
Advancing AI-Scientist Understanding: Multi-Agent LLMs with Interpretable Physics Reasoning

Yinggan Xu¹ Hana Kimlee² Yijia Xiao¹ Di Luo¹

Abstract

Large Language Models (LLMs) are playing an increasingly important role in physics research by assisting with symbolic manipulation, numerical computation, and scientific reasoning. However, ensuring the reliability, transparency, and interpretability of their outputs remains a major challenge. In this work, we introduce a novel multi-agent LLM physicist framework that fosters collaboration between AI and human scientists through three key modules: a reasoning module, an interpretation module, and an AI-scientist interaction module. Recognizing that effective physics reasoning demands logical rigor, quantitative accuracy, and alignment with established theoretical models, we propose an interpretation module that employs a team of specialized LLM agents—including summarizers, model builders, visualization tools, and testers—to systematically structure LLM outputs into transparent, physically grounded science models. A case study demonstrates that our approach significantly improves interpretability, enables systematic validation, and enhances human-AI collaboration in physics problem-solving and discovery. Our work bridges free-form LLM reasoning with interpretable, executable models for scientific analysis, enabling more transparent and verifiable AI-augmented research.

1. Introduction

Large Language Models (LLMs) have become increasingly popular for tackling complex physics problems, emerging as valuable assistants to scientists (Zhang et al., 2024). However, interpreting the solutions they generate remains a sig-

nificant challenge due to the inherent complexity of physics problems. Identifying potential flaws often demands substantial effort from experts, as LLM-generated solutions can obscure their underlying reasoning.

Several key issues contribute to this interpretability gap. First, the reasoning trajectories employed by LLMs are often highly complex and diverse. Depending on the inference techniques used, ranging from direct outputs to tool-assisted reasoning, the underlying processes may be partially hidden or require considerable effort to trace. Second, the numerical complexity involved in many physics problems poses a significant verification challenge, making it difficult for humans to independently validate the results. Third, the absence of an interpretable underlying mechanism can lead to seemingly correct outcomes even when the LLM’s understanding of the physics is flawed.

To address these challenges, we develop a novel multi-agent LLM physicists framework that enhances the interpretability, transparency, and verifiability of LLM outputs in physics problem-solving. Unlike prior approaches that treat LLMs as black-box solvers, this framework decomposes the reasoning pipeline into three coordinated modules: a reasoning module, an interpretation module, and an AI-scientist interaction module. We propose an innovative LLM interpretation module, consist of a suite of specialized agents including summarizers, model builders, and testers, which translates opaque LLM outputs into structured, executable, and physically grounded science models. This interpretable interface bridges the gap between AI-generated reasoning and human scientific intuition by supporting validation through code execution, visual inspection, and human-in-the-loop critique. Extensive case studies on textbook-level problems from SciBench demonstrate the framework’s ability to detect flaws, test consistency, and enable interactive validation, thereby offering a new pathway for interpretable, verifiable, and collaborative AI-assisted physics discovery.

2. Related Works

2.1. LLM for Physics

Researchers have begun exploring the potential of Large Language Models (LLMs) as reasoning tools in the physics

¹University of California, Los Angeles ²NSF Center for Quantum Network.

Correspondence to: Di Luo <diluo@ucla.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

domain (Anand et al., 2024; Ding et al., 2023; Pan et al., 2024; Pang et al., 2024; Wang et al., 2023b). Studies have demonstrated that LLMs can solve complex word problems requiring calculation and inference, often achieving near human-level accuracy, especially with effective prompting techniques such as few-shot learning using similar examples (Ding et al., 2023), leveraging reinforcement learning from human feedback (RLHF) (Anand et al., 2024) or implementing agentic system (Pang et al., 2024).

While much of this research focuses on general physics reasoning, recent efforts have applied LLMs to highly specialized domains. Pan et al. (Pan et al., 2024) demonstrated that GPT-4 can perform advanced theoretical derivations, such as deriving Hartree–Fock equations, highlighting LLMs’ potential to automate and accelerate research workflows in theoretical physics. However, as most physics reasonings are complex and domain-specific, existing approaches offer limited support for human scientists to interpret and validate LLM-generated results. The lack of intuitive interfaces for understanding these outputs places a significant cognitive burden on researchers, limiting the practical usability of LLMs in scientific discovery.

2.2. Verifiable Generation

A parallel line of research focuses on improving the verifiability and interpretability of LLM outputs. One common approach involves grounding generated content in external sources and providing detailed citations (Hennigan et al., 2023; Shen et al., 2024; Li et al., 2024). Other methods enhance transparency by generating with more structured and intuitive processes (Cecchi & Babkin, 2024) or enable self-explanatory reasoning (Huang et al., 2023).

However, physics reasoning differs fundamentally from tasks based purely on factual retrieval or general logical reasoning. Unlike citation-based fact-checking, physics problem-solving requires structured derivations, adherence to established theoretical frameworks, and quantitative validation. Despite advances in interpretable generation, the challenge of making LLM-generated physics reasoning both understandable and verifiable remains largely unexplored.

3. System Design

Building on prior research in LLM-assisted physics reasoning and verifiable AI generation, we propose an interpretation module that enhances both interpretability and validation in physics reasoning. We focus on physics reasoning within the context of problem-solving, which represents its most fundamental form. Our approach employs an agentic system composed of specialized agents, each with a distinct role in structuring the reasoning process. This inference-agnostic pipeline can generate science models for a broad

range of problem-solving scenarios, regardless of the implementation of the reasoning module. By explicitly modeling the reasoning process, our system deepens AI-scientist understanding, facilitating more transparent, interpretable, and verifiable AI-augmented scientific reasoning. To clearly articulate our approach, we structure our system into three key modules: a reasoning module, which processes physics problems using naive, tool-using, or agentic LLMs; an interpretation module, which refines AI reasoning into structured science models, executable code, and validation tools; and an AI-scientist interaction module, which facilitates human oversight by enabling experts to analyze, critique, and refine AI-generated reasoning.

