8	92.46	80.75	92.95	90.94	90.53	**93.46**	92.38	91.71	92.00	91.84
GLM-5.3	0.10	99.30	95.03	104.53	104.78	108.94	110.13	111.05	**111.21**	111.14
Kimi K3	84.19	110.69	111.26	111.42	108.86	110.19	109.35	**111.99**	111.98	111.53
DeepSeek V4 Pro	94.81	99.61	102.31	97.99	98.85	101.56	102.61	101.75	102.17	**102.62**
Qwen3.8 Max	0.00	0.00	0.00	88.29	**100.70**	90.06	24.29	98.69	97.23	100.49
Biology — Enzyme function prediction										
GPT-5.6 Sol	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†
Claude Opus 4.8	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	55.20	55.19	55.21	**55.26**
GLM-5.3	64.01	66.19	64.14	72.88	72.58	72.59	**74.49**	73.92	72.61	72.58
Kimi K3	0.00†	0.00†	3.02	24.08	24.07	**65.79**	33.65	0.00†	65.78	65.79
DeepSeek V4 Pro	0.00†	**54.57**	35.84	0.00†	0.00†	54.57	54.57	54.55	48.08	54.57
Qwen3.8 Max	52.72	55.01	54.86	54.74	61.31	61.28	62.65	62.64	62.69	**69.06**
Biology — Peptide-spectrum matching										
GPT-5.6 Sol	0.00†	0.00†	56.70	11.73	60.64	63.66	64.47	68.48	70.18	**70.22**
Claude Opus 4.8	**0.04**	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†
GLM-5.3	57.29	58.31	62.49	**63.19**	60.99	57.36	0.00†	60.72	62.33	61.42
Kimi K3	0.00†	0.02	62.60	69.98	75.61	72.85	75.67	75.86	**79.81**	79.76
DeepSeek V4 Pro	31.31	0.00†	38.26	28.38	38.47	38.78	38.42	38.51	41.18	**41.38**
Qwen3.8 Max	0.04	0.00†	0.03	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	**58.88**
Biology — RNA secondary structure										
GPT-5.6 Sol	19.49	28.57	28.51	28.68	28.59	30.72	36.54	0.00†	37.89	**40.92**
Claude Opus 4.8	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	23.93	**27.48**	19.80	27.36
GLM-5.3	23.85	0.00†	0.00†	50.64	**50.82**	50.69	50.79	50.69	50.60	50.61
Kimi K3	23.81	23.99	23.78	23.75	23.82	0.00†	0.00†	0.00†	67.78	**67.81**
DeepSeek V4 Pro	23.75	0.00†	0.00†	0.00†	0.00†	0.00†	34.87	33.37	44.30	**45.21**
Qwen3.8 Max	23.84	23.84	34.13	34.84	36.28	35.83	38.88	39.07	38.13	**40.75**
Biology — Multiple sequence alignment										
GPT-5.6 Sol	65.17	82.09	86.99	85.15	94.86	91.14	92.90	94.48	**95.00**	94.92
Claude Opus 4.8	13.61	**42.94**	38.38	31.22	24.62	24.45	28.77	40.98	42.81	42.60
GLM-5.3	85.79	57.34	14.30	87.08	97.18	94.67	89.77	91.62	99.19	**100.75**
Kimi K3	68.65	62.94	68.80	80.81	**82.49**	72.69	79.37	50.92	50.77	81.19
DeepSeek V4 Pro	20.78	20.75	35.60	51.41	80.79	80.59	81.94	82.09	**84.01**	84.00
Qwen3.8 Max	15.67	9.58	9.59	46.97	45.72	47.51	37.92	51.16	57.37	**58.20**

Continued on next page

Table 11 (continued)

