PhysVer: Constraining LLM Engineering Reasoning with Domain-Specific Physical Models

Anonymous ACL submission

Abstract

Large language models are increasingly used for scientific and engineering question answering, yet their reasoning often produces plausible but physically incorrect numerical predictions and risk assessments. Existing test-time methods largely rely on the LLM's own signals, making physical hallucinations difficult to detect. We propose PhysVer, a test-time verification framework that constrains LLM engineering reasoning with domain-specific physical models. PhysVer treats a trained domain model as an external verifier independent of the LLM and selects among candidate reasoning chains according to their physical consistency. We also study a verifier-visible Feedback variant, where physical predictions are injected into the prompt for answer revision. We introduce LAM-Bench to evaluate laser–arc hybrid welding. On glm-4-flash, PhysVer improves chain-of-thought (CoT) accuracy from 36% to 75.5%, while Feedback reaches 84.9%. These results show that trusted domain models can provide model-independent verification signals for reliable engineering reasoning without LLM fine-tuning, and further support a bidirectional loop between LLM reasoning and domain model refinement.

1 Introduction

LLM agents are increasingly used as natural-language interfaces for scientific and engineering analysis, as shown in recent work on tool-augmented scientific reasoning such as SciAgent(Ma et al., 2024; Zhang et al., 2025b). Unlike open-domain question answering, engineering tasks require answers to be not only linguistically plausible but also physically consistent(Shi et al., 2026). Numerical predictions should respect material properties and physical constraints, and risk assessments such as "is this process parameter safe?" should agree with quantities observable from simulation or experiment(Al Rashid and Koç, 2023). However, LLMs can generate coherent reasoning chains while making incorrect claims about key physical quantities, threshold decisions, or parameter comparisons(Tyen et al., 2024). We refer to such errors as physical hallucinations: answers that appear plausible in language but are inconsistent with an underlying physical model or simulation.

Existing test-time methods improve LLM reasoning in several ways. Chain-of-thought prompting encourages models to generate intermediate reasoning steps(Wei et al., 2022), while self-consistency samples multiple reasoning paths and aggregates their answers(Wang et al., 2022). Self-verification and process reward models further score answers, rationales, or intermediate steps using language-model-based signals(Weng et al., 2023; Lightman et al., 2024). Tool-augmented methods allow LLMs to call external environments, programs, or scientific tools, as in ReAct(Yao et al., 2023), PAL(Gao et al., 2023), Toolformer(Schick et al., 2023), and SciAgent(Ma et al., 2024). These methods improve reasoning and tool use, but they do not explicitly verify whether an LLM's physical claims agree with an independent domain model. As a result, a reasoning chain may be internally consistent as text while still violating physical constraints.

Our work starts from a simple observation: many engineering domains already have trained domain models for predicting target physical processes. These models may be neural operators(Lu et al., 2021a) for parametric PDEs, such as the Fourier Neural Operator(Li et al., 2021) and the general Neural Operator(Kovachki et al., 2023), simulation-trained predictors, mechanistic solvers, or task-specific regression and classification models. Compared with full-scale simulation or experimental measurement, such models are often cheaper to query and can output task-relevant physical quantities(Schweiger et al., 2020). This makes them natural external verifiers for LLM engineering reasoning: instead of relying on the LLM's self-evaluation

or textual preferences(Jiang et al., 2026), the system can check whether candidate answers are consistent with predicted physical quantities.

Based on this idea, we propose PhysVer, a test-time verification framework that constrains LLM engineering reasoning with domain-specific physical models(Lu et al., 2021b). We wrap a trained domain model as a verifier V_ϕ. Given process parameters, the verifier predicts task-relevant physical quantities(Karniadakis et al., 2021), such as peak temperature, melt-pool depth, or composition deviation. These quantities are then used to evaluate whether the LLM's final answer and explicit physical claims are physically consistent. In our experiments, we instantiate V_ϕ with an offline-trained PCS-FNO model, but the framework is model-agnostic: any domain model that outputs relevant physical quantities can serve as the external verifier.

PhysVer uses a verifier-hidden best-of-N(Khalifa et al., 2026) mechanism at inference time. The LLM samples multiple candidate reasoning chains, each containing a final answer and parseable physical claims. The verifier does not reveal its predictions to the LLM. Instead, it scores candidates in the background according to physical consistency and returns the highest-scoring answer. To study a verifier-visible setting, we also introduce a Feedback variant(Qiao et al., 2024; Gou et al., 2024), where the LLM first produces a draft answer and then revises it after the verifier's physical predictions are injected into the prompt(Bayazıt and Si, 2026). Thus, PhysVer treats the domain model as a hidden judge, while Feedback treats it as a readable physical tool.

To evaluate the framework, we introduce LAM-Bench, a domain-grounded engineering reasoning benchmark for laser–arc hybrid welding with simulation-derived physical labels. On glm-4-flash(GLM et al., 2024), chain-of-thought prompting achieves only 36% overall accuracy, and self-consistency reaches 36.7%. In contrast, verifier-hidden PhysVer achieves 75.5%, while verifier-visible Feedback reaches 84.9%. Further ablations show that the improvement mainly comes from external physical verification rather than simply increasing the number of LLM samples. These results suggest that trusted domain models can provide model-independent verification signals for reliable engineering reasoning without fine-tuning the LLM.

Finally, the same simulation contract can support a bidirectional loop between LLM reasoning and domain model refinement. On the inference side, domain models constrain the physical consistency of LLM outputs; on the training side, LLMs can assist domain model improvement through data selection, error analysis, and iterative refinement. This paper focuses on the inference-side verification problem and leaves the full training-side loop as a framework extension.