3.1. LLM Reasoning Module: Establishing the Problem Context

The reasoning module serves as the entry point to the pipeline, handling diverse physics problems and their solutions from different sources, including: naive LLMs that generate direct, unstructured solutions, tool-using LLMs that incorporate computational resources to refine their responses, and agentic systems that coordinate multiple AI components for enhanced reasoning. While these reasoning modules can be powerful, they often involve complex, opaque processes that may not be fully visible to human scientists. For example, tool-using mechanisms or multi-agent debates can lead to solutions that are difficult to interpret, making it challenging to trace the reasoning behind the results.

3.2. LLM Interpretation Module: Structuring and Validating AI Reasoning

To enhance the interpretability and reliability of AI-generated physics solutions, we introduce an interpretation module, which systematically structures AI reasoning into explicit, verifiable science models and provides intuitive feedback for human scientists. Our module refines raw AI outputs into structured representations, aligning them with scientific intuition and enabling validation through interactive tools and automated checks.

This module consists of specialized agents that structure reasoning, build executable models, and enhance human interpretability.

- **LLM Summarizer** The summarizer agent processes diverse inputs such as direct solutions, tool usage details, and chat history into a structured, concise format. By preserving core reasoning and reducing redundancy, this agent improves clarity and ensures smoother downstream processing for subsequent agents.
- **LLM Model Builder** To ensure interpretable physics generation, our approach explicitly constructs and val-

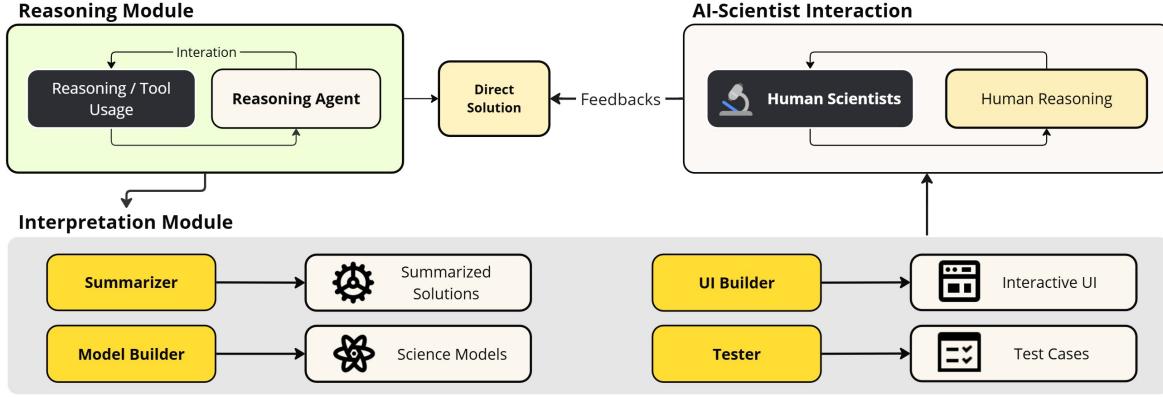


Figure 1. An overview of the augmented reasoning with interpretation module.

idates the underlying science model, which is often implicit in solutions. This module consists of two key components:

Theory Model Builder: The correctness of an AI-generated physics solution depends on the validity of its underlying conceptual model, which LLMs often leave implicit. This agent explicitly extracts, organizes, and refines the model by identifying key physical quantities, governing equations, and problem constraints. It also uses gater agents classifies the problem type, invokes relevant idealized concepts (e.g. mass point in mechanics) for conceptual coherence.

Code Model Builder: Translating theory models into executable code is essential for validation and downstream applications of the theory model. This agent converts structured science models into computational processes, ensuring consistency between theoretical assumptions and computational implementation.

- **Visualization Builder** To support human intuition-driven assessment, the visualization builder generates interactive representations of the coding model. This allows scientists to apply established validation techniques, such as testing extreme conditions and symmetry constraints, to assess solution consistency.
- **LLM Auxiliary Tester** While human scientists excel at verification, LLMs can assist this process by performing automated sanity checks like extreme case analysis, providing an additional layer of quality control. Though not a substitute for human judgment, this agent enhances the reliability of AI-generated solutions by identifying inconsistencies.

By structuring AI reasoning into explicit science models, executable simulations, and interactive validation tools, the interpretation module improves interpretability, verifiability, and alignment with scientific reasoning.

3.3. AI-Scientist Interaction: Fostering Collaborative Reasoning

Ultimately, our system is designed to augment—not replace—human scientific reasoning. The AI-scientist interaction module ensures that human experts remain central to the validation and refinement process by providing multiple touchpoints for engagement. Scientists can examine and verify the science model to explicitly assess AI reasoning, interact with the visualization interface to dynamically explore and test solutions, and critique AI-generated logic through intuitive representations. By fostering an interpretable reasoning process, this module ensures that AI remains an assistive tool that enhances scientific inquiry while preserving human oversight and expertise.

4. Case Study and Experiments

We demonstrate the effectiveness of our interpretation module using a mechanics problem from SciBench (Wang et al., 2023a). In this case, a potato is launched from a potato gun with air resistance, and the task requires an LLM to analyze the object’s motion via the energy conservation law. For our experiments, we utilize ChatGPT-4o (Achiam et al., 2023) integrated with a Python programming tool as the reasoning module. We use the same prompt templates in SciBench for our reasoning module to solve this problem.

