SymPyBench: A Dynamic Benchmark for Scientific Reasoning with Executable Python Code

Shima Imani, Seungwhan Moon, Adel Ahmadyan, Lu Zhang, Kirmani Ahmed, Babak Damavandi

Meta Reality Labs

Correspondence: shanemoon@meta.com

Abstract

We introduce SymPyBench, a large-scale synthetic benchmark of 15K university-level physics problems (90/10% train/test split). Each problem is *fully parameterized*, supporting an effectively infinite range of input configurations, and is accompanied by structured, step-by-step reasoning and executable Python code that produces the ground-truth solution for any parameter set. The benchmark contains three question types: MC-Symbolic (multiple-choice with symbolic options), MC-Numerical (multiple-choice with numerical options), and free-form (open-ended responses). These diverse formats test complementary reasoning skills. By leveraging the dynamic, code-driven nature of the benchmark, we introduce three novel evaluation metrics in addition to standard accuracy: Consistency Score, Failure Rate, and Confusion Rate, that quantify variability and uncertainty across problem variants. Experiments with state-of-the-art instruction-tuned language models reveal both strengths and limitations in scientific reasoning, positioning SymPyBench as a foundation for developing more robust and interpretable reasoning systems.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive capabilities across a wide range of natural language processing tasks (Kojima et al., 2022; Anthropic; Bai et al., 2023; Grattafiori et al., 2024). Despite this progress, their proficiency in domain-specific, structured reasoning, particularly within scientific disciplines such as physics, remains limited (Ahn et al., 2024; Lewkowycz et al., 2022; Chang et al., 2024).

Solving physics problems requires the integration of multiple reasoning steps, the precise application of physical laws, and careful mathematical rigor (Larkin and Reif, 1979; Hegde and Meera, 2012; Reif and Heller, 1982). While existing benchmarks are valuable for evaluating factual recall and fundamental scientific knowledge, they do not fully capture the complexity of structured, step-by-step reasoning that is essential in physics and related domains[1]. Moreover, these benchmarks do not support systematic variation of numerical parameters or linguistic formulations, which limits their ability to effectively evaluate and audit model performance.

To address these limitations, we introduce **SymPyBench**, a dynamic benchmark for physics-based reasoning comprising 15,045 problem instances paired with executable Python code. Our contributions are:

Dynamical Generalization. SymPyBench features systematically parameterized physics problems, where each question can be instantiated with varied input variables. Every instance is accompanied by step-by-step reasoning and executable Python code that produces the corresponding ground-truth solution. The benchmark includes three question types that test complementary reasoning skills. *MC-Symbolic* questions are multiple-choice with symbolic options and primarily evaluate symbolic and algebraic reasoning. *MC-Numerical* questions are multiple-choice with numerical answers, testing a model's ability to perform calculations and apply formulas accurately. *free-form* questions are open-ended and assess the model's ability to generate solutions without any hints, often involving multiple sub-questions or intermediate steps. An example of our benchmark is shown in Figure 1.

Metrics Beyond Accuracy. SymPyBench enables systematic evaluation of LLMs through controlled perturbations of problem inputs and linguistic expressions, allowing researchers to probe model behaviors and reveal reasoning patterns. Un-

[1]Examples from prior benchmarks in Appendix 11.

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track), pages 105–118

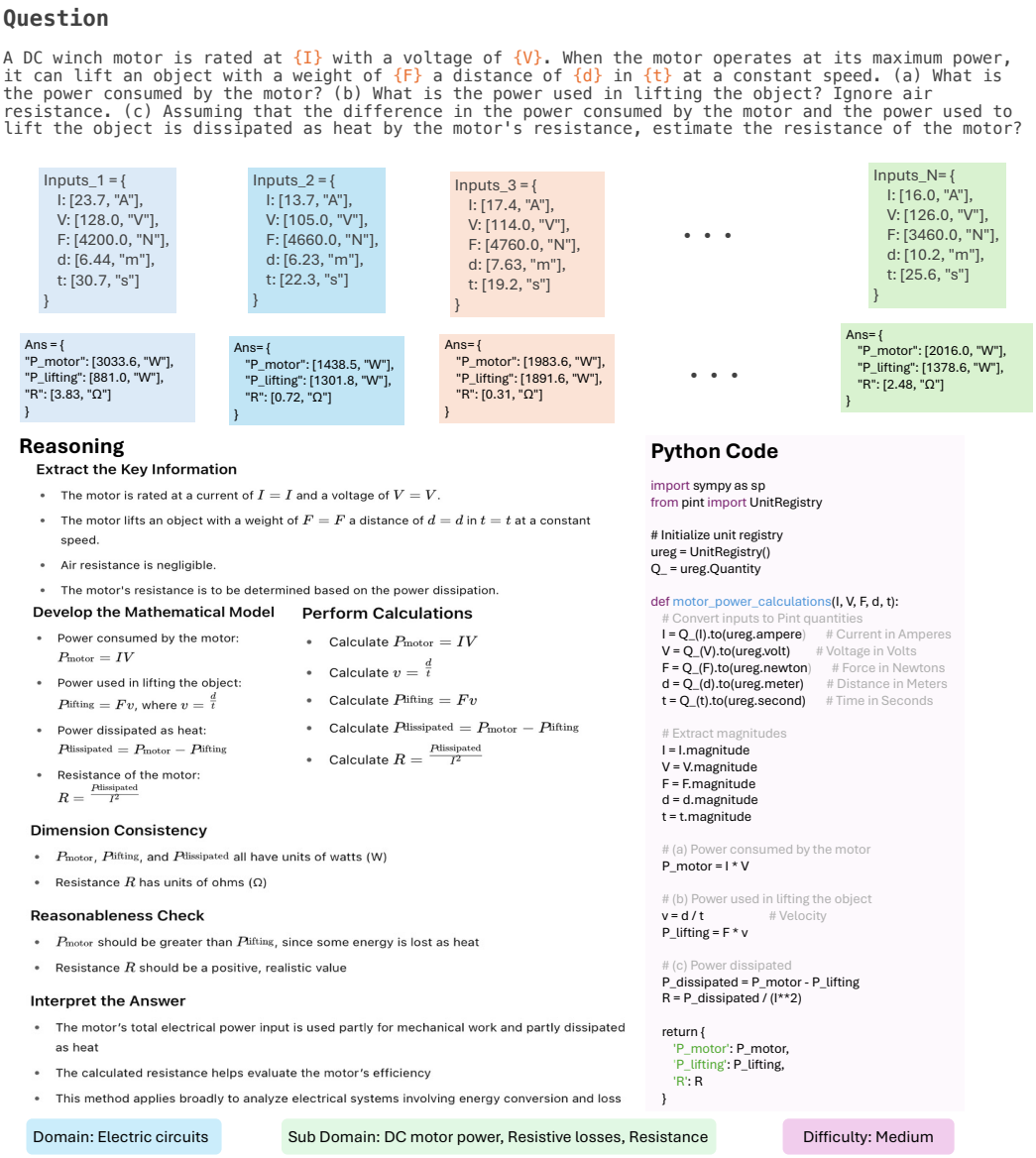

Figure 1: An example from the **SymPyBench** dataset illustrating a free-form physics question. The figure illustrates a parameterized problem with variable input parameters, the final answer, detailed step-by-step reasoning, and the associated executable Python code. The question includes metadata such as domain, subdomain, and difficulty.