Our contributions are as follows:

- **PhysVer**, a test-time verification framework that constrains LLM engineering reasoning with domain-specific physical models as external verifiers.
- A verifier-hidden best-of-N mechanism that selects candidate reasoning chains by physical consistency, together with a verifier-visible **Feedback** variant for answer revision.
- **LAM-Bench**, a benchmark for laser–arc hybrid welding with simulation-derived physical labels.
- Empirical evidence that external physical verification improves LLM engineering reasoning without LLM fine-tuning, together with a bidirectional-loop perspective that connects LLM reasoning and domain model refinement.

2 Related Work

This work studies how domain-specific physical models can serve as external verifiers for reliable LLM engineering reasoning. We focus on three recent and closely related directions: tool-augmented scientific agents, verifier-based test-time reasoning, and scientific or numerical claim verification.

2.1 Tool-Augmented Scientific Agents

Recent work has moved from general tool use toward scientific agents that retrieve, select, and apply domain tools for scientific reasoning(Chen et al., 2025). SciAgent(Ma et al., 2024) constructs tool-augmented scientific reasoning data and evaluates LLMs on SciToolBench, showing that scientific tools can improve problem solving in specialized domains. TAT further studies tool utilization in scientific reasoning by improving tool awareness, tool usage, and cross-domain generalization(Xie et al., 2025). Agentic Reasoning(Wu et al., 2025) integrates external agents such as web search, code

execution, and structured memory into complex reasoning workflows.

PhysVer shares the same motivation that LLMs should not solve scientific and engineering tasks in isolation. However, the external module plays a different role. In tool-augmented(Gridach et al., 2025) scientific agents, tools are usually used to retrieve information, execute computations, or produce intermediate results. In PhysVer, the domain model is primarily used as a physical verifier. In the verifier-hidden setting, the LLM never sees the verifier's predictions; the verifier only ranks candidate reasoning chains according to physical consistency. Thus, the domain model does not directly answer the question, but constrains the LLM's answer.

2.2 Verifier-Based Test-Time Reasoning

Verifier-based reasoning has become an important direction for improving LLM reliability. S2R teaches LLMs to self-verify and self-correct during inference(Ma et al., 2025). R-PRM improves process reward modeling by introducing reasoning-driven process-level evaluation(She et al., 2025).

PhysVer is related to this line of work in its post-generation verification structure, but differs in the source of verification. Existing self-verification and PRM methods typically rely on language-model outputs(Prince et al., 2024), learned reward models, human preference signals, or domain annotations. PhysVer instead uses a domain-specific physical model that is decoupled from the LLM. The verifier does not judge whether a rationale sounds plausible; it checks whether the candidate answer and its explicit physical claims agree with predicted physical quantities. This distinction is important in engineering tasks, where textual plausibility and physical correctness can diverge.

2.3 Scientific and Numerical Claim Verification

Recent work has also focused on scientific and numerical claim verification. Think Right, Not More studies test-time scaling for numerical claim verification by sampling multiple reasoning paths and training a verifier to select reliable paths(Chungkham et al., 2025). PhysReason introduces a physics-based reasoning benchmark and identifies bottlenecks in theorem application, physical process understanding, calculation, and condition analysis(Zhang et al., 2025a). QuantumQA further introduces physical consistency verification for quantum-mechanics question answering(Qu et al., 2026).

These works are close to PhysVer in motivation, because they all address the reliability of scientific or numerical reasoning. The key difference is the grounding source. Existing claim-verification work usually relies on textual evidence, retrieved documents, learned verifiers, or problem-solution consistency checks. PhysVer grounds verification in physical quantities predicted by a domain model. For claims such as "the peak temperature is about 5400 K" or "this parameter setting exceeds the safety threshold," the relevant evidence is not only textual; it is a physical quantity predicted from process parameters. This makes PhysVer closer to engineering simulation and domain-model-based verification.

3 Method

This section presents the proposed **domain-model-guided verification** framework for reliable LLM engineering reasoning. The key idea is to wrap a trained domain-specific physical model as an external verifier V_ϕ, which provides physical consistency signals that are decoupled from the language model. As shown in Figure 1, the same verifier supports two inference-time strategies. The core method, **PhysVer**, uses the verifier as a hidden judge to select the most physically consistent reasoning chain from multiple LLM candidates. We also study a verifier-visible **Feedback** variant, where verifier predictions are injected into the prompt to guide one-step answer revision.

The framework is connected by a unified simulation contract. The field-level domain model and LAM-Bench share the same process parameters, physical quantities, and scalar extraction map Φ. This contract also enables a bidirectional-loop view: on the inference side, the domain model constrains LLM reasoning through physical consistency; on the training side, LLM agents can assist future domain model improvement through data selection, error analysis, and iterative refinement. This paper focuses on the inference-side verification problem.

3.1 Problem Formulation

Taking laser–arc hybrid welding as the example domain, we consider an engineering physical process parameterized by θ. In this work, $\theta = (P, V)$, where P denotes laser power and V denotes scanning speed. Given a natural-language question q and a task type $t \in \mathcal{T}$, the system is required to

