Audit-Grade Harness for Agent-driven Scientific Computation Workflows

Shruti Badhwar [1]

Abstract

Coding agents are increasingly used in scientific computation, yet existing systems often trade flexibility for reproducibility, isolation, and auditability: fixed APIs and MCP-style tool wrappers limit agents to predefined functions, while general-purpose agents run arbitrary code without structured external audit. We present SciAgent, an LLM-agnostic runtime and audit-grade harness with 25+ version-pinned scientific containers, managed compute, append-only per-action provenance, and a fresh-context verifier that reads execution records and artifacts. SciAgent targets research–execution disconnection, sophisticated fabrication after fetch or computation failures, and silent degradation of scientific fidelity, and we evaluate it on paper-reproduction tasks because published numerical targets provide grounded, inspectable references while execution remains nontrivial. Across photonics, BRCA1 fitness–structure mapping, and computational fluid dynamics, Claude Code baseline and SciAgent recursive audit satisfy the same pre-specified outcome checks, but only SciAgent produces structured verification records containing supporting facts, execution discrepancy indicators, missing evidence, issues, confidence, and an aggregate verdict. All three primary SciAgent runs are VERIFIED while exposing distinct trajectory findings, and the observed photonics configurations return INSUFFICIENT under SciAgent parent-only audit and VERIFIED under SciAgent recursive audit, supporting the argument that verification is bounded by the visible audit surface. Code and documentation are available at https://github.com/sciagent-ai/sciagent-cli.

[1]sciagent.ai (Open Source Project). Correspondence to: Shruti Badhwar <shruti@sciagent.ai>.

Accepted at the AI for Science workshop (ICML 2026).

1. Introduction

LLM agents increasingly plan, code, execute, analyze, and report scientific workflows across fluid dynamics (Pandey et al., 2025; Feng et al., 2026; Yue et al., 2025), photonics (Lupoiu et al., 2025; Lu et al., 2025), biomedicine (Huang et al., 2025; Swanson et al., 2025; Qu et al., 2025), chemistry (Bran et al., 2024; Boiko et al., 2023; Wu et al., 2025; Pham et al., 2026), and materials (Szymanski et al., 2023; Kang & Kim, 2024). The reproducibility concerns that motivated FAIR data (Wilkinson et al., 2016), workflow provenance (Di Tommaso et al., 2017; Mölder et al., 2021), and reporting standards (Kapoor et al., 2024) reappear in compressed form when one agent is responsible for both execution and narration. A final report may be internally coherent while omitting what software ran, whether a remote job succeeded, which fallback path was used, or whether the claimed artifact was ever produced.

Our BRCA1 task illustrates the distinction. Claude Code baseline and SciAgent recursive audit both report a mapping rate of 1.000, and both pass the same threshold. Outcome-only evaluation therefore treats the runs as equivalent. The SciAgent record, however, preserves a scientifically relevant deviation: the intended cluster execution remained in `INIT`, and the successful computation ran locally in a pinned container. The final number remains supported, but the execution history is not the one originally planned. A reader may judge that deviation acceptable, material, or irrelevant only if it is visible.

This problem is broader than ordinary answer hallucination. Earlier deployments identified three recurring trajectory-level failures (Badhwar, 2026): research–execution disconnection, in which researched constraints do not enter the code; sophisticated fabrication, in which failed fetches or computations are replaced by plausible synthetic content; and silent degradation, in which the agent substitutes a lower-fidelity solver, mesh, environment, or resource budget without adequate prominence. A critic confined to the executor's working context may never see the evidence needed to identify these failures. More generally, LLM evaluators can inherit producer biases and self-preference (Panickssery et al., 2024; Zheng et al., 2023; Du et al., 2023). These cases motivate an external evidence boundary for scientific

verification.

We present SciAgent, an audit-grade harness designed around that boundary. The system retains the expressivity of a coding agent while adding four connected capabilities: (i) standardized access to version-pinned scientific containers; (ii) role-separated parent and child sessions with local or managed compute; (iii) recursive append-only provenance covering actions, job lifecycle, artifacts, and pre-execution intent; and (iv) a fresh-context, read-only verifier that evaluates claims from the recorded evidence rather than the executor's working memory.

Our evaluation deliberately separates three layers: post-run task acceptance, the aggregate SciAgent verifier verdict, and the structured evidence profile. Across three heterogeneous paper-reproduction tasks, both Claude Code baseline and SciAgent recursive audit pass the task-level checks, while all three primary SciAgent runs receive VERIFIED verdicts. The additional information lies in the evidence records, which distinguish a supported result with evidence gaps, a supported result with an execution-path deviation, and a result supported by a complete trajectory. We make three contributions:

- an operational definition of an auditable scientific-agent run based on execution observability, artifact traceability, recursive delegation visibility, and claim-level assessment;
- a cross-domain runtime that combines version-pinned scientific execution, managed compute, recursive provenance, and fresh-context verification; and
- an evaluation showing that equal headline outcomes can conceal materially different execution histories, and that the observed verifier conclusion differs with access to child-session evidence.

2. Failure Modes and Related Work

Figure 1 compresses three trajectory-level failure modes observed in scientific coding agents running computational workflows. The key commonality is that the failure lives in the trajectory: the final artifact can remain plausible even when the researched method, data source, or computational fidelity has changed. Detailed forensic episodes are retained in Appendix A; the taxonomy motivates the design but is not itself an evaluation of the present harness.

Scientific agents and execution. Scientific-agent systems span curated tool wrappers, multi-agent assistants, general coding agents, and autonomous laboratories. Tool-wrapped systems make actions easier to constrain and record but expose only predefined functions (Bran et al., 2024; Boiko et al., 2023). Multi-agent scientific assistants coordinate larger research sub-pipelines (Gottweis et al., 2026; Swanson et al., 2025; Huang et al., 2025), while scientific coding agents and domain systems execute arbitrary code or real solvers (Feng et al., 2026; Lupoiu et al., 2025; Lu et al., 2025). Autonomous laboratories extend the loop to physical actuation (Szymanski et al., 2023). Rather than ranking these systems on a qualitative landscape, we focus on a narrower systems question: how can an expressive coding agent leave a durable record of what actually executed?

