BEYOND EXECUTION: OPERATIONALITY AND SCIENTIFIC FIDELITY ARE DISTINCT CAPABILITIES OF SCIENTIFIC CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scientific coding agents are increasingly judged by whether the artifacts they produce execute. For simulation code, execution is only operational evidence: a case can run while instantiating the wrong physical parameter, geometry, boundary condition, or numerical intent. We ask whether *operationality* and *scientific fidelity* are the same capability. We encode five canonical OpenFOAM problems as hidden machine-readable scientific contracts and evaluate six contemporary language models under real solver execution and deterministic requirement checks. Both mismatch directions occur: executable artifacts violate the intended science, and scientifically faithful artifacts fail to run. Across the six evaluated models, the two rankings have a Spearman correlation of $\rho = -0.03$, and model ordering can reverse between the criteria; the reversal between the best executor and the most faithful model persists across three independent generations. We then intervene on the information the agent receives: given only ordinary runtime feedback, operationality rises by $+0.70$ while scientific fidelity changes by $+0.04$, and 79% of doubly-failing artifacts whose execution is repaired remain scientifically wrong. Execution feedback repairs operationality but provides little signal for hidden scientific errors. Evaluation that collapses the two axes into a single execution signal can change which model is judged best.

1 INTRODUCTION

LLM coding agents turn natural-language requests into executable programs. Scientific computing makes this capability consequential in a particular way: the user is specifying not only software behavior but a model of the world—geometry, governing equations, material and transport parameters, boundary and initial conditions, discretization, and run duration. The generated program is useful only if it implements that scientific intent.

Execution-based evaluation is the standard for modern code and scientific-agent benchmarks (Jimenez et al., 2024; Tian et al., 2024; Chen et al., 2025). It is a large advance over surface-form metrics, but for simulation code it conflates two properties that need not coincide. A wrong viscosity can preserve syntax and solver stability while changing the Reynolds regime; a changed geometric scale can still produce a valid mesh; a plausible output profile can be insensitive to the very parameter that was mis-specified. The software stack accepts a program that solves a different problem from the one intended.

This paper asks whether *operationality* (the software stack accepts and runs the artifact) and *scientific fidelity* (the artifact implements the intended experiment) are the same capability. If they are, execution is a sound proxy and the distinction is bookkeeping. If they are separable, then execution-centric evaluation can mis-measure agents, and can change comparative conclusions about which model is better. We show the second is true, and we show it three ways.

First, we measure the two properties independently. Five canonical OpenFOAM problems are encoded as hidden machine-readable scientific contracts, generated across six specification conditions by six contemporary language models, and scored by deterministic requirement checks and real OpenFOAM execution. Figure 1 summarizes the distinction: an artifact can succeed or fail independently along the operational and scientific dimensions, and the evaluated models exhibit different profiles across

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.

the two. The model with the highest execution rate (GLM-5.3) is mid-pack on scientific fidelity, while the most scientifically faithful model (DeepSeek-V4-Pro) has the second-lowest execution rate. That pairwise reversal persists across three independent generations.

Second, we intervene on the information the agent receives. Given only ordinary runtime feedback, the parser, mesh, and solver errors a real agent would see, models repair 17 of 25 cases that were scientifically faithful but failed to execute, restoring both criteria ($E0S1 \to E1S1$), while the mean scientific-fidelity rate moves by only 0.04. Execution feedback sharply repairs operationality while leaving hidden scientific errors largely untouched, which is intervention-based evidence that the two capabilities respond to different information.

Third, we stress-test the evaluator itself: a blind two-reviewer audit (inter-rater $\kappa = 0.88$) exposes one genuine semantic-equivalence issue, and all headline conclusions survive the corrected evaluator.

Existing evaluation practice supplies both signals but does not ask whether they measure the same capability. Execution-based benchmarks treat a successful run as success; physics-aware benchmarks add correctness checks alongside executability. Our question is different: whether the two criteria agree, whether choosing one over the other changes comparative conclusions, and how the two dimensions respond when the agent receives software information without hidden scientific-contract feedback.

Our contributions are:

- **A two-axis evaluation formulation.** We measure operationality and scientific fidelity separately using real solver execution and frozen deterministic scientific contracts, and we show the four resulting artifact states, including scientifically faithful artifacts that fail to run, are all populated in practice.
- **A comparative consequence.** Across six models the two criteria produce different capability profiles and orderings, including a DeepSeek-V4-Pro/GLM-5.3 reversal that persists across three independent generations, so the choice of criterion changes which model is judged best.
- **Intervention-based explanatory evidence.** Execution-only feedback sharply improves operationality ($\Delta E = +0.70$) with little change in scientific fidelity ($\Delta S = +0.04$), providing intervention-based evidence for why the capabilities diverge.

2 Operationality and Scientific Fidelity

Let a simulation task carry scientific requirements $\mathcal{C} = \{c_1, \ldots, c_m\}$ covering, as applicable, the governing model, physical parameters and units, geometry, boundary conditions, numerical controls, and stopping criteria. An LLM maps a user-facing specification s to a solver artifact A. We evaluate two properties independently:

$$E(A) = \mathbb{1}[A \text{ completes the required solver execution}], \tag{1}$$

$$S(A, \mathcal{C}) = \mathbb{1}[A \text{ satisfies every applicable requirement in } \mathcal{C}]. \tag{2}$$

They are not nested. A scientifically faithful artifact can fail to run because of a software-specific configuration, and an executable artifact can violate the intended science. Every artifact therefore falls into one of four states (Figure 1a):

$E1S1$	operational and scientifically faithful;
$E1S0$	executable but violates the scientific contract;
$E0S1$	scientifically faithful but operationally unsuccessful;
$E0S0$	fails both dimensions.

The two off-diagonal quadrants are the informative ones. $E1S0$ is the failure mode that execution-based evaluation cannot see, and $E0S1$ is the failure mode that execution-based evaluation misreads as incapability.