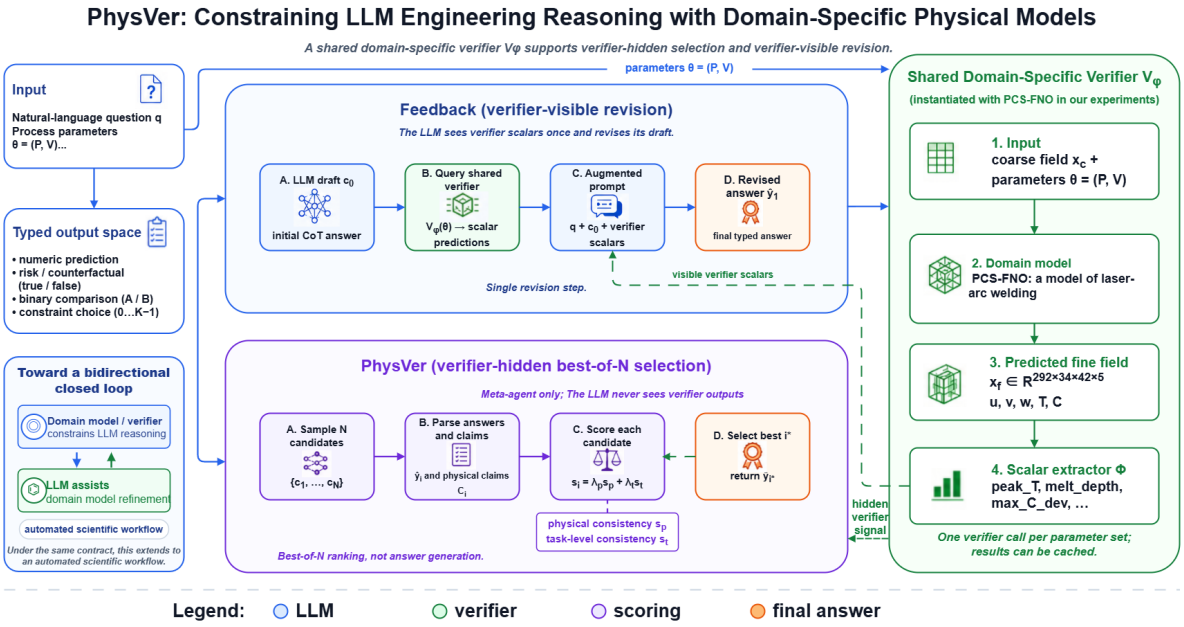

Figure 1: Overview of **domain-model-guided verification** for reliable LLM engineering reasoning. The input consists of a natural-language question and process parameters. A shared domain-specific verifier V_ϕ predicts task-relevant physical quantities from the same process parameters. **PhysVer** treats the verifier as a hidden judge and selects the best candidate reasoning chain according to physical consistency. **Feedback** treats the verifier as a readable physical tool and injects verifier predictions into the prompt for answer revision. The same simulation contract further supports a bidirectional-loop view between LLM reasoning and domain model refinement.

produce a typed answer $\hat{y}$ that matches the task specification. LAM-Bench covers five task types:

- numerical prediction, where $\hat{y} \in \mathbb{R}$;
- risk diagnosis, where $\hat{y} \in \{\text{true}, \text{false}\}$;
- counterfactual judgment, where $\hat{y} \in \{\text{true}, \text{false}\}$;
- binary comparison, where $\hat{y} \in \{A, B\}$;
- constrained selection, where $\hat{y} \in \{0, \ldots, K-1\}$.

Unlike standard chain-of-thought prompting, our setting requires the LLM reasoning chain to expose not only a final answer but also parseable physical claims. Let c denote a reasoning chain generated by the LLM. A parser π extracts the typed answer and explicit physical claims:

$$(\hat{y}, \mathcal{C}) = \pi(c), \tag{1}$$

where $\mathcal{C}$ contains claims such as `peak_T` $\approx$ `5400 K`. These claims are not treated as final answers; instead, they are used for consistency checking against the external verifier.

Besides the LLM L, the system can query an offline-trained domain model f_ϕ and a deterministic scalar extraction map Φ. The domain model predicts physical fields or quantities from process parameters, while Φ extracts task-relevant physical quantities. Together, they define the external verifier:

$$V_\phi(\theta) = \Phi\big(f_\phi(\theta)\big) \in \mathbb{R}^Q. \tag{2}$$

If the domain model directly outputs physical quantities, then $V_\phi = f_\phi$. Given a question instance (q, θ, t), an inference algorithm $\mathcal{A}$ produces the final typed answer:

$$\hat{y} = \mathcal{A}(q, \theta, t;\ L, V_\phi). \tag{3}$$

This work does not fine-tune the LLM. All improvements come from how the external verifier V_ϕ is used at inference time.

3.2 Domain Model as an External Physical Verifier

Existing self-verification and process-reward methods usually rely on LLM-generated signals, language-model-based verifiers, or human preference supervision(Choudhury, 2025). In contrast, we use a domain-specific physical model as an external verifier. The verifier does not generate natural language and does not judge whether a rationale sounds plausible. It predicts task-relevant

physical quantities from process parameters and uses them to check whether the LLM answer and physical claims are consistent.

The verifier should not be viewed as a perfect oracle. Its reliability depends on the accuracy of the underlying domain model, the coverage of the training data, and the scalar extraction map Φ. In this work, V_ϕ is treated as a fixed domain-model component that has been trained offline and independently validated. We do not study how to train the domain model, nor do we treat verifier quality evaluation as the main contribution. Instead, we focus on how an existing trusted domain model can be connected to LLM inference as an external physical verification signal.

In implementation, V_ϕ is queried once for each parameter setting involved in a question. For comparison, counterfactual, and constrained-selection tasks, a single question may contain multiple parameter settings; their verifier outputs are cached and reused. Although our experiments instantiate V_ϕ with PCS-FNO, the framework is model-agnostic: any domain model that outputs task-relevant physical quantities can serve as the verifier.

3.3 PhysVer: Verifier-Hidden Best-of-N Selection

PhysVer is the core inference strategy of this work. It is designed for the setting where the LLM is responsible for natural-language understanding, candidate reasoning, and answer formatting, while the domain model provides an independent physical consistency signal. The verifier is hidden from the LLM: its predictions are not exposed in the prompt and are only used for post-generation candidate selection.