like existing benchmarks that rely on a single problem instance, our dynamic design creates multiple problem variants, enabling a more nuanced assessment of model performance. We introduce novel metrics (*Consistency Score*, *Failure Rate*, and *Confusion Rate*) to capture variability and uncertainty in model reasoning across variants. By analyzing performance across multiple variants, we can determine whether a model consistently applies the correct solution strategy or exhibits inconsistent behavior, failing to generalize across similar problems, thereby providing a more comprehensive understanding of its strengths and weaknesses.

2 Related Work

The development of science benchmarks such as ScienceQA (Lu et al., 2022), SciBench (Wang et al., 2023), and physics-specific datasets like PhysBench from MMLU (Hendrycks et al., 2021) has been instrumental in advancing the evaluation of LLMs on structured reasoning tasks. These benchmarks provide valuable testbeds for assessing baseline scientific knowledge and reasoning skills. Several physics, focused resources, including PhysBench (Hendrycks et al., 2021), SciEval (Sun et al., 2024), and JEEBench (Arora et al., 2023), primarily adopt multiple-choice formats, which enable

Dataset	Number of Problems	Academic Level	Step-by-step Reasoning	Numerical Variation Q&A	Textual Variation Q&A	Python Code	Unit Validation
ScienceQA (Lu et al., 2022)	4,546	Elem. & Highschool	✗	✗	✗	✗	✗
SciBench (Wang et al., 2023)	594	Highschool	✗	✗	✗	✗	✗
SciEval (Sun et al., 2024)	1,657	Mixed	✗	Partial	✗	✗	✗
JEEBench (Arora et al., 2023)	512	Highschool	✗	✗	✗	✗	✗
MMLU Physics (Hendrycks et al., 2021)	124	Highschool	✗	✗	✗	✗	✗
PhysicsQA (Jaiswal et al., 2024)	370	Highschool	✓	✗	✗	✗	✗
SymPyBench (Ours)	**15,045**	Undergraduate	✓	✓	✓	✓	✓

Table 1: **High-level comparison of physics benchmarks.** 'Partial' indicates limited (2–3 variations) or inconsistent support.

standardized evaluation at scale. Many existing benchmarks lack detailed, step-by-step solutions and do not explicitly support symbolic computation, which are essential in scientific disciplines such as physics. Table 1 summarizes the key differences between SymPyBench and existing scientific reasoning datasets across several dimensions.

While benchmarks are foundational, robust evaluation protocols are equally critical to understand model behavior under variation. Typical evaluations of LLMs often report a single performance metric per dataset, reflecting best-case results under idealized or carefully curated settings. This obscures important dimensions of robustness and reliability (Zhu et al., 2024; Bommasani et al., 2023).

PromptBench (Zhu et al., 2024) provides a flexible toolkit for robustness testing, with modules for prompt creation, adversarial generation, and analysis. However, its adversarial prompts can alter input semantics, reducing realism. HELM (Bommasani et al., 2023) uses a broader evaluation across metrics like fairness and efficiency, but its robustness tests are limited to minor surface changes and basic contrastive examples (Gardner et al., 2020). Building on these efforts, **SymPyBench** introduces a dynamic, parameterized benchmark designed to evaluate model consistency and generalization under controlled variations.

3 Methodology

We begin by collecting open-source problem sets covering a broad range of undergraduate-level physics topics. The topic distribution reflects the emphasis typically found in standard Bachelor of Science in Physics curricula. All content is sourced from openly available, Creative Commons–licensed materials.[2]

Problem Extraction. We apply OCR via Tesseract to extract text from each problem and detect any reliance on figures, diagrams, tables, or references to other questions. Dependencies are identified using keyword search and pattern matching. Problems flagged as dependent are excluded, resulting in a dataset of predominantly self-contained, text-based questions suitable for our benchmark. While some dependencies may remain undetected, this step effectively filters most problems that rely on visual or contextual information and prepares the dataset for subsequent processing.

Structured Representation. For each problem, we generate a structured textual representation using the LLaMA-3.2-90B-Vision-Instruct model (Grattafiori et al., 2024). The model is prompted to reformat the original problem into a standardized schema, as illustrated in Figure 2. This process consists of five stages, producing the following components:

1. **Question:** A clean, self-contained restatement of the problem in natural language.
2. **Reasoning Step:** A detailed, step-by-step textual explanation outlining the relevant physical principles and intermediate computations.
3. **Input Variables:** Numerical quantities with their associated physical units (e.g., $v1$: `[2, m/s]`).
4. **Output Variables:** Target quantities with their expected final values and units (e.g., v_a: `[-3, m/s]`).
5. **Constants:** Known physical constants such as gravitational acceleration (e.g., g: `[9.8, m/s`^{2}`]`).

At the conclusion of this process, each question is represented in a structured JSON format encom-

[2]The dataset is released under a CC-BY-NC license. Portions of the data were generated using LLaMA 3.2 and 3.3 models and are subject to their respective licenses(https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/LICENSE; https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/LICENSE).