4.1. LLM Reasoning Module and Summarizer

Our workflow begins by refining the generated solution through a summarization step. The original inference trajectory includes complex details, including multiple code executions and internal thought processes, which can be difficult for human experts to interpret. Although the direct solution appears to be straightforward, its opaque derivation limits transparency and hinders scientific understanding by human . Our summarizer condenses both the final output and the inference trajectory into a structured form (see Fig.

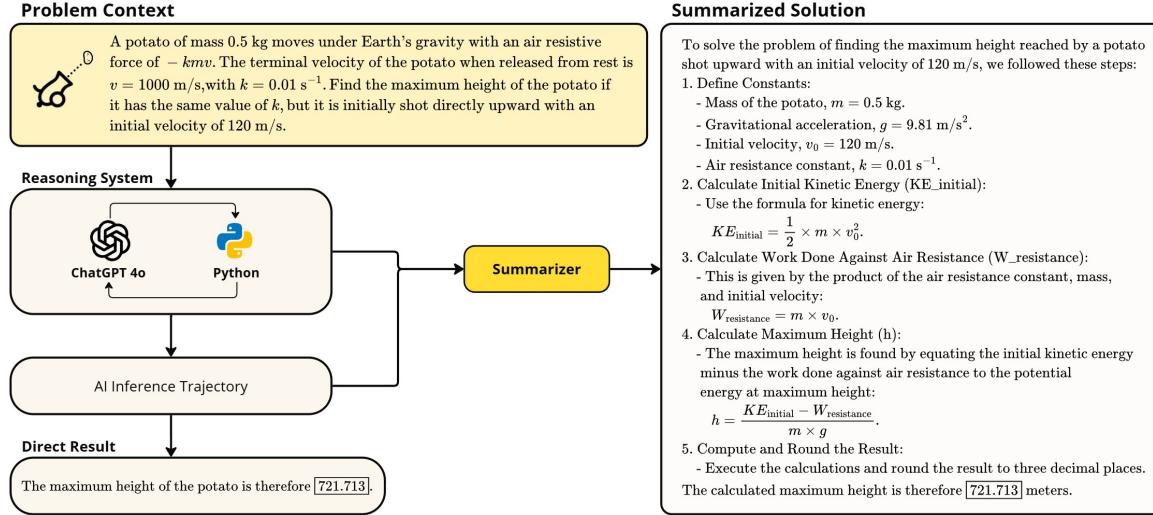


Figure 2. Transformation of a directly generated solution into a summarized solution

- 2). It distills the reasoning trajectory into a step-by-step format for improved interpretability.

4.2. Model Construction

Given a problem context and its summarized solution, the interpretation module constructs a corresponding science model in Python and generates an interactive user interface (UI) for scientists to inspect and validate the solution.

The theoretical model is aligned with the fundamental physics principles familiar to human scientists and serves as a reference for downstream model construction. The Python-based model enables reproduction of numerical results and facilitates modifications to test alternative conditions. The code model follows a predefined template to ensure consistency and a structured format for interpretation and execution. The built models are sent to the downstream agents for testing and user interface construction. We provide full demonstrations and more case studies in the appendix.

The science model and its interfaces are only practical for human scientists when they are faithful to the original reasoning result. To ensure that the science model and UI accurately reflect the original reasoning, we evaluate the consistency of our module using a subset of problems from the SciBench dataset. This subset contains problems from three textbooks: Fundamentals of Physics(Halliday et al., 2013), Statistical Thermodynamics(Engel & Reid, 2010), and Classical Dynamics of Particles and Systems(Thornton & Marion, 2021). For a meaningful assessment, we carefully selected 50 problems, excluding those that involve only basic computations or contain incorrect reference solutions.

We evaluate consistency on two key dimensions:

Summarized Solution

To solve the problem of finding the maximum height reached by a potato shot upward with an initial velocity of 120 m/s, we followed these steps:

1. Define Constants:
 - Mass of the potato, $m = 0.5 \text{ kg}$.
 - Gravitational acceleration, $g = 9.81 \text{ m/s}^2$.
 - Initial velocity, $v_0 = 120 \text{ m/s}$.
 - Air resistance constant, $k = 0.01 \text{ s}^{-1}$.
2. Calculate Initial Kinetic Energy (KE_{initial}):
 - Use the formula for kinetic energy:
$$KE_{\text{initial}} = \frac{1}{2} \times m \times v_0^2.$$
3. Calculate Work Done Against Air Resistance (W_{resistance}):
 - This is given by the product of the air resistance constant, mass, and initial velocity:
$$W_{\text{resistance}} = m \times v_0.$$
4. Calculate Maximum Height (h):
 - The maximum height is found by equating the initial kinetic energy minus the work done against air resistance to the potential energy at maximum height:
$$h = \frac{KE_{\text{initial}} - W_{\text{resistance}}}{m \times g}.$$
5. Compute and Round the Result:
 - Execute the calculations and round the result to three decimal places.
 - The calculated maximum height is therefore [721.713] meters.

Model	Cons.	Incons.
ChatGPT-4o-mini	47	3
ChatGPT-4o	46	4

Table 1. Numerical Consistency of Different Base Models.

- **Numerical Consistency:** The science model should yield numerical results that agree with the original reasoning output.
- **Theoretical Consistency:** The constructed model should be physically coherent and correctly reflect the solution's underlying principles.

Numerical consistency is verified via program execution, while theoretical consistency is assessed by a ChatGPT-4o model acting as a grader. The grader classifies each solution into three categories: highly consistent, moderately consistent, or inconsistent. We evaluate our model builders using two different underlying LLMs for agents: ChatGPT-4o and ChatGPT-4o-mini.

Table 1 summarizes the numerical consistency of the base models. Although most solutions are consistent, discrepancies—stemming from reasoning failures or incorrect numerical outcomes—provide valuable feedback for further investigation by human experts.