Given a question q, the LLM samples N candidate reasoning chains at temperature $T > 0$:

$$\{c_i\}_{i=1}^{N} \sim L(q; T). \tag{4}$$

For each candidate, the parser extracts the final typed answer and physical claims:

$$(\hat{y}_i, \mathcal{C}_i) = \pi(c_i). \tag{5}$$

The verifier then scores each candidate according to physical-claim consistency and task-level consistency. The final answer is selected as:

$$i^\star = \arg\max_{i:\, s_i \geq 0} s_i, \qquad \hat{y}_{\mathrm{pv}} = \hat{y}_{i^\star}. \tag{6}$$

Physical-claim consistency. For two scalar values u and v, we define the relative error and the consistency kernel as:

$$\begin{aligned} \mathrm{re}(u, v) &= \frac{|u - v|}{\max(|u|, |v|, 1)}, \\ \kappa(u, v) &= \exp\big(-2\,\mathrm{re}(u, v)\big). \end{aligned} \tag{7}$$

For a candidate reasoning chain c_i, the physical-claim consistency score is:

$$s_p(c_i) = \begin{cases} \dfrac{1}{|\mathcal{C}_i|} \displaystyle\sum_{(k,v)\in\mathcal{C}_i} \kappa\big(v, [V_\phi]_k\big), & \mathcal{C}_i \neq \emptyset, \\ 0.5, & \mathcal{C}_i = \emptyset. \end{cases} \tag{8}$$

Here, $[V_\phi]_k$ denotes the verifier prediction for physical quantity k. If a candidate does not contain any explicit physical claim, we assign a neutral score of 0.5 rather than fully penalizing it.

The task-level score $s_t(c_i) \in [0, 1]$ measures whether the candidate final answer agrees with the answer induced by the verifier. For discrete decision tasks, a deterministic rule ρ_t maps the verifier output to a task-level answer:

$$\tilde{y} = \rho_t(V_\phi, t), \tag{9}$$

and the task-level consistency is:

$$s_t(c_i) = \mathbb{1}[\hat{y}_i = \tilde{y}]. \tag{10}$$

For numerical prediction tasks, the task-level consistency is:

$$s_t(c_i) = \kappa(\hat{y}_i, [V_\phi]_q), \tag{11}$$

where $[V_\phi]_q$ is the verifier prediction for the target physical quantity of the question.

The final candidate score is:

$$s_i = \lambda_p s_p(c_i) + \lambda_t s_t(c_i), \qquad \lambda_p = 1, \quad \lambda_t = 1.5. \tag{12}$$

If the parser fails to recover a valid final answer, we set $s_i = -1$. The main experiments use this argmax selection rule for PhysVer. We analyze the effect of the sampling scale in Section 4.4.

3.4 Feedback: Verifier-Visible One-Step Revision

Feedback is a verifier-visible variant that studies whether an LLM can revise its answer when physical predictions are explicitly provided. The LLM first generates a draft reasoning chain at temperature $T = 0$:

$$c_0 \sim L(q; T = 0), \qquad (\hat{y}_0, \mathcal{C}_0) = \pi(c_0). \tag{13}$$

The verifier is then queried to obtain $V_\phi(\theta)$, and the relevant physical quantities are injected into the prompt. The LLM performs one-step revision:

$$\hat{y}_{\text{fb}} = \pi\left(L\big(\text{augment}(q, c_0, V_\phi(\theta)); T = 0\big)\right). \tag{14}$$

Unlike PhysVer, Feedback exposes verifier predictions to the LLM. Thus, Feedback treats the domain model as a readable physical tool, while PhysVer treats it as a hidden judge. Feedback requires two LLM calls and one verifier query per question, without sampling or candidate ranking. It is especially useful for numerical prediction tasks, but it still depends on whether the LLM correctly accepts and uses the injected physical quantities.

3.5 Verifier Instance: PCS-FNO

In our experiments, we instantiate f_ϕ with PCS-FNO(Anonymous, 2026):

$$f_\phi : (x_{\text{c}}, \theta) \mapsto \hat{x}_{\text{f}}, \qquad V_\phi(\theta) = \Phi(\hat{x}_{\text{f}}), \tag{15}$$

where x_{c} is the coarse-grid input field, $\hat{x}_{\text{f}}$ is the predicted fine-resolution field, and Φ extracts task-relevant scalars such as `peak_T`, `melt_depth`, and `max_C_dev`. Here, `peak_T` denotes the maximum temperature over the predicted field, `melt_depth` measures the depth of the melt-pool region under the task-specific threshold, and `max_C_dev` denotes the maximum local deviation of Mg concentration from its reference value.

PCS-FNO is used in this work only as an instance of a trained domain model for constructing the external verifier V_ϕ. It is trained offline on a publicly available laser–arc hybrid welding simulation dataset(Chen et al., 2020, 2018) and predicts fine-resolution physical fields from the given process parameters and input fields. The scalar extraction map Φ then derives task-relevant quantities such as `peak_T`, `melt_depth`, and `max_C_dev`. This paper does not study the training procedure of PCS-FNO, nor does it include the LLM in the domain model training loop. Instead, PCS-FNO is treated as a fixed physical prediction module that provides external verification signals for LLM reasoning. The focus of this work is how to turn an existing domain model satisfying the V_ϕ interface into an external physical verifier for LLM engineering reasoning.

