Physics Reasoner: Knowledge-Augmented Reasoning for Solving Physics Problems with Large Language Models

Xinyu Pang[1], Ruixin Hong[1], Zhanke Zhou[2], Fangrui Lv[1],
Xinwei Yang[1], Zhilong Liang[1], Bo Han[2], Changshui Zhang[1]
[1]Institute for Artificial Intelligence, Tsinghua University (THUAI);
[1]Beijing National Research Center for Information Science and Technology (BNRist);
[1]Department of Automation, Tsinghua University, Beijing, P.R.China
[2]TMLR Group, Hong Kong Baptist University
`{pangxy22, hrx20, lvfr23, yangxw21, liangzl20}@mails.tsinghua.edu.cn,`
`{cszkzhou, bhanml}@comp.hkbu.edu.hk,`
`zcs@mail.tsinghua.edu.cn`

Abstract

Physics problems constitute a significant aspect of reasoning, necessitating complicated reasoning ability and abundant physics knowledge. However, existing large language models (LLMs) frequently fail due to a lack of knowledge or incorrect knowledge application. To mitigate these issues, we propose Physics Reasoner, a knowledge-augmented framework to solve physics problems with LLMs. Specifically, the proposed framework constructs a comprehensive formula set to provide explicit physics knowledge and utilizes checklists containing detailed instructions to guide effective knowledge application. Namely, given a physics problem, Physics Reasoner solves it through three stages: problem analysis, formula retrieval, and guided reasoning. During the process, checklists are employed to enhance LLMs' self-improvement in the analysis and reasoning stages. Empirically, Physics Reasoner mitigates the issues of insufficient knowledge and incorrect application, achieving state-of-the-art performance on SciBench with an average accuracy improvement of 5.8%.[1]

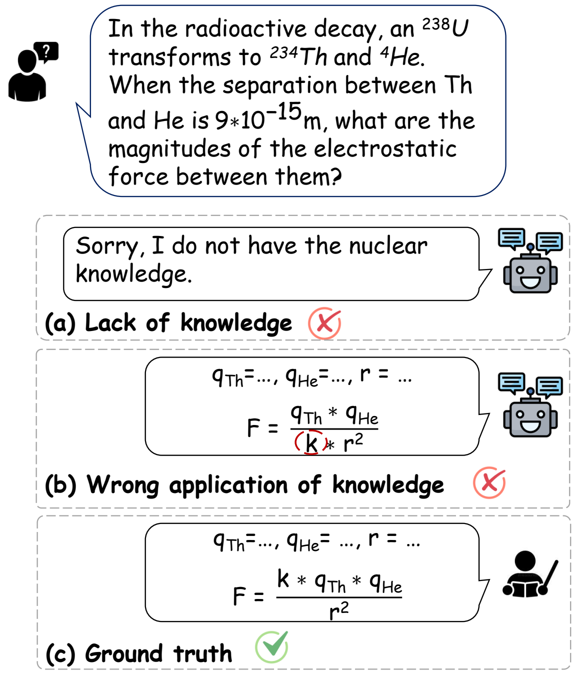

In the radioactive decay, an ^{238}U transforms to ^{234}Th and ^{4}He. When the separation between Th and He is $9*10^{-15}$m, what are the magnitudes of the electrostatic force between them?
Sorry, I do not have the nuclear knowledge.
(a) Lack of knowledge
q_{Th}=..., q_{He}=..., r = ... $F = \frac{q_{Th} * q_{He}}{k * r^2}$
(b) Wrong application of knowledge
q_{Th}=..., q_{He}= ..., r = ... $F = \frac{k * q_{Th} * q_{He}}{r^2}$
(c) Ground truth

Figure 1: Exemplar error cases. Given a complex physics problem, LLMs may make mistakes due to a lack of physics knowledge, as illustrated in (a), or incorrect application of knowledge, as shown in (b). The ground truth answer to this question is shown in (c).

1 Introduction

Physics problems are essential for evaluating the capabilities of large language models (LLMs), assessing the models' comprehension of the natural world, and their ability to navigate complex scenarios. Inadequate performance on physics problems by LLMs could imply a potential deficiency in their understanding of fundamental real-world concepts such as spatial relationships (Yang et al., 2023) and molecular structures (Flam-Shepherd et al., 2021), significantly hindering their effectiveness in real-world applications and scientific discoveries.

However, solving physics problems is highly challenging, demanding a high level of abstraction to translate given information into solvable expressions. This process necessitates the acquisition and application of physics knowledge to find an accurate solution *w.r.t.* a given question.

Several methods are proposed to enhance LLMs' reasoning on physics tasks (Li et al., 2024; Chen et al., 2023c). For instance, Chain-of-Thought

---

[1] The code is publicly available at `https://github.com/Xinyu-Pang/Physics_Reasoner`

Proceedings of the 31st International Conference on Computational Linguistics, pages 11274–11289
January 19–24, 2025. ©2025 Association for Computational Linguistics

(CoT) (Wei et al., 2022) asks models to solve step by step, while Chameleon (Lu et al., 2023b) integrates various external tools. However, even equipped with advanced methods, LLMs still struggle to solve physics problems. Using the CoT strategy, GPT-3.5-turbo achieves only 6.8% accuracy on the physics section of SciBench (Wang et al., 2023a) benchmark. Considering the knowledge-intensive nature of these physics problems, we raise the research question: *Can LLMs master physics knowledge and solve physics problems?*

To answer this question, this work starts by analyzing the common errors, which reveal two primary factors behind the frequent failures. Namely, (1) LLMs lack the necessary physics knowledge, including key concepts and formulae, as illustrated in Fig. 1(a); and (2) even when provided with relevant knowledge, LLMs still struggle to apply it to solve problems correctly, as shown in Fig. 1(b).

To mitigate these problems, we propose Physics Reasoner, a novel framework for physics problem reasoning. Conceptually, this framework aims to make up for the knowledge deficiency of LLMs by incorporating (i) a comprehensive formula set for knowledge acquisition, which encompasses 122 formulae, each accompanied by detailed annotations; (ii) detailed checklists for evaluating the correctness of *knowledge application*, which contain detailed instructions for identifying and correcting errors, thereby assisting LLMs in effectively applying the acquired knowledge. Furthermore, Python code is integrated into the solving process to guarantee precise and reliable calculations.

Technically, Physics Reasoner consists of three stages: problem analysis, formula retrieval, and guided reasoning. Given a physics problem, it first comprehends the problem and extracts known variables. After the initial extraction, a checklist is used to review the correctness and completeness of the extraction. Next, it retrieves relevant formulae from a pre-constructed formula set and performs reasoning. Finally, it reviews and refines the reasoning process with another checklist to aid LLMs in applying the retrieved knowledge effectively.