Why the two properties must be measured separately. Execution tests a solver-facing interface: files must parse, the mesh must be accepted, and the solver must reach its completion condition. Scientific fidelity tests the mapping from the requested experiment to the instantiated one. A correct

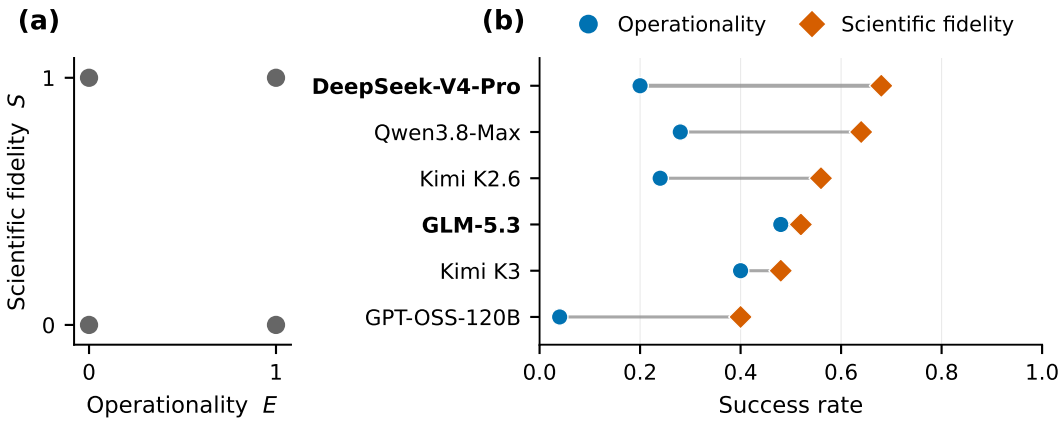


Figure 1: Operationality and scientific fidelity expose different capability profiles. **(a)** The two binary criteria define four artifact states according to whether a generated case executes (E) and satisfies the scientific contract (S); the off-diagonal states separate executable-but-scientifically-wrong artifacts ($E1S0$) from scientifically faithful artifacts that fail to execute ($E0S1$). **(b)** Within each model the two rates differ substantially on the 25-task matched panel (circles: operationality; diamonds: scientific fidelity; the representative reversal pair DeepSeek-V4-Pro and GLM-5.3 in bold). DeepSeek-V4-Pro combines low operationality ($E = 0.20$) with the highest scientific fidelity ($S = 0.68$), whereas GLM-5.3 has the highest operationality ($E = 0.48$) but middling fidelity ($S = 0.52$), so the evaluation criterion changes the model ordering (Figure 2).

physical specification can be rendered non-executable by a software detail, while a numerically stable case can encode the wrong viscosity, length scale, boundary value, or time horizon. Neither execution rate nor output plausibility is a surrogate for $S(A, \mathcal{C})$.

Scientific contracts. Our contracts are concrete rather than ontological. A requirement is included only when it can be tied to the user specification and checked from solver-native artifacts or a cheap physical relation: a transport coefficient and its dimensions, a cylinder diameter recovered from the mesh, a no-slip wall type, an inlet value, an end-time lower bound, or Reynolds-number consistency. This makes the evaluator auditable: a failed scientific-fidelity label traces to one or more explicit checks rather than an opaque model score (Appendix B).

A conditional statistic, kept in its place. Where execution counts are small we also report the *Executable Scientific Error Rate*,

$$\mathrm{ESER} = \Pr[S(A, \mathcal{C}) = 0 \mid E(A) = 1], \tag{3}$$

the fraction of executed artifacts that violate the contract. ESER conditions on a post-treatment event—an omission can change both whether a case runs and whether it is correct—so it is not by itself a causal contrast across conditions. We use the unconditional fidelity rate and matched comparisons as primary, and ESER as a supporting diagnostic.

3 Benchmark and Evaluation Protocol

Frozen benchmark. We construct five standard problems: lid-driven cavity flow, plane Poiseuille flow, planar Couette flow, cylinder flow at $Re = 40$, and slab heat conduction. Each problem has a hidden machine-readable contract containing only requirements needed for evaluation. From each full specification we derive five controlled omissions—`NO_BOUNDARY`, `NO_PHYSICS`, `NO_GEOMETRY`, `NO_NUMERICS`, and `NO_STOPPING`—which together with `FULL` give 30 frozen instances. The benchmark, contracts, generic generation prompt, metrics, and exclusion rules were frozen before formal generation (hashes in Appendix J).

Generation. Each model receives only the frozen coding instruction and the user-facing instance specification; it does not receive the omission label, hidden contract, evaluator, or expected result. We

Condition	N	Execute	Sci. correct	Exec. wrong	ESER
FULL	10	3/10	8/10	0/3	0.0%
NO_BOUNDARY	10	5/10	6/10	1/5	20.0%
NO_PHYSICS	10	**6/10**	**1/10**	**5/6**	**83.3%**
NO_GEOMETRY	10	4/10	6/10	2/4	50.0%
NO_NUMERICS	10	7/10	9/10	1/7	14.3%
NO_STOPPING	10	3/10	4/10	2/3	66.7%
Overall	60	28/60	34/60	11/28	39.3%

Table 1: Controlled OpenFOAM results (Kimi K3, two generations per instance). Artifact completeness and structural validity are 10/10 in every condition and omitted for space. Scientific correctness is checked against the full hidden contract and does not require execution; ESER is conditional on solver execution.

evaluate six contemporary language models spanning five providers: Kimi K3 (Moonshot AI, 2026), Kimi K2.6, DeepSeek-V4-Pro, Qwen3.8-Max, GLM-5.3, and GPT-OSS-120B. Kimi K3 additionally anchors the controlled study with two isolated generations per instance (60 total). Generation uses temperature 0 and a completion budget only; truncation and provider failures are recorded separately from scientific outcomes (Appendix J).

Deterministic evaluation and real solver runs. Generated cases are parsed into OpenFOAM artifacts and checked deterministically; no LLM judges scientific correctness. We record artifact completeness, structural validity, mesh validity, execution, and contract correctness. Execution requires `blockMesh`, `checkMesh`, and solver exit code zero with an `End` marker in the log under OpenFOAM v2412 (OpenCFD, 2024). Contract checks extract viscosity, thermal diffusivity, geometry scale, boundary conditions, solver selection, time controls, and nondimensional consistency from the generated files.

Model panel and repeats. The primary comparison uses the 25 task instances on which all six models produced a usable generation, so no model is compared over a different task mixture. For the decisive reversal we additionally run three independent repeats of DeepSeek-V4-Pro and GLM-5.3 on the same frozen tasks.

4 Execution and Scientific Fidelity Measure Different Capabilities

4.1 Executable does not mean scientifically correct

Table 1 reports the controlled Kimi K3 study. All 60 generations are artifact-complete and structurally valid; 42 mesh successfully and 28 complete the required solver execution. Among those 28 executions, 11 violate at least one scientific requirement (ESER $= 11/28 = 39.3\%$, bootstrap 95% CI $[21.4, 57.1]$). The strongest intervention is omission of a consequential physical parameter: under `NO_PHYSICS`, 6/10 generations still execute, but only 1/10 satisfies the intended contract, and 5/6 executed cases are scientifically wrong. These are not syntax-level defects—the artifacts have already passed the solver run—but scientific choices the runtime accepts: $\nu = 0.01$ instead of 0.001 (changing Re from 100 to 10), or thermal diffusivities an order of magnitude off. Assumption disclosure does not help: all 50 underspecified generations explicitly state an assumption, none requests clarification, and disclosed assumptions are frequently wrong.