4 Experiments

This section evaluates PhysVer and Feedback on **LAM-Bench**, using aluminum laser–arc hybrid welding as the example engineering domain. The experiments are designed to answer four questions: (i) whether an external physical verifier improves LLM engineering reasoning; (ii) whether verifier-hidden PhysVer improves over test-time methods that rely only on LLM-internal signals; (iii) whether the gains come from physical verification rather than simply increasing the number of LLM samples; and (iv) whether PhysVer and Feedback are complementary across task types. Unless otherwise stated, the main results are reported on `glm-4-flash`, a lightweight base LLM suitable for large-scale test-time sampling. Cross-model results are reported in Section A.5.

4.1 LAM-Bench and Evaluation Protocol

We introduce **LAM-Bench**, a domain-grounded engineering reasoning benchmark for laser–arc hybrid welding. Unlike large-scale general-purpose QA benchmarks, LAM-Bench is designed to evaluate whether LLM answers are consistent with simulation-derived physical quantities in a specific engineering process. The benchmark contains 139 questions covering five task types: numerical prediction, risk diagnosis, counterfactual judgment, binary comparison, and constrained selection. All ground-truth labels are derived from high-fidelity CFD simulation outputs, rather than from PCS-FNO or any other domain model. PCS-FNO is used only as an external verifier during inference and is never used as the evaluation oracle.

LAM-Bench is built around three task-relevant physical quantities: `peak_T`, `melt_depth`, and `max_C_dev`. Most questions are constructed from held-out process conditions. Counterfactual questions may additionally refer to paired settings with the same scanning speed V and different laser power P in order to define parameter changes. Regardless of how a question is constructed, the evaluation label is always derived from simulation-based physical quantities. Each logical question is associated with 3-4 paraphrases to test robustness to linguistic variation.

4.2 Evaluation Metrics

Let M denote the number of questions, y_n the simulation-derived ground truth for question n, and $\hat{y}_n$ the system output. Overall accuracy is defined as:

$$\text{Acc} = \frac{1}{M}\sum_{n=1}^{M} \mathbb{1}[\text{correct}(t_n, \hat{y}_n, y_n)]. \tag{16}$$

Task type	#	Example
Numerical prediction	36	What is the peak temperature at $P = 1500$ W and $V = 0.015$ m/s?
Risk diagnosis	33	Does (P, V) make the melt depth exceed 38 voxels?
Counterfactual judgment	18	If P changes from 1000 W to 1500 W while V is fixed, does `peak_T` exceed 6000 K?
Binary comparison	25	Which setting has a deeper melt pool, $(1000, 0.015)$ or $(1500, 0.020)$?
Constrained selection	27	Among four candidates, which one has `peak_T` closest to 5000 K?

Table 1: Task types in LAM-Bench.

For numerical prediction tasks, we use the same relative error definition as in Section 3.3. A numerical answer is considered correct if $\mathrm{re}(\hat{y}_n, y_n) < 0.15$. Numerical MAPE is defined as:

$$\mathrm{MAPE} = \frac{100}{M_{\mathrm{num}}} \sum_{n \in \mathcal{N}_{\mathrm{num}}} \mathrm{re}(\hat{y}_n, y_n)\ \%. \quad (17)$$

We report **parse-failure rate** rather than hallucination rate. It measures the proportion of outputs from which the parser cannot recover a valid typed answer. This is more precise than calling it hallucination, since physical hallucination refers to physically inconsistent reasoning, while parse failure only measures invalid or unparseable outputs.

We also report verifier-alignment score (VAS), which measures how well explicit physical claims generated by the LLM agree with simulation-derived quantities:

$$\mathrm{VAS} = \exp(-2 \cdot \overline{\mathrm{re}}_{\mathrm{claim}}), \quad (18)$$

where $\overline{\mathrm{re}}_{\mathrm{claim}}$ is the average relative error over all parsed physical claims. Paraphrase variance (PV) measures the variance of accuracy across paraphrases of the same logical question; lower PV indicates better robustness to linguistic variation.

4.3 Main Results

Table 2 reports the main results on `glm-4-flash`. Direct prompting achieves 32.4% accuracy, and CoT improves slightly to 36.0%. Self-Consistency reaches only 36.7% despite using eight samples, showing that additional LLM sampling alone does not reliably fix engineering reasoning errors.

In contrast, **PhysVer** improves accuracy to 75.5% in the stricter verifier-hidden setting. Since the LLM never sees the verifier output, this result shows that external physical verification can improve reasoning reliability without directly feeding physical predictions back into the model. The verifier-visible **Feedback** variant reaches 84.9% accuracy. Feedback and PhysVer correspond to different settings: Feedback treats the domain model as a readable physical tool, while PhysVer treats it as a hidden physical judge. Thus, Feedback demonstrates the practical value of exposing domain-model predictions to LLM agents, whereas PhysVer more directly tests whether external physical signals can improve candidate selection.

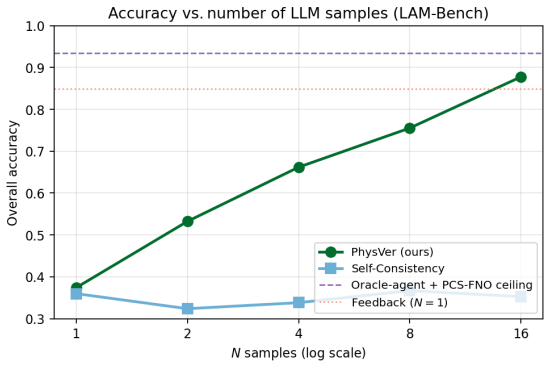

Figure 2: Accuracy as a function of the number of sampled reasoning chains N. PhysVer benefits from larger candidate pools because the external verifier can rank candidates by physical consistency, while Self-Consistency remains nearly flat.

Hybrid is reported only as an analysis of task-level complementarity. It is not treated as the main method.