Model	1	2	3	4	5	6	7	8	9	10
Biology — Phylogenetic tree inference										
GPT-5.6 Sol	109.07	110.26	109.23	111.04	110.99	111.03	111.04	111.04	111.04	**111.04**
Claude Opus 4.8	106.44	80.48	108.06	108.20	108.83	**109.72**	107.87	108.09	108.58	109.16
GLM-5.3	97.87	52.74	**106.25**	87.60	80.94	105.82	67.49	66.98	72.53	57.16
Kimi K3	98.26	77.56	104.51	107.78	107.77	108.54	109.68	110.43	**111.05**	110.54
DeepSeek V4 Pro	0.00†	59.23	76.78	57.29	76.54	83.12	83.15	84.97	85.11	**92.66**
Qwen3.8 Max	97.73	80.27	80.28	80.32	**106.49**	106.47	106.12	105.37	106.46	106.43
Chemistry — Hartree–Fock self-consistent field										
GPT-5.6 Sol	120.90	118.40	120.48	120.61	120.99	121.07	120.64	**121.17**	120.59	**121.17**
Claude Opus 4.8	45.03	44.57	116.49	117.79	118.55	**120.74**	120.73	120.73	120.73	120.73
GLM-5.3	3.35	3.35	20.94	20.94	0.00†	12.98	0.24	0.24	0.00†	**25.62**
Kimi K3	118.89	118.89	118.89	118.89	**120.13**	119.46	119.62	89.48	119.51	119.62
DeepSeek V4 Pro	44.62	0.00†	115.90	116.08	117.22	117.53	118.22	118.76	118.86	**118.87**
Qwen3.8 Max	0.13	0.13	117.23	114.33	115.13	116.69	116.69	117.45	**117.59**	**117.59**
Chemistry — Molecular ADMET prediction										
GPT-5.6 Sol	93.00	98.16	0.00†	0.00†	98.95	102.28	**103.59**	103.55	102.87	103.37
Claude Opus 4.8	49.41	49.42	49.41	**103.77**	103.52	103.47	103.44	103.32	103.42	103.39
GLM-5.3	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	**106.04**
Kimi K3	49.42	0.00†	0.00†	0.00†	0.00†	**106.79**	106.51	0.00†	104.73	104.55
DeepSeek V4 Pro	98.51	0.00†	107.20	107.08	106.66	108.66	109.24	109.34	107.83	**109.35**
Qwen3.8 Max	0.00†	0.00†	65.17	0.00†	107.28	101.78	109.99	107.52	**110.81**	110.79
Chemistry — Flexible macrocycle docking										
GPT-5.6 Sol	76.50	72.17	55.51	90.16	90.12	92.00	91.73	91.88	91.86	**92.12**
Claude Opus 4.8	7.95	32.76	59.65	82.37	**87.78**	75.78	87.57	87.67	87.57	87.66
GLM-5.3	0.00†	56.13	56.16	0.00†	55.37	65.93	74.90	67.46	**74.96**	74.92
Kimi K3	0.00	0.69	0.69	0.95	0.69	0.87	0.03	0.03	0.00†	**53.22**
DeepSeek V4 Pro	45.63	61.30	38.75	58.19	50.66	54.69	75.70	73.44	75.76	**75.81**
Qwen3.8 Max	0.54	60.99	65.30	**82.56**	29.05	39.31	78.12	49.63	67.11	66.35
Chemistry — CCSD amplitude solver										
GPT-5.6 Sol	98.09	104.08	107.52	108.41	109.61	110.96	110.71	111.65	112.91	**113.07**
Claude Opus 4.8	12.44	76.74	101.29	102.75	105.37	105.82	79.72	**106.32**	106.31	105.99
GLM-5.3	0.04	102.43	112.83	112.81	112.63	112.75	112.42	112.81	112.79	**112.97**
Kimi K3	104.55	104.65	104.26	103.12	111.87	114.40	114.36	**114.47**	0.00†	114.46
DeepSeek V4 Pro	107.13	95.39	106.86	99.11	104.53	106.86	108.29	**108.32**	108.22	107.83
Qwen3.8 Max	0.00†	102.48	102.19	102.41	108.00	107.72	108.72	108.22	112.26	**113.27**
Chemistry — Perturbative triples correction										
GPT-5.6 Sol	0.00†	0.00†	0.00†	93.39	100.73	100.06	102.51	102.89	**103.47**	101.63
Claude Opus 4.8	9.15	85.72	**96.49**	53.79	41.70	86.15	63.09	75.36	67.30	71.50
GLM-5.3	**28.73**	23.84	24.17	23.91	23.96	20.83	22.26	24.16	24.19	23.86
Kimi K3	0.35	86.27	86.28	70.43	85.81	102.61	98.23	107.79	106.51	**108.75**
DeepSeek V4 Pro	71.28	86.80	87.22	90.09	91.27	99.95	99.47	**99.96**	99.73	99.57
Qwen3.8 Max	29.00	29.01	**29.02**	29.02	29.02	29.02	29.02	29.01	29.02	29.01
Engineering — Distribution-grid restoration										
GPT-5.6 Sol	25.13	124.51	**130.28**	0.00†	130.28	130.28	130.28	130.28	130.28	130.28
Claude Opus 4.8	25.13	0.00†	124.51	124.51	**124.51**	124.51	124.51	124.51	124.51	124.51
GLM-5.3	**127.85**	124.51	124.51	124.51	124.51	124.51	124.51	124.51	124.51	124.51
Kimi K3	25.13	**124.51**	124.51	1.52	1.50	1.51	1.95	25.13	25.13	24.40
DeepSeek V4 Pro	1.17	1.48	25.13	25.13	25.13	25.13	25.13	94.26	122.71	**122.71**
Qwen3.8 Max	**25.13**	25.13	3.65	25.13	24.99	25.03	25.04	25.04	25.06	25.13