4.2 The opposite mismatch is equally common

Executable-but-wrong ($E1S0$) is only half the picture. The reverse mismatch, scientifically faithful but non-executable ($E0S1$), is at least as common. Across the six-model panel, large fractions of each model's artifacts are scientifically correct yet fail to run: 14/25 for DeepSeek-V4-Pro, 11/25 for Qwen3.8-Max and Kimi K2.6, 7/25 for Kimi K3. These are cases the contract accepts but the software stack rejects, for reasons as mundane as a missing `pFinal` residual-control entry or a duplicated

Model	E	S	ESER	$E0S1$	$E1S0$
GLM-5.3	0.48	0.52	4/12	5	4
Kimi K3	0.40	0.48	5/10	7	5
Qwen3.8-Max	0.28	0.64	2/7	11	2
Kimi K2.6	0.24	0.56	4/6	12	4
DeepSeek-V4-Pro	0.20	0.68	2/5	14	2
GPT-OSS-120B	0.04	0.40	1/1	10	1

Table 2: Six-model comparison on the 25-task matched panel: execution rate E, scientific-fidelity rate S, executable scientific error rate (ESER), and the two mismatch quadrants $E0S1$ (faithful but non-executable) and $E1S0$ (executable but wrong). E and S are not monotone in each other across models.

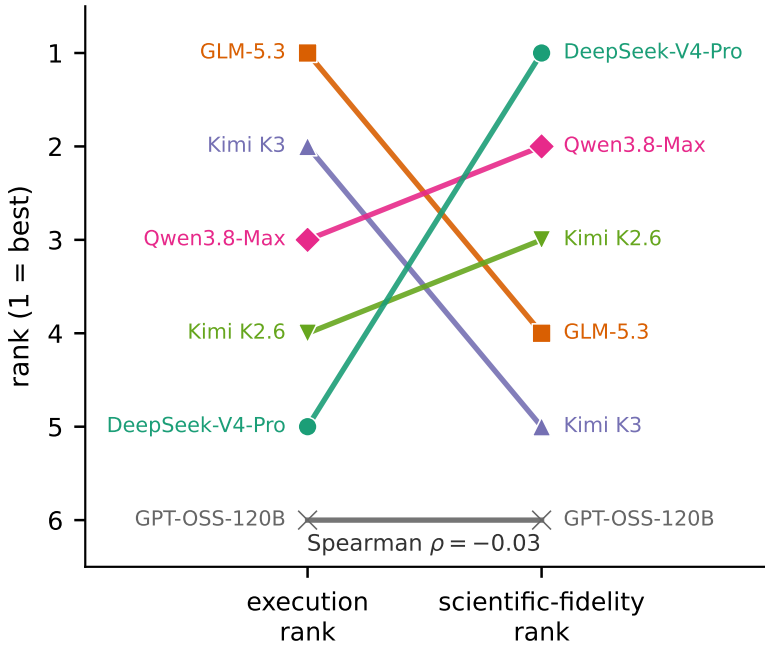


Figure 2: Model ranks under the two evaluation criteria on the 25-task matched panel (rank 1 = best). The order changes substantially: the rank correlation is Spearman $\rho = -0.03$. GLM-5.3 ranks first by execution but only fourth by scientific fidelity; DeepSeek-V4-Pro shows the opposite pattern. The choice of evaluation criterion changes which model is judged best.

blockMesh face. An execution-only benchmark scores all of them as failures, indistinguishable from genuine scientific errors.

4.3 Execution and fidelity rankings disagree

Figure 1b compares operationality and scientific fidelity within each model, and Figure 2 compares the two rankings directly. The within-model gaps vary substantially: the two criteria do not scale together. Quantitatively, the execution and scientific-fidelity rankings have a Spearman correlation of $\rho = -0.03$ (bootstrap 95% CI $[-0.46, 0.73]$); the stronger claim is that criterion choice can change model ordering. The clearest example is the pair at the extremes. GLM-5.3 has the highest execution rate ($E = 0.48$) but middling scientific fidelity ($S = 0.52$); DeepSeek-V4-Pro has the second-lowest execution rate ($E = 0.20$) but the highest scientific fidelity ($S = 0.68$). An execution-centric evaluation ranks GLM-5.3 first and DeepSeek-V4-Pro fifth; a fidelity-centric evaluation reverses them. Table 2 reports the full comparison; the models occupy genuinely different positions rather than scaling one shared axis, and no model dominates on both.

Initial	Main outcome	Count	Interpretation
$E0S1$	$E1S1$	17/25	software failure repaired, science preserved
$E0S0$ (execution repaired)	$E1S0$	11/14	software repaired, scientific error remains

Table 3: Decisive repair transitions under execution-only feedback. Faithful-but-broken cases are largely repaired without corrupting the science; cases broken in both dimensions are made to run while their scientific error persists.

4.4 The reversal persists across independent generations

A single generation per model could make the reversal a sampling accident. We therefore generated three independent repeats of DeepSeek-V4-Pro and GLM-5.3 on the frozen tasks. Across repeats, GLM-5.3 executes more often ($E = 0.45$) and DeepSeek-V4-Pro is more scientifically faithful ($S = 0.68$); a task-level bootstrap over the matched set gives $P(E_{\mathrm{GLM}} > E_{\mathrm{DS}}) = 0.89$, $P(S_{\mathrm{DS}} > S_{\mathrm{GLM}}) = 0.98$, and $P(\text{full reversal}) = 0.87$. The reversal persists across three independent generations.

5 Execution Feedback Selectively Repairs Operationality

The ranking disagreement shows that the two metrics disagree; it does not by itself show that they measure different capabilities. The disagreement could instead reflect that some models are simply better at everything and happen to score higher on one axis. To separate the capabilities we intervene on the information the agent receives.

5.1 Intervention design

For every artifact that fails to execute ($E0$), we give the same model only ordinary runtime feedback, the parser, `blockMesh`, `checkMesh`, or solver error a real agent would observe, and allow at most two repair rounds, $A_0 \to A_1 \to A_2$. The model never receives the hidden contract, the expected physical values, the omission label, or any correctness signal. The repair instruction asks the model to fix the failure while preserving the scientific setup implied by the specification; values the specification omits, which the model had to assume, remain free to change, so the intervention does not mechanically freeze the science. At every step we independently re-evaluate (E_k, S_k) with the frozen contract checker. We run the intervention on three models spanning the capability plane (DeepSeek-V4-Pro, GLM-5.3, Kimi K3), yielding 46 valid repair targets. This is the information available to a scientific coding agent with execution access, and nothing more.

5.2 Execution feedback repairs operationality, not fidelity

Figure 3 shows the resulting trajectories in the operationality–fidelity plane. The arrows move strongly rightward and barely upward. Across the 46 valid repair targets, operationality rises from 0 to 0.70 ($\Delta E = +0.70$) while scientific fidelity moves from 0.55 to 0.60 ($\Delta S = +0.04$). The mechanism is clearest for DeepSeek-V4-Pro: its execution rate climbs from 0.20 to 0.92 while its scientific fidelity stays at 0.68. Its Round-1 "low execution" was not low capability; it was a collection of repairable software defects on top of scientifically sound cases.