To verify the effectiveness of the proposed framework, we conduct extensive experiments on the SciBench benchmark. Notably, Physics Reasoner outperforms existing methods by an average of 5.8% in accuracy, reduces reasoning errors, and further boosts LLMs' physical reasoning abilities.

Our contributions are summarized as follows:

- We identify the two major limitations of solving physics problems, *i.e.*, (i) lack of knowledge and (ii) incorrect application of knowledge (Sec. 3).
- We propose a novel reasoning framework that integrates knowledge acquisition with the formula set and application with checklists (Sec. 4).
- We conduct extensive experiments on four datasets. The Physics Reasoner achieves an average improvement of 5.8% in accuracy (Sec. 5).

2 Related work

2.1 LLMs for Scientific Reasoning

The rapid progress of LLMs has significantly advanced the field of scientific reasoning. Numerous representative benchmarks have emerged to evaluate scientific reasoning abilities across a wide array of subjects, including math (Zhou et al., 2024; Cobbe et al., 2021; Lu et al., 2023a), physics (Bakhtin et al., 2019; Wang et al., 2023c), and chemistry (Guo et al., 2023; Lu et al., 2022).

LLMs have demonstrated strong abilities in solving scientific problems. Recent approaches that gather annotation data and fine-tune LLMs have yielded remarkable results (Lu et al., 2023c; Lewkowycz et al., 2022; Zhang et al., 2024). Also, strategies utilizing prompts with frozen LLMs have proven effective, significantly reducing training overhead while maintaining or even enhancing performance. For instance, Wang et al. (2023b) proposes self-consistency, which combines different CoT reasoning paths for more probable answers.

However, despite these advancements, few methods are specifically designed to address LLMs' knowledge deficiency and their inability to effectively apply the knowledge. This gap underscores the need for dedicated approaches to enhance the effectiveness of solving physics-related challenges.

2.2 Physics Problem Reasoning

To evaluate LLMs' physical reasoning abilities, several benchmarks have been proposed (Bakhtin et al., 2019; Bisk et al., 2020). For instance, NEWTON (Wang et al., 2023c) consists of labeled object-centric data encompassing 8 physical attributes.

However, these works primarily focus on simple physical attributes of everyday objects, neglecting complex physics problem reasoning. Solving complicated physics problems requires sophisticated reasoning skills with the acquisition and application of extensive physics knowledge.

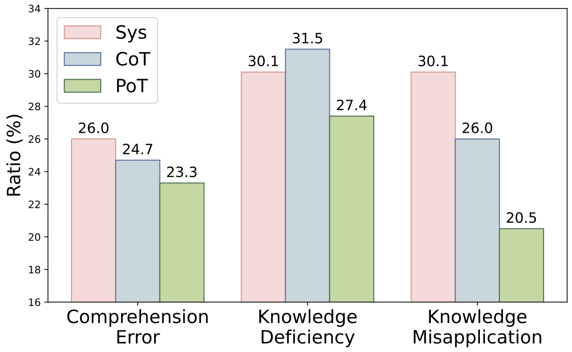


Figure 2: Distribution of the proposed three error types across Sys, CoT, and PoT baselines.

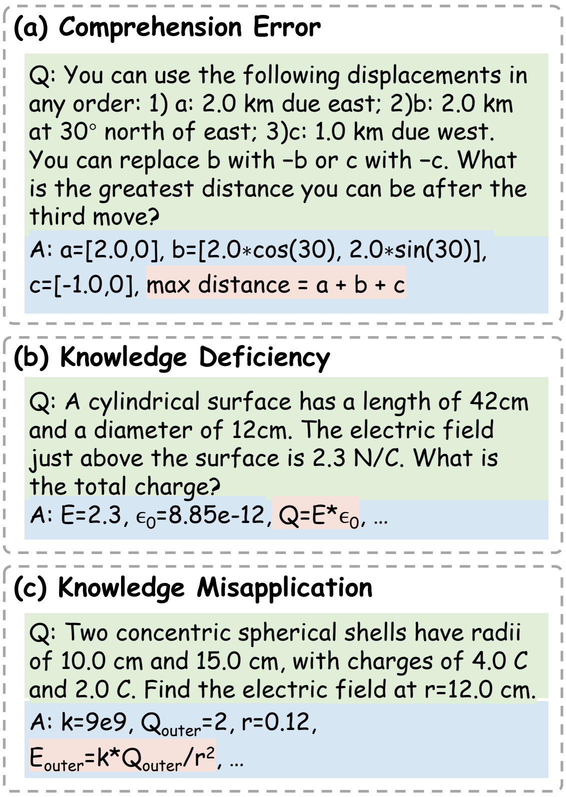
2.3 Self-improvement

A widely accepted way to enhance LLMs' abilities is self-improvement through verification feedback, which enhances their capabilities in multiple aspects, including security (Cao et al., 2024; Li et al., 2023), reasoning (Hong et al., 2023), and other critical areas. For example, high-quality feedback can be used for model fine-tuning (Ouyang et al., 2022; Chen et al., 2023a; Zelikman et al., 2022). Besides, Feedback can be used to re-rank model outputs (Lightman et al., 2023; He et al., 2023; Ni et al., 2023), to modify current reasoning process (Shinn et al., 2023; Chen et al., 2023b), and to guide generation (Yao et al., 2023; Xie et al., 2023). Furthermore, Ribeiro et al. (2021) proposes to examine the capacity of LLMs using a checklist, including general capabilities and test types.

However, for complex problems like physics problems, general self-improvement methods often perform poorly and may even have adverse effects, leading LLMs to incorrect reasoning processes.

3 Limitations of Physics Reasoning

LLMs often encounter difficulties and make errors when tackling complex physics problems. To investigate the primary factors behind these errors, we manually analyze numerous error cases in the fund dataset from the SciBench benchmark using three representative baselines, *i.e.* System (Sys), Chain of Thought (CoT), and Program of Thought (PoT) (Chen et al., 2022). The detailed introduction of these baselines can be found in Sec. 5.3. Following the paradigm in SciBench, experiments are conducted using GPT-3.5-turbo (Peng et al., 2023). Two undergraduate students with strong backgrounds in physics are enlisted to categorize these errors into three types as follows:



(a) Comprehension Error

Q: You can use the following displacements in any order: 1) a: 2.0 km due east; 2)b: 2.0 km at 30° north of east; 3)c: 1.0 km due west. You can replace b with −b or c with −c. What is the greatest distance you can be after the third move?