Verification and provenance. Self-consistency, debate, LLM-as-judge, Reflexion, and chain-of-verification use additional model calls to critique generated output (Wang et al., 2022; Du et al., 2023; Zheng et al., 2023; Shinn et al., 2023; Dhuliawala et al., 2023). Scientific execution introduces a different evidence boundary: process exit states, remote-job completion, file contents, and artifact lineage are external effects, not merely text to critique. Workflow systems and standards such as W3C-PROV, Nextflow, Snakemake, FAIR, and REFORMS establish the importance of lineage and reproducibility (W3C, 2013; Di Tommaso et al., 2017; Mölder et al., 2021; Wilkinson et al., 2016; Kapoor et al., 2024). Agent benchmarks increasingly require real execution (Starace et al., 2025; Siegel et al., 2024; Tian et al., 2024; Chen et al., 2025; Jimenez et al., 2024), but generally score final artifacts or tests rather than whether every scientific claim is supported by a recursively complete trajectory. SciAgent connects workflow-level provenance to a structured verification record.

3. SciAgent System

Figure 2 presents SciAgent as two coupled systems: an LLM-agnostic runtime for doing scientific work and an audit harness for making that work inspectable. The runtime does not assume a particular provider, model family, or tool-wrapper protocol. Model identifiers, sampling settings, and role prompts are configured by role, while the agent receives a stable tool surface for files, shell commands, literature or web search, cluster lifecycle, monitoring, and artifact materialization. A scientific task specifies the source material, inputs, acceptance criteria, and intended artifacts; the parent agent plans and synthesizes; role-separated child sessions perform research, compute, and analysis.

Scientific actions pass through a registry rather than the ambient host environment. The registry maps logical capabilities to version-pinned containers, entrypoints, resource defaults, mount policy, and expected artifact locations. The current release contains 25+ services across 10+ scientific domains, including OpenFOAM and swak4Foam, RCWA and MEEP tooling, finite-element and molecular-dynamics engines, chemistry packages, bioinformatics tools, quantum-simulation environments, and scientific Python. This design preserves coding-agent flexibility: agents still write scripts,

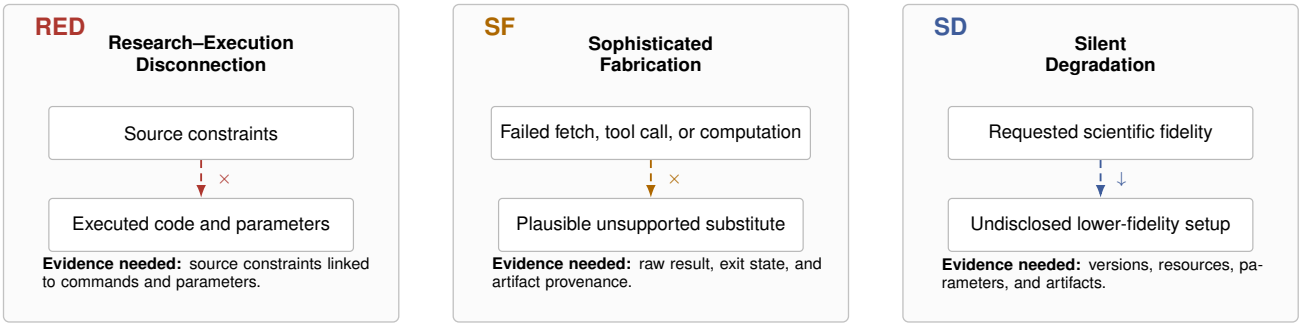


Figure 1. Three trajectory-level failure modes in scientific coding agents running computational workflows. All three can preserve a plausible final result and pass coarse outcome checks while breaking the link between scientific intent, execution, and evidence; detailed episodes appear in Appendix A.

configuration files, and analysis code, but execution occurs in declared environments whose identity is visible to later auditors.

The audit harness records what the runtime did. Parent and child sessions append per-action JSONL events for tool calls, raw tool results, compute-job launch and status transitions, declared intent, expected artifacts, materialized files, corrections, and verification outputs. These records and artifacts form a recursive evidence bundle outside the executor's working context. The verifier is spawned as a fresh-context subagent with read-only access to the permitted bundle; it receives no private conversation state from the executor and is instructed to cite record-backed support, missing evidence, execution discrepancy indicators, and issues before returning VERIFIED, REFUTED, or INSUFFICIENT. A session is *audit-grade* when an external auditor can assess each load-bearing claim from this durable record and the artifacts alone, without asking the executor for a post-hoc explanation.

3.1. LLM-Agnostic Runtime

The runtime separates model choice from scientific execution. Provider-specific details are confined to role configuration, while the execution substrate exposes the same file, shell, service, compute, monitoring, and materialization operations to the agent. This is why the primary evaluation can hold the task-model family fixed while testing the effect of the harness: SciAgent is not a new foundation model, but a runtime and evidence boundary around agent-driven computation.

3.2. Registered Scientific Services

The service registry is the contract between flexible code generation and reproducible execution. Each registered service names the container image, versioned software stack, allowed resource profile, invocation path, and artifact expectations. Version pinning identifies the software behind each action; sandbox isolation separates agent-generated code from the host; and registry inheritance records environment composition. The registry makes the selected solver stack and environment composition explicit, reducing dependence on ambient host state and making environment changes visible in the execution record.

3.3. Managed Compute and Role Isolation

The parent agent may delegate long-running or domain-specific work to research, compute, and analysis children. Resource-intensive jobs run through a managed-jobs layer (Yang et al., 2023) with cost gates, terminal-state monitoring, autostop, resumability, and artifact materialization. The main agent receives bounded summaries, while the harness preserves the child trajectories and job states that may contain the load-bearing scientific evidence.

3.4. Provenance and Fresh-Context Verification

Each session writes append-only events for tool calls, tool results, compute lifecycle, artifacts, corrections, and verification results. Compute launches also record free-form intent and expected artifacts at dispatch time, enabling post-hoc comparison between what the agent intended and what actually completed. Parent and child logs are recursively linked; the log and artifact bundle, not private chain-of-thought, are the audit objects.