The transition structure sharpens the point. Of the scientifically-faithful-but-non-executable cases ($E0S1$), 68% are repaired to both execute and remain faithful ($E0S1 \to E1S1$), and fidelity is lost in 4 of the 25 (16%). Of the cases that fail both dimensions ($E0S0$), execution feedback fixes the crash but leaves the science wrong: among those whose execution is repaired, 79% remain scientifically incorrect ($E0S0 \to E1S0$). Ordinary runtime feedback therefore directly exposes many software failures but provides little signal for hidden scientific-contract violations. This asymmetry is the direct mechanistic evidence that operationality and scientific fidelity are separable capabilities. Table 3 summarizes the two decisive transitions.

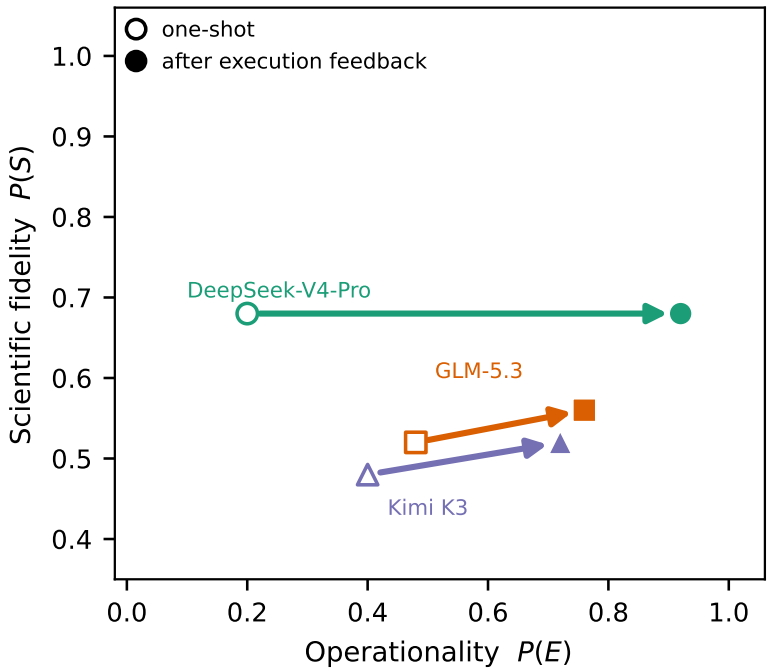


Figure 3: Operationality versus scientific fidelity before (open markers) and after (filled markers) repair, for the three repair-panel models. Only ordinary runtime and software feedback is supplied during repair; the hidden scientific contract and any correctness signal are never supplied. Arrows move rightward (operationality) far more than upward (fidelity), $\Delta E \gg \Delta S$: DeepSeek-V4-Pro rises from $E = 0.20$ to $E = 0.92$ with fidelity unchanged at 0.68.

Initial	Scientific state	Runtime signal	Repair outcome	Final
$E0S1$	contract satisfied	missing `pFinal` entry	software configuration repaired	$E1S1$
$E0S0$	wrong viscosity	software failure only	execution repaired; viscosity unchanged	$E1S0$

Table 4: Two representative repair trajectories (both DeepSeek-V4-Pro, cavity). Runtime feedback names the software defect in the first case; in the second, the scientific mismatch is silent because the erroneous case is a valid solver input.

5.3 What execution feedback reveals

The aggregate asymmetry has a concrete microstructure. Table 4 shows two representative repairs (full trajectories in Appendix F). In the first, DeepSeek-V4-Pro's cavity case is scientifically faithful but the solver aborts on a missing `pFinal` residual-control entry; the runtime error names the defect, and the model adds the missing solver block ($E0S1 \to E1S1$). In the second, an underspecified case uses a wrong viscosity and also fails to run; runtime feedback repairs the execution failure, but because the error log says nothing about the viscosity, the repaired case runs with the same wrong Re ($E0S0 \to E1S0$). Runtime errors directly identify many software defects, but a solver-valid scientific mismatch produces no error to repair.

5.4 Implication for evaluation

The intervention explains the observational result. Models with many $E0S1$ cases (DeepSeek-V4-Pro, Qwen3.8-Max, Kimi K2.6) look weak under one-shot execution because their scientifically sound cases are fragile to software defects, while models with many $E1S0$ cases look strong because execution cannot see their scientific errors. Execution-centric evaluation conflates these two very different situations into a single number. One-shot execution mixes scientific translation ability with software-interface reliability: runtime feedback removes a substantial fraction of repairable

software friction, but it does not reveal many solver-valid scientific mismatches, so models appear different depending on which dimension a benchmark measures. Measuring the two axes separately, or supplying execution feedback during evaluation, recovers the distinction.

6 Evaluator Validation and Robustness

We stress-test the deterministic evaluator with a blind two-reviewer audit rather than an LLM judge.

The two reviewers scored a frozen stratified sample of 50 artifacts, blind to model identity, the automatic labels, and the study's conclusions. The reviewers agree with each other at 0.97 requirement-level (Cohen's $\kappa = 0.88$), and the evaluator agrees with their consensus on the consequential scientific requirements: geometry 0.98, physics 0.94, boundary 0.96. The audit also exposed one genuine defect: the initial checker required the literal `noSlip` wall type, treating the scientifically equivalent `fixedValue uniform (0 0 0)` too literally. We corrected the affected rule in a versioned evaluator with regression tests and recomputed all analyses; the principal capability-separation, ranking, and intervention conclusions were unchanged (the correction moves ten artifacts from $E0S0$ to $E0S1$ and leaves the executable-but-wrong count, the ESER, the ranking reversal, and the repair asymmetry untouched).

The residual disagreement is confined to numerics and stopping, where the evaluator applies preregistered admissibility semantics (Courant ≤ 1, timestep and duration bounds) while the blind reviewers applied exact-match instruction fidelity; most of these cases sit in conditions where the exact value was deliberately withheld from the model. Full disagreement taxonomy, contract semantics, and before/after metrics are in Appendix G.

7 Related Work

Executable code evaluation. Code-generation evaluation has moved from lexical similarity toward functional execution, including HumanEval, APPS, SWE-bench, and repository-scale generation (Chen et al., 2021; Hendrycks et al., 2021; Jimenez et al., 2024; Zhao et al., 2025). This is appropriate when tests capture the intended program behavior. Our setting highlights a regime where they cannot: a scientific solver accepts a program that solves a different problem.