A: a=[2.0,0], b=[2.0∗cos(30), 2.0∗sin(30)], c=[-1.0,0], max distance = a + b + c

(b) Knowledge Deficiency

Q: A cylindrical surface has a length of 42cm and a diameter of 12cm. The electric field just above the surface is 2.3 N/C. What is the total charge?

A: E=2.3, ϵ_0=8.85e-12, Q=E*ϵ_0, ...

(c) Knowledge Misapplication

Q: Two concentric spherical shells have radii of 10.0 cm and 15.0 cm, with charges of 4.0 C and 2.0 C. Find the electric field at r=12.0 cm.

A: k=9e9, Q_{outer}=2, r=0.12, E_{outer}=k*Q_{outer}/r^2, ...

Figure 3: Example for each error type, where the red highlighted parts indicate errors.

- **Comprehension Error** refers to misunderstanding of the problem, including misinterpreting the context and identifying incorrect assumptions.
- **Knowledge Deficiency** indicates a lack of the necessary knowledge required to solve the problems effectively and accurately.
- **Knowledge Misapplication** involves the incorrect use of relevant knowledge, such as misinterpretation of key concepts, incorrect formula translation, and other related errors.

Each error type is illustrated with an example in Fig. 3. To further explore the frequency of these error types, we compile statistics on error cases on the fund dataset across three baselines. The error distribution for each error type is shown in Fig. 2, suggesting the two main reasons for LLMs' recurring shortcomings: (i) insufficient physics knowledge in LLMs; (ii) challenges in correctly applying the knowledge to solve physics problems. Motivated by the observation, we propose Physics Reasoner to mitigate these issues, detailed in Sec. 4.

4 Physics Reasoner

Physics problem reasoning involves solving problems with domain knowledge and necessary steps.

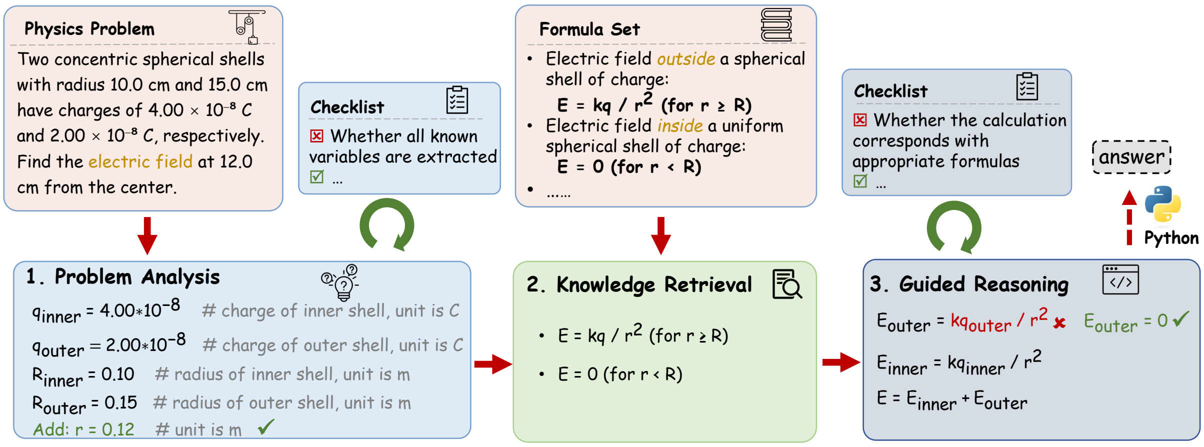

Physics Problem
Two concentric spherical shells with radius 10.0 cm and 15.0 cm have charges of 4.00 × 10⁻⁸ C and 2.00 × 10⁻⁸ C, respectively. Find the electric field at 12.0 cm from the center.
Checklist
☒ Whether all known variables are extracted
☑ ...
Formula Set
• Electric field outside a spherical shell of charge:
E = kq / r² (for r ≥ R)
• Electric field inside a uniform spherical shell of charge:
E = 0 (for r < R)
• ......
Checklist
☒ Whether the calculation corresponds with appropriate formulas
☑ ...
answer
Python
1. Problem Analysis
q_inner = 4.00*10^-8 # charge of inner shell, unit is C
q_outer = 2.00*10^-8 # charge of outer shell, unit is C
R_inner = 0.10 # radius of inner shell, unit is m
R_outer = 0.15 # radius of outer shell, unit is m
Add: r = 0.12 # unit is m ✓
2. Knowledge Retrieval
• E = kq / r² (for r ≥ R)
• E = 0 (for r < R)
3. Guided Reasoning
E_outer = kq_outer / r² ✗ E_outer = 0 ✓
E_inner = kq_inner / r²
E = E_inner + E_outer

Figure 4: Illustration of Physics Reasoner, solving a physics problem using LLMs with the help of the formula set and checklists. The approach contains three stages: problem analysis, knowledge retrieval, and guided reasoning.

The inputs include a problem P and a constructed formula set F. The desired output is the calculated result R in the form of the target unit.

We introduce details of the workflow of our approach, construction of the formula set, and design of checklists in Sec. 4.1, 4.2 and 4.3 respectively.

4.1 Workflow

Physics Reasoner is divided into 3 stages: problem analysis, formula retrieval, and guided reasoning.

As shown in Fig. 4, given a physics problem P, along with our formula set F, Physics Reasoner first comprehends P and then extracts known variables $V = \{v_0, v_1, ..., v_n\}$ from it as incomplete Python code. Here, each variable v_i is roughly defined in a single line of code, followed by comments explaining the specific meaning of the variable and its corresponding unit. After this initial extraction, a checklist CL_{PA} is used to review and refine the analysis process. The details of this checklist are introduced in Sec. 4.3.

The next step is formula retrieval, which is conducted hierarchically. Formulae are categorized into various fields of physics. Each formula is accompanied by a brief description, formula content, and definitions of involved variables. Based on problem text P, the LLM first determines relevant fields and retrieves formulae from them. It then selects formulae $F = \{f_0, f_1, ..., f_m\}$ related to P and writes them as Python code comments, supplementing the extracted variables V.

Subsequently, Physics Reasoner completes the Python code C with precise reasoning using extracted variables V and formulae F, printing the target variable in the required unit at the end. Following this, another checklist CL_{GR} is utilized for guiding and refining the reasoning process. The refined code C' is then executed, and the output variable is taken as the predicted answer.

4.2 Formula Set Construction

This section outlines the principles and process of constructing our formula set, aiming to provide explicit knowledge of physics formulae to enhance knowledge acquisition in LLMs.