Table 2 presents the theoretical consistency results. ChatGPT-4o demonstrates a higher degree of theoretical consistency, with no instances classified as completely inconsistent. This suggests that LLMs can effectively structure physics problems into theory models for interpretability.

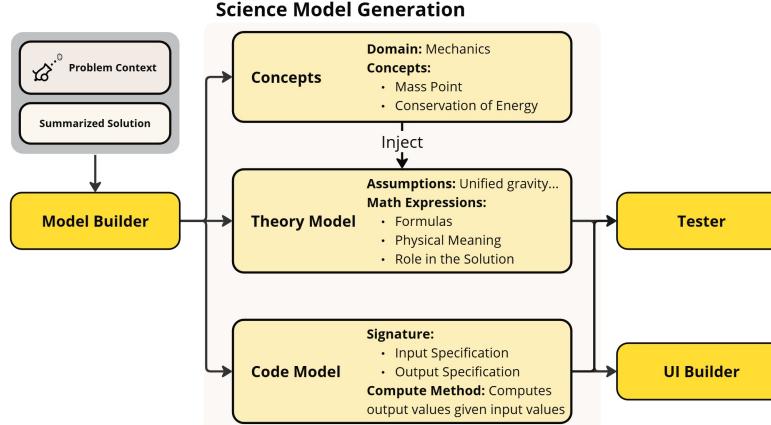


Figure 3. The model builder generates science models from summarized solutions, giving rise to interpretable reasoning

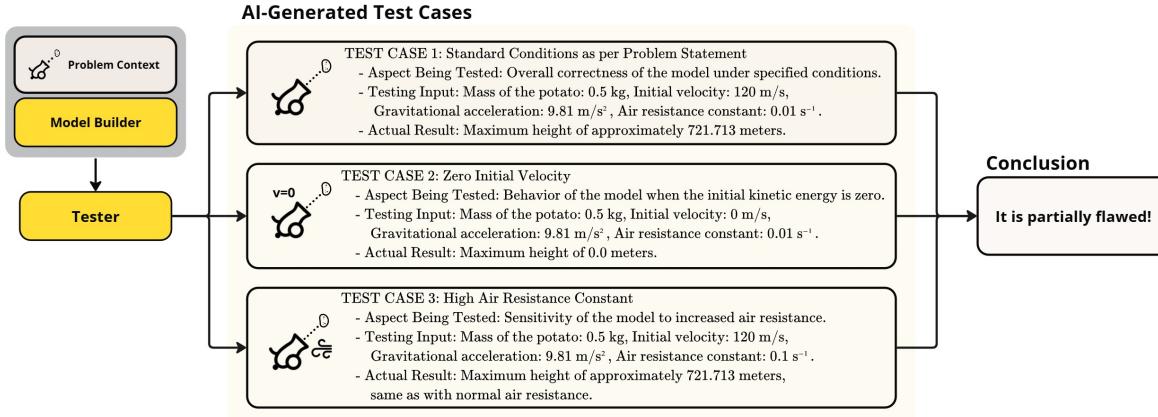


Figure 4. The Tester agent automatically generates test cases based on human-like reasoning principles

Model	High	Mod.	Incons.
ChatGPT-4o-mini	43	3	4
ChatGPT-4o	47	3	0

Table 2. Theoretical Consistency of Different Base Models.

4.3. LLM Auxiliary Tester

In addition to generating solutions, the auxiliary tester enhances validation by automatically generating diverse test cases and analyzing their outcomes using the science model. Although LLM-generated test cases are common in software engineering (Tufano et al., 2020; Li et al., 2022), they also provide valuable insights when applied to science models. Our experiments show that LLMs naturally adopt human-like reasoning in test case generation, such as evaluating extreme scenarios. This enables them to provide more informative feedback beyond the science model and the interactive UI.

As depicted in Fig. 4, the tester agent uncovers partial

flaws in the model by exploring various input conditions, by reconsidering the original input and tuning the initial velocities and the air resistance constant. The tester agent's conclusion well aligns with the ground truth that the solution was indeed incorrect due to an erroneous underlying science model.

4.4. Interactive UI

Inspired by previous work on enabling LLMs to generate user interfaces through coding (Wu et al., 2024), we introduce an interactive interface built using Gradio (Abid et al., 2019). The UI Builder agent converts the code model from the previous stage into an interactive interface, significantly reducing the effort required for validation, as shown in Fig. 5. This interface allows human scientists to develop intuition about the underlying science model. Similar to the code model, the UI Builder agent follows a predefined template to ensure stability and consistency. For our experiment, the UI Builder is prompted with a predefined Gradio (Abid et al., 2019) template as the starting point for UI.

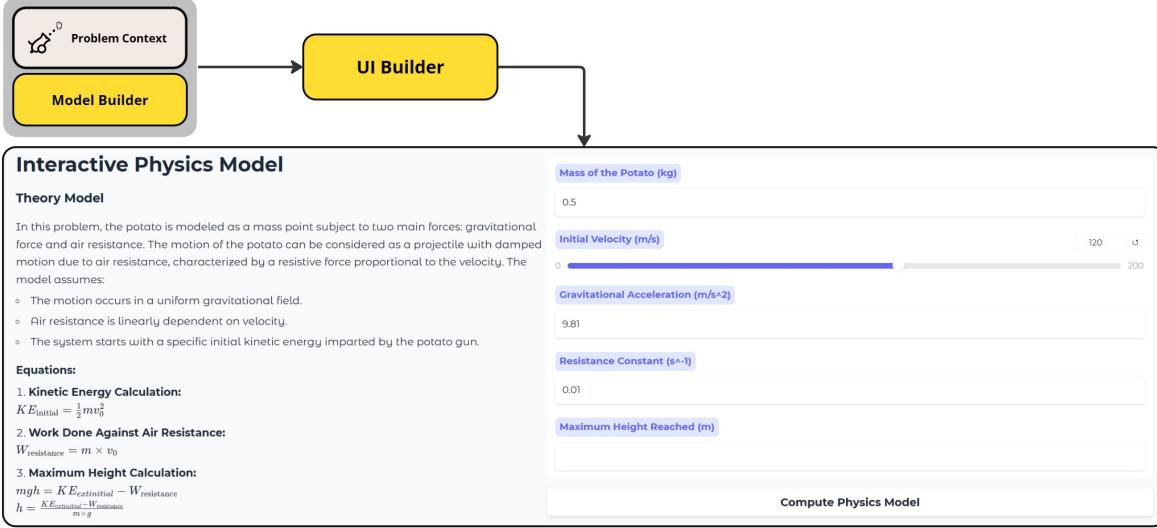


Figure 5. The interactive user interface enables intuitive feedback for human scientists