Scientific coding and research agents. SciCode evaluates research-level scientific programming with scientist-written tests (Tian et al., 2024); ScienceAgentBench evaluates executable programs for data-driven discovery tasks (Chen et al., 2025); RECODE-H adds interactive human feedback (Miao et al., 2026); Paper2Code generates research repositories from papers (Seo et al., 2026). These works establish that executable scientific artifacts matter. We ask a different question: whether solver-native artifacts preserve the physical requirements of the request, and whether execution changes the answer to "which model is better."

Physics-aware simulation generation. PhysCodeBench combines executability with physical accuracy for symbolic simulation (Xie et al., 2026), and LLM agents have been coupled to CFD solvers directly (Chen et al., 2024; Yue et al., 2026). These works establish that scientific correctness can be evaluated in addition to executability. Our question is whether the two criteria measure the same model capability and whether choosing one or the other changes comparative conclusions; we then use execution feedback as an intervention to test how the two dimensions respond to additional software information. The checks themselves are complementary to this line: we manipulate the information in the specification and check the resulting low-level solver configuration against prespecified requirements, isolating errors that survive both generation and execution.

Specification quality and evaluation validity. SciCode-Verified reports that benchmark defects and missing scientific information can cause correct solutions to be rejected (Hu et al., 2026). Work on ambiguity and clarification (Liu et al., 2023; Mu et al., 2024) studies the *failure mechanism and mitigation* of underspecification, asking models to recognize and resolve missing information. We study the complementary *measurement* question: holding a hidden reference contract fixed, we remove selected information from the model-facing specification and measure what the model reconstructs and whether the executable artifact still matches the intended task.

Execution feedback and self-debugging. A growing line of work feeds runtime errors back to the model for repair. Our repair study uses this idea instrumentally rather than as a product: it asks what execution feedback *contains*, and finds that it exposes operational but not scientific defects.

8 Discussion and Limitations

The two studies together clarify what a one-shot execution score actually measures. Execution mixes two things: whether the model translated the scientific request correctly, and whether it produced a configuration the software stack accepts. Different models trade these differently, which is why the rankings diverge. The repair intervention separates them: supplying ordinary runtime feedback removes a large share of the repairable software friction ($\Delta E = +0.70$) while leaving hidden scientific mismatches largely untouched ($\Delta S = +0.04$). What remains after repair is much closer to scientific-fidelity capability alone.

The experiments support a concrete change to scientific-agent evaluation: keep the user-level quantities that define the experiment, and verify them after generation whenever they are mechanically recoverable. This does not require a general-purpose science judge. Even partial contracts catch errors that execution cannot, provided each check traces to the task. The two evidence channels disagree in both directions in our data, so treating either as ground truth discards information, and reporting them separately makes failures actionable: an execution failure calls for software repair, an executable contract violation calls for scientific correction. The practical consequence for benchmarks is direct: report operationality and scientific fidelity as separate metrics rather than collapsing them into a single execution-success label.

Our scope is bounded. The controlled experiments use OpenFOAM and five canonical problems, deliberately chosen for low solver cost rather than industrial scale; the panel is six models; and the matched task set is 25 instances, so individual rank orderings among close models carry uncertainty even though the headline reversal persists across repeats. Canonical problems are likely represented in training data, so the measured profiles characterize well-rehearsed solver configuration; whether the faithful-but-fragile ($E0S1$) mechanism transfers to non-canonical problems is an open question. Scientific contracts test only the requirements they encode and do not replace open-ended scientific judgment, mesh-convergence studies, or constitutive-model adequacy. We study classical solver configuration, not iterative human–agent collaboration. A historical four-backend study (OpenFOAM, SU2, CalculiX, OpenPhase) shows the same ordering at the workflow level and is reported for context in Appendix I; all controlled headline claims use the fully auditable frozen benchmark.

9 Conclusion

Execution and scientific fidelity are separable capabilities of scientific coding agents. Across six contemporary language models the two criteria give different capability profiles and can reverse model ordering, and execution-only feedback sharply repairs operationality while leaving hidden scientific errors largely unchanged. Execution is necessary operational evidence, but it does not substitute for scientific-fidelity evaluation: when both are measurable, scientific coding agents should be evaluated and reported on both axes.

REPRODUCIBILITY STATEMENT

The frozen 30-instance benchmark and hidden contracts, the exact generation prompt and payload construction, all raw generations, parsed OpenFOAM cases, the deterministic contract checker with unit and regression tests, per-case and per-requirement results, statistical outputs, execution-stage logs, repair trajectories, repeat generations, and the human-audit package are retained in the release. The benchmark SHA256 (Appendix J) binds the analyzed task set to the frozen pre-generation artifact.

OpenFOAM version, solver commands, timeouts, and the post-freeze evaluator v1→v2 correction are documented in Appendices J and G. The cross-system Study-I is retained as historical breadth evidence only.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for providing feedback on research methodology and experiments, providing feedback on mathematical arguments, and interpreting results. We have not used generative AI tools for generating synthetic data sets, formulating mathematical claims or implementing methods, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for literature discovery, brainstorming, drafting and editing parts of the manuscript, editing software code and figures, and simulated peer-review feedback. We have reviewed all AI-assisted work and AI-assisted code was re-run with its outputs checked against the experimental artifacts reported in this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Chen, M. et al. Evaluating Large Language Models Trained on Code. arXiv:2107.03374, 2021.

Chen, Y., Zhu, X., Zhou, H., and Ren, Z. MetaOpenFOAM: An LLM-Based Multi-Agent Framework for CFD. arXiv:2407.21320, 2024.

Chen, Z. et al. ScienceAgentBench: Toward Rigorous Assessment of Language Agents for Data-Driven Scientific Discovery. *ICLR*, 2025.

Hendrycks, D. et al. Measuring Coding Challenge Competence With APPS. *NeurIPS*, 2021.

Hu, S., Huang, L., Deng, Y., and Chen, K. SciCode-Verified: How Benchmark Defects Underestimated the Scientific-Coding Ability of Language Models. arXiv:2608.04975, 2026.

Jimenez, C. E. et al. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *ICLR*, 2024.

Liu, A. et al. We're Afraid Language Models Aren't Modeling Ambiguity. *EMNLP*, 2023. arXiv:2304.14399.

Miao, C. et al. RECODE-H: A Benchmark for Research Code Development with Interactive Human Feedback. *ICLR*, 2026.

Moonshot AI. Kimi K3: Open Frontier Intelligence. arXiv:2607.24653, 2026.

Mu, F. et al. ClarifyGPT: A Framework for Enhancing LLM-Based Code Generation via Requirements Clarification. *Proc. ACM Softw. Eng. (FSE)*, 2024.

OpenCFD. OpenFOAM v2412 Release. Release documentation, 2024.

Seo, M., Baek, J., Lee, S., and Hwang, S. J. Paper2Code: Automating Code Generation from Scientific Papers in Machine Learning. *ICLR*, 2026.