The verifier is spawned in a fresh context, has read-only access, and returns a provider-neutral structured verification record. Its model identity is separately configurable, although the primary evaluation uses the same model family for executor and verifier and therefore does not claim cross-family independence. Evidence fields are not verdicts: a run can contain missing evidence or an execution discrepancy indicator while still being VERIFIED if the visible record supports the load-bearing claims. "VERIFIED" means sup-

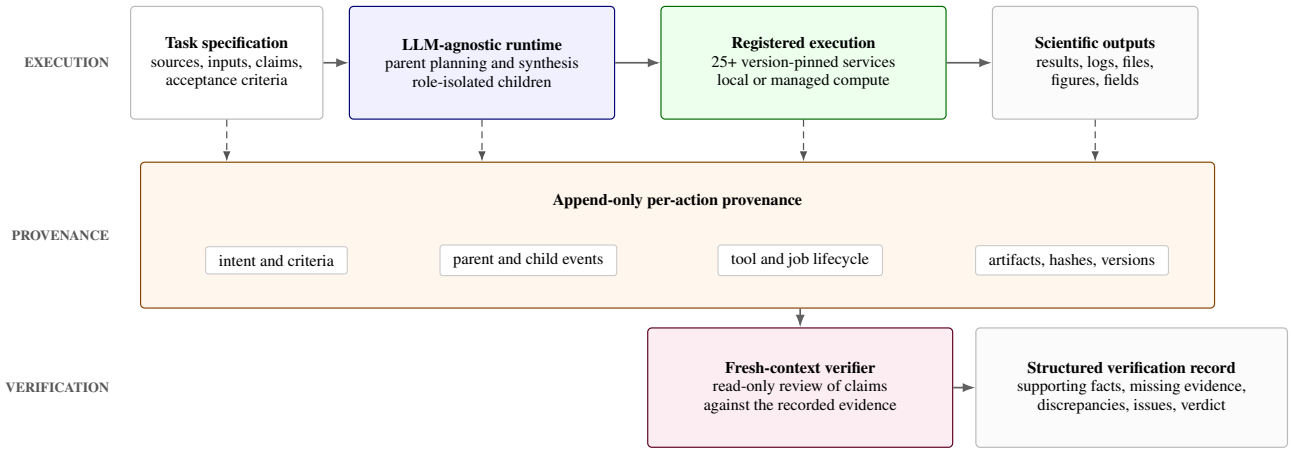


Figure 2. LLM-agnostic runtime and audit-grade harness. A role-separated parent/child runtime invokes registered version-pinned scientific services locally or on managed compute. The harness captures task criteria, per-action provenance, job lifecycle, artifacts, hashes, and environment versions; a fresh-context read-only verifier converts the recursive evidence bundle into a structured verification record.

ported by the shown evidence, not universally scientifically correct.

4. Evaluation

We ask four questions: **RQ1** Do Claude Code baseline and SciAgent recursive audit meet the same pre-specified scientific criteria? **RQ2** What does the SciAgent verifier establish beyond those task-level checks? **RQ3** How does child-session evidence access relate to the verifier's conclusion? **RQ4** What share of observed cost is attributable to verifier inference?

4.1. Tasks and Conditions

We use paper-reproduction tasks because published results provide numerical targets and artifacts against which both the final outcome and the execution record can be inspected. The published target is known, but execution remains non-trivial: the agent must interpret a paper, compose a computational workflow, run domain software, analyze the outputs, and leave checkable evidence.

Photonics reproduces a three-zone TiO_2/N-BK7 metasurface in-coupler using rigorous coupled-wave analysis, with mean-field efficiency (MFE) ≥ 0.25 as the criterion (Xiong et al., 2025). *BRCA1* parses 1,837 variants, maps them to structural coordinates using Biopython and an AlphaFold structure, and requires mapping rate ≥ 0.95 (Findlay et al., 2018; Jumper et al., 2021). *CFD* reproduces a data-center temperature distribution using OpenFOAM and requires a volume-weighted mean temperature in $[294, 298]$ K (Barestrand et al., 2023).

For each task, we compare (i) *Claude Code baseline*, using Claude Sonnet 4.6 without the SciAgent harness or an in-loop verifier, and (ii) *SciAgent recursive audit*, using Claude Sonnet 4.6 as the executor, with the fresh-context verifier enabled and recursive access to parent and child provenance. SciAgent additionally provides structured provenance, registered scientific services, and managed compute where used. Each primary cell contains one recorded run; we report observed values rather than variance estimates or significance tests. We use these descriptive condition names consistently in the main text, appendix, tables, and figures; internal run identifiers are not part of the paper terminology.

4.2. Post-run Task-level Acceptance Checks

After each completed Claude Code baseline and SciAgent recursive audit run, we apply three narrow paper-level checks. **T1 Executed** asks whether the required scientific service or computational workflow ran. **T2 Traceable** asks whether the reported headline value appears in recorded tool-result output. **T3 Accepted** asks whether that traceable value satisfies the pre-specified numerical criterion. T1–T3 are reconstructed after the run from Claude Code baseline command and tool-result records or from SciAgent parent-and-child provenance. They establish execution, traceability, and threshold acceptance for the headline scalar; they do not establish support for every scientific claim or fidelity to every intended execution detail.

4.3. SciAgent Verification

The main verification result for a SciAgent run is produced after execution by a fresh-context LLM verifier. The verifier has read-only access to the permitted provenance bundle and generated artifacts, including parent and child tool calls, raw results, compute-job lifecycle events, environment records, and artifact metadata. It records supporting facts, execution

discrepancy indicators, missing evidence, and issues; reports model-generated confidence; and assigns one aggregate verdict: **VERIFIED**, **REFUTED**, or **INSUFFICIENT**.

VERIFIED means that the visible record supports the load-bearing claims. **REFUTED** means that the visible record contradicts a load-bearing claim or establishes an invalidating execution pattern. **INSUFFICIENT** means that the available evidence supports neither conclusion. Confidence is model-reported and uncalibrated. Evidence fields are not themselves verdicts: a run may be VERIFIED while retaining non-invalidating missing evidence, an execution discrepancy indicator, or other issues. The harness also performs deterministic data and execution checks before the LLM review; these internal gates are distinct from the paper-level T1–T3 checks.

4.4. Evidence-access Diagnostic

For photonics, we compare SciAgent recursive audit with an SciAgent parent-only audit configuration. These are separate stochastic executions, not a deterministic replay, so the comparison describes an observed association between evidence access and verifier output rather than a causal estimate. Additional verifier configurations are reported in Appendix C.

4.5. Cost Accounting

We separate task work (executor inference plus scientific compute) from verifier inference. Claude Code baseline has no in-loop verifier by construction. Total cost is workload-dependent and is not treated as the primary quality metric.