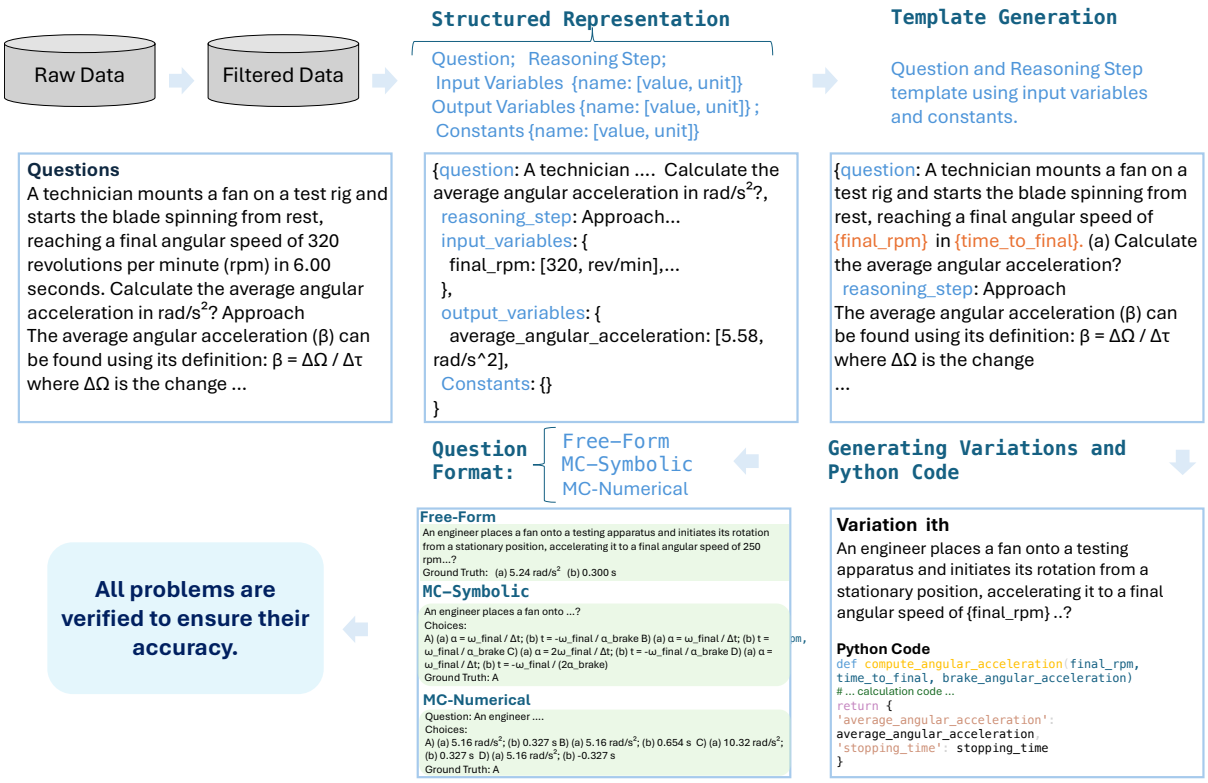

Figure 2: High-level pipeline diagram summarizing the workflow of creating SymPyBench.

passing these five components. This structured representation serves as the foundation for generating parameterized code and enables systematic variation across problem instances.

Template Generation. Building on the structured representation, we prompt the LLM to generate parameterized problem templates. Specifically, the model populates both the question text and the step-by-step solution using symbolic placeholders drawn from the `Input Variables`, `Output Variables`, and `Constants` fields (e.g., v_1, F, g). This ensures that every symbolic reference in the reasoning aligns with the corresponding schema entry, maintaining coherence across intermediate steps and enabling consistent substitution of values across problem variations, as illustrated in Figure 2.

Generating Variations and Python Code. This step consists of two phases: generating textual variations and synthesizing executable Python code.

Textual Variation Generation. Given the parameterized template from the previous step, we prompt the model to generate three textual variations of each question. These variations diversify the linguistic phrasing while preserving the underlying problem structure and symbolic placeholders, ensuring linguistic diversity in the benchmark.

Python Code Synthesis. We then prompt the model to generate executable Python code that solves each problem. The prompt specifies: (1) the function signature, with `Input Variables` and `Constants` keys as input parameters, (2) the expected return format, a dictionary mapping `Output Variables` keys to their computed values, and (3) the `Reasoning Steps` to guide the solution logic. This structured guidance enables the model to systematically translate the symbolic solution into executable code, as illustrated in Figure 2. We employ few-shot prompting with high-quality examples to improve code generation reliability and consistency.

To validate correctness, we execute each generated Python function by substituting the original numerical values and units from the structured representation. If the computed outputs match the expected `Output Variables` (accounting for numerical tolerance and unit equivalence), we retain the code in our dataset; otherwise, we discard it. By iteratively refining prompts, we are able to generate correct code for approximately 88% of problems. For the remaining cases, the generated code does not pass our validation tests.

To ensure correctness and dimensional consistency, all generated code relies on well-established

Mechanics	**33.80%**	**Electricity and Magnetism**	**26.76%**	**Modern Physics**	**12.68%**
Kinematics		*Electric Current*		*Quantum Mechanics*	
SUVAT Equations		*Electric Field*		*Special Relativity*	
Projectile Motion		*Lorentz Force*		*Photon Energy*	
Thermodynamics	**8.45%**	**Waves and Oscillations**	**11.27%**	**Optics**	**7.04%**
Kinetic Theory of Gases		*Wave Motion*		*Geometric Optics*	
Ideal Gas Law		*Frequency*		*Polarization*	
RMS Speed		*Doppler Effect*		*Electromagnetic Waves*	

Table 2: Distribution of problems across physics domains and their top three subdomains in SymPyBench.

tools (Newell et al., 1972; Meurer et al., 2017; pin, 2025). Specifically, Pint enforces unit consistency across all numerical operations, preventing unit-related errors, while SymPyBench facilitates symbolic algebra, equation solving, and analytical manipulation, enabling precise mathematical handling throughout the benchmark.

Question Format Generation To enable robust and diverse evaluation, we generate problems in three distinct formats: free-form, multiple-choice symbolic (MC-Symbolic), and multiple-choice numerical (MC-Numerical). Our dataset comprises 71.52% free-form questions, 14.24% MC-Symbolic questions, and 14.24% MC-Numerical questions. Note that MC-Symbolic and MC-Numerical formats are generated only for problems with a single sub-question, as many problems in our dataset contain multiple sub-questions. For numerical instantiation, we sample random values for all `Input Variables` with controlled perturbation (typically ± 20 to 50% of the original values) and substitute them into each question.

Free-Form Questions. Free-form questions are directly derived from the textual variations generated in the previous step. In this format, models must produce numerical answers with appropriate units.

MC-Symbolic Questions. For each MC-Symbolic problem, we generate a multiple-choice question to assess symbolic reasoning capabilities. The correct symbolic answer is obtained by prompting the model to generate an algebraic expression that matches the output of the reference Python implementation. To validate correctness, we substitute N random sets of input variables (typically $N = 20$) into both the symbolic expression and the Python code, verifying that outputs match across all test cases.