Tian, M. et al. SciCode: A Research Coding Benchmark Curated by Scientists. *NeurIPS*, 2024.

Xie, T. et al. PhysCodeBench: Benchmarking Physics-Aware Symbolic Simulation of 3D Scenes via Self-Corrective Multi-Agent Refinement. arXiv:2604.23580, 2026.

Yue, L. et al. Foam-Agent: A Large Language Model-Based Multi-Agent Framework for Automating Computational Fluid Dynamics Workflows. *Comput. Methods Appl. Mech. Eng.*, 461:119271, 2026.

Zhao, W. et al. Commit0: Library Generation from Scratch. *ICLR*, 2025.

A Complete Controlled Benchmark

The benchmark freezes five canonical problems, each with a hidden machine-readable contract and six specification conditions. Table 5 defines the problems; Table 6 defines the conditions. The model-facing specification for an omitted condition is the FULL specification with the listed lines removed and nothing else changed; the hidden contract is identical across conditions.

Problem	Solver	Defining scientific quantities (checked by the hidden contract)
Lid-driven cavity	icoFoam	$\nu = 0.001$ m^2/s with dimensions [0 2 −1 0 0 0 0]; square domain 0.1×0.1 m; moving lid $U = (1, 0, 0)$, no-slip fixed walls, empty front/back; $Re = 100$; Courant ≤ 1; $t \geq 1$ s
Plane Poiseuille	icoFoam	$\nu = 0.001$; channel 1.0×0.1 m; inlet $U = (0.1, 0, 0)$, outlet $p = 0$, no-slip walls; $Re = 10$; Courant ≤ 1; $t \geq 5$ s
Planar Couette	icoFoam	$\nu = 0.01$; channel 0.5×0.1 m; moving top wall $U = (1, 0, 0)$, no-slip bottom, cyclic inlet/outlet; $Re = 10$; Courant ≤ 1; $t \geq 2$ s
Cylinder flow, $Re = 40$	icoFoam	$\nu = 0.0025$; domain 2.0×1.0 m; cylinder diameter 0.1 m; inlet $U = (1, 0, 0)$, outlet $p = 0$, no-slip cylinder, symmetry top/bottom; $Re = 40$; Courant ≤ 1; $t \geq 2$ s
Slab conduction	laplacianFoam	thermal diffusivity $DT = 0.001$ m^2/s with dimensions [0 2 −1 0 0 0 0]; slab 0.1×0.01 m; left $T = 373$ K, right $T = 273$ K, insulated sides; $\Delta t \leq 0.1$ s; $t \geq 10$ s

Table 5: The five canonical problems and the scientific quantities each hidden contract checks. Geometry uses domain extent (or cylinder diameter) plus a 2D single-cell-z requirement.

Condition	Information removed from the model-facing spec
FULL	nothing (complete specification)
NO_BOUNDARY	the boundary value (lid/inlet/top velocity, or the left temperature)
NO_PHYSICS	the physical property value (ν or DT)
NO_GEOMETRY	the domain length scale (or cylinder domain extent)
NO_NUMERICS	the mesh resolution and timestep
NO_STOPPING	the run duration (endTime)

Table 6: The six specification conditions. Each omission removes only the model-facing information; the hidden contract is unchanged.

Every contract check is tied to the FULL specification and computable from solver-native artifacts: the solver name, the bounding box and 2D setup, the physical-property value and its dimension set, the boundary-condition types and values, the Courant number $\mathrm{Co} = U\,\Delta t/\Delta x$ using the streamwise cell spacing, the timestep and endTime bounds, and the Reynolds number $Re = UL/\nu$ recomputed from the artifact.

B Scientific-Contract Evaluator

The evaluator is deterministic: no LLM judges scientific correctness. It parses the generated files as OpenFOAM dictionaries, checks structural validity, runs `blockMesh`, `checkMesh`, and the solver, and evaluates each contract requirement against the parsed files.

Requirement families and semantics. Solver/model, geometry, physical properties, boundary conditions, and derived consistency are checked for equality within frozen contract-specific relative

tolerances (0.5%–15% across contracts; 10 of 21 equality requirements use 1%). Numerics (Courant, timestep) and stopping (endTime) are checked as validity *bounds* rather than exact values: the scientifically meaningful requirement is a stable, sufficiently long, adequately resolved run, so a finer mesh, a smaller timestep, or a longer run passes. This admissibility semantics is a deliberate construct choice, fixed before the human audit, and is the source of the numerics/stopping disagreement reported in Appendix G.

Units and equivalent representations. Physical properties must carry the correct dimension set (e.g., $[0\ 2\ -1\ 0\ 0\ 0\ 0]$ for kinematic viscosity); this confirms the property is the intended physical quantity. Boundary conditions are checked by type and value. Evaluator v2 additionally recognizes the scientific equivalence `fixedValue uniform (0 0 0)` $\equiv$ `noSlip` on a wall (zero velocity); v1 required the literal `noSlip` type.

Evaluator v1→v2 change. The human audit (Appendix G) found one genuine defect: v1 failed wall boundaries written as `fixedValue uniform (0 0 0)`. v2 accepts this equivalence. The change is a pure addition (it can only turn a fail into a pass for this pattern), is covered by three regression tests, and is confined to the boundary family. No contract content, threshold, or tolerance changed.

C Full Six-Model Results

Table 7 reports per-model metrics on the 25-task matched panel under evaluator v2; Table 8 breaks execution and scientific fidelity down by specification condition.

Model	E	S	ESER	$E0S1$	$E1S0$	$E0S0$
GLM-5.3	0.48	0.52	4/12	5	4	8
Kimi K3	0.40	0.48	5/10	7	5	8
Qwen3.8-Max	0.28	0.64	2/7	11	2	7
Kimi K2.6	0.24	0.56	4/6	12	4	7
DeepSeek-V4-Pro	0.20	0.68	2/5	14	2	6
GPT-OSS-120B	0.04	0.40	1/1	10	1	14

Table 7: Per-model operationality E, scientific fidelity S, ESER (executed-but-wrong / executed), and quadrant counts on the 25-task matched panel (evaluator v2).

Model	FULL	-BC	-PHYS	-GEOM	-NUM	-STOP
GLM-5.3	3/3	2/1	1/1	3/2	1/4	2/2
Kimi K3	1/4	3/2	3/0	0/2	3/4	0/0
Qwen3.8-Max	1/4	2/1	1/2	1/1	1/5	1/3
Kimi K2.6	0/3	1/2	1/3	1/1	1/4	2/1
DeepSeek-V4-Pro	2/4	1/2	0/3	1/2	0/4	1/2
GPT-OSS-120B	0/4	0/1	0/0	1/0	0/3	0/2

Table 8: Per-model per-condition counts on the matched panel, written executed / scientifically-correct (so "3/2" means 3 of the model's generations in that condition executed and 2 satisfied the contract). The NO_PHYSICS column shows the starkest split: Kimi K3 executes 3 but is correct 0, while DeepSeek-V4-Pro executes 0 but is correct 3.