4.2.1 Collection Principles

Covering common physics problems. We expect the formula set to cover common college-level physics problems. To achieve this, we select three prevalent college physics textbooks as collection sources and identify four representative fields.

Clarifying detailed introduction and variable definition. Another principle is to explicitly explain the correct conditions for formula applications and provide clear definitions of involved variables. Even with the formula known, correct reasoning is impossible without applying it in an appropriate context or accurately substituting the variables. To enhance knowledge acquisition, each formula in our set is accompanied by a detailed introduction and precise variable definitions.

4.2.2 Formula Collection

We choose three physics textbooks to collect formulae following SciBench: *Fundamentals of Physics* (Halliday et al., 2013), *Statistical Thermodynamics* (Engel and Reid, 2019), and *Classical Dynamics of Particles and Systems* (Greiner and Bromley, 2003). These textbooks are representative of the field of physics, containing abundant knowledge of formulae.

Formula Example

name: "Kepler's Third Law"
content: "T ** 2 = (4 * pi ** 2 / G * M) * a ** 3",
variables:
"T": "Period of the orbit",
"G": "Gravitational constant",
"M": "Mass of the central body",
"a": "Semi-major axis of the orbit"

Figure 5: An example of our formula annotation.

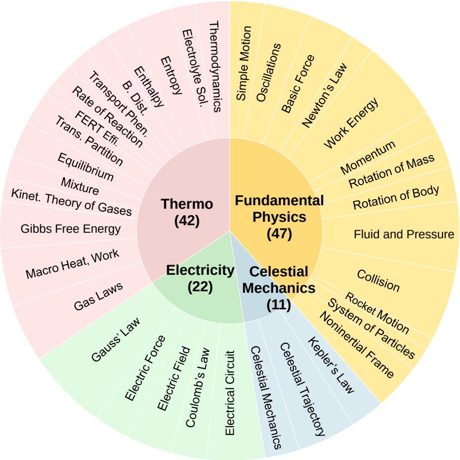


Figure 6: Subfields in our formula set.

Initially, we collect 1124 formulae from these textbooks and refine the collection by removing: (i) repetitive formulae, such as those differing only in the order of multiplication and division; (ii) intermediate calculation processes that do not constitute scientific formulae; and (iii) conclusions from individual example problems that are not universal. After the refinement, we identify 122 representative formulae. As shown in Fig. 6, we categorize these formulae into 36 subfields, which are further classified into four major fields: Fundamental Physics, Celestial Mechanics, Electricity, and Thermodynamics. For each formula, we include a detailed introduction and definitions of the involved variables to enhance knowledge application. Fig. 5 provides an example of a formula annotation.

4.2.3 Comparison with Existing Dataset

To the best of our knowledge, there are few datasets of physics formulae designed for complex physics problem reasoning. Previous formula-based datasets mainly consist of commonsense formulae. For instance, in the MAWPS-F (Liu et al., 2023) dataset, the most frequently used formula is $total\,amount = unit\,amount * total\,number$, which is basic commonsense knowledge.

Formula Set	Physics	English	Theorem
Math23K-F (Liu et al., 2023)	✗	✓	✗
MAWPS-F (Liu et al., 2023)	✗	✓	✗
FormulaQA (Li et al., 2024)	✓	✗	✗
Ours	✓	✓	✓

Table 1: Comparison with other formula sets.

The most relevant work to ours is FormulaQA (Li et al., 2024), a question-answering numerical reasoning dataset. It contains annotated problems selected from Chinese junior high school physics examinations. However, the formulae in it are primarily abstract expressions derived from individual example problems rather than scientific theorems in physics, such as $Degree\,of\,temperature\,increase = Final\,temperature - Initial\,temperature$. This makes them heavily dependent on the problem and limits their potential for extensibility.

A detailed comparison between our formula set and other existing datasets is presented in Table 1.

4.3 Checklist

This section outlines the principles and procedures for designing effective checklists, which include detailed instructions for identifying and correcting prone errors. These checklists are utilized to help LLMs self-improve by enabling them to apply physics knowledge more accurately.

4.3.1 Design Principles

Focusing on common mistakes of physics problems. We propose checklists to help avoid common mistakes LLMs often make. By thoroughly analyzing numerous incorrect processes in physics problems, we identify and summarize representative errors to include in the checklists. For example, LLMs often confuse vectors with scalars, leading to incorrect reasoning. The checklist includes steps to verify the text and distinguish vectors from scalars.

Enhancing LLMs' application concisely. Our goal is using checklists to enhance LLMs' accurate application of domain knowledge. Merely acquiring the necessary physics knowledge is not enough to successfully solve a complex problem. Additionally, we found that providing too many instructions can distract LLMs and hinder their ability to apply knowledge effectively. Therefore, our checklists are designed to be both instructive and concise, offering clear guidance without leading to distraction.

Checklist for Problem Analysis

You are an excellent physics teacher. Given a physics problem and its extracted known variables, please examine the variables using the following checklist and revise it:

Checklist:
- Understand the problem, whether all known variables are extracted.
- Whether the variable names are clear and can distinguish the different objects to which they may belong.
- Whether the variables have been converted to SI units.
- Check if the variables have confused vectors with scalars.

Return the correct known variables in the format of Python code, encasing it within triple back-ticks for clarity.

{Problem text}
{Initial extracted variables}

Figure 7: Checklist for problem comprehension, verifying extracted variables and units

Checklist for Guided Reasoning

You are an excellent physics teacher. Given a physics problem and its Python code, please review the code using the following checklist and improve it:

Checklist:
- If the problem text is correctly understood and solved.
- Whether the calculation corresponds with correct, appropriate formulas.
- Whether the variables and constants are correctly defined before use, ensuring that constants are not confused.
- If the problem is solved, whether the code prints the first target variable at the end.
- Check the unit, whether the target variable is printed in the required unit.

Please reiterate your code and return the improved code. Encasing the code within triple back-ticks for clarity.

{Problem text}
{Initial solving process}

Figure 8: Checklist for calculation process, verifying physics knowledge application.

4.3.2 Design Procedure

Since the process of solving physics problems is sequential, an error in an earlier step may lead to mistakes in subsequent reasoning. We observe that errors primarily occur during problem comprehension and knowledge application. Motivated by this, we propose separate checklists for both problem comprehension and calculation stages, trying to ensure the accuracy of each step.

We design checklists based on the errors observed. By testing several LLM-based reasoning methods on abundant physics problems, we identify key points that need to be carefully checked and include them in the checklists. Additionally, common errors vary across different fields. For instance, in electronics problems, LLMs often confuse Coulomb's constant k with permittivity of free space ϵ_0 when calculating electronic fields. To address such issues, we design specialized checklists for various fields. Namely, our checklist examples are shown in Fig. 7 and Fig. 8.