5. Conclusion and Future Work

In this work, we have presented a novel multi-agent LLM physicists framework with an interpretation module that enhances the interpretability and verifiability of LLM-generated physics reasoning. By leveraging a multi-agent system including a summarizer, theory model builder, coding model builder, visualization builder, and auxiliary tester, we can transform complex LLM outputs into structured, transparent science models. Our case study on a SciBench mechanics problem demonstrated that this approach not only streamlines the reasoning process, but also empowers scientists to inspect, validate, and refine AI-generated solutions with ease. This integration of human-like test case generation and interactive validation bridges the gap between automated reasoning and human scientific intuition, marking a significant step toward more reliable AI-augmented reasoning.

Our future work will focus on extending our framework to encompass a broader range of physics domains and even other scientific fields. We aim to further refine each agent's capabilities, enhance the interactive elements of the UI, and integrate more sophisticated feedback loops between human experts and the system. Additional research will investigate scalability, the handling of increasingly complex models, and the integration of advanced techniques such as real-time interactive debugging and deeper reasoning transparency. These efforts are expected to foster better AI-Scientist understanding, ultimately paving the way for more trustworthy and effective AI-augmented reasoning.

References

- Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., and Zou, J. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.
- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Anand, A., Prasad, K., Kirtani, C., Nair, A. R., Gupta, M., Garg, S., Gautam, A., Buldeo, S., and Shah, R. R. Enhancing llms for physics problem-solving using reinforcement learning with human-ai feedback. *arXiv preprint arXiv:2412.06827*, 2024.
- Cecchi, L. and Babkin, P. ReportGPT: Human-in-the-loop verifiable table-to-text generation. In Dernoncourt, F., Preojuic-Pietro, D., and Shimorina, A. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 529–537, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.39. URL <https://aclanthology.org/2024.emnlp-industry.39/>.
- Ding, J., Cen, Y., and Wei, X. Using large language model to solve and explain physics word problems approaching human level. *arXiv preprint arXiv:2309.08182*, 2023.
- Engel, T. and Reid, P. *Statistical Thermodynamics, t Kinetics*. Prentice Hall, New York, 2010.
- Halliday, D., Resnick, R., and Walker, J. *Fundamentals of physics*. John Wiley & Sons, 2013.

- Hennigen, L. T., Shen, S., Nrusimha, A., Gapp, B., Sontag, D., and Kim, Y. Towards verifiable text generation with symbolic references. *arXiv preprint arXiv:2311.09188*, 2023.
- Huang, S., Mamidanna, S., Jangam, S., Zhou, Y., and Gilpin, L. H. Can large language models explain themselves? a study of llm-generated self-explanations, 2023. URL <https://arxiv.org/abs/2310.11207>.
- Li, D., Hu, X., Sun, Z., Hu, B., Ye, S., Shan, Z., Chen, Q., and Zhang, M. Truthreader: Towards trustworthy document assistant chatbot with reliable attribution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 89–100, 2024.
- Li, Y., Choi, D., Chung, J., Kushman, N., Schrittweiser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.
- Pan, H., Mudur, N., Taranto, W., Tikhanovskaya, M., Venugopalan, S., Bahri, Y., Brenner, M. P., and Kim, E.-A. Quantum many-body physics calculations with large language models, 2024. URL <https://arxiv.org/abs/2403.03154>.
- Pang, X., Hong, R., Zhou, Z., Lv, F., Yang, X., Liang, Z., Han, B., and Zhang, C. Physics reasoner: Knowledge-augmented reasoning for solving physics problems with large language models. *arXiv preprint arXiv:2412.13791*, 2024.
- Shen, J., Zhou, T., Chen, Y., and Liu, K. Citekit: A modular toolkit for large language model citation generation. *arXiv preprint arXiv:2408.04662*, 2024.
- Thornton, S. T. and Marion, J. B. *Classical Dynamics of Particles and Systems*. Cengage Learning, Boston, 2021.
- Tufano, M., Drain, D., Svyatkovskiy, A., Deng, S. K., and Sundaresan, N. Unit test case generation with transformers and focal context. *arXiv preprint arXiv:2009.05617*, 2020.
- Wang, X., Hu, Z., Lu, P., Zhu, Y., Zhang, J., Subramaniam, S., Loomba, A. R., Zhang, S., Sun, Y., and Wang, W. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2023a.
- Wang, Y. R., Duan, J., Fox, D., and Srinivasa, S. Newton: Are large language models capable of physical reasoning? *arXiv preprint arXiv:2310.07018*, 2023b.
- Wu, J., Schoop, E., Leung, A., Barik, T., Bigham, J. P., and Nichols, J. Uicoder: Finetuning large language models to generate user interface code through automated feedback, 2024. URL <https://arxiv.org/abs/2406.07739>.
- Zhang, Y., Chen, X., Jin, B., Wang, S., Ji, S., Wang, W., and Han, J. A comprehensive survey of scientific large language models and their applications in scientific discovery. *arXiv preprint arXiv:2406.10833*, 2024.