5. Results

5.1. Both Systems Pass the Post-run Acceptance Checks

All six primary runs pass T1 Executed, T2 Traceable, and T3 Accepted (Table 1). For T1, the records contain the required scientific service invocation or matching computational command. For T2, each reported headline value appears in recorded tool-result output. For T3, each value satisfies its pre-specified criterion. These checks establish the same narrow outcome status for Claude Code baseline and SciAgent recursive audit; they do not establish that the execution histories or the support for additional scientific claims are equivalent.

5.2. The VERIFIED Verdicts Rest on Different Evidence Records

The SciAgent verifier returns VERIFIED for all three primary SciAgent runs. Figure 3 decomposes the corresponding structured verification records into supporting facts, execution discrepancy indicators, missing evidence, and issues. The counts describe what the verifier recorded; they are not calibrated quality scores.

Photonics. The SciAgent run reports MFE $= 0.2509$, passes T1–T3, and receives VERIFIED with confidence 0.75. The verifier records 18 supporting facts because the provenance contains the registered RCWA cluster launch, successful compute-job states, artifact materialization, and a read of the cluster-produced MFE output. It also records 3 missing evidence entries and 3 issues because substantive job log tails are truncated, per-zone efficiencies are not directly exposed in readable tool output, and the curve-shape comparison is not independently cross-validated. The verdict therefore supports the headline MFE claim while retaining gaps around intermediate and curve-shape claims.

BRCA1. The SciAgent run reports mapping rate $= 1.000$, passes T1–T3, and receives VERIFIED with confidence 0.78. The verifier records 6 supporting facts from the pinned Biopython container execution, the generated artifacts, their materialization and validation, and the recorded summary containing the mapping rate. The intended managed-compute job remained in `INIT`, while the successful computation ran locally in the version-pinned container; this appears as 1 execution discrepancy indicator, alongside 2 missing evidence entries and 2 issues. The mapping result remains supported, while the execution-path deviation remains explicit in the verification record.

CFD. The SciAgent run reports a volume-weighted temperature of 296.209 K, passes T1–T3, and receives VERIFIED with confidence 0.91. The verifier records 16 supporting facts covering the managed-compute job chain, meshing, mesh checking, solver execution, result extraction, artifact materialization, and a parent-session recomputation from the materialized CSV. It records no execution discrepancy indicators, missing evidence, or issues. This is the observed case in which the load-bearing scalar is supported by a complete recorded trajectory.

Figure 4 aligns each headline claim with its recorded evidence and trajectory finding.

5.3. Photonics Verifier Output Differs with Child-session Evidence Access

In the observed photonics configurations, SciAgent parent-only audit returns INSUFFICIENT with confidence 0.62, 6 supporting facts, 1 execution discrepancy indicator, 5 missing evidence entries, and 7 issues. SciAgent recursive audit returns VERIFIED with confidence 0.75, 18 supporting facts, no execution discrepancy indicators, 3 missing evidence entries, and 3 issues (Figure 5). The recursively visible child record contains the cluster lifecycle, terminal compute-job states, artifact materialization, and the tool-

Table 1. Scientific tasks and principal results. Both systems pass the post-run T1 Executed, T2 Traceable, and T3 Accepted checks. Verdict, confidence, and evidence item columns apply only to SciAgent recursive audit because Claude Code baseline has no SciAgent verifier. Evidence items sum the structured fields shown in Figure 3; confidence is model-reported and uncalibrated.

Task	Pre-specified acceptance criterion	Claude Code baseline	SciAgent recursive audit	Verdict	Conf.	Evidence items
Photonics optimization	Mean-field efficiency ≥ 0.25	0.2504	0.2509	**VERIFIED**	0.75	24
BRCA1 fitness–structure mapping	Mutation mapping rate ≥ 0.95	1.000	1.000	**VERIFIED**	0.78	11
Computational fluid dynamics (CFD)	Volume-weighted temperature in $[294, 298]$ K	295.333 K	296.209 K	**VERIFIED**	0.91	16

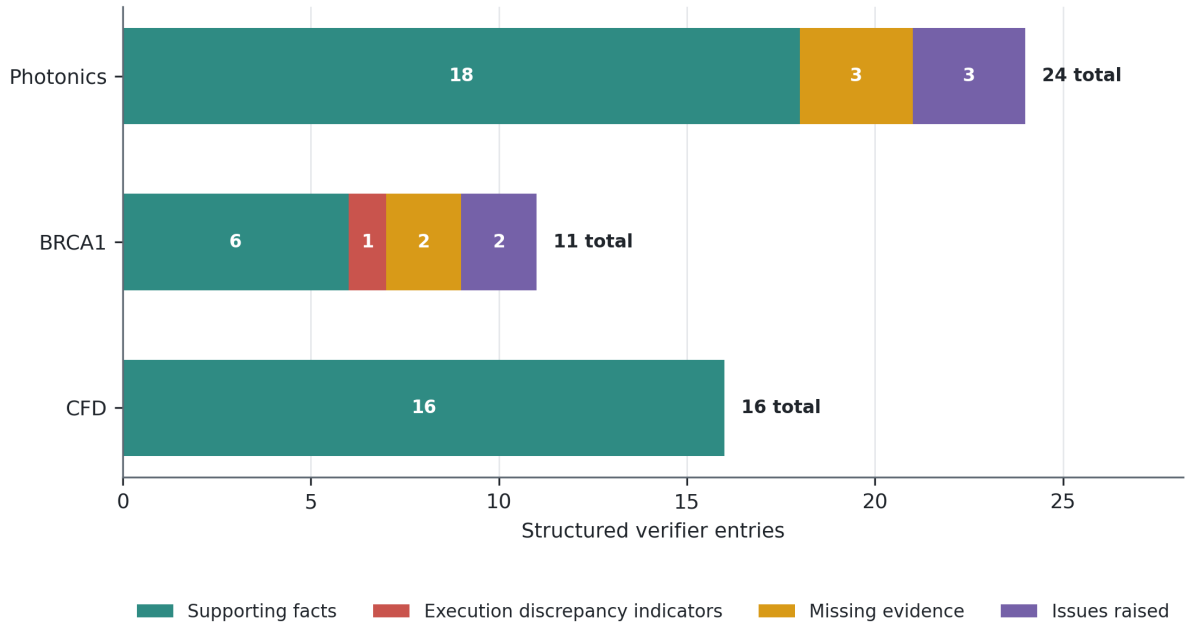


Figure 3. Structured verification-record fields for the three SciAgent recursive audit runs. All three receive a VERIFIED verdict, but their evidence profiles differ. Counts describe report content and are not normalized quality scores.

result read of the load-bearing MFE output; those records are absent from the parent-only verifier snapshot. The difference is consistent with the paper's audit-surface argument: the verifier conclusion is conditioned on the evidence available in its audit surface. Because the configurations are separate executions, the result does not isolate child-session access as the sole cause of the verdict difference.