5 Experiments

5.1 Dateset

We conduct experiments on three physics datasets from the SciBench benchmark, systematically examining the reasoning capabilities required to solve complex problems. These datasets cover a wide range of fields in physics, including motion, kinetic energy, electronics, and so on. The diversity enhances the comprehensiveness of evaluating LLMs' abilities in physics problem reasoning.

The datasets are manually collected from various college-level physics textbooks and are selected for challenging, free-response answers. Each dataset is divided into two parts, P_s and P_w. P_w contains most of the problems without solutions, while P_s comprises several problems with detailed solutions, which can be used as examples in a few-shot setting. Detailed statistics are presented in Table 3.

5.2 Implementation Details

We test on both open-source and close-source LLMs, including GPT-4 (OpenAI, 2023), GPT-3.5 (Peng et al., 2023), and Llama 3 (Meta, 2024). For evaluation metrics, we follow Wang et al. (2023a) to compare the predicted answers with the correct answers, allowing a relative tolerance of 5%. Few-shot examples are randomly selected within each textbook, ensuring consistency with SciBench. The temperature is set to zero to reduce randomness. Llama3 is executed on an NVIDIA A800-SXM4-80GB GPU, with a time cost of approximately two hours for these datasets. The other two models are accessed through OpenAI API.

5.3 Baselines

We evaluate multiple baselines under the few-shot setting, following the SciBench evaluation

Method	GPT-3.5-turbo				Llama3-70B				GPT-4-turbo			
	fund	**thermo**	**class**	**Avg.**	**fund**	**thermo**	**class**	**Avg.**	**fund**	**thermo**	**class**	**Avg.**
System	13.7	11.9	2.1	9.3	5.5	14.9	14.9	11.8	42.5	41.8	25.5	36.6
CoT	17.8	9.0	4.3	10.3	27.4	16.4	19.2	21.0	50.7	26.9	25.5	34.4
PoT	28.8	13.4	12.8	18.3	45.2	**34.3**	14.9	31.5	63.0	46.3	31.9	47.1
PoT + Self-Correction	30.1	11.9	10.6	17.6	34.3	20.9	14.9	23.4	24.7	25.4	19.2	23.1
PoT + Self-Refine	30.1	19.4	14.9	21.5	30.1	26.9	17.0	24.7	53.4	23.9	31.9	36.4
Physics Reasoner (Ours)	**38.4**	**25.4**	**23.4**	**29.1**	**50.7**	29.9	**25.5**	**35.4**	**70.0**	**50.8**	**38.3**	**53.0**

Table 2: Accuracy results (%) on three SciBench physics datasets: fund, thermo, and class under few-shot setting. **Boldface** numbers highlight the best results; underlined numbers represent the second-best.

Dataset	**Field**	**# P**	**# S**
fund	electronics	83	10
thermo	thermodynamics	84	17
class	classical dynamics	54	7

Table 3: Dataset Statistics. $\#P$ and $\#S$ represent the number of total problems and problems with detailed solutions, respectively.

paradigm to ensure a rigorous evaluation process:

- **System** refers to directly feeding the problem with instructions describing question types.
- **CoT** points to the Chain of Thought strategy, where the model outputs the reasoning process step by step before providing a predicted answer.
- **PoT** guides LLMs to write and then execute a Python program to determine the final answer.
- **PoT + Self-Correction** Huang et al. (2023) asks LLMs to rectify initial PoT responses.
- **PoT + Self-Refine** requests the model to provide feedback on its initial output by PoT and uses the feedback to refine itself (Madaan et al., 2023). The iteration number is set to 1 in this case.

6 Result Analysis

6.1 Reasoning Quality

Table 2 presents the accuracy of different models on the test set of the three datasets. Based on the results, we have the following observations:

Physics Reasoner achieves superior accuracy. As shown in the table, Physics Reasoner outperforms all baselines, improving average accuracy from 21.5% to 29.1% on GPT-3.5-turbo, from 31.5% to 35.4% on Llama3:70b, and from 47.1% to 53.0% on GPT-4-turbo. It is worth noting that Physics Reasoner can be regarded as an enhanced form of PoT. The improvements indicate that Physics Reasoner benefits from the formula set to provide physics knowledge and from the checklists, which enhance knowledge application.

Physics Reasoner gains considerable advancement in complex problems. We observe that Physics Reasoner performs better on fund dataset compared to the other datasets. One possible explanation is that fund contains problems in more fundamental fields, such as gravitation, which is more frequently encountered in LLMs' training data, leading to potentially better performance. Also, as shown in Table 7, a larger proportion of problems in fund requires formulae usage and the average number of formulae per problem is also higher. These differences highlight Physics Reasoner's capability in complex physics problem reasoning.

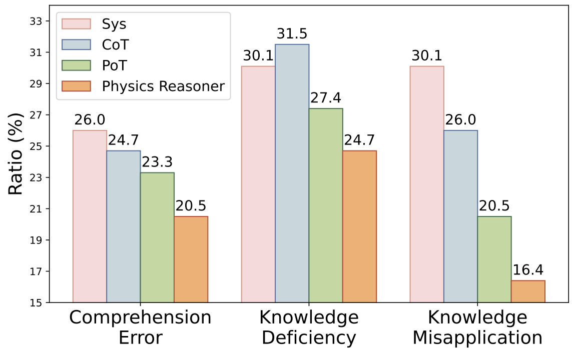


Figure 9: Proportion of each error type for Physics Reasoner and other three baselines.

We further conduct manual analysis of error cases for Physics Reasoner on the fund dataset with the GPT-3.5-turbo model following the error types defined in Sec. 3. The results are shown in Figure 9, revealing that Physics Reasoner significantly reduces all three types of errors.

6.2 Tool-Argumented Baselines

For a more thorough comparative analysis, we also conducted supplementary experiments with two representative tool-argumented baselines, *i.e.* CREATOR (Qian et al., 2023) and Chameleon (Lu et al., 2023b), utilizing GPT-3.5-turbo. The parameters are aligned with the source code separately, and both the few-shot examples and metrics are consistent with our main experiments. The accuracy results (%) are represented in Table 4.

Method	**fund**	**thermo**	**class**	**Avg.**
CREATOR	21.9	17.9	8.5	16.1
Chameleon	32.9	14.9	19.2	22.3
Physics Reasoner (Ours)	**38.4**	**25.4**	**23.4**	**29.1**

Table 4: Results (%) of tool-argumented baselines.