After validating the correct answer, we generate three distractors by prompting the model to produce small, plausible modifications to the correct expression (e.g., sign changes, term omissions, or altered variable combinations). Each distractor is designed to be algebraically similar yet unambiguously incorrect. To mitigate positional bias, we randomize the order of the four answer choices.

MC-Numerical Questions. For MC-Numerical problems, we substitute the sampled numerical values into the MC-Symbolic format, yielding four numerical options.

Dataset Quality All problems are manually reviewed to ensure correctness. Table 3 shows the percentage of error for each step in each step.

Dataset Composition. SymPyBench features a diverse set of problems with three types of variations: (1) linguistic variation with three distinct phrasings, (2) format variation with three question formats (free-form, MC-Symbolic, MC-Numerical), and (3) numerical variation with theoretically infinite instantiations. In addition, each problem is annotated with relevant keywords, including domain, sub-domain, and difficulty level. The distribution of problems across high-level physics topics is shown in Table 2. More analysis is provided in appendix.

4 Experimental Results and Insights Beyond Accuracy

We evaluate a range of state-of-the-art instruction-tuned LLMs on SymPyBench to assess their scientific reasoning capabilities under dynamic and perturbed conditions. To this end, we measure several key metrics:

Exact Match Accuracy: The proportion of problems for which the model produces a completely correct end-to-end solution:

$$\text{Exact Match Accuracy} = \frac{\text{Number of fully correct solutions}}{\text{Total number of problems}}$$

Stage	Type of Error Checked	Error Rate (Percentage of data filtered)
1. Filtered Data	Dependency to previous questions or visual information	∼5% (manually checked)
2. Structured Representation	Incorrect JSON structure	∼4.5%
3. Template Generation	Variable mismatch with Stage 3; Incorrect JSON structure	∼1%
4. Generating Variations	Incorrect JSON structure	<1%
5. Python Code	Function signature mismatch; incorrect output; unit errors	∼12%
6. Final Manual Review	Human inspection	All remaining verified

Table 3: Data Filtered Due to Errors at Each Stage of the Collection and Processing Pipeline

Model	Partial Accuracy↑	Exact Match Accuracy ↑	Consistency Score↑	Complete Failure Rate ↓	Confusion rate ↓
Qwen2.5-7B-Instruct	24.26%	16.44%	5.66%	41.51%	15.09%
Qwen2.5-72B-Instruct	66.57%	61.69%	37.74%	15.09%	11.32%
Llama-3.3-70B-Instruct	59.05%	54.17%	28.30%	15.09%	7.55%
Llama3.1-405b-instruct	42.79%	34.45%	17.46%	30.16%	14.29%
Llama4-maverick-17b-128e-instruct	69.92%	64.17%	34.92%	9.52%	11.11%
Llama4-scout-17b-16e-instruct	56.49%	50.17%	20.63%	14.29%	19.05%
OpenAI GPT (gpt-4-turbo)	60.59%	53.73%	33.33%	18.18%	12.12%
Gemini-2.0-Flash	**71.43%**	64.49%	34.38%	**9.38%**	12.50%
Anthropic Sonnet-3.7	70.81%	**65.48%**	**42.42%**	18.18%	**6.06%**

Table 4: Performance Metrics across LLMs on SymPyBench. Includes Partial Accuracy, Exact Match Accuracy, Consistency Score, Complete Failure Rate, and Confusion Rate.

Many problems in our dataset are composed of multiple subproblems (e.g., parts a, b, c; see Appendix 9). To calculate exact match accuracy, we require the model to correctly solve all parts of a problem. However, because many problems are subdivided in this way, we also introduce **Partial Accuracy** as a complementary metric.

Partial Accuracy: The fraction of subproblems within a structured solution that the model answers correctly.

$$\text{Partial Accuracy} = \frac{\text{Number of correct subproblems}}{\text{Total number of subproblems}}$$

In addition to accuracy, we evaluate the model's robustness using several complementary metrics:

Consistency Score: The proportion of problem groups where the model consistently provides the correct answer across all perturbed variants (i.e., versions of each problem modified by numerical, textual, or format changes), reflecting the stability and reliability of the model's performance. A high consistency score indicates that the model can reliably solve problems even with slight variations, showcasing its generalization ability.

$$\text{Consistency Score} = \frac{\text{\# groups with all correct variants}}{\text{\# total problem groups}}$$

Confusion Rate: The confusion rate indicates the fraction of problem groups where the model's accuracy across variants is around 40%-60%, reflecting uncertainty in the model's reasoning. It provides insight into situations where the model may be guessing or uncertain about the correct approach.

$$\text{Confusion Rate} = \frac{\text{\# groups with } \sim 50\% \text{ accuracy}}{\text{\# total problem groups}}$$

Complete Failure Rate: This metric tracks the proportion of problem groups where the model answers all variants incorrectly. A high Complete Failure Rate indicates areas of consistent failure, providing valuable diagnostic information for improving model performance.

$$\text{Complete Failure Rate} = \frac{\text{\# groups with all incorrect variants}}{\text{\# total problem groups}}$$

4.1 Results

Table 4 provides a comprehensive evaluation of large language models on SymPyBench. Three models emerge as clear leaders: Anthropic Sonnet-3.7, Gemini-2.0-Flash (Anthropic; Google DeepMind), and Llama4-Maverick-17B-128E achieve Exact Match Accuracy exceeding 64% with correspondingly high Partial Accuracy, demonstrating reliable multi-step reasoning capabilities.

Among the top performers, Sonnet-3.7 distinguishes itself with the highest Consistency Score (42.42%) and lowest Confusion Rate (6.06%), demonstrating superior robustness to paraphrased and perturbed problem variants, and Gemini-2.0-Flash achieves the highest Partial Accuracy (71.43%). The Confusion Rate, which quantifies

the proportion of problem groups where model accuracy across variants hovers around 50%, reflects reasoning uncertainty; stronger models such as Sonnet-3.7 consistently display lower confusion rates. GPT-4-Turbo presents solid overall metrics (60.59% partial, 53.73% exact) but underperforms in consistency (33.33%) compared to the top tier. Qwen2.5-72B-Instruct performs competitively (66.57% partial, 61.69% exact) but shows higher complete failure rates (15.09%) than Maverick and Gemini. These results underscore the importance of evaluating mathematical reasoning not only in terms of accuracy, but also with respect to consistency, robustness, and failure modes, qualities that are often overlooked by traditional metrics, yet are essential for ensuring real-world reliability in scientific problem-solving.