4.4 Effect of Sampling Scale

Figure 2 compares PhysVer and Self-Consistency as the number of samples N changes. Self-Consistency provides limited improvement as N increases, suggesting that sampling more chains does not guarantee physical correctness. PhysVer improves more consistently with larger N, indicating that additional candidates become useful only when an external physical verifier can identify the more physically consistent ones. Thus, the benefit of PhysVer comes not from sampling alone, but from verifier-guided selection.

4.5 Task-Level Analysis and Complementarity

Figure 3 and Table 3 show task-level accuracy. Feedback performs best on numerical prediction,

Method	Acc.↑	ParseFail↓	VAS↑	MAPE↓	PV↓	#LLM
Direct	0.324	0.050	0.000	87.92	0.211	139
CoT	0.360	0.000	0.000	76.59	0.176	139
Self-Consistency, $N = 8$	0.367	0.000	0.135	56.96	0.144	1112
PhysVer, $N = 8$	0.755	0.000	**0.432**	27.61	**0.085**	1112
Feedback	0.849	0.000	0.414	**2.00**	0.087	278
Hybrid, analysis routing	0.899	0.000	0.423	**2.00**	0.074	1390
Oracle-agent + PCS-FNO	0.935	0.000	–	1.58	–	0
Oracle-agent + simulation oracle	1.000	0.000	–	0.00	–	0

Table 2: Main results on `glm-4-flash` over 139 LAM-Bench questions. Hybrid is included only as an analysis of task-level complementarity, not as the main method.

Method	Numerical	Risk	Counterfactual	Comparison	Selection
Direct	0.000	0.455	0.611	0.640	0.111
CoT	0.111	0.545	0.556	0.560	0.148
Self-Consistency, $N = 8$	0.028	0.700	0.444	0.600	0.148
Feedback	**1.000**	0.939	0.944	**0.920**	0.407
PhysVer	0.528	**1.000**	**1.000**	0.800	**0.556**
Oracle + PCS-FNO	1.000	1.000	1.000	0.960	0.700
Oracle + trilinear	0.667	0.758	0.833	0.880	0.519
Oracle + simulation oracle	1.000	1.000	1.000	1.000	1.000
# Questions	36	33	18	25	27

Table 3: Task-level accuracy across five LAM-Bench task types.

since verifier predictions are explicitly injected into the prompt and can be directly used for answer revision. PhysVer is stronger on several decision-oriented tasks, such as risk diagnosis, counterfactual judgment, and constrained selection, where candidate answers need to be consistent with physical rules and thresholds.

We also report a task-level routing variant as an analysis of complementarity. This routing uses Feedback for numerical prediction and binary comparison, and PhysVer for the other decision-oriented tasks:

$$\hat{y}_{\text{hyb}} = \begin{cases} \hat{y}_{\text{fb}}, & t \in \mathcal{T}_{\text{fb}}, \\ \hat{y}_{\text{pv}}, & t \notin \mathcal{T}_{\text{fb}}. \end{cases} \tag{19}$$

Hybrid is not treated as the core method. It is reported only to show that verifier-visible revision and verifier-hidden selection are complementary.

5 Conclusion

This paper introduced PhysVer, a test-time verification framework that constrains LLM engineering reasoning with domain-specific physical models. PhysVer uses a domain model as a hidden verifier to select candidate reasoning chains by physical consistency. We also studied a verifier-visible Feedback variant for answer revision.

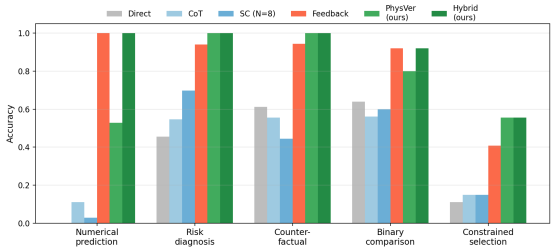

Figure 3: Task-level accuracy across five LAM-Bench task types. Feedback is strongest on numerical prediction, while PhysVer is more effective on several decision-oriented tasks.

On LAM-Bench, PhysVer improves CoT accuracy from 36.0% to 75.5% on glm-4-flash, while Feedback reaches 84.9%. These results show that existing domain models can serve as LLM-independent verification signals to improve engineering reasoning reliability without fine-tuning the LLM.

Limitations

PhysVer requires an available domain model, so its effectiveness depends on the model's parameter coverage and prediction accuracy. Our experiments focus on laser–arc hybrid welding and mainly evaluate inference-time verification. The full loop in which LLMs assist domain model refinement is left for future work.

References

Ans Al Rashid and Muammer Koç. 2023. Numerical predictions and experimental validation on the effect of material properties in filament material extrusion. *Journal of Manufacturing Processes*, 94:403–412.

Anonymous. 2026. Pcs-fno: Physics-coupled spectral fourier neural operator for small-data multiphysics prediction. Anonymous manuscript under review.

Barış Bayazıt and Xujie Si. 2026. RESpecbench: How reliable is LLM-as-a-judge? rigorous evaluation of specification generation with automated verification.

Xuyang Chen, Gang Yu, Xiuli He, Shaoxia Li, and Zhiyong Li. 2020. Numerical study of heat transfer and solute distribution in hybrid laser-mig welding. *International Journal of Thermal Sciences*, 149:106182.

Xuyang Chen, Gang Yu, Xiuli He, Shaoxia Li, and Haibin Miao. 2018. Effect of droplet impact on molten pool dynamics in hybrid laser-mig welding of aluminum alloy. *The International Journal of Advanced Manufacturing Technology*, 96:209–222.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. 2025. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations*.