Method	**fund**	**thermo**	**class**	**Avg.**
CREATOR	571	533	400	502
Chameleon	538	520	289	449
Physics Reasoner (Ours)	**470**	**435**	**253**	**386**

Table 5: Token cost of tool-argumented baselines (in thousands of tokens, k tokens).

Physics Reasoner outperforms other baselines across all datasets, achieving an average improvement of 6.73%. The consistent advantage underscores the significant efficacy of our approach.

6.3 Token Cost Comparison

We count the token cost of different methods with GPT-3.5-turbo in Table 5. The total tokens of Physics Reasoner are fewer than the previous tool-argument methods. It is worth noting that our approach is not iterative and requires a fixed 4-step process per problem. In contrast, Chameleon requires 5-6 steps and CREATOR demands 4-5 steps, leading to a higher overall token expenditure.

6.4 Ablation Study

We further investigate the effectiveness of each component in Physics Reasoner, focusing on three key components: the formula set, the checklist for problem comprehension, and the checklist for calculation, which are sequentially removed.

Table 6 presents the results of the ablation study conducted with GPT-3.5-turbo on the fund dataset, offering the greatest diversity of questions. We observe that the inclusion of each component leads to a consistent improvement, underscoring the necessity of each component in our method.

6.5 Necessity of Formulae in Problem Solving

From the analysis of incorrect examples in Sec. 3, we observe that LLMs often make mistakes due to a lack of relevant knowledge. After the construction of our formula set, we are interested in whether the formulae are utilized in the reasoning process. Formula usage is summarized in Table 7, from which we observe the following key points:

Most problems necessitate formulae. The vast majority of problems in the fund and class datasets rely on formulae from our formula set, as do approximately half of the problems in the thermo dataset. Overall, most problems across the three datasets require the use of physics formulae from our formula set for solutions.

Formula Set	**CL_VE**	**CL_SP**	**Accuracy**
✓	✓	✓	38.4
✓	✓	✗	35.2
✓	✗	✗	30.1
✗	✗	✗	28.8

Table 6: Ablation study results (%) of Physics Reasoner on fund dataset with GPT-3.5-turbo, in which CL_VE represents checklist for variable extraction, CL_SP represents checklist for solving process.

Dataest	**Num. Form.**	**% Form.**	**Avg. Num.**
fund	58	79.45	1.58
thermo	33	49.25	0.63
class	35	74.47	0.89

Table 7: Formula usage across datasets: "Num. Form." for the number of problems with formulae, "% Form." for the percentage of problems with formulae, and "Avg. Num." for the average number of formulae per problem.

The number of formulae required for each problem varies. While some simple problems can be solved with a single formula, others may require up to four. Generally, problems with fewer formulae are easier to solve. Complex problems require a thorough application of multiple formulae and detailed analysis, increasing their difficulty.

6.6 Checklists Enhance Effective Knowledge Application

To assess the efficacy of checklists in knowledge application, we carefully examine the results of Physics Reasoner and two self-improvement methods, *i.e.* self-correction and self-refine on the fund dataset using the GPT-4-turbo model. The proportion of initial errors that are corrected and misled problems are shown in Table 8.

Our method corrects 16.67% of incorrect reasoning processes, whereas the other two baselines correct only 8.7% and 0.0%. Additionally, the number of misled problems notably decreases with our method, indicating that Physics Reasoner significantly enhances knowledge application.

7 Conclusion

In this paper, we propose Physics Reasoner, a novel framework for physics problem reasoning that inte-

Method	Wrong2Correct↑	Correct2Wrong↓
PoT+Self-Correction	0.0	6.3
PoT+Self-Refine	8.7	26.0
Physics Reasoner (Ours)	**16.7**	**3.9**

Table 8: Ratio (%) to correct incorrect problems (Wrong2Correct) and mislead answers (Correct2Wrong) of different methods on fund dataset with GPT-4-turbo.

grates knowledge acquisition through a formula set and knowledge application with meticulously designed checklists. We compile a representative formula set and design detailed checklists to address issues related to the lack of knowledge or incorrect knowledge application. This approach significantly improves accuracy in solving physics problems and achieves state-of-the-art performance. Given the limited research on physics problem reasoning, we advocate for further studies to explore the potential of this field, aiming to advance the effectiveness of LLMs in complex problem-solving scenarios.

Limitations

In this paper, we propose Physics Reasoner, a novel framework to mitigate knowledge gaps and knowledge misapplication in physics problem reasoning. Despite the promising performance of our approach, we acknowledge that there are areas for improvement and opportunities for future research.

Firstly, we conduct experiments on a limited number of datasets from SciBench using three models. Our results may not fully reflect the reasoning abilities of LLMs under different conditions. Future research should examine a broad range of physics problem datasets and model types to provide a more comprehensive analysis.

Secondly, our formula set covers a limited range of physics fields due to the manual effort required for its construction. Expanding our formula set to include more physics fields would be beneficial and could provide deeper insights to further study.

Aknowledgements

We gratefully acknowledge the anonymous reviewers for their thoughtful feedback. This work was also supported by the National Science and Technology Major Project (No. 2022ZD0114903), the Natural Science Fundation of China (NSFC. No. 62476149), and the Guoqiang Institute of Tsinghua University, with Grant No. 2020GQG0005.

Ethical Considerations

This article adheres to the ACL Code of Ethics. According to our knowledge, our work constitutes foundational research, and we do not identify any significant risks related to malicious harm, environmental impact, fairness, or privacy concerns.

References

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross B. Girshick. 2019. PHYRE: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5083–5094.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Chentao Cao, Zhun Zhong, Zhanke Zhou, Yang Liu, Tongliang Liu, and Bo Han. 2024. Envisioning outlier exposure by large language models for out-of-distribution detection. In *ICML*.

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. 2023a. Improving code generation by training with natural language feedback. *CoRR*, abs/2303.16749.

Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. 2023b. Iterative translation refinement with large language models. *CoRR*, abs/2306.03856.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *CoRR*, abs/2211.12588.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023c. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7889–7901. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro

Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Thomas Engel and Philip Reid. 2019. *Thermodynamics, statistical thermodynamics, and kinetics: physical chemistry*. Prentice Hall Upper saddle River.

Daniel Flam-Shepherd, Kevin Zhu, and Alán Aspuru-Guzik. 2021. Keeping it simple: Language models can learn complex molecular distributions. *CoRR*, abs/2112.03041.

Walter Greiner and D Allan Bromley. 2003. *Classical mechanics: systems of particles and Hamiltonian dynamics*. Springer.