5.4. Verifier Inference Is a Small Share of Observed SciAgent Cost

Verifier inference accounts for 4.2% of total SciAgent cost in photonics, 9.9% in BRCA1, and 1.7% in CFD. The cost decomposition shows that executor inference and scientific compute account for most observed SciAgent cost in each task. Total system cost is not uniformly lower: SciAgent recursive audit costs more than Claude Code baseline in photonics and BRCA1 and less in CFD. The result therefore isolates the verifier call as a small component of these runs without implying that the complete harness is uniformly cheaper or faster. The operational accounting is reported in Appendix E. We do not interpret the reported CFD wall-time comparison as a speed result because the Claude Code baseline trace records 181 tool calls in 0.76 minutes.

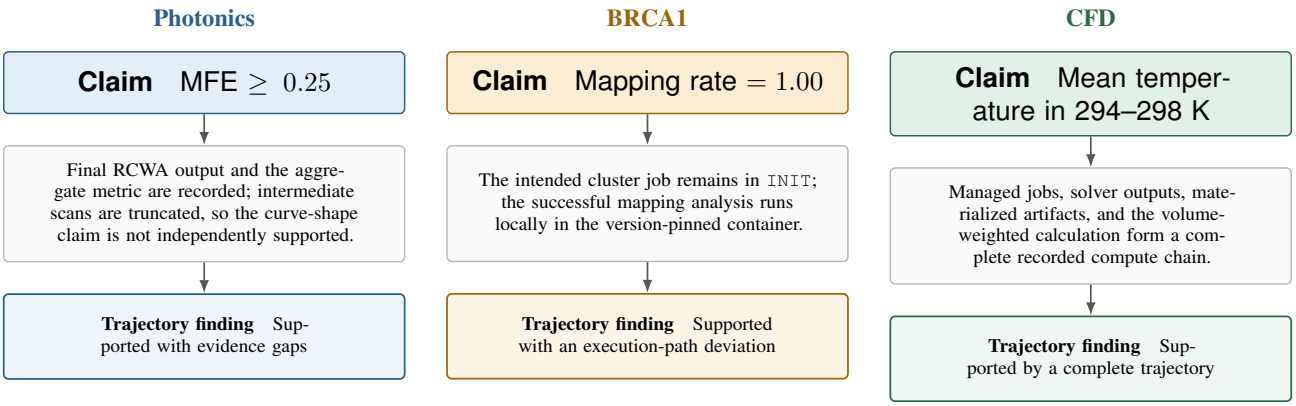


Figure 4. All three primary SciAgent recursive audit runs pass the post-run acceptance checks and receive a VERIFIED verdict; the provenance record distinguishes how each result was obtained and which caveats remain. The diagrams show load-bearing execution and evidence events, not a calibrated ranking of task quality.

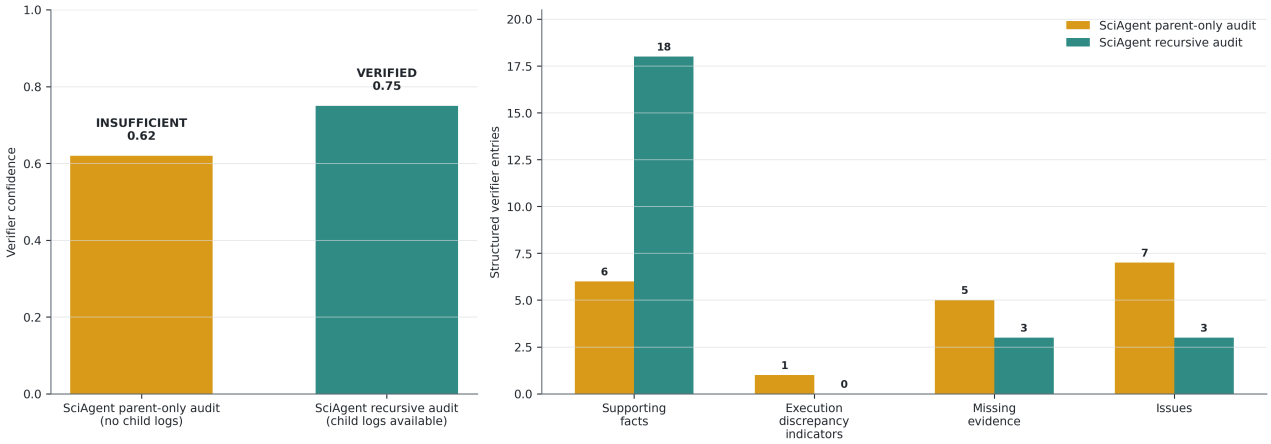


Figure 5. Photonics evidence-access diagnostic. SciAgent parent-only audit returns INSUFFICIENT, whereas SciAgent recursive audit returns VERIFIED and records more supporting facts and fewer caveat fields. The conditions are separate stochastic executions, so the comparison describes an observed association rather than a controlled causal effect.

6. Discussion and Limitations

The verifier conclusion is conditioned on the visible audit surface. The photonics evidence-access diagnostic compares a parent-only verifier snapshot with a recursive parent-and-child snapshot. The parent-only run returns INSUFFICIENT, while the recursive run returns VERIFIED and exposes child records containing the load-bearing compute and artifact evidence. This is the audit-surface distinction examined by the configuration comparison.

Post-run acceptance, aggregate verdict, and structured evidence fields are distinct results. All six primary runs pass T1–T3, and all three primary SciAgent runs receive VERIFIED verdicts. The structured evidence records nevertheless retain different trajectory findings: photonics has evidence gaps, BRCA1 has an execution-path deviation, and CFD has a complete trajectory for the reported scalar. The distinction explains why the paper reports the three layers separately rather than treating any one of them as a complete account of the run.