Continued on next page

Table 11 (continued)

Model	1	2	3	4	5	6	7	8	9	10
Engineering — Wind turbine control										
GPT-5.6 Sol	49.36	48.54	47.86	55.55	**56.17**	52.16	52.97	55.98	55.27	51.92
Claude Opus 4.8	9.31	9.42	0.00†	0.00†	0.00†	48.28	56.02	52.28	60.44	**72.00**
GLM-5.3	0.00†	51.30	63.73	61.55	101.68	101.77	100.85	102.69	98.99	**102.78**
Kimi K3	72.84	68.91	79.82	79.19	81.97	**84.54**	79.59	79.02	82.41	73.92
DeepSeek V4 Pro	64.43	61.74	48.62	61.53	66.24	73.01	**74.59**	73.51	72.38	72.49
Qwen3.8 Max	38.68	59.05	58.61	**61.85**	60.18	61.31	61.06	57.75	59.73	60.82
Engineering — Learned image compression										
GPT-5.6 Sol	73.06	107.72	109.23	107.07	107.91	108.88	114.18	112.99	115.92	**116.48**
Claude Opus 4.8	12.29	15.23	14.49	13.95	37.46	37.31	38.45	38.82	**41.15**	41.12
GLM-5.3	0.00†	98.41	101.54	109.24	120.56	122.26	122.16	122.14	123.09	**123.12**
Kimi K3	0.00†	0.00†	0.00†	93.89	109.94	116.13	114.45	115.08	115.02	**119.57**
DeepSeek V4 Pro	40.66	0.00†	110.92	134.99	131.01	131.32	136.69	138.54	138.28	**138.78**
Qwen3.8 Max	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	0.00†	132.70	136.51	**141.55**
Engineering — Road map matching										
GPT-5.6 Sol	111.26	112.46	112.91	113.63	113.71	113.70	113.66	113.71	113.71	**113.73**
Claude Opus 4.8	107.96	109.47	110.11	110.09	110.41	110.41	110.36	111.86	111.87	**111.89**
GLM-5.3	0.00†	0.00	100.28	**100.65**	100.58	44.52	83.66	92.45	92.38	93.66
Kimi K3	58.56	108.44	102.64	105.70	81.40	109.66	**109.68**	109.68	108.56	105.86
DeepSeek V4 Pro	0.00†	103.30	78.88	113.63	111.77	113.64	113.64	113.68	113.68	**113.82**
Qwen3.8 Max	96.69	110.90	110.89	112.17	112.74	110.50	112.82	112.85	**112.97**	112.96
Engineering — Blind audio separation										
GPT-5.6 Sol	12.57	12.86	17.21	**19.04**	15.44	19.00	18.51	18.98	18.37	19.03
Claude Opus 4.8	10.94	10.47	**20.20**	17.97	17.36	19.86	18.92	19.53	16.70	19.54
GLM-5.3	28.18	15.10	28.79	33.49	35.79	34.38	35.02	39.99	44.75	**45.87**
Kimi K3	**29.23**	25.91	22.60	24.45	20.66	26.86	25.04	22.33	0.00†	22.37
DeepSeek V4 Pro	**16.24**	15.31	13.09	16.13	14.05	15.77	12.48	12.57	11.75	11.24
Qwen3.8 Max	8.21	9.59	9.83	**10.69**	7.59	9.36	10.43	9.59	10.41	9.95
Computer Science — Low-bit CPU kernels										
GPT-5.6 Sol	45.45	56.04	72.65	78.22	74.02	75.86	72.09	40.27	57.33	**78.54**
Claude Opus 4.8	48.20	65.50	51.84	79.46	67.45	**80.81**	78.37	64.90	70.29	76.85
GLM-5.3	0.04	0.03	0.00†	0.00†	0.00†	0.00†	65.62	66.08	74.22	**96.69**
Kimi K3	0.03	0.11	0.04	0.00†	91.21	80.83	93.56	106.96	**118.96**	114.60
DeepSeek V4 Pro	0.03	0.03	43.40	95.78	83.76	95.97	86.31	72.44	83.98	**96.16**
Qwen3.8 Max	0.03	0.03	0.00†	81.75	78.29	83.02	84.35	80.39	**112.65**	96.67
Computer Science — Approximate nearest neighbors										
GPT-5.6 Sol	35.79	45.87	45.54	45.17	46.66	46.19	53.70	55.73	55.55	**55.84**
Claude Opus 4.8	0.01	16.62	21.85	21.89	21.77	21.81	21.68	21.91	21.88	**47.08**
GLM-5.3	43.42	44.22	56.14	61.95	68.14	73.33	73.10	72.47	72.93	**74.07**
Kimi K3	41.13	41.94	0.00†	60.31	60.48	52.53	51.74	56.54	56.62	**67.12**
DeepSeek V4 Pro	33.75	0.00†	39.66	54.62	57.12	57.75	57.08	57.63	**57.81**	56.98
Qwen3.8 Max	22.67	0.00†	55.13	54.93	0.00†	50.67	54.24	**77.82**	77.23	0.00†
Computer Science — Order-to-Cash reconciliation										
GPT-5.6 Sol	0.00†	54.32	53.43	54.31	54.81	56.86	56.86	59.75	**61.05**	60.75
Claude Opus 4.8	0.00†	9.36	9.36	9.35	9.36	9.37	9.36	9.34	9.34	**9.37**
GLM-5.3	0.00†	59.54	61.08	60.87	61.60	68.02	68.44	**68.48**	68.45	67.87
Kimi K3	62.96	62.81	62.98	59.88	70.43	77.30	**77.63**	75.56	77.52	76.58
DeepSeek V4 Pro	0.00†	69.19	63.38	70.22	70.12	70.18	69.74	**70.46**	70.42	70.15
Qwen3.8 Max	0.00†	66.16	51.75	73.36	73.57	74.53	74.33	68.77	73.88	**74.71**