Our in-depth analysis reveals that models such as Maverick are far more likely to succeed when guided by multiple-choice formats, particularly MC-Symbolic, compared to open-ended free-form questions. The structured nature of multiple-choice formats reduces the complexity of open-ended generation and enables models to focus on selecting the correct answer, which often leads to higher accuracy. While these models may still make conceptual errors, our results indicate that a substantial portion of their failures in free-form settings stem from challenges in generating complete and well-formatted solutions, as well as difficulties in arithmetic computation and physics-specific skills such as unit conversion. However, it is important to note that multiple-choice formats can provide additional cues or scaffolding that help the model arrive at the correct answer, even in the presence of partial understanding. Weaker models like 405B, however, show lower accuracy across all formats, suggesting more fundamental gaps in both conceptual understanding and execution. These findings underscore the diagnostic power of our benchmark. By systematically varying question formats, we are able to disentangle the underlying sources of model errors, yielding actionable insights for the advancement of scientific language models. Complete results, along with additional examples and extended analysis, are provided in Appendix 8.

5 Conclusion and Future Work

We introduce SymPyBench, a benchmark for evaluating the scientific reasoning capabilities of large language models in physics, with a focus on generalization across diverse problem variations. Our benchmark assesses model robustness and introduces new metrics to capture output stability beyond standard accuracy. Future work will expand SymPyBench to include multimodal reasoning tasks and interdisciplinary STEM domains, enabling the evaluation of models with complex, cross-domain scientific reasoning capabilities. This will drive the development of AI systems with robust, transparent, and reliable scientific reasoning.

Limitations

While SymPyBench provides a dynamic and rigorous evaluation of large language models on university-level physics problems, its current focus on a single domain limits the generalizability to other scientific fields and reasoning tasks. Moreover, as a synthetic benchmark, it does not fully capture the ambiguity, and incomplete information often present in real-world physics challenges. We plan to enhance SymPyBench by broadening its scope to include additional scientific disciplines and more realistic, open-ended problems that better reflect the complexity of practical applications.

6 Ethics Statement

Potential Risks. While our benchmark and analysis focus on research purposes, potential risks include misuse of data for generating incorrect or misleading solutions, or overreliance on automated reasoning without human oversight.

Use of AI Assistant. We used an AI assistant for grammar and style checking to improve the clarity of the paper.

References

2025. Pint: Define, operate, and manipulate physical quantities. https://pint.readthedocs.io/. Accessed May 7, 2025.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Anthropic. Claude (sonnet). https://www.anthropic.com/claude/sonnet.

Daman Arora, Himanshu Gaurav Singh, and 1 others. 2023. Have llms advanced enough? a challenging problem solving benchmark for large language models. *arXiv preprint arXiv:2305.15074*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Rishi Bommasani, Percy Liang, and Tony Lee. 2023. Holistic evaluation of language models. *Annals of the New York Academy of Sciences*, 1525(1):140–146.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, and 1 others. 2020. Evaluating models' local decision boundaries via contrast sets. *arXiv preprint arXiv:2004.02709*.

Google DeepMind. Gemini. https://gemini.google.com/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Balasubrahmanya Hegde and BN Meera. 2012. How do they solve it? an insight into the learner's approach to the mechanism<? format?> of physics problem solving. *Physical Review Special Topics—Physics Education Research*, 8(1):010109.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Raj Jaiswal, Dhruv Jain, Harsh Parimal Popat, Avinash Anand, Abhishek Dharmadhikari, Atharva Marathe, and Rajiv Ratn Shah. 2024. Improving physics reasoning in large language models using mixture of refinement agents. *arXiv preprint arXiv:2412.00821*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jill H Larkin and F Reif. 1979. Understanding and teaching problem-solving in physics. *European journal of science education*, 1(2):191–203.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, and 8 others. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Allen Newell, Herbert Alexander Simon, and 1 others. 1972. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ.

Frederick Reif and Joan I Heller. 1982. Knowledge structure and problem solving in physics. *Educational psychologist*, 17(2):102–127.

Liangtai Sun, Yang Han, Zihan Zhao, Da Ma, Zhennan Shen, Baocai Chen, Lu Chen, and Kai Yu. 2024. Scieval: A multi-level large language model evaluation benchmark for scientific research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19053–19061.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2023. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254):1–22.

7 Examples from SymPyBench

Here are some examples from SymPyBench, where each question is shown along with step-by-step reasoning and the corresponding Python code. We populate the problems with numerical values, and the relevant variables are generated as part of the pipeline.

7.1 Example A

Question

In a simplified atomic model, the most probable distance between the nucleus and an electron is $r = 3.33e - 11\,\text{m}$. The nucleus contains 1.3 protons. Determine the electric field due to the nucleus at the electron's position.
Here are constants:

$$\text{Permittivity of free space} = 8.85 \times 10^{-12}\,\frac{\text{C}^2}{\text{N}\cdot\text{m}^2}$$

$$\text{Elementary charge} = 1.6 \times 10^{-19}\,\text{C}$$

Solution

Identify Relevant Concepts

- The electric field due to a point charge is given by

$$\vec{E} = \frac{1}{4\pi\epsilon_0}\frac{q}{r^2}\hat{r}$$

  where ϵ_0 is the permittivity of free space, q is the charge, and r is the distance from the charge.

- The goal is to calculate the electric field at the electron's position.

Set Up the Problem

- The electric field at a distance r from a point charge is given by the formula above.
- The direction of the electric field is radially outward from the nucleus.

Execute the Solution

- Substituting the given values into the formula

Evaluate Your Answer

- The electric field is expected to be radially outward from the nucleus due to its positive charge.
- If r were very small, the electric field would be very large, and if r were large, the electric field would approach zero, which is physically reasonable.