Sanjiban Choudhury. 2025. Process reward models for llm agents: Practical framework and directions. *ArXiv*, abs/2502.10325.

Primakov Chungkham, Venktesh V, Vinay Setty, and Avishek Anand. 2025. Think right, not more: Test-time scaling for numerical claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24345–24363, Suzhou, China. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, and 40 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. 2025. Agentic ai for scientific discovery: A survey of progress, challenges, and future directions. *Preprint*, arXiv:2503.08979.

Chunyang Jiang, Yonggang Zhang, Yiyang Cai, Chi-Min Chan, Yulong Liu, Mingming Chen, Wei Xue, and Yike Guo. 2026. Semantic voting: A self-evaluation-free approach for efficient llm self-improvement on unverifiable open-ended tasks. *Preprint*, arXiv:2509.23067.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2026. Process reward models that think. *Transactions on Machine Learning Research*. J2C Certification.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2023. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2021. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021a. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021b. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3).

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. 2025. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pages 22632–22654, Vienna, Austria. Association for Computational Linguistics.

Yubo Ma, Zhibin Gou, Junheng Hao, Ruochen Xu, Shuohang Wang, Liangming Pan, Yujiu Yang, Yixin Cao, and Aixin Sun. 2024. SciAgent: Tool-augmented language models for scientific reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15701–15736, Miami, Florida, USA. Association for Computational Linguistics.

Michael H. Prince, Henry Chan, Aikaterini Vriza, Tao Zhou, Varuni K. Sastry, Matthew T. Dearing, Ross J. Harder, Rama K. Vasudevan, and Mathew J. Cherukara. 2024. Opportunities for retrieval and tool augmented large language models in scientific facilities. *npj Computational Materials*, 10(1):251.

Shuofei Qiao, Honghao Gui, Chengfei Lv, Qianghuai Jia, Huajun Chen, and Ningyu Zhang. 2024. Making language models better tool learners with execution feedback. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3550–3568, Mexico City, Mexico. Association for Computational Linguistics.

Songxin Qu, Tai-Ping Sun, Yun-Jie Wang, Huan-Yu Liu, Cheng Xue, Xiao-Fan Xu, Han Fang, Yang Yang, Yu-Chun Wu, Guo-Ping Guo, and Zhao-Yun Chen. 2026. Quantumqa: Enhancing scientific reasoning via physics-consistent dataset and verification-aware reinforcement learning.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

G. Schweiger, H. Nilsson, J. Schoeggl, W. Birk, and A. Posch. 2020. Modeling and simulation of large-scale systems: A systematic comparison of modeling paradigms. *Applied Mathematics and Computation*, 365:124713.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-PRM: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13438–13451, Suzhou, China. Association for Computational Linguistics.

Qi Shi, Ye Yang, Jiaqi Di, Lin Li, Jie Liu, Youmin Hu, Jinhua Xiao, and Zhongxu Hu. 2026. Trust-aware kg-rag: a framework for trustworthy question answering for engineering equipment. *Advanced Engineering Informatics*, 74:104801.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13894–13908, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, Singapore. Association for Computational Linguistics.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. 2025. Agentic reasoning: A streamlined framework for enhancing LLM reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28489–28503, Vienna, Austria. Association for Computational Linguistics.

Wenjing Xie, Xiaobo Liang, Juntao Li, Wanfu Wang, Kehai Chen, Qiaoming Zhu, and Min Zhang. 2025. From awareness to adaptability: Enhancing tool utilization for scientific reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8811–8831, Vienna, Austria. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Xinyu Zhang, Yuxuan Dong, Yanrui Wu, Jiaxing Huang, Chengyou Jia, Basura Fernando, Mike Zheng Shou, Lingling Zhang, and Jun Liu. 2025a. PhysReason: A comprehensive benchmark towards physics-based reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16593–16615, Vienna, Austria. Association for Computational Linguistics.

Y. Zhang, S. A. Khan, A. Mahmud, H. Yang, A. Lavin, M. Levin, J. Frey, J. Dunnmon, J. Evans, A. Bundy, S. Dzeroski, J. Tegner, and H. Zenil. 2025b. Exploring the role of large language models in the scientific method: From hypothesis to discovery. *npj Artificial Intelligence*, 1(1):14.

A Additional Results

A.1 Example Workflow

Table 4 gives a text-based example workflow of PhysVer and Feedback on a LAM-Bench risk-diagnosis question. The example illustrates how the LLM generates multiple candidate reasoning chains, how the hidden physical verifier scores their physical consistency, and how the verifier-visible Feedback variant revises the answer when physical predictions are exposed.

A.2 Baselines and Inference Settings

We compare PhysVer and Feedback with several test-time baselines. **Direct** uses a single completion at $T = 0$. **CoT** uses a single chain-of-thought completion at $T = 0$. **Self-Consistency** samples $N = 8$ reasoning chains at $T = 0.8$ and aggregates answers by taking the median for numerical tasks and majority vote for discrete tasks. **PhysVer** also samples $N = 8$ reasoning chains at $T = 0.8$, but selects the final answer using the hidden physical verifier. **Feedback** first generates a draft answer and then injects verifier-predicted physical quantities into the prompt for one-step revision. We also report **Oracle-agent** as a reference upper bound. Oracle-agent does not call the LLM; it directly maps verifier outputs to task answers using deterministic task rules.

A.3 Verifier Signal Ablation

To analyze the role of external physical signals in PhysVer, we compare different verifier signal sources: no verifier, trilinear interpolation, PCS-FNO, and simulation oracle. This experiment is not intended to re-evaluate the training quality of PCS-FNO. Instead, it asks how PhysVer changes when the external physical signal becomes weaker or stronger.