Continued on next page

Table 11 (continued)

Model	1	2	3	4	5	6	7	8	9	10
Computer Science — SQL cardinality and planning										
GPT-5.6 Sol	39.80	42.65	53.94	56.55	**66.30**	64.10	64.57	65.10	66.04	64.18
Claude Opus 4.8	7.53	9.58	5.12	9.51	**11.27**	10.61	8.81	9.52	8.02	7.57
GLM-5.3	40.03	59.75	66.46	65.36	66.36	65.81	60.92	**100.15**	76.98	99.90
Kimi K3	**32.17**	16.11	13.01	13.94	10.67	10.83	7.92	9.71	12.37	12.24
DeepSeek V4 Pro	12.94	13.40	14.10	9.84	10.72	18.33	**18.92**	18.55	18.64	18.18
Qwen3.8 Max	12.31	23.30	24.88	10.75	10.90	27.32	23.71	13.32	**30.75**	28.95
Computer Science — Flash Attention										
GPT-5.6 Sol	51.62	65.55	68.13	70.07	80.84	81.54	80.35	81.97	**85.98**	85.41
Claude Opus 4.8	46.12	48.80	58.98	59.47	60.16	61.27	**69.11**	0.00†	0.00†	60.68
GLM-5.3	72.08	80.97	87.81	87.17	90.51	87.18	89.02	90.70	96.26	**96.57**
Kimi K3	0.00†	0.00†	47.05	46.76	66.40	0.00†	95.70	98.53	97.50	**98.95**
DeepSeek V4 Pro	36.62	64.32	65.60	68.53	69.87	68.68	70.05	**70.71**	69.51	69.61
Qwen3.8 Max	51.26	91.06	94.13	94.77	96.88	102.22	99.18	106.87	**107.29**	106.12

F Extended Related Work

Agent benchmarks initially focused on whether a model could complete a single interactive task. The field now follows longer processes in which agents modify repositories, build or optimize executable systems, conduct research, accumulate reusable skills, and coordinate with one another. As these tasks have become longer and more stateful, duration, resource use, and evaluation integrity have become part of the benchmark itself.

F.1 Interactive task and workflow completion

Interactive-agent benchmarks established the basic paradigm of evaluating an agent through actions and observable consequences. AgentBench (Liu et al., 2024) brought together eight environments that exercise reasoning, decision making, and tool use over multiple turns. WebArena (Zhou et al., 2024) then grounded web-agent evaluation in self-hosted websites whose back-end state can be inspected after an episode, while OSWorld (Xie et al., 2024) extended executable state checking to open-ended tasks across real desktop applications. WorkArena (Drouin et al., 2024) focused this line on common enterprise knowledge-work tasks in ServiceNow, pairing a realistic software environment with browser-based observations and actions.

Subsequent benchmarks made the interaction itself more stateful and consequential. AppWorld (Trivedi et al., 2024) asks agents to generate and execute code against a world of interconnected applications; its state-based tests check both the requested result and unintended collateral changes. ToolSandbox (Lu et al., 2025) combines stateful tools, an on-policy user simulator, and dynamic checks for intermediate milestones as well as the final state. In τ-bench (Yao et al., 2024), success depends on tool calls, user interaction, and compliance with domain policies, and repeated-trial metrics expose the reliability of the resulting behavior. TheAgentCompany (Xu et al., 2025) broadens the setting to consequential workplace assignments that can require browsing, coding, communication, and the use of several enterprise applications.