Python code:

```
import sympy as sp
from pint import UnitRegistry
ureg = UnitRegistry()
Q_ = ureg.Quantity

def electric_field_at_electron(r, e, number_of_protons,
      epsilon_0):
    # Convert inputs to Pint quantities
    r = Q_(r).to(ureg.meter) # Ensure meters
    e = Q_(e).to(ureg.coulomb) # Ensure coulombs
    number_of_protons = Q_(number_of_protons).to(ureg.
          dimensionless)
    epsilon_0 = Q_(epsilon_0).to(ureg.farad / ureg.meter) #
          Ensure F/m

    r = r.magnitude
    e = e.magnitude
    number_of_protons = number_of_protons.magnitude
    epsilon_0 = epsilon_0.magnitude

    # Define symbolic variables
    q = e * number_of_protons
    E = sp.Symbol('E', real=True, positive=True)

    # Calculate the electric field
    E = (1 / (4 * sp.pi * epsilon_0)) * (q / r**2)

    return {
        'E': E.evalf()
    }
```

7.2 Example B

Question

Consider a solid metal cube with an edge length of $L = 0.0237\,\text{m}$.
(a) Determine the lowest energy level for an electron within this metal.
(b) Calculate the energy difference between this level and the next higher energy level.
Here are constants:

$$\text{Reduced Planck's constant } \hbar = 1.05 \times 10^{-34}\,\text{J}\cdot\text{s}$$

$$\text{Electron mass } m_e = 9.11 \times 10^{-31}\,\text{kg}$$

$$\text{Ground state quantum numbers: } n_x = n_y = n_z = 1$$

$$\text{Next state quantum numbers: } n_x = 2,\ n_y = 1,\ n_z = 1$$

Solution

Identify Relevant Concepts

- Model the electron as a particle in a 3D box.
- Energy levels are given by:

$$E(n_x, n_y, n_z) = \frac{\pi^2\hbar^2}{2m_eL^2}(n_x^2 + n_y^2 + n_z^2)$$

Set Up the Problem

- Ground state: $n_x = n_y = n_z = 1$
- Next higher level: $n_x = 2, n_y = 1, n_z = 1$

Execute the Solution

- Compute:

$$E_1 = \frac{\pi^2\hbar^2}{2m_eL^2}(1^2 + 1^2 + 1^2)$$

$$E_2 = \frac{\pi^2\hbar^2}{2m_eL^2}(2^2 + 1^2 + 1^2)$$

- Energy difference:

$$\Delta E = E_2 - E_1$$

Evaluate Your Answer

- Positive energy difference is expected since next level is higher.
- Larger cube size would reduce energy spacing, consistent with quantum model.

Python code:

```
import sympy as sp
from pint import UnitRegistry
ureg = UnitRegistry()
Q_ = ureg.Quantity

def electron_energy_levels(L, h_bar, m_e, n_x, n_y, n_z,
      n_x_next, n_y_next, n_z_next):
    L = Q_(L).to(ureg.meter)
    h_bar = Q_(h_bar).to(ureg.joule * ureg.second)
    m_e = Q_(m_e).to(ureg.kilogram)

    L = L.magnitude
    h_bar = h_bar.magnitude
    m_e = m_e.magnitude

    pi = sp.pi

    def energy(n_x, n_y, n_z, L, h_bar, m_e):
        return (pi**2 * h_bar**2 / (2 * m_e * L**2)) * (n_x**2
              + n_y**2 + n_z**2)

    E1 = energy(n_x, n_y, n_z, L, h_bar, m_e)
    E2 = energy(n_x_next, n_y_next, n_z_next, L, h_bar, m_e)
    DeltaE = E2 - E1

    return {
        'E1': E1.evalf(),
        'E2': E2.evalf(),
        'DeltaE': DeltaE.evalf()
    }
```

8 Detailed Model Performance Analysis

In addition to standard metric evaluation, we conducted an in-depth analysis of model performance across multiple dimensions. For this analysis, we evaluate three model configurations: `llama4-maverick-17b-128e` (Maverick), `llama4-scout-17b-16e` (Scout), and `llama3.1-405b` (405B), presenting a comprehensive performance comparison.

8.1 Performance by Textual Variant

We analyze model performance across three textual variants, which differ in their surface phrasing while preserving the underlying question. Table 5 presents a detailed breakdown.

All models exhibit consistent performance across templates, with minimal variation.

8.2 Performance by Question Type

Table 6 presents accuracy by question format: free-form, MC-Numerical, and MC-Symbolic.

A striking divergence emerges across models and question types. Maverick and Scout excel at MC-Symbolic questions (95.70% and 81.51%, respectively), while showing substantially lower performance on MC-Numerical questions. Manual analysis of 100 randomly sampled examples revealed that the performance gap between MC-Symbolic and MC-Numerical stems primarily from errors in numerical computation and unit conversion, rather than conceptual understanding deficits.

Free-form questions present a fundamentally different challenge and exhibit consistently lower performance across all models. This gap is attributable to two key factors: (1) structural complexity: free-form questions contain an average of two to three interconnected sub-questions that must be solved sequentially, with errors in early steps propagating to subsequent parts; and (2) evaluation stringency: models must generate complete, correctly formatted solutions rather than simply selecting from provided options. This combination of increased reasoning depth and answer generation requirements makes free-form questions substantially more demanding than their multiple-choice counterparts.

Surprisingly, 405B exhibits relatively balanced multiple-choice performance (59.42% MC-Numerical, 57.21% MC-Symbolic) but dramatically lower free-form accuracy (24.95%).

8.3 Performance Across Response Iterations

We examine model stability across five response iterations to assess consistency. Table 7 summarizes the results. All models demonstrate stable performance across iterations.

8.4 Cross-Type Error Analysis

To investigate whether errors are question-format-specific or stem from deeper conceptual misunderstandings, we conducted a cross-type analysis on a subset of problems that appear in all three formats (free-form, MC-Numerical, and MC-Symbolic). This allows us to examine whether difficulty in one format predicts difficulty in others, revealing the nature of model errors.

Conditional Accuracy Analysis A critical question is whether model errors reflect format-specific challenges (e.g., numerical computation, answer generation) or fundamental conceptual misunderstandings. To distinguish these error types, we compute *conditional accuracy*: for problems where a model fails in one format, what is its accuracy on the same problem presented in a different format?

If errors were primarily conceptual, models should fail consistently across all formats of the same problem, yielding low conditional accuracy. Conversely, high conditional accuracy indicates that the model understands the concept but fails due to format-specific requirements. Table 8 presents this analysis.

Interpretation and Key Insights The conditional accuracy patterns reveal fundamentally dif-

Model	Textual Variant	Partial Accuracy (%)	Exact Match Accuracy (%)
Llama4-maverick-17b-128e-instruct	Variant I	69.81	64.16
	Variant II	69.17	63.54
	Variant III	70.82	64.86
Llama4-scout-17b-16e-instruct	Variant I	55.81	49.56
	Variant II	57.15	50.57
	Variant III	56.44	50.33
Llama3.1-405b-instruct	Variant I	42.18	34.03
	Variant II	43.36	34.10
	Variant III	42.78	35.24

Table 5: Accuracy by textual variant across models. Each variant represents a different surface phrasing of the same underlying question.