Taicheng Guo, Kehan Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2023. What can large language models do in chemistry? A comprehensive benchmark on eight tasks. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

David Halliday, Robert Resnick, and Jearl Walker. 2013. *Fundamentals of physics*. John Wiley & Sons.

Hangfeng He, Hongming Zhang, and Dan Roth. 2023. Rethinking with retrieval: Faithful large language model inference. *CoRR*, abs/2301.00303.

Ruixin Hong, Hongming Zhang, Xinyu Pang, Dong Yu, and Changshui Zhang. 2023. A closer look at the self-verification abilities of large language models in logical reasoning. *CoRR*, abs/2311.07954.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *CoRR*, abs/2310.01798.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Xiao Li, Sichen Liu, Bolin Zhu, Yin Zhu, Yiwei Liu, and Gong Cheng. 2024. Formulaqa: A question answering dataset for formula-based numerical reasoning. *CoRR*, abs/2402.12692.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *CoRR*, abs/2305.20050.

Jiayu Liu, Zhenya Huang, Zhiyuan Ma, Qi Liu, Enhong Chen, Tianhuang Su, and Haifeng Liu. 2023. Guiding mathematical reasoning via mastering commonsense formula knowledge. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1477–1488. ACM.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023a. Mathvista: Evaluating math reasoning in visual contexts with gpt-4v, bard, and other large multimodal models. *CoRR*, abs/2310.02255.

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. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023b. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023c. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. `https://ai.meta.com/blog/meta-llama-3/`.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-Tau Yih, Sida I. Wang, and Xi Victoria

Lin. 2023. LEVER: learning to verify language-to-code generation with execution. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 26106–26128. PMLR.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Andrew Peng, Michael Wu, John Allard, Logan Kilpatrick, and Steven Heidel. 2023. Gpt-3.5 turbo fine-tuning and api updates. `https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates`.

Cheng Qian, Chi Han, Yi R Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. *arXiv preprint arXiv:2305.14318*.

Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2021. Beyond accuracy: Behavioral testing of NLP models with checklist (extended abstract). In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4824–4828. ijcai.org.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yi Ru Wang, Jiafei Duan, Dieter Fox, and Siddhartha S. Srinivasa. 2023c. NEWTON: are large language models capable of physical reasoning? In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9743–9758. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Decomposition enhances reasoning via self-evaluation guided decoding. *CoRR*, abs/2305.00633.

Yijun Yang, Tianyi Zhou, Kanxue Li, Dapeng Tao, Lusong Li, Li Shen, Xiaodong He, Jing Jiang, and Yuhui Shi. 2023. Embodied multi-modal agent trained by an LLM from a parallel textworld. *CoRR*, abs/2311.16714.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Dongzhan Zhou, Shufei Zhang, Mao Su, Hansen Zhong, Yuqiang Li, and Wanli Ouyang. 2024. Chemllm: A chemical large language model. *CoRR*, abs/2402.06852.

Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. 2024. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

A Details for Formula Set Construction

A.1 Collection Details

Classification: We chose three textbooks as the sources for our formula collection to guarantee correctness and comprehensiveness. Two undergraduate students with strong backgrounds in physics summarize these textbooks and re-categorize them into four representative fields, *i.e.* fundamental physics, celestial mechanics, electricity, and thermodynamics (thermo). Furthermore, each field is classified into several subfields, which are more refined and specific topics.

Selection: We focus on scientific theorems, removing problem-specific formulae and formulae in different formats with the same meanings.

Annotations: Each formula is annotated in three parts: a brief description of the formula, formula content, and clear definitions of involved variables.

A.2 Detailed Statistics

The detailed statistics of our formula set, including the distribution and categorization of formulae across various physics fields and subfields, are shown in Table 9.

B Details for Experiment

B.1 Experimental Setups

Few-shot Examples: we conduct all experiments under few-shot settings to better enable LLMs to solve physics problems. The examples are chosen from problems with detailed solutions in each dataset. Following SciBench, we use 3 examples for the thermo dataset and 4 examples for the other two datasets.

Evaluation Criteria: the prediction is considered correct if it deviates from the correct answer by no more than 5%. For system and CoT, we prompt the LLMs to output the predicted answer in "\\boxed{ }". For other methods, the output Python code is executed which prints the predicted answer. If the output cannot be directly converted to a float, we capture the first number as the result.

B.2 Prompt Templates for Physics Reasoner

The prompt templates for three stages in Physics Reasoner are shown in Table 10. In the formula retrieval stage, our approach first decides relevant subfields and then retrieves formulae from them.

Field	Subfield	Number
Fundamental Physics	Simple Motion	3
	Oscillations	4
	Basic Force	4
	Newton's Law	2
	Work Energy	7
	Momentum	2
	Rotation of Mass	3
	Rotation of Body	3
	Fluid and Pressure	6
	Collision	6
	Rocket Motion	1
	System of Particles	2
	Noninertial Frame	4
Celestial Mechanics	Kepler's Law	3
	Celestial Trajectory	5
	Celestial Mechanics	3
Electricity	Electrical Circuit	5
	Coulomb's Law	3
	Electric Field	4
	Electric Force, Energy	3
	Gauss' Law	7
Thermo	Gas Laws	5
	Macro Heat, Work	5
	Gibbs Free Energy	4
	Kinetic Theory of Gases	2
	Mixture	2
	Equilibrium	3
	Translational Partition	3
	FRET Efficiency	1
	Rate of Reaction	2
	Transport Phenomena	2
	Boltzman Distribution	2
	Enthalpy	2
	Entropy	4
	Electrolyte Solutions	2
	Thermodynamics	3
All		122

Table 9: Detailed distribution of our formula set.

B.3 Prompt Templates for Baselines

The prompt templates for the five baselines are shown in Table 11. For system, CoT, and PoT, we follow the settings in SciBench to ensure the effectiveness of comparisons. For the other two baselines, *i.e.* self-correction and self-refine, we adapt their original prompts to fit the physics problem reasoning task.

B.4 Case Study

We show a representative example by Physics Reasoner method in Fig. 10. In this problem, it successfully comprehends the problem, retrieves proper formulae, and effectively corrects mistakes with checklists.

C Further Experiments

C.1 Inclusion of Additional Datasets

To confirm Physics Reasoner's generalization ability, we also conduct experiments on a more challenging dataset, *i.e.*, TheoremQA. The accuracy results (%) with gpt-3.5-turbo are represented in Table 12. Following the TheoremQA approach, we compare the predicted answers with the correct answers, allowing a relative tolerance of 4%. The temperature is set to zero to eliminate randomness.