Command-line benchmarks provide another route to realistic, executable workflows. Terminal-Bench (Merrill et al., 2026) packages hard terminal tasks in isolated environments with human-written solutions and task-specific tests. Long-Horizon-Terminal-Bench (Li et al., 2026c) extends this setting with substantially longer workflows and fine-grained subtask rewards, making partial progress visible when complete success remains rare. Across these benchmarks, the principal evaluation unit is an episode: a final answer, an environment state, a completed workflow, or a verified set of subgoals.

Relation to SysLoopBench. SysLoopBench uses executable environments, persistent workspaces, state-based checks, and bounded interaction during development. The resulting workspace is retained as a runnable implementation whose task behavior is measured again on official workloads after development.

F.2 Executable artifact development

Software and research benchmarks increasingly retain the artifact created during an episode and evaluate it directly. These artifacts range from transformed repositories and newly constructed systems to optimized implementations and verifiable research results.

F.2.1 Repository transformation and software evolution

SWE-bench (Jimenez et al., 2024) made issue resolution in real repositories a standard agentic coding task. Each instance begins from a historical repository snapshot and a GitHub issue, and the submitted patch is evaluated with tests that distinguish repaired behavior from regressions. SWE-bench Pro (Deng et al., 2025) expands this formulation to 1,865 longer, frequently cross-file problems from 41 public, held-out, and commercial repositories, adding a contamination-resistant view of enterprise software work. FeatureBench (Zhou et al., 2026a) changes the unit from an issue to a removed feature: its executable evaluation reconstructs feature-level tests and dependency relationships while retaining other features as regression obligations.

A newer group evaluates repository changes that unfold across multiple releases or rounds of requirements. SWE-EVO (Le et al., 2025) derives multi-change tasks from release histories and reports

partial progress as well as final correctness. SWE-CI (Chen et al., 2026a) follows real codebases across long-running continuous integration histories and measures how functional correctness and maintainability survive successive changes. EvoCode-Bench (Shen et al., 2026) presents evolving requirements over 5–15 rounds in one persistent workspace and reruns cumulative tests at every round, directly measuring specification retention and regression. RoadmapBench (Xu et al., 2026b) asks an agent to move a repository from a source version toward a target release containing multiple coordinated features. LoopsBench (Li et al., 2026a) instead represents a project as a dependency graph of development units: tests are released along the ready frontier, while completed units remain part of the regression suite. SWE Refactor Bench (Hong et al., 2026) reaches the whole-repository scale through build-, framework-, and language-stack migrations, combining behavioral tests with migration audits that check the intended structural change.

Together, this progression moves repository evaluation from local repair to feature development, version advancement, sequential requirement change, and whole-codebase transformation. The central artifact remains an evolved repository, but the accepted evidence has expanded from a small test delta to cumulative behavior, dependency-aware progress, regression preservation, and structural conformance.

Relation to SysLoopBench. SysLoopBench uses version-pinned codebases, persistent development state, executable regression checks, and multi-file tasks. Each task begins from a declared capability state, and official evaluation measures the completed implementation for functionality and task-specific quality–efficiency behavior.

F.2.2 Whole-system construction and behavioral reconstruction

Whole-system benchmarks ask an agent to create a complete executable artifact from a higher-level description. Commit0 (Zhao et al., 2025) gives agents an API specification, an empty implementation, static-analysis feedback, and interactive unit tests with which to construct an entire software library. NL2Repo-Bench (Ding et al., 2025) starts from natural-language requirements and an empty workspace, and evaluates whether the result is an installable, cross-module Python repository using tests derived from a reference implementation. RepoZero (Zhang et al., 2026b) also casts generation as repository reproduction, but supplies API specifications and uses black-box output equivalence—including cross-language settings—to check the reconstructed repository.

Other benchmarks vary the information channel available during construction. ProgramBench (Yang et al., 2026) provides documentation and query access to a compiled reference executable without revealing its source; agents design a replacement codebase whose end-to-end behavior is compared with the executable on fuzz-generated inputs. SpecBench (Zhao et al., 2026) supplies a natural-language specification and visible tests that isolate individual features, then evaluates the complete system on held-out compositions of those features. These designs expose complementary forms of construction evidence: conformance to a declared API, consistency across a multi-file repository, black-box behavioral equivalence, and generalization from visible components to their unseen interactions.

Relation to SysLoopBench. SysLoopBench tasks follow this pattern: a constrained starting project, declared implementation resources, and a complete executable artifact as the outcome. Each task's executable expert reference defines the target behavior and provides the comparison point for evaluating the developed system.

F.2.3 Objective-driven artifact optimization

Optimization benchmarks evaluate whether execution feedback can be converted into a measurably better artifact while preserving required behavior. KernelBench (Ouyang et al., 2025) studies this loop at the GPU-kernel level: generated kernels must match PyTorch outputs and are credited for speedups over the framework baseline. Repository-scale benchmarks make the search space substantially larger. GSO (Shetty et al., 2025) pairs real codebases with performance tests and expert optimizations; SWE-Perf (He et al., 2025) derives 140 instances from performance-improving pull requests and packages target functions, performance tests, expert patches, and executable environments; and SWE-fficiency (Ma et al., 2025) evaluates profiling and optimization on real data-science, machine-learning, and high-performance-computing workloads while retaining covering correctness tests.