Table 9 breaks contract satisfaction down by requirement family. The models do not fail in the same place: DeepSeek-V4-Pro and Qwen3.8-Max are strongest on the consequential physical properties, Kimi K3 is weakest on them, and the derived-consistency family (Reynolds-number consistency) separates the models most.

Finally, Table 10 gives the outcome of every artifact on the matched panel, so that each entry in Table 7 can be audited cell by cell. The four symbols are the quadrant outcomes: **C** ($E1S1$, executes and satisfies the contract), **E** ($E1S0$, executes but violates the contract), **S** ($E0S1$, satisfies the contract but does not execute), **X** ($E0S0$, neither). Column sums reproduce the quadrant counts in Table 7 exactly.

Model	solver	geom.	phys.	bound.	numerics	stopping	derived
DeepSeek-V4-Pro	0.88	0.86	0.88	0.92	0.88	0.84	0.84
GLM-5.3	1.00	0.94	0.88	0.96	0.96	0.96	0.79
GPT-OSS-120B	0.92	0.82	0.84	0.87	0.84	1.00	0.47
Kimi K2.6	1.00	0.94	0.86	0.97	1.00	1.00	0.79
Kimi K3	1.00	0.97	0.78	0.98	0.96	1.00	0.74
Qwen3.8-Max	1.00	0.94	0.96	0.97	1.00	1.00	0.68

Table 9: Requirement-family pass rate per model on the matched panel (evaluator v2). The cross-model spread is largest for derived consistency (0.37) and smallest for solver and boundary, confirming that models lose scientific fidelity in different places.

Instance	GLM-5.3	Kimi K3	Qwen3.8-Max	Kimi K2.6	DeepSeek-V4-Pro	GPT-OSS-120B
cavity/FULL	X	S	S	S	S	S
cavity/−BC	X	C	C	S	X	S
cavity/−GEOM	C	S	X	X	S	X
cavity/−NUM	S	C	S	X	S	S
cavity/−PHYS	X	E	X	S	S	X
cavity/−STOP	X	X	S	X	X	X
channel/FULL	C	C	S	S	S	S
channel/−BC	X	E	X	X	S	X
channel/−GEOM	C	S	S	S	S	X
channel/−NUM	C	E	S	S	S	S
channel/−PHYS	E	E	S	S	S	X
channel/−STOP	C	X	S	E	S	S
conduction/FULL	C	S	C	S	C	S
conduction/−BC	E	X	E	E	E	X
conduction/−GEOM	E	X	E	E	E	E
conduction/−NUM	S	C	C	C	S	X
conduction/−PHYS	X	E	C	E	X	X
conduction/−STOP	C	X	C	C	C	S
couette/FULL	C	S	S	X	C	S
couette/−BC	E	C	X	S	S	X
couette/−GEOM	X	X	X	X	X	X
couette/−NUM	X	S	S	S	X	S
couette/−PHYS	S	X	X	S	S	X
cylinder/−BC	S	X	X	X	X	X
cylinder/−NUM	S	S	S	S	S	X

Table 10: Per-instance outcome of every generated artifact on the 25-task matched panel (evaluator v2). C $= E1S1$, E $= E1S0$, S $= E0S1$, X $= E0S0$. Summing each column reproduces Table 7.

D Ranking Robustness

The ranking-distortion result is checked against the main threats. (i) *Matched tasks:* all models are compared on the same 25 task IDs. (ii) *Truncation/channel:* excluding truncated or provider-failed generations, and regenerating Kimi K3 through the same relay channel as the other models, leaves the distortion in place (channel-consistent Spearman $\rho = 0.20$). (iii) *Leave-one-model-out:* excluding GPT-OSS-120B (weakest on both axes) the remaining five models give Spearman $\rho = -0.67$, so the distortion is not carried by a single weak model. (iv) *Repeats:* three independent repeats of DeepSeek-V4-Pro and GLM-5.3 give $P(E_{\mathrm{GLM}} > E_{\mathrm{DS}}) = 0.89$, $P(S_{\mathrm{DS}} > S_{\mathrm{GLM}}) = 0.98$, $P(\text{reversal}) = 0.87$ by task-level bootstrap. (v) *Evaluator version:* the headline Spearman is -0.03 under v1 and -0.03 under v2. The primary point estimate ($\rho = -0.03$) has a wide bootstrap interval (95% CI $[-0.46, 0.73]$) because the panel is small; the pairwise reversal is the robust form of the claim.

E EXECUTION-FEEDBACK PROTOCOL

The repair scaffold gives the same model ordinary runtime feedback only. Table 11 records the information the model does and does not see. Table 12 gives the full transition counts; Table 13 gives the per-model breakdown.

Information	Generation	Repair
user specification	yes	yes
generated artifact	—	yes
runtime log (error region)	no	yes
hidden scientific contract	no	no
scientific evaluator feedback	no	no
expected scientific answer	no	no

Table 11: Information access during generation and repair. The repair agent sees the specification, its own artifact, and the runtime error—nothing about the hidden contract or its correctness.

Initial	Final	Count
$E0S1$	$E1S1$	17
$E0S1$	$E0S1$	4
$E0S1$	$E0S0$	3
$E0S1$	$E1S0$	1
$E0S0$	$E1S0$	11
$E0S0$	$E0S0$	4
$E0S0$	$E1S1$	3
$E0S0$	$E0S1$	3

Table 12: Full transition counts over the 46 valid repair targets (two GLM cases with no valid repair attempt excluded).

Model	$E0S1 \to E1S1$	n
DeepSeek-V4-Pro	0.86	14
Kimi K3	0.57	7
GLM-5.3	0.25	4

Table 13: Per-model repair rate of faithful-but-non-executable cases. GLM-5.3's $E0S1$ cases are mostly the hard cylinder O-mesh, which it does not repair.

F REPRESENTATIVE REPAIR TRAJECTORIES

Faithful but broken, repaired (DeepSeek-V4-Pro, cavity FULL; $E0S1 \to E1S1$). The initial case satisfies the contract but the solver aborts with `Entry 'pFinal' not found in system/fvSolution/solvers`. Given only that error, the model adds the missing `pFinal` and `UFinal` solver blocks. The repaired case executes and still satisfies the contract. The fix is a pure software completion.

Broken in both, science stays wrong (DeepSeek-V4-Pro, cavity NO_PHYSICS; $E0S0 \to E1S0$). The initial case uses a wrong viscosity (the omitted quantity) and also fails to run. Runtime feedback fixes the execution failure, but because the error log says nothing about the viscosity, the repaired case runs with the same wrong Re.