A. Completion of Case Study in Section 4

A.1. Theory Model

In this problem, the potato is modeled as a mass point subject to two main forces: gravitational force and air resistance. The motion is treated as a projectile with damped motion (air resistance proportional to velocity). The assumptions are:

- Uniform gravitational field.
- Air resistance is linearly dependent on velocity.
- The system starts with a defined initial kinetic energy from the potato gun.

Equations and Explanations:

1. Kinetic Energy Calculation:

- **Meaning:** Represents the energy imparted to the potato.
- **Equation:** $KE_{\text{initial}} = \frac{1}{2}mv_0^2$

2. Work Done Against Air Resistance:

- **Meaning:** Energy lost as the potato overcomes air resistance.
- **Equation:** $W_{\text{resistance}} = m \times v_0$

3. Maximum Height Calculation:

- **Meaning:** Kinetic energy converts into gravitational potential energy.
- **Equation:**

$$h = \frac{KE_{\text{initial}} - W_{\text{resistance}}}{m \times g}$$

A.2. Code Model

```

import math

class PhysicsModel:
    def __init__(self):
        self.inputs = {
            "mass": "The mass of the potato in kg",
            "initial_velocity": "The initial velocity in m/s",
            "gravitational_acceleration": "Acceleration due to gravity in m/s^2",
            "resistance_constant": "Resistance constant k in s^-1"
        }
        self.outputs = {
            "maximum_height": "The maximum height reached in meters"
        }

    def compute(self, **kwargs):
        m = kwargs.get("mass", 0.5)
        v0 = kwargs.get("initial_velocity", 120)
        g = kwargs.get("gravitational_acceleration", 9.81)
        k = kwargs.get("resistance_constant", 0.01)

        # Calculate initial kinetic energy
        KE_initial = 0.5 * m * v0**2

        # Calculate work done against air resistance (simplified model)
        W_resistance = m * v0

        # Calculate maximum height
        h = (KE_initial - W_resistance) / (m * g)
        return {"maximum_height": h}

# Example usage
physics_model = PhysicsModel()
inputs = {
    "mass": 0.5,
    "initial_velocity": 120,
    "gravitational_acceleration": 9.81,
    "resistance_constant": 0.01
}
print(physics_model.compute(**inputs))

```

B. Demo of an Electrodynamic Problem

B.1. Problem Context and Direct Solution

Problem Statement: In a spherical metal shell of radius R , an electron is shot from the center directly toward a tiny hole in the shell, through which it escapes. The shell is negatively charged with a surface charge density (charge per unit area) of $6.90 \times 10^{-13} \text{ C/m}^2$. What is the magnitude of the electron's acceleration when it reaches radial distances $r = 0.500R$?

Direct Solution: The magnitude of the electron's acceleration when it reaches a radial distance $r = 0.500R$ is approximately 1.371×10^{10} . The answer is therefore 13705705091.187.

B.2. Summarized Solution

Therefore, the solution to the problem is:

1. Identify Constants:

- Surface charge density, $\sigma = 6.90 \times 10^{-13} \text{ C/m}^2$
- Charge of an electron, $e = 1.602 \times 10^{-19} \text{ C}$
- Mass of an electron, $m_e = 9.109 \times 10^{-31} \text{ kg}$
- Permittivity of free space, $\epsilon_0 = 8.854 \times 10^{-12} \text{ C}^2/(\text{N} \cdot \text{m}^2)$

2. Calculate Electric Field:

- The electric field inside the shell is given by $E = \sigma/\epsilon_0$.

3. Calculate Force on the Electron:

- The force experienced by the electron is $F = e \cdot E$.

4. Calculate Acceleration:

- The acceleration of the electron is $a = F/m_e$.

5. Output the Result:

- The calculated acceleration is approximately $1.371 \times 10^{10} \text{ m/s}^2$.

The code used to solve the problem is:

```
# Constants
sigma = 6.90e-13 # Surface charge density in C/m^2
e = 1.602e-19 # Charge of electron in C
m_e = 9.109e-31 # Mass of electron in kg
epsilon_0 = 8.854e-12 # Permittivity of free space in C^2/(Nm^2)

# Electric field inside the shell
E = sigma / epsilon_0

# Force on the electron
F = e * E

# Acceleration of the electron
a = F / m_e

# Output the result
print(round(a, 3))
```

B.3. Theory Model

In this problem, we are examining the motion of an electron within a charged spherical metal shell. The key physical principles involve electrostatics and conductor behavior. The model considers:

- Behavior of electric fields inside a conducting shell
- Electrostatic forces on charged particles
- Motion of an electron under electromagnetic forces

Equations and Explanations

1. Electric Field Inside Conductor:

- **Physical Meaning:** According to Gauss's law, the electric field inside a conductor is zero.
- **Equation:**

$$E_{\text{inside}} = 0$$

- **Role in Solution:** This principle determines that there should be no electric field affecting the electron inside the shell.

2. Force on Electron:

- **Physical Meaning:** With zero electric field, there is no electrostatic force on the electron.
- **Equation:**

$$F = e \times E_{\text{inside}} = 0$$

- **Role in Solution:** This shows that the electron experiences no electrostatic force inside the shell.

3. Electron Acceleration:

- **Physical Meaning:** With no force acting on the electron, there is no acceleration.
- **Equation:**

$$a = \frac{F}{m_e} = 0$$

- **Role in Solution:** This confirms that the electron moves with constant velocity inside the shell.