Other benchmarks extend iterative optimization beyond repository-level performance patches to numerical algorithms, heuristic solvers, and heterogeneous engineering artifacts. AlgoTune (Press et al., 2025) asks agents to iteratively edit, compile, profile, and verify numerical programs, comparing valid implementations with mature scientific library routines. ALE-Bench (Imajuku et al., 2025) uses heuristic-programming problems whose continuous objectives and private contest instances support repeated algorithm-engineering experiments. AutoLab (Xu et al., 2026c) spans systems, models, CUDA, and challenge problems; each task starts from a correct but deliberately suboptimal baseline and gives the agent a fixed wall-clock budget for closed-loop refinement. The reference points across this literature include framework implementations, expert patches, established libraries, human contest performance, and task-specific operating points.

Relation to SysLoopBench. Agents iteratively revise the supplied starting system using development feedback. Final evaluation measures the frozen system on held-out workloads against the task's expert reference.

F.2.4 Research reproduction, extension, and discovery

Research-agent benchmarks cover several distinct stages of scientific work. At the foundation is the ability to run an existing research artifact. SUPER (Bogin et al., 2024) evaluates dependency setup, configuration, execution, and progress on machine-learning and natural-language-processing repositories. CORE-Bench (Siegel et al., 2024) asks agents to reproduce reported results from released code and data across multiple disciplines, using executable checks at several difficulty levels. In both cases, the benchmark scores successful execution and faithful recovery of an existing result.

A second line evaluates closed-loop experimentation against an objective. MLAgentBench (Huang et al., 2023) places agents in machine-learning repositories where they read, modify, run, and iteratively improve experimental methods. MLE-bench (Chan et al., 2025) packages 75 offline Kaggle competitions, evaluates submitted models with competition metrics, and uses the human leaderboard to make performance interpretable. MLGym (Nathani et al., 2025) supplies 13 open-ended AI research environments with baseline implementations and objective task scores. RE-Bench (Wijk et al., 2024) similarly provides open-ended research-engineering environments, but adds strong reference solutions and human expert attempts at multiple time budgets. These benchmarks make experimental iteration, compute allocation, and improvement over a known baseline directly measurable.

Paper-scale benchmarks expand the required artifact from a successful run to a complete implementation and evidence package. PaperBench (Starace et al., 2025) asks agents to construct a codebase and replication experiments from a paper, with author-developed rubrics decomposing success into thousands of assessable criteria. RExBench (Edwards et al., 2026) begins from a paper, its codebase, and an expert-specified hypothesis, and evaluates whether the agent implements the proposed extension against hidden executable success criteria. MLRC-Bench (Zhang et al., 2025) targets open machine-learning research competitions and measures how much of the gap from a provided baseline to the top human result an agent closes.

Recent suites push toward a more complete research lifecycle. AIRS-Bench (Lupidi et al., 2026) draws tasks from state-of-the-art papers and evaluates idea generation, implementation, experiment analysis, and iterative refinement without providing the papers' baseline code. ResearchGym (Garikaparthi et al., 2026) retains datasets, evaluators, and baseline implementations while withholding the published method, requiring agents to formulate hypotheses and attempt to surpass the baseline through experiments. NatureBench (Wang et al., 2026b) constructs containerized scientific problems from Nature-family papers and compares candidate methods with published state-of-the-art results. Together, these suites span repository setup, computational reproduction, metric-driven experimentation, specified extension, and open-ended method development.

Relation to SysLoopBench. Research-derived tasks in SysLoopBench use executable experiments and domain-native metrics. The evaluator measures each submitted system against the task's expert reference and reports performance and efficiency under the original scientific objective.

F.3 Skills, harnesses, and multi-agent organization

Recent benchmarks also evaluate reusable procedures, agent controllers, and team coordination.

F.3.1 Skills and lifelong transfer

Voyager (Wang et al., 2023) established an influential system pattern in which an embodied agent builds, repairs, and retrieves an executable skill library during open-ended exploration. LifelongAgentBench (Zheng et al., 2025) turns experience accumulation into an explicit benchmark by sequencing interdependent tasks across database, operating-system, and knowledge-graph environments. SkillsBench (Li et al., 2026b) isolates skill utility through paired conditions with a matched skill package and without that package, measuring its downstream effect with deterministic verifiers.