Executable but wrong, invisible to feedback (Kimi K3, cavity NO_PHYSICS; $E1S0$). The case executes on the first attempt, so no repair is triggered. Its viscosity is ten times the intended value; nothing in the successful run reveals this, and the artifact remains $E1S0$.

G HUMAN EVALUATION AND EVALUATOR AUDIT

A frozen stratified sample of 50 artifacts (all four quadrants, all five problems, all six models) was scored by two independent blind reviewers against the FULL specification, blind to model identity, the automatic labels, and the study's conclusions.

Reliability and agreement. Human-human agreement is 0.966 requirement-level (Cohen's $\kappa = 0.88$) and 0.96 overall ($\kappa = 0.90$). Blind first-pass agreement between a single reviewer and the frozen v1 evaluator is 0.70 overall; the evaluator passes 26 and the reviewer passes 13, so the evaluator is the more lenient judge.

Disagreement taxonomy. The 38 requirement-level disagreements are: (i) numerics and stopping (19), where the evaluator enforces validity bounds and the reviewers applied exact-match instruction fidelity, mostly in conditions where the exact value was withheld from the model; (ii) the non-isomorphic `solver_2d` rubric field (10), which merges the solver choice and the 2D setup that the evaluator checks separately; (iii) boundary (5), where the evaluator required the literal `noSlip` and the artifact used the equivalent `fixedValue uniform (0 0 0)`; and (iv) one genuinely ambiguous cylinder-geometry case.

Correction and impact. Issue (iii) is a genuine evaluator defect, corrected in evaluator v2 with three regression tests. After correction, the aligned human-to-evaluator agreement is 0.914 per-requirement (boundary and geometry 1.00), and the reviewer-consensus-to-v2 agreement is 0.898 (geometry 0.98, physics 0.94, boundary 0.96). The correction moves ten artifacts from $E0S0$ to $E0S1$ and leaves the executable-but-wrong count, the ESER, the ranking reversal, and the repair asymmetry unchanged. We do not claim humans and the evaluator are identical judges; the useful result is that independent stress testing found a real bug, the bug was fixed, and the headline conclusions survived.

H FOUR-QUADRANT EXAMPLES

$E1S1$ **(operational and faithful).** Kimi K3, cavity NO_BOUNDARY: correct lid velocity reconstructed, no-slip walls, runs to completion.

$E1S0$ **(executable but wrong).** Kimi K3, cavity NO_PHYSICS: runs cleanly, but uses $\nu = 0.01$ instead of 0.001, moving the flow from $Re = 100$ to $Re = 10$. The model discloses the assumption; the disclosure does not make it correct.

$E0S1$ **(faithful but not operational).** DeepSeek-V4-Pro, cavity FULL: the contract is fully satisfied, but `fvSolution` omits the `pFinal` solver block, so the solver aborts. Execution-based evaluation scores this as a failure; it is a software defect on a scientifically correct case.

$E0S0$ **(fails both).** Kimi K3, cavity NO_STOPPING: the case neither runs nor satisfies the contract (wrong viscosity and a missing stopping control).

I STUDY I: CROSS-SYSTEM BREADTH

The cross-system study evaluates generated workflows for four open-source simulation systems (OpenFOAM and SU2 for CFD, CalculiX for finite-element solid mechanics, and OpenPhase for phase-field simulation), each specified in three surface forms. Structural metrics saturate: all 12 base workflows are file-complete, syntactically valid, and mesh-ready; solver execution succeeds in 10/12 and expected output in 9/12, while only 6/12 provide validation-ready evidence under the original protocol. These are historical single-run records retained for breadth; the original execution repository is unavailable, so they are not pooled with the controlled study and carry no controlled claim.

J EXACT PROMPTS AND REPRODUCIBILITY

Generation prompt. The frozen instruction is reproduced verbatim below.

```
You are an expert computational scientist and OpenFOAM user.
You are given a scientific simulation specification from
a user.  Your task is to construct a complete, runnable
OpenFOAM case that implements the scientific problem
described in the specification.
Target environment:  OpenFOAM version 2412 (OpenFOAM.com
distribution) on Linux, running in serial (single core).
Requirements:  1.  Return every required case file with
its relative path and complete content (directories
0/, constant/, system/, including system/blockMeshDict,
system/controlDict, system/fvSchemes and system/fvSolution).
2.  Output format:  for each file, print a line of the
form ### FILE: <relative/path> immediately followed by a
fenced code block containing the complete file content.
Do not truncate file contents.  Return the case as text
only:  do not create, write or execute any files and do not
use any tools.  3.  The case must implement the scientific
problem described by the specification:  geometry, physical
properties, boundary conditions, numerical settings and run
duration must match the specification.  4.  Do not invent
scientific information unnecessarily.  If information that
is necessary to define the scientific problem is missing
from the specification, either request clarification, or
explicitly state each assumption you make (what you assumed
and which value you chose) before giving the case files.  5.
Do not claim that the simulation has executed or that it
produced results; you do not have execution evidence.
```

Repair instruction. The frozen repair instruction is reproduced verbatim below. It is preceded by the specification, the model's own case, and the failing-stage error output.

```
The OpenFOAM case you produced failed to run.  Below is the
output that OpenFOAM produced when the case was executed
(the failing stage and its error output).
Your task:  repair the case so that it runs correctly
under OpenFOAM v2412, while preserving the scientific
setup implied by the specification (geometry, physical
properties, boundary conditions, solver, and time settings
must not be changed relative to the specification).  Fix the
software/configuration problems that caused the failure.
Return the COMPLETE corrected case:  every required
file with its relative path and full content, using the
same format as before (a line ### FILE: <relative/path>
immediately followed by a fenced code block).  Do not
truncate file contents.  Do not claim the case has executed.
```

Reproducibility. The release retains the frozen 30-instance benchmark and hidden contracts (`benchmark.jsonl` SHA256 `135819567...983`), all raw generations, parsed OpenFOAM cases, the deterministic contract checker with unit and regression tests, per-case and per-requirement results, statistical outputs, execution-stage logs, repair trajectories, repeat generations, and the blinded audit package. Execution uses OpenFOAM v2412 (conda-forge) with timeouts of 180 s mesh, 120 s checkMesh, 600 s solver, and 120 s sampling; a run counts as executed only on solver exit code 0 with an `End` marker. Generation uses temperature 0 through an OpenAI-compatible relay; reasoning-heavy models use a 64000-token completion budget (a budget, not a sampling parameter). The evaluator is versioned: v1 is the frozen reference, v2 adds the boundary-equivalence fix.

Statistical details. Bootstrap intervals are percentile intervals over generation-level resamples (10,000 draws, fixed seed), reported descriptively given clustering by five base problems. The ranking bootstrap resamples the 25 matched tasks and recomputes both rankings. Repeat analyses pool three independent generations per model on the same frozen tasks. Truncated and provider-failed generations are excluded from rates, never scored as scientific failures.