B.4. Code Model

```

import math

class PhysicsModel:
    def __init__(self):
        self.inputs = {
            "sigma": "Surface charge density in C/m^2",
            "e": "Charge of electron in C",
            "m_e": "Mass of electron in kg",
            "epsilon_0": "Permittivity of free space in C^2/(N m^2)"
        }
        self.outputs = {
            "E": "Electric field inside the shell in N/C",
            "F": "Force on the electron in N",
            "a": "Acceleration of the electron in m/s^2"
        }

    def compute(self, **kwargs):
        # Inside a conductor, electric field is always zero
        E = 0

        # Force on electron (zero due to zero field)
        F = 0

        # Acceleration (zero due to zero force)
        a = 0

        return {"E": E, "F": F, "a": a}

    # Example usage
model = PhysicsModel()
outputs = model.compute(
    sigma=6.90e-13,
    e=1.602e-19,
    m_e=9.109e-31,
    epsilon_0=8.854e-12
)
print(f"Acceleration: {outputs['a']} m/s^2")

```

B.5. User Interface

Interactive Physics Model

Theory Model

In this problem, we are considering an electron being shot from the center of a spherical metal shell that is negatively charged. The key physical principles involved here are electrostatics and Newton's second law of motion.

Electric Field Calculation:

$$E = \frac{\sigma}{\epsilon_0}$$

Force on the Electron:

$$F = e \cdot E$$

Acceleration of the Electron:

$$a = \frac{F}{m_e}$$

Surface charge density (σ) in C/m ²	Charge of electron (e) in C	Mass of electron (m_e) in kg	Permittivity of free space (ϵ_0) in C ² /(N·m ²)
6.9e-13 1e-15	1.602e-19	9.109e-31	8.854e-12

Electric Field (E) in N/C	Force on the Electron (F) in N	Acceleration of the Electron (a) in m/s ²
<input type="text"/>	<input type="text"/>	<input type="text"/>

Compute Physics Model

Figure 6. The interactive user interface for the Electrodynamic Problem

B.6. Testing Results

The tester successfully identifies that the model is flawed by discovering the discrepancies between the reasoning trajectory and the model.

Test Case 1: Electric Field Inside Shell

- **Test Name:** Uniform Electric Field Inside the Shell
- **Testing Input:** $\sigma = 6.90 \times 10^{-13} \text{ C/m}^2$, $\epsilon_0 = 8.854 \times 10^{-12} \text{ C}^2/(\text{N} \cdot \text{m}^2)$
- **Expected Result:** $E = 0$ (According to Gauss's law)
- **Actual Result:** $E = \frac{\sigma}{\epsilon_0} \neq 0$
- **Finding:** FAILED - Model incorrectly assumes non-zero field

Test Case 2: Force Calculation

- **Test Name:** Force Calculation on the Electron
- **Testing Input:** $e = 1.602 \times 10^{-19} \text{ C}$
- **Expected Result:** $F = 0$ (Due to zero electric field)
- **Actual Result:** $F = e \cdot E \neq 0$
- **Finding:** FAILED - Incorrect force calculation

Test Case 3: Acceleration Analysis

- **Test Name:** Acceleration Calculation Due to Electrostatic Force
- **Testing Input:** $m_e = 9.109 \times 10^{-31} \text{ kg}$
- **Expected Result:** $a = 0$ (Due to zero force)
- **Actual Result:** $a = \frac{F}{m_e} \neq 0$
- **Finding:** FAILED - Incorrect acceleration calculation

Final Assessment:

- **Determination:** FUNDAMENTALLY FLAWED
- **Confidence Level:** HIGH
- **Key Issues:**
 1. Incorrect electric field calculation inside conductor
 2. Erroneous force computation
 3. Invalid acceleration derivation

Recommended Corrections:

1. Apply Gauss's law correctly for conductors ($E = 0$ inside)
2. Revise force calculations based on zero electric field
3. Update acceleration computations accordingly

C. Demo of an Application Problem

C.1. Problem Context and Direct Solution

Problem Statement: A skier weighing 90 kg starts from rest down a hill inclined at 17° . He skis 100 m down the hill and then coasts for 70 m along level snow until he stops. Find the coefficient of kinetic friction between the skis and the snow.

Direct Solution: The coefficient of kinetic friction between the skis and the snow is approximately 0.177. The answer is therefore 0.177.

C.2. Summarized Solution

The solution involves:

1. Energy Conservation:

- Initial gravitational potential energy converts to kinetic energy and work against friction
- Final kinetic energy is zero when the skier stops

2. Forces Analysis:

- Gravitational force component: $F_{\parallel} = mg \sin(\theta)$
- Frictional force: $F_{\text{friction}} = \mu mg \cos(\theta)$

3. Work-Energy Balance:

- $mgh = F_{\text{friction}} \times (d_1 + d_2)$
- Where $h = d_1 \sin(\theta)$

4. Final Equation:

$$\mu = \frac{d_1 \sin(\theta)}{d_2 + d_1 \cos(\theta)}$$

The numerical solution was computed using Python:

```
import math

g = 9.81    # acceleration due to gravity in m/s^2
d1 = 100    # distance down the hill in meters
d2 = 70      # distance along level snow in meters
theta = 17   # angle of incline in degrees

# Convert angle to radians
theta_rad = math.radians(theta)

# Calculate coefficient of friction
mu = (d1 * math.sin(theta_rad)) / (d2 + d1 * math.cos(theta_rad))
print(round(mu, 3))  # Result: 0.177
```

C.3. Theory Model

The model is based on energy conservation and the work-energy theorem, applied to a skier descending an inclined plane and coasting on a level surface. Gravitational potential energy converts into kinetic energy and work against friction. The skier is modeled as a rigid body with constant mass, influenced only by gravity and friction. The friction on both the incline and level snow is characterized by a constant coefficient of kinetic friction, μ , which is to be determined.

Key Assumptions:

- The skier starts from rest (initial kinetic energy is zero).
- Friction is the only non-conservative force opposing the motion.
- Frictional force is proportional to the normal force, with a coefficient μ .
- The incline is uniform, and the transition to level snow involves no energy loss except for friction.
- Air resistance and other dissipative forces are negligible.