Later benchmarks focus on acquiring and evolving the reusable object. SkillCraft (Chen et al., 2026b) lets agents compose atomic tools into executable skills that can be cached and reused within and across tasks. SkillFlow (Zhang et al., 2026c) begins from an empty skill library, carries it through related task sequences, and measures whether discovering and patching skills improves future performance. SkillLearnBench (Zhong et al., 2026) compares continual skill-generation methods at three levels—the generated skill, the execution trajectory, and the task outcome. EvoAgentBench (Gao et al., 2026) extracts operational abilities from training-side traces and separately studies their encoding, routing, and uptake on held-out tasks. The progression is from demonstrating reusable procedural code, to measuring the value of a supplied skill, to evaluating whether experience can produce a library that transfers.

Relation to SysLoopBench. SysLoopBench retains the workspace and conversation throughout each development episode. This continuity allows the agent to draw on earlier revisions and execution feedback when developing the task-specific system.

F.3.2 Harness evaluation and evolution

Harness research studies the prompts, tools, memory, routing, and control flow that mediate between a policy model and an environment. Harness-Bench (Yao et al., 2026) holds tasks and budgets common while measuring how fixed harnesses change completion rates, process quality, efficiency, and failure modes across models. HarnessOpt-Bench (Ursekar et al., 2026) makes the harness itself the submission: an optimizer starts from a seed harness, spends a fixed number of evaluation calls changing its components, and is scored by normalized gain on an inaccessible test partition. Evo-Bench (Huang et al., 2026) extends the optimization horizon for a shared executable harness and tests whether its gains transfer across task suites and policy models.

Relation to SysLoopBench. All six models in our evaluation use the OpenCode harness under the configuration described in Section 4.1. The reported results measure system development by these models within this shared harness.

F.3.3 Multi-agent coordination and team execution

Multi-agent frameworks such as AutoGen (Wu et al., 2023), MetaGPT (Hong et al., 2024), and ChatDev (Qian et al., 2024) established conversation, role specialization, and software-development workflows as practical ways to compose language-model agents. Benchmarks that study coordination as a construct add controlled tasks and process evidence. LLM-Coordination (Agashe et al., 2025) evaluates environment comprehension, theory-of-mind reasoning, joint planning, and coordination with unseen partners in pure coordination games. MultiAgentBench (Zhu et al., 2025a) compares collaborative and competitive behavior under star, chain, tree, and graph topologies, using milestone-level measures in addition to final success. Collab-Overcooked (Sun et al., 2025) uses tasks that require natural-language cooperation and evaluates proactive communication, responsiveness, and trajectory efficiency. Cemri et al. (2025) further analyze traces across multi-agent systems and identify recurring failures in system design, inter-agent alignment, verification, and termination.

Relation to SysLoopBench. Our evaluation uses the OpenCode harness with subagent delegation disabled. Each agent develops its system in a persistent workspace over ten rounds, using execution feedback and development history to guide successive revisions.

F.4 Scale, validity, and reproducibility

As agent tasks grow longer, results become more sensitive to time and resource budgets and to the feedback exposed during execution. The following work studies how to calibrate these choices and make the resulting measurements reproducible.

F.4.1 Autonomy horizon and resource calibration

Step count and token count are weak proxies for the difficulty or economic scale of autonomous work. HCAST (Rein et al., 2025) grounds software tasks in completion times measured from human experts. Building on these human-duration data and shorter task sets, Kwa et al. (2025) estimate a model's 50%-success time horizon and track how that horizon changes over time. RE-Bench (Wijk et al., 2024) complements this population-level view by comparing agent and expert performance at multiple time budgets within the same open-ended research environments. More generally, Kapoor et al. (2024) argue that accuracy, reliability, and resource use jointly determine whether an agent system is useful.

Relation to SysLoopBench. SysLoopBench records wall-clock limits, verifier attempts, model calls, tool access, concurrency, and hardware conditions for each episode. Retained trajectories and intermediate snapshots connect task performance with resource use, progress, stagnation, regression, and recovery during development.

F.4.2 Protected evaluation, validity, and reproducibility

Rigorous standardized evaluation specifies the scenario, system configuration, resource regime, outcome measure, and evidence record. HELM (Liang et al., 2022) demonstrates the value of explicit scenarios, controlled adaptations, and multi-metric reporting. For agents, Kapoor et al. (2024) distinguish model evaluation from evaluation of a model–agent system and emphasize cost, robustness, holdout design, and reproducibility. The Agentic Benchmark Checklist (Zhu et al., 2025b) makes task validity and outcome validity separate review obligations and shows how environment and reward defects can change the measured result. AlphaDiana (Zhou et al., 2026b) operationalizes system-level attribution by recording the model, harness, benchmark, environment, scorer, budget, and trajectory.

Recent benchmarks test validity with executable checks. SpecBench (Zhao et al., 2026) uses visible atomic tests and hidden compositional tests to quantify whether local feedback supports complete behavior. Shao et al. (2026) catalog ways an agent can obtain a score without demonstrating the intended capability, including solution exposure, evaluator access, generator inference, and feedback manipulation. Protected workloads and negative controls can then be used to detect these paths.