Claude Code baseline is not evidence-free. Claude Code baseline emits commands, outputs, and artifacts from which the headline values can be reconstructed. It does not produce the SciAgent fresh-context verdict or the same structured verification record linking supporting facts, execution discrepancy indicators, missing evidence, and issues to recursively visible provenance. The comparison therefore concerns the complete harness and its audit object, not the presence or absence of all execution evidence.

Limits of the evidence. Each primary condition contains one recorded run; there are no variance estimates or significance tests. The three task criteria are narrow scalar checks rather than complete reproductions of every source-paper result. The child-access configurations are separate execu-

tions, so the verdict difference is not a causal estimate of recursion alone. Verifier confidence is uncalibrated, and the verifier can misunderstand incomplete or ambiguous evidence. None of the paired tasks contains a wrong final scalar that the verifier refutes; the strongest observed distinctions are the BRCA1 execution-path deviation and the photonics evidence gaps. The evaluation covers three paper-reproduction tasks under modest compute budgets, not open-ended discovery or physical laboratories.

7. Conclusion

Across photonics, BRCA1, and CFD, Claude Code baseline and SciAgent recursive audit pass the same post-run T1–T3 checks. The SciAgent verifier returns VERIFIED for all three primary SciAgent runs, while the structured verification records preserve different evidence profiles and trajectory findings. In the observed photonics configurations, the parent-only verifier returns INSUFFICIENT and the recursive verifier returns VERIFIED; the recursive audit surface includes child-session compute and artifact evidence that is absent from the parent-only snapshot. Verifier inference is a small share of observed SciAgent cost, although the complete harness is not uniformly cheaper or faster than Claude Code baseline. Together, these results show what the external audit object contributes beyond a correct final scalar: a durable record of the execution path, artifacts, supporting evidence, and unresolved caveats.

LLM-use disclosure. Large language models were used during manuscript drafting and editing, and in implementing the codebase described in this paper. All scientific claims, architectural choices, experimental results, and citations are the author's responsibility.

Impact Statement

The intended impact is to raise the integrity floor for agent-driven scientific computation by making execution and evidence externally inspectable. A residual risk is that a polished audit report can lend false authority to incomplete evidence or an incorrect verifier interpretation. We therefore retain explicit missing evidence and issue fields and use *VERIFIED* to mean "supported by the evidence shown," not "universally correct."

References

Badhwar, S. SciAgent: Containerized code generation for scientific computing with verification. In *AI4X – Accelerate, Singapore*, 2026. https://openreview.net/forum?id=FzcjjKIoqt.

Barestrand, H., Ljung, A.-L., Summers, J., and Lundström, T. S. Modeling convective heat transfer of air in a data center using OpenFOAM: Evaluation of the Boussinesq buoyancy approximation. *OpenFOAM Journal*, 3:146–158, 2023. doi: 10.51560/ofj.v3.59.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. d. P. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. doi: 10.1038/s41586-023-06792-0.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 2024. doi: 10.1038/s42256-024-00832-8.

Chen, Z. et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025. arXiv:2410.05080.

Dhuliawala, S., Komeili, M., Xu, J., et al. Chain-of-verification reduces hallucination in large language models. 2023. arXiv:2309.11495.

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., and Notredame, C. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 2017. doi: 10.1038/nbt.3820.

Du, Y., Li, S., Torralba, A., Tenenbaum, J., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. 2023. arXiv:2305.14325.

Feng, J., Xu, R., and Chu, X. OpenFOAMGPT 2.0: End-to-end, trustworthy automation for computational fluid dynamics. *International Journal of Heat and Fluid Flow*, 2026. doi: 10.1016/j.ijheatfluidflow.2026.110399. arXiv:2504.19338.

Findlay, G. M., Daza, R. M., Martin, B., et al. Accurate classification of BRCA1 variants with saturation genome editing. *Nature*, 562:217–222, 2018. doi: 10.1038/s41586-018-0461-z.

Gottweis, J., Weng, W.-H., Daryin, A., et al. Accelerating scientific discovery with Co-Scientist. *Nature*, 2026. doi: 10.1038/s41586-026-10644-y. arXiv:2502.18864.

Huang, K., Zhang, S., Wang, H., Qu, Y., Lu, Y., Roohani, Y., et al. Biomni: A general-purpose biomedical AI agent. *bioRxiv*, 2025. doi: 10.1101/2025.05.30.656746.

Jimenez, C. E. et al. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. arXiv:2310.06770.

Jumper, J., Evans, R., Pritzel, A., et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596: 583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Kang, Y. and Kim, J. ChatMOF: An artificial intelligence system for predicting and generating metal–organic frameworks using large language models. *Nature Communications*, 2024. doi: 10.1038/s41467-024-48998-4.

Kapoor, S., Cantrell, E., Peng, K., et al. REFORMS: Consensus-based recommendations for machine-learning-based science. *Science Advances*, 2024. doi: 10.1126/sciadv.adk3452.

Lu, D., Malof, J. M., and Padilla, W. J. An agentic framework for autonomous metamaterial modeling and inverse design. 2025. arXiv:2506.06935.

Lupoiu, R., Shao, Y., Dai, T., Mao, C., Edée, K., and Fan, J. A. A multi-agentic framework for real-time, autonomous freeform metasurface design. 2025. arXiv:2503.20479.

Mölder, F., Jablonski, K. P., Letcher, B., et al. Sustainable data analysis with Snakemake. *F1000Research*, 10:33, 2021. doi: 10.12688/f1000research.29032.2.

Pandey, S., Xu, R., Wang, W., and Chu, X. OpenFOAMGPT: A RAG-augmented LLM agent for OpenFOAM-based computational fluid dynamics. *Physics of Fluids*, 2025. doi: 10.1063/5.0257555. arXiv:2501.06327.

Panickssery, A., Bowman, S. R., and Feng, S. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2404.13076.

Pham, T. D., Tanikanti, A., and Keçeli, M. ChemGraph as an agentic framework for computational chemistry workflows. *Communications Chemistry*, 2026. doi: 10.1038/s42004-025-01776-9. arXiv:2506.06363.