Model	Type	Partial Accuracy (%)	Exact Match Accuracy (%)	Data%
Llama4-maverick-17b-128e-instruct	Free-Form	65.72	57.69	71.52
	MC Numerical	65.23	65.23	14.24
	MC Symbolic	95.70	95.70	14.24
Llama4-scout-17b-16e-instruct	Free-Form	53.28	44.44	71.52
	MC Numerical	47.56	47.56	14.24
	MC Symbolic	81.51	81.51	14.24
Llama3.1-405b-instruct	Free-Form	36.61	24.95	71.52
	MC Numerical	59.42	59.42	14.24
	MC Symbolic	57.21	57.21	14.24

Table 6: Accuracy by question type across models.

ferent error sources across models:

- **MC-Numerical vs. MC-Symbolic Gap:** We assess cases where models fail on MC-Numerical questions and evaluate their accuracy on the corresponding MC-Symbolic versions. All models show substantially higher conditional accuracy on MC-Symbolic (95.00%, 79.55%, 60.93%), indicating that most MC-Numerical errors are due to computational issues (e.g., arithmetic mistakes, unit conversion), rather than misunderstanding the underlying concepts or formulas. When explicit computation is removed, model performance improves markedly.

- **Free-Form vs. Multiple-Choice Gap:** We examine cases where models fail on free-form questions and measure their accuracy on the corresponding MC-Numerical and MC-Symbolic formats. For example, Maverick achieves 95.45% accuracy on MC-Symbolic and 60.18% on MC-Numerical for problems it fails in free-form. This substantial improvement in accuracy with guided formats suggests that many free-form errors may stem from challenges in generating complete and well-formatted solutions, rather than from fundamental conceptual or arithmetic misunderstandings. However, it is important to note that multiple-choice formats can provide additional cues or scaffolding that help the model arrive at the correct answer, even in the presence of partial understanding. Thus, while the observed gap is indicative of generation and formatting challenges, it does not fully rule out the possibility of underlying conceptual gaps. Further analysis is needed to disentangle these effects.

- **Model Differences:** Scout exhibits a similar but slightly weaker pattern, with conditional accuracies of 81.25% (MC-Symbolic given free-form failure) and 79.55% (MC-Symbolic given MC-Numerical failure). This indicates that most errors are still format-specific, though some conceptual gaps may remain. As with Maverick, the improvement in accuracy with guided formats highlights the benefit of scaffolding and structured problem presentation for model performance.

- **Conceptual Inconsistency in 405B:** In contrast, 405B shows lower conditional accuracies (60–65%), indicating that a significant portion of its errors are due to inconsistent rea-

Model	Iteration	Partial Accuracy (%)	Exact Match Accuracy (%)
Llama4-maverick-17b-128e-instruct	0	70.11	64.65
	1	70.29	64.24
	2	70.27	64.40
	3	69.49	63.99
	4	69.44	63.58
Llama4-scout-17b-16e-instruct	0	56.89	50.33
	1	56.96	51.16
	2	56.74	50.33
	3	56.28	49.67
	4	55.56	49.34
Llama3.1-405b-instruct	0	42.78	34.69
	1	42.75	34.27
	2	43.03	34.85
	3	42.77	34.19
	4	42.63	34.27

Table 7: Accuracy by iteration for all models. All values are percentages.

Model	Given Failure On	Accuracy On	Success Rate (%)
Llama4-maverick-17b-128e-instruct	Free-Form	MC-Symbolic	95.45
	Free-Form	MC-Numerical	60.18
	MC-Numerical	MC-Symbolic	95.00
Llama4-scout-17b-16e-instruct	Free-Form	MC-Symbolic	81.25
	Free-Form	MC-Numerical	46.33
	MC-Numerical	MC-Symbolic	79.55
Llama3.1-405b-instruct	Free-Form	MC-Symbolic	64.48
	Free-Form	MC-Numerical	54.06
	MC-Numerical	MC-Symbolic	60.93

Table 8: Conditional accuracy: given failure on one format (Condition), what is the average accuracy on another format (Target) for the same problems? High values indicate format-specific rather than conceptual errors. All values are percentages.

soning or incomplete conceptual understanding, even when the format is simplified.

Implications for Model Development: These findings suggest different improvement strategies for different model capabilities. For Maverick and Scout, gains would come from better solution generation, numerical precision, and unit handling, as their conceptual reasoning is already strong. For 405B, improvements require addressing fundamental reasoning consistency before tackling format-specific issues. The conditional accuracy metric thus serves as a diagnostic tool, revealing whether a model needs better conceptual understanding or improved execution.

9 Distribution of Sub-Questions

Real-world physics problems often consist of multiple interconnected sub-questions that build upon each other, requiring students to demonstrate cumulative understanding. To reflect this complexity, many problems in SymPyBench are structured as multi-part questions. Figure 3 shows the distribution of problems across different numbers of sub-questions per instance.

10 Case Studies: Example-Level Insights

While aggregate metrics provide a high-level view of model performance, deeper insights emerge when examining specific examples and their variations. In this section, we analyze representa-

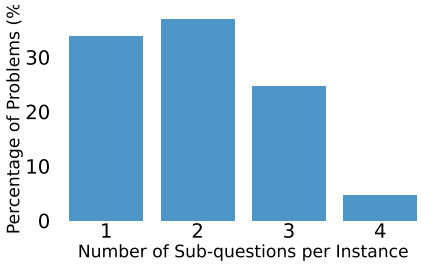

Figure 3: Distribution of SymPyBench problems by number of sub-questions per instance.

tive cases that highlight recurring success patterns, consistent failure modes, and surprising behaviors across paraphrased or perturbed inputs.

These quantitative and qualitative observations shed light on the limitations of current models in terms of robustness, generalization, and interpretability.

Case Study I: Sensitivity to Input Variations

We analyze three semantically equivalent versions of a physics question requiring symbolic reasoning and precise numerical calculation as shown in Figure 4. The first part of the task is to compute the average kinetic energy of a gas molecule nitrogen molecules at a given temperature. We analyze the result of Qwen2.5-7B model.