Measurement reliability is especially important when the score depends on systems performance. Chen et al. (2026c) re-execute reference patches for GSO (Shetty et al., 2025), SWE-Perf (He et al., 2025), and SWE-fficiency (Ma et al., 2025) across machines and show how reference validity, timing instability, and aggregation rules can affect benchmark conclusions. KernelBench-Verified (Zhang et al., 2026a) strengthens kernel evaluation with a more realistic framework baseline, multiple hidden input distributions, and memory measurements, revealing optimizations that exploit narrow correctness tests. Wang et al. (2026a) likewise use held-out tasks and matched-budget test-time-search controls to measure harness transfer across tasks and policy models.

Relation to SysLoopBench. SysLoopBench adopts development–evaluation separation and evaluator-side reexecution from this literature. It preserves raw outputs, task-level measurements, environment identities, and failure metadata so that results can be reproduced and reanalyzed.

F.5 LANDSCAPE SUMMARY

Table 12 summarizes the main focus of each benchmark direction and how its evaluation scope has developed.

Table 12: Major directions in agent evaluation, their main focus, representative work, and field progression.

Direction	Main focus	Representative benchmarks	Field progression
Interactive task and workflow completion	Episode outcomes and environment states in executable digital settings	AgentBench (Liu et al., 2024); WebArena (Zhou et al., 2024); OSWorld (Xie et al., 2024); τ-bench (Yao et al., 2024); Long-Horizon-Terminal-Bench (Li et al., 2026c)	Diverse interactive environments → stateful web, desktop, and tool–user workflows → long tasks with progress-sensitive scoring
Repository transformation and software evolution	Patches, features, releases, and evolving codebases	SWE-bench (Jimenez et al., 2024); FeatureBench (Zhou et al., 2026a); SWE-EVO (Le et al., 2025); LoopsBench (Li et al., 2026a); SWE Refactor Bench (Hong et al., 2026)	Issue repair → feature-scale change → multi-release and dependency-structured evolution → whole-repository migration
Whole-system construction and reconstruction	Installable repositories and behaviorally complete programs	Commit0 (Zhao et al., 2025); NL2Repo-Bench (Ding et al., 2025); RepoZero (Zhang et al., 2026b); ProgramBench (Yang et al., 2026); SpecBench (Zhao et al., 2026)	API-guided completion → repository synthesis from requirements → black-box reconstruction and compositional validation
Objective-driven artifact optimization	Correct, quantitatively improved executable artifacts	KernelBench (Ouyang et al., 2025); GSO (Shetty et al., 2025); AlgoTune (Press et al., 2025); ALE-Bench (Imajuku et al., 2025); AutoLab (Xu et al., 2026c)	Kernel and repository speedups → numerical and algorithmic tuning → heterogeneous closed-loop engineering
Research reproduction, extension, and discovery	Reproduced results, experiment systems, and research methods	CORE-Bench (Siegel et al., 2024); MLE-bench (Chan et al., 2025); RE-Bench (Wijk et al., 2024); RExBench (Edwards et al., 2026); ResearchGym (Garikaparthi et al., 2026)	Executable reproduction → metric-driven research engineering → specified extensions and open-ended method discovery
Skills and lifelong transfer	Reusable procedural knowledge and executable skill libraries	Voyager (Wang et al., 2023); SkillsBench (Li et al., 2026b); SkillFlow (Zhang et al., 2026c); SkillLearnBench (Zhong et al., 2026); EvoAgentBench (Gao et al., 2026)	Reusable skill libraries → isolated measurement of skill utility → continual acquisition, routing, and transfer
Harness evaluation and evolution	Prompt–tool–memory–control programs around a model	Harness-Bench (Yao et al., 2026); HarnessOpt-Bench (Ursekar et al., 2026); Evo-Bench (Huang et al., 2026)	Fixed harness effects → budgeted seed-harness optimization → transfer across tasks and policy models
Multi-agent coordination	Communication, joint planning, topology, and team-process quality	LLM-Coordination (Agashe et al., 2025); MultiAgentBench (Zhu et al., 2025a); Collab-Overcooked (Sun et al., 2025)	Coordination games → topology-controlled collaboration → language-mediated teamwork with process-sensitive metrics

Taken together, these benchmarks show three concurrent shifts: agents operate in more realistic environments, their work products have grown from episode outcomes to persistent executable artifacts, and evaluation draws on richer evidence about correctness, quality, efficiency, resources, and provenance. Within this landscape, SYSLOOPBENCH follows the long-horizon evolution of a task-specific implementation. It connects progress over supplied starting code to attainment relative to mature executable references, with independent workloads used to assess whether the improvements persist.