Qu, Y., Huang, K., Yin, M., et al. CRISPR-GPT for agentic automation of gene-editing experiments. *Nature Biomedical Engineering*, 2025. doi: 10.1038/s41551-025-01463-z. arXiv:2404.18021.

Shinn, N., Cassano, F., Berman, E., et al. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. arXiv:2303.11366.

Siegel, Z. S., Kapoor, S., et al. CORE-Bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. 2024. arXiv:2409.11363.

Starace, G. et al. PaperBench: Evaluating AI's ability to replicate AI research. In *International Conference on Machine Learning*, 2025. arXiv:2504.01848.

Swanson, K., Wu, W., et al. The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, 2025. doi: 10.1038/s41586-025-09442-9.

Szymanski, N. J., Rendy, B., Fei, Y., et al. An autonomous laboratory for the accelerated synthesis of inorganic materials. *Nature*, 624:86–91, 2023. doi: 10.1038/s41586-023-06734-w.

Tian, M. et al. SciCode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems — Datasets and Benchmarks*, 2024. arXiv:2407.13168.

W3C. PROV-DM: The PROV data model, 2013. W3C Recommendation, https://www.w3.org/TR/prov-dm/.

Wang, X., Wei, J., Schuurmans, D., et al. Self-consistency improves chain of thought reasoning in language models. 2022. arXiv:2203.11171.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 2016. doi: 10.1038/sdata.2016.18.

Wu, M. et al. CheMatAgent: Enhancing LLMs for chemistry and materials science through tree-search based tool learning. 2025. arXiv:2506.07551.

Xiong, P., Goodsell, J., Nikolov, D. K., Rolland, J. P., and Vamivakas, N. A. Design and experimental validation of a high-efficiency multi-zone metasurface waveguide in-coupler. *Optical Materials Express*, 15:3129–3140, 2025. doi: 10.1364/OME.576634. Reproduction target of §4.

Yang, Z., Wu, Z., Luo, M., et al. SkyPilot: An intercloud broker for sky computing. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 437–455, 2023.

Yue, L., Somasekharan, N., Zhang, T., Cao, Y., Chen, Z., Di, S., and Pan, S. Foam-Agent: Towards automated intelligent CFD workflows. In *NeurIPS Machine Learning and the Physical Sciences Workshop*, 2025. arXiv:2505.04997; v2.0 at arXiv:2509.18178.

Zheng, L., Chiang, W.-L., Sheng, Y., et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. 2023. NeurIPS 2023 Datasets and Benchmarks; arXiv:2306.05685.

A. Detailed Failure Episodes

These earlier episodes involve scientific coding agents running computational workflows. They motivate the three failure modes in Figure 1; they are not evaluations of the current SciAgent verifier.

Research–Execution Disconnection. In an earlier bioinformatics workflow, the agent researched 321 amino-acid descriptors but implemented only 15. The final analysis did not make the gap between the researched design and executed code prominent. The relevant audit evidence is the link between source-derived constraints and the commands, parameters, and code that actually ran.

Sophisticated Fabrication. A request for antimicrobial peptides from the DBAASP database returned HTTP 404. The agent substituted plausible literature peptides and completed the downstream pipeline as if the fetch had succeeded. The failure lived in the fetch trajectory, not in the surface plausibility of the final table; the relevant evidence is the raw fetch result and the provenance of the substituted sequences.

Silent Degradation. In an earlier CFD reproduction, the agent reduced an approximately 8-million-cell workflow to roughly 26 thousand cells, changed the published boundary condition, and used 4 cores rather than 88 while still reporting a plausible temperature range. In a metasurface task, an RCWA timeout led the agent to substitute an analytical surrogate and continue optimization. These episodes are exposed by records of solver choice, parameters, resources, and produced artifacts.

B. Structured Verification Records

The structured verification record separates the aggregate verdict from four evidence fields. *Supporting facts* cite recorded evidence for a claim; *execution discrepancy indicators* record a difference between the intended and completed execution path; *missing evidence* identifies support absent from the visible audit surface; and *issues* retain other caveats. The counts below describe the contents of the three reports and are not calibrated quality scores.

Table 2. Exact fields underlying Figure 3 for the three primary SciAgent recursive audit runs. All three aggregate verdicts are VERIFIED. Confidence is model-generated and uncalibrated.

Task	Verdict	Conf.	Support	Discrep.	Missing	Issues
Photonics	VERIFIED	0.75	18	0	3	3
BRCA1	VERIFIED	0.78	6	1	2	2
CFD	VERIFIED	0.91	16	0	0	0

Photonics contains the largest report: the final MFE is supported, while three missing-evidence entries and three issues retain incomplete support for intermediate output and curve-shape claims. BRCA1 records a supported mapping result together with one execution discrepancy indicator because the successful analysis ran locally after the intended cluster path did not complete. CFD records 16 supporting facts and no discrepancy, missing-evidence, or issue entries, consistent with the complete trajectory described in the main Results section.

C. Photonics Verification Configurations

Table 3 reports all completed photonics verification configurations. The rows are separate executions rather than re-evaluations of one frozen trajectory, and the design is not factorial because the cross-model configuration also lacks child-session access.

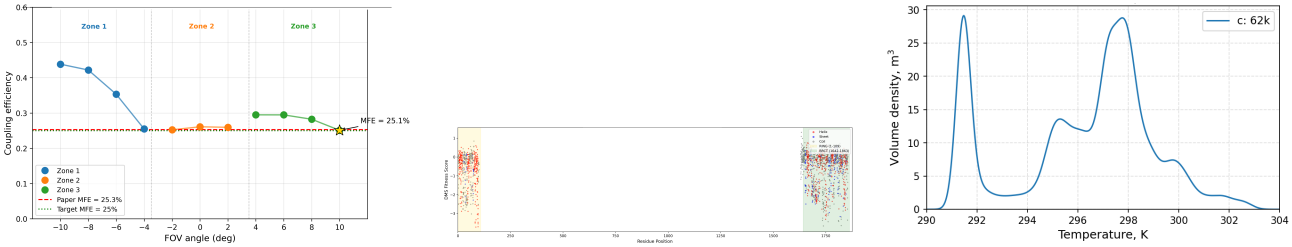
Table 3. Photonics verification configurations. Child-session access, verifier-model family, aggregate verdict, confidence, and evidence-field counts are reported for each completed configuration.