As shown in Figure 4, in the first variation, the model's answer is off by several orders of magnitude, indicating a fundamental flaw in its solution strategy. In the second variation, the response is approximately correct aside from a minor numerical discrepancy, reflecting improved accuracy under this paraphrased input. In the third variation, the kinetic energy is overestimated by more than three times, likely due to an arithmetic error or a misinterpretation of symbolic expressions. Without SymPyBench, such nuanced insights into model behavior that emerge from the same underlying question with different input realizations would remain inaccessible, limiting our ability to diagnose failure modes and evaluate robustness.

Our observation is not limited to smaller models like Qwen2.5-7B. Even Qwen2.5-72B, despite using the correct physics formula and providing step-by-step reasoning, often produces numerically inconsistent results.

Consider the following example:

Qwen2.5-72B: Incorrect Electric Field Calculatio

Question: Determine the magnitude of the electric field E generated by a point charge of 2.09×10^{-9} C at a distance of 0.00567 m. Use Coulomb's constant $k = 8.99 \times 10^{9}\,\mathrm{N \cdot m^2/C^2}$.

Qwen2.5-72B Answer: Applies $E = \frac{kq}{r^2}$, computes $E \approx 587.5\,\mathrm{N/C}$.

Ground Truth: $E \approx 584{,}440\,\mathrm{N/C}$

The model uses the correct formula and walks through intermediate steps, but its final numeric output is off by nearly three orders of magnitude. This suggests that the issue is not conceptual misunderstanding but internal instability in arithmetic or symbolic execution. Similar inconsistencies were observed across other problems with slight input perturbations.

Case Study II: Measuring Hallucination

A key strength of our benchmark is its ability to systematically induce and detect hallucinations in LLM model responses by dynamically altering or concealing critical components of a physics problem, such as input variables or domain-specific constants. This functionality enables controlled testing of model behavior under uncertainty. When confronted with incomplete information: *Does the model seek clarification? Does it make reasonable assumptions and state them explicitly? Does it hallucinate values and proceed as if the input were fully specified?* Our benchmark is uniquely designed to probe these behaviors, allowing us to quantify reasoning integrity in under-specified scenarios and assess the robustness of models under challenging conditions.

We illustrate this with a representative example:

Example of Hallucination

Question: Determine the average time required for a glucose molecule to diffuse a distance of 0.00991 m in water.

This question omits the diffusion coefficient D, a necessary constant for computing the answer via the physical equation $t = \frac{x^2}{2D}$. However, when prompted with this version, Gemini does not request the missing value or flag the input as incomplete. Instead, the model fabricates a response by

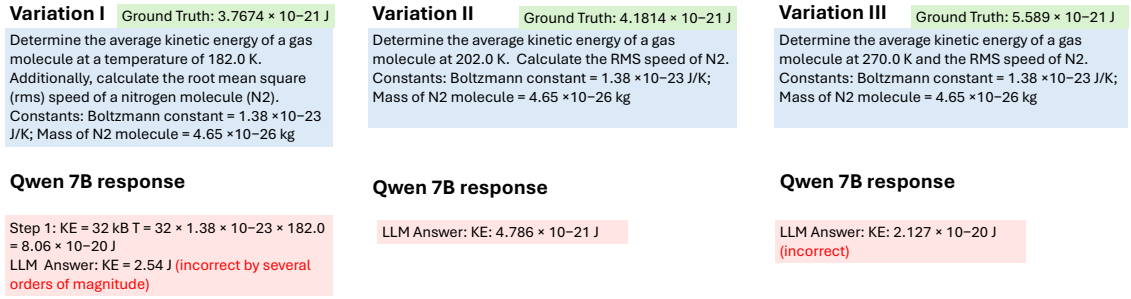

Figure 4: Three variation of the same question with different input variables. Only the Qwen-7B model's final step responses are shown.

assuming an image is provided, stating: 'Based on the image,' and then extrapolates from a hallucinated example involving diffusion over 0.010 m in 7.5×10^4 seconds. It uses the proportionality $t \propto x^2$ to calculate:

$$t_2 = t_1 \cdot \left(\frac{x_2^2}{x_1^2} \right),$$

with

$$t_1 = 7.5 \times 10^4\,\text{s},\ x_1 = 0.010\,\text{m},\ x_2 = 0.00991\,\text{m}$$

leading to an incorrect answer.

This response reflects a hallucinated reasoning chain. Instead of applying the correct physics or querying for D, the model infers a scenario that was never presented. Such behavior can be quantitatively evaluated in our benchmark by selectively omitting critical variables and analyzing how often models hallucinate versus recognize underspecified inputs.

In future work, we plan to formalize this capability and systematically benchmark hallucination rates across model families. This expands the scope of our dataset beyond correctness and robustness, making it a valuable tool for studying reasoning integrity under *partial* or *ambiguous* inputs.

Case Study III: Implicit Simplification Bias

Another class of error arises in problems requiring more advanced topics. For example in question related to relativistic mechanics even when the scenario clearly demands relativistic treatment, the Qwen2.5-72B frequently defaults to oversimplified Newtonian expressions, for example using $a = \frac{F}{m}$ or $a = \frac{F}{\gamma m}$ without accounting for the orientation of the force relative to velocity. We call this behavior implicit simplification bias where the model superficially identifies relevant physical variables but fails to apply the correct governing equations when deeper conceptual distinctions are required. This suggests that such biases are not merely a consequence of model size but rather reflect fundamental gaps in their understanding of domain-specific complexities, highlighting the need for explicit training in these advanced areas.

11 Comparison with Existing Scientific Reasoning Benchmarks

To illustrate the limitations of current scientific reasoning benchmarks, we provide representative examples from ScienceQA (Lu et al., 2022) and SciEval (Sun et al., 2024). These datasets primarily focus on selecting the correct answer from multiple choices, without requiring explicit, step-by-step reasoning or handling parameterized problem variations.

For example, consider the ScienceQA dataset:

```
"question": "Select the solid."
"choices": ["rain", "water in a fishbowl", "
    hammer"]
"answer": 2
```

Or the SciEval benchmark:

```
"question": "How can momentum be decreased?"
"choices": [
  "A. Decrease mass or velocity, or transfer
      momentum through collision.",
  "B. Keep mass and velocity constant, avoid
      collisions.",
  "C. Increase mass and velocity, avoid
      collisions.",
  "D. Increase mass, decrease velocity, and
      avoid collisions."
]
"answer": ["A"]
```

Table 1 provides a high-level comparison of existing physics benchmarks, illustrating how SymPyBench addresses these limitations.