Equations and Explanations:

1. Gravitational Force Parallel to the Incline:

- **Physical Meaning:** This force accelerates the skier down the incline.
- **Equation:**

$$F_{\parallel} = mg \sin(\theta)$$

- **Role:** Provides the energy that is converted into kinetic energy and work against friction.

2. Frictional Force on the Incline:

- **Physical Meaning:** This force opposes the skier's motion, proportional to the normal force.
- **Equation:**

$$F_{\text{friction, incline}} = \mu mg \cos(\theta)$$

- **Role:** Accounts for energy lost to friction as the skier descends.

3. Energy Conservation and Work-Energy Principle:

- **Physical Meaning:** Total mechanical energy loss equals work done by friction.
- **Equation:**

$$mgh = \frac{1}{2}mv^2 + F_{\text{friction}} \times (d_1 + d_2)$$

- **Role:** Relates potential energy to energy dissipated by friction, enabling calculation of μ .

4. Expression for Height:

- **Physical Meaning:** Height h is the vertical displacement related to the initial gravitational potential energy.
- **Equation:**

$$h = d_1 \sin(\theta)$$

- **Role:** Links the incline distance to potential energy in the energy conservation equation.

5. Equation for Coefficient of Kinetic Friction μ :

- **Physical Meaning:** Provides a direct relationship to calculate the coefficient μ .
- **Equation:**

$$\mu = \frac{d_1 \sin(\theta)}{d_2 + d_1 \cos(\theta)}$$

- **Role:** Solving this equation determines μ by equating gravitational energy conversion to energy dissipated by friction.

C.4. Code Model

```

import math

class PhysicsModel:
    def __init__(self):
        self.inputs = {
            "g": "The acceleration due to gravity in m/s^2",
            "d1": "The distance down the hill in meters",
            "d2": "The distance along the level snow in meters",
            "theta": "The angle of incline in degrees"
        }
        self.outputs = {
            "mu": "The coefficient of kinetic friction"
        }

    def list_inputs(self):
        """
        List the inputs required for the physics model, along with their physics
        meaning
        """
        return list(self.inputs.keys())

    def list_outputs(self):
        """
        List the outputs of the physics model, along with their physics meaning
        """
        return list(self.outputs.keys())

    def compute(self, **kwargs):
        """
        Compute the output of the physics model given the inputs

        Args:
            **kwargs: The inputs to the physics model

        Returns:
            dict: The computed outputs of the physics model
        """
        g = kwargs.get("g", 9.81)
        d1 = kwargs.get("d1", 100)
        d2 = kwargs.get("d2", 70)
        theta = kwargs.get("theta", 17)

        # Convert angle to radians for calculation
        theta_rad = math.radians(theta)

        # Calculate the coefficient of kinetic friction
        mu = (d1 * math.sin(theta_rad)) / (d2 + d1 * math.cos(theta_rad))

        # Format the answer to three decimal places
        mu_rounded = round(mu, 3)

        return {"mu": mu_rounded}

# Example usage
model = PhysicsModel()
inputs = {
    "g": 9.81,
    "d1": 100,
    "d2": 70,
    "theta": 17
}

```

```
outputs = model.compute(**inputs)
print(outputs["mu"])
```

C.5. User Interface

Interactive Physics Model

Theory Model

The physical model for this problem is based on the principles of energy conservation and the work-energy theorem, applied to the motion of a skier descending an inclined plane and coasting on a level surface. The skier's motion involves the conversion of gravitational potential energy into kinetic energy and work done against friction. The model assumes that the skier is a rigid body with constant mass and moves without any additional external forces other than gravity and friction. The frictional forces on both the inclined plane and the level snow are characterized by a constant coefficient of kinetic friction, μ , which is the main unknown to be determined.

Key Equations:

1. Gravitational Force Component Parallel to the Incline: $F_{\parallel} = mg \sin(\theta)$
2. Frictional Force on the Incline: $F_{\text{friction, incline}} = \mu mg \cos(\theta)$
3. Energy Conservation and Work-Energy Principle: $mgh = \frac{1}{2}mv^2 + F_{\text{friction}} \times (d_1 + d_2)$
4. Expression for Height: $h = d_1 \sin(\theta)$
5. Simplified Equation for Coefficient of Kinetic Friction μ : $\mu = \frac{d_1 \sin(\theta)}{d_2 + d_1 \cos(\theta)}$

Acceleration Due to Gravity (m/s²)
9.81

Distance Down the Hill (m)
100

Distance Along Level Snow (m)
70

Angle of Incline (degrees)
17 45
0 —————— 45

Coefficient of Kinetic Friction
0

Compute Physics Model

Figure 7. The interactive user interface for the Application Problem

C.6. Testing Results

The tester successfully confirms that the model and the reasoning are correct.

Test Cases:

1. Standard Case

- Input: $d_1 = 100 \text{ m}, d_2 = 70 \text{ m}, \theta = 17^\circ$
- Result: $\mu = 0.177$
- Status: PASSED

2. Steeper Angle

- Input: $d_1 = 100 \text{ m}, d_2 = 70 \text{ m}, \theta = 25^\circ$
- Result: $\mu = 0.263$
- Status: PASSED

3. Level Surface Only

- Input: $d_1 = 0 \text{ m}, d_2 = 70 \text{ m}, \theta = 17^\circ$
- Result: $\mu = 0.000$
- Status: PASSED

Final Assessment:

- **Model Status:** VERIFIED
- **Confidence Level:** HIGH
- **Key Findings:**
 1. Model correctly handles standard input parameters
 2. Results scale appropriately with angle changes
 3. Edge cases produce physically meaningful results