Table 6 shows the results. Without a verifier, the method reduces to Self-Consistency. Trilinear interpolation provides a weaker but still useful physical signal and substantially improves over no verifier. PCS-FNO further improves performance, showing that a stronger domain model provides more useful candidate-ranking signals. Replacing the verifier with the simulation oracle yields additional gains, but the improvement is limited, suggesting that PCS-FNO already provides an effective external verification signal.

A.4 Robustness and Error Modes

To analyze the concrete gains over CoT, we compute rescue and demote cases. Let $\mathcal{R}$ denote the set of questions that are answered correctly by a verifier-based method but incorrectly by CoT. Let $\mathcal{D}$ denote the set of questions that are answered correctly by CoT but incorrectly by the verifier-based method. The net gain is:

$$\Delta_{\text{net}} = \frac{|\mathcal{R}| - |\mathcal{D}|}{M} \times 100\%. \tag{20}$$

The results show that both Feedback and PhysVer produce many more rescue cases than demote cases. PhysVer mainly improves by selecting physically consistent candidates from the sampled pool. Feedback mainly improves by revising numerical answers after explicit physical quantities are injected into the prompt.

A.5 Cross-Model Generalization

Finally, we evaluate the methods across four base LLMs: `glm-4-flash`, `gpt-4o-mini`, `gpt-4.1-mini`, and `gpt-4o`. Table 8 reports overall accuracy. Feedback remains strong across models, suggesting that explicit physical feedback transfers reliably across base LLMs. PhysVer depends more on the quality of the candidate pool. When the base model can sample candidates that contain correct or near-correct answers, the hidden verifier can select more reliable outputs. When the candidate pool is weak, PhysVer is limited by candidate availability.

B Artifact Use and License

LAM-Bench is constructed in this paper and will be released for research use after the review period. The released artifacts will include the benchmark questions, task labels, evaluation scripts, and necessary derived files for reproducing the reported results. External simulation data and related resources are cited where appropriate, and their original terms of use should be followed. We do not redistribute third-party artifacts beyond derived benchmark annotations and evaluation files needed for this study. LAM-Bench contains engineering questions, task labels, and simulation-derived physical quantities; it does not contain personally identifying information or intentionally offensive content.

Stage	Example
User query	For laser power $P = 1800$ W and scan speed $V = 0.0210$ m/s, would the peak temperature exceed the safety threshold of 6000 K? Reply with yes or no.
LLM candidates	The LLM samples multiple candidate reasoning chains: **Candidate 1:** The peak temperature may be around 5600 K, which is below 6000 K. Final answer: no. **Candidate 2:** The peak temperature is likely around 6600 K, which exceeds 6000 K. Final answer: yes. **Candidate 3:** The temperature is close to the threshold, but probably safe. Final answer: no.
Hidden verifier	The domain-specific verifier is queried with the same process parameters. It predicts the task-relevant physical quantity: $V_\phi(P = 1800, V = 0.0210): \quad \texttt{peak_T} \approx 6579\ \mathrm{K}.$ According to the task rule, the verifier-induced answer is yes, since $6579 > 6000$.
PhysVer selection	PhysVer does not expose the verifier prediction to the LLM. Instead, the meta-agent compares each candidate's physical claims and final answer with the verifier signal. Candidate 2 receives the highest physical-consistency score and is selected as the final answer: yes.
Feedback revision	In the verifier-visible setting, the verifier prediction is injected into the prompt. The LLM receives the physical quantity peak_T ≈ 6579 K and revises the answer to yes.

Table 4: Text-based example workflow of verifier-guided engineering reasoning. PhysVer uses the verifier as a hidden judge for candidate selection, while Feedback exposes the verifier prediction as a readable physical tool for answer revision.

Method	Configuration	Verifier visibility
Direct	Single completion, $T = 0$	None
CoT	Single reasoning chain, $T = 0$	None
Self-Consistency	$N = 8$, $T = 0.8$, answer aggregation	None
PhysVer	$N = 8$, $T = 0.8$, physical-consistency selection	Hidden judge
Feedback	Draft + verifier injection + revision	Visible tool
Oracle-agent	Verifier output + task rule, no LLM	Not applicable

Table 5: Inference methods and test-time configurations.

Signal	Acc.	VAS	MAPE	Num. Acc.
None / SC	0.367	0.135	56.96%	0.028
Trilinear	0.619	0.432	29.40%	0.444
PCS-FNO	**0.755**	**0.432**	27.61%	0.528
Simulation oracle	0.799	0.432	27.57%	0.528

Table 6: Verifier signal ablation for PhysVer with $N = 8$.

Task	PhysVer rescue/demote	Feedback rescue/demote
Numerical prediction, 36 questions	15 / 0	32 / 0
Risk diagnosis, 33 questions	15 / 0	15 / 2
Counterfactual, 18 questions	8 / 0	7 / 0
Binary comparison, 25 questions	7 / 1	10 / 1
Constrained selection, 27 questions	14 / 3	11 / 4
Total	**59 / 4**	**75 / 7**
Net gain	+39.6%	+48.9%

Table 7: Rescue and demote cases relative to CoT over 139 questions.

Method	**glm-4-flash**	**gpt-4o-mini**	**gpt-4.1-mini**	**gpt-4o**
CoT	0.360	0.353	0.410	0.374
Self-Consistency, $N = 8$	0.367	0.360	0.388	0.338
Feedback	0.849	0.842	0.899	0.892
PhysVer, $N = 8$	0.755	0.655	0.683	0.683
Hybrid, analysis routing	0.899	0.748	0.842	0.813

Table 8: Cross-model results on LAM-Bench. Hybrid is included only as an analysis of task-level complementarity.