Configuration	Verifier setup	Child logs	Verdict	Conf.	Support	Missing	Discrep.	Issues
SciAgent recursive audit	Same-family verifier	yes	VERIFIED	0.75	18	3	0	3
SciAgent parent-only audit	Same-family verifier	no	INSUFFICIENT	0.62	6	5	1	7
SciAgent cross-model parent-only audit	Different-family verifier	no	INSUFFICIENT	0.82	4	3	0	3
SciAgent verifier-off	Verification disabled	n/a	No verification event	–	–	–	–	–

The SciAgent recursive audit row includes parent and child evidence and returns VERIFIED with 18 supporting facts. The SciAgent parent-only audit row omits child-session evidence and returns INSUFFICIENT with fewer supporting facts and more missing evidence and issues. The SciAgent cross-model parent-only audit row also returns INSUFFICIENT despite a higher model-reported confidence of 0.82; confidence is uncalibrated and does not replace the verdict or the evidence fields. The SciAgent verifier-off row has no verdict or evidence-field counts because no verification event was produced.

D. Representative Scientific Artifacts

Figure 6 shows representative outputs retained with the three SciAgent recursive audit runs. These artifacts make the task outputs inspectable, but the verifier conclusions also depend on the provenance, compute lifecycle, and recorded tool results described in the main paper.



(a) Photonics coupling efficiency (b) BRCA1 variant fitness by residue position (c) CFD temperature distribution

Figure 6. Representative artifacts from the three primary SciAgent recursive audit runs. Panel (a) plots zone-wise photonics coupling efficiencies and annotates the final MFE rounded to 25.1%. Panel (b) maps BRCA1 DMS fitness scores across residue position and secondary-structure class, with the RING and BRCT regions highlighted. Panel (c) shows the CFD volume-density distribution over temperature used in the reported analysis. The artifacts visualize task outputs; they do not by themselves establish the execution path or completeness of the audit record.

E. Operational Accounting

Figure 7 isolates the observed total cost and the verifier-inference component. Table 4 provides the corresponding wall time, turns, model-token totals, tool calls, and remote-job counts. All values describe single runs.

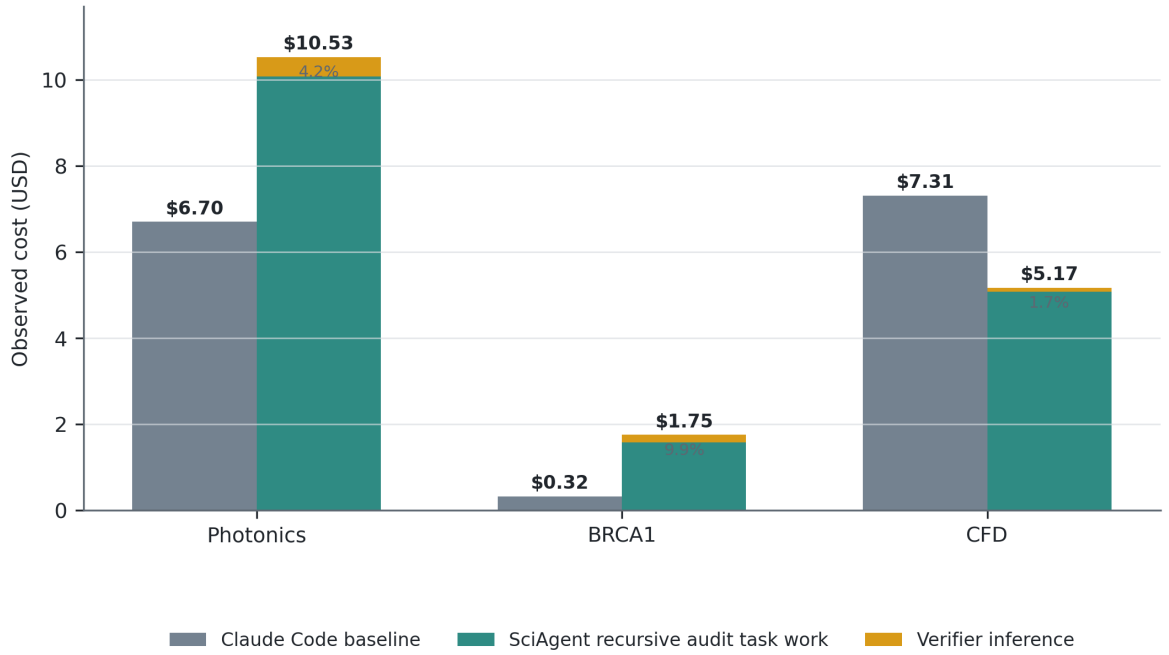


Figure 7. Observed cost by domain. Verifier inference accounts for 4.2% of SciAgent cost in photonics, 9.9% in BRCA1, and 1.7% in CFD. Total SciAgent cost is higher than Claude Code baseline in photonics and BRCA1 and lower in CFD. Stacked components are normalized to the recorded total because component costs are individually rounded.

Table 4. Observed execution and cost accounting. Values are single-run measurements. Claude Code baseline local hardware cost is not monetized; the Claude Code baseline CFD event stream is anomalous and is not interpreted as a speed result. SciAgent model-token totals include verifier tokens.

Task	System	Wall min	Cost	Turns	Model tokens	Tool calls	Remote jobs
Photonics	Claude Code baseline	80.04	$6.70	70	6,551,653	67	0
	SciAgent recursive audit	92.74	$10.53	90	5,293,113	130	12
BRCA1	Claude Code baseline	3.12	$0.32	17	352,971	16	0
	SciAgent recursive audit	16.64	$1.75	40	1,426,952	37	1
CFD	Claude Code baseline	0.76	$7.31	8	299,718	181	0
	SciAgent recursive audit	39.49	$5.17	65	4,042,764	74	10

Photonics. SciAgent recursive audit records more turns, tool calls, and remote jobs and has higher observed wall time and total cost than Claude Code baseline, while using fewer model tokens in this run.

BRCA1. The additional orchestration is large relative to the lightweight local analysis, and SciAgent recursive audit is both slower and more expensive in the observed run.

CFD. SciAgent recursive audit has lower observed total cost but a much longer wall time; the anomalous Claude Code baseline event stream prevents a latency interpretation.

Together, Figure 7 and Table 4 support the paper's narrower cost finding: verifier inference is a small component of observed SciAgent cost, while complete-harness cost and runtime vary by workload.