Methods	**Accuracy**
CoT	7.6
PoT	14.5
PoT+Self-Refine	21.4
PoT+Self-Correction	22.1
CREATOR	16.8
Chameleon	22.9
Physics Reasoner	26.7

Table 12: Additional results on TheoremQA.

Compared to other baselines, Physics Reasoner enhances accuracy from 22.9% to 26.7%, resulting in an average improvement of 3.8%. The gains observed on both Scibench and TheoremQA demonstrate that Physics Reasoner outperforms other methods across various physics datasets. Additionally, Physics Reasoner exhibits strong generalization, boosting reasoning ability for challenging and infrequent physics problems.

C.2 Direct Comparison with Other Self-Revision Techniques

We have also conducted experiments comparing our approach with other self-revision techniques, namely self-correction and self-refine, using the *fund* dataset. The accuracy results (%) are shown in Table 13.

Methods	**Accuracy**
PoT + Self-Correction	30.1
PoT + Self-Correction + Formula Set	31.5
PoT + Self-Refine	30.1
PoT + Self-Correction + Formula Set	32.9
Physics Reasoner	38.4

Table 13: Comparison with other self-revision baselines with our Formula Set on the fund dataset.

The results indicate that Checklists are more effective than other self-revision techniques. Checklists are carefully designed to assist LLMs in self-improvement by enabling more accurate application of physics knowledge. In contrast, other self-revision techniques may overlook common mistakes in physics problems, resulting in inaccurate correction.

Prompt Templates for Physics Reasoner	

Problem Analysis	You are an excellent student majoring in physics. Given a physics problem, please extract known variables from it in the format of Python code. The problem is intended to cover general topics within [FIELDS], sourced from a college-level textbook, and it requires analytical reasoning in physics.
	The extracted variables should be well-named to represent the objects to which they belong and should be well-annotated with comments that provide sufficient explanations of the variables defined, including their units. Encase the Python code within triple backticks for clarity.
	
	{ Example Problem}
	{ Example Analysis}
	
	{ Problem}

Subfield determination	You are an excellent student majoring in physics, given a physics problem, that could be related to several subfields of physics, including: [POTENTIAL SUBFIELDS], please determine and return relevant subject(s) from the provided list.
	
	{ Example Problem}
	{ Example Subfield(s)}
	
	{ Problem}

Formula Retrieval	You are an excellent student majoring in physics, given a physics problem, please identify and retrieve relevant formulas. Annotate these relevant formulas in Python comments format and encase the Python code within triple backticks for clarity.
	
	{ Example Problem}
	{ Example Potential Formulas in Chosen Subfiled(s)}
	{ Example Retrieved Formulas}
	
	{ Problem}
	{ Potential Formulas in Chosen Subfiled(s)}

Guided Reasoning	You are an excellent student majoring in physics, given a physics problem and its incomplete Python code, please think and complete the code. Print the target variable at the end if the problem has been solved. If there exists more than 1 target variable, print the first one. Remember to check whether the formulas are correctly used. And check whether the variable is printed with the required unit. Return the complete Python code and ensure it within triple backticks for clarity.
	
	{ Example Problem}
	{ Example Incomplete Code}
	{ Example Code}
	
	{ Problem}
	{ Incomplete Code}

Table 10: Prompt templates for Physics Reasoner.

Prompt Templates for Baselines	

Sys	Please provide a clear and step-by-step solution for a scientific problem in the categories of Chemistry, Physics, or Mathematics. The problem will specify the unit of measurement, which should not be included in the answer. Express the final answer as a decimal number with three digits after the decimal point. Conclude the answer by stating "The answer is therefore \\boxed{[ANSWER]}.
	
	{ Example Problem}
	{ Example Answer}
	
	{ Problem}

CoT	Please provide a clear and step-by-step solution for a scientific problem in the categories of Chemistry, Physics, or Mathematics. The problem will specify the unit of measurement, which should not be included in the answer. Express the final answer as a decimal number with three digits after the decimal point. Conclude the answer by stating "The answer is therefore \\boxed{[ANSWER]}.
	
	{ Example Problem}
	{ Example Solution}
	
	{ Problem}

PoT	# Please provide a clear and step-by-step solution for a scientific problem in the categories of Chemistry, Physics, or Mathematics. The problem will specify the unit of measurement. And potential related formulas are given. Please translate the solution steps into Python code and encase the Python code within triple backticks for clarity. #
	
	{ Example Problem}
	{ Example Code}
	
	{ Problem}

Self-Correction (i) Feedback	Review your previous answer and find problems with your answer. {Problem} {Code}
Self-Correction (ii) Correction	Based on the problems you found, improve your code. Please reiterate your code, with your final answer a single numerical number, in the form \\boxed{answer}. Encase the Python code within triple backticks for clarity. {Code} {Feedback}

Self-Refine	{Example Code}
	# There is an error in the code above because of lack of understanding of the question. What is the error? To find the error, go through semantically complete blocks of the code, and check if everything looks good.
	{Example Error}
	# Okay! Here is the rewrite:
	{Example Refined code}
	
	{Code}
	# There is an error in the code above because of lack of understanding of the question. What is the error? To find the error, go through semantically complete blocks of the code, and check if everything looks good.

Table 11: Prompt templates for baselines.

Problem:
A particle of mass m and velocity u_1 makes a head-on collision with another particle of mass 2m at rest. If the coefficient of restitution is such to make the loss of total kinetic energy a maximum, what are the velocities v_1 after the collision? The unit of the answer is u_1.

Problem Analysis (Previous):

```
# Given parameters
m = Symbol('m') # mass of the first particle
u1 = Symbol('u1') # velocity of the first particle
m2 = 2 * m # mass of the second particle
u2 = 0 # velocity of the second particle at rest
```

Review:
The variables are defined using symbolic math (Symbol) from the sympy library, which is not necessary for this problem. It is better to define the variables as numerical values. Also, the variable names can be more descriptive.

Problem Analysis (Refined):

```
# Given parameters
mass_particle1 = 1 # mass of the first particle
velocity_particle1 = 1 # velocity of the first particle
mass_particle2 = 2 * mass_particle1 # mass of the second particle
velocity_particle2 = 0 # velocity of the second particle at rest
```

Formula Retrieval:

```
# Potential formulas
# v1' = (m1 - m2) * v1 / (m1 + m2) + 2 * m2 * v2/ (m1 + m2), calculate velocity of
m1 in 1d elastic collision of two particles
# v2' = 2 * m1 * v1 / (m1 + m2) + (m2 - m1) * v2 / (m1 + m2), calculate velocity of
m2 in 1d elastic collision of two particles
# v = (m1 * v1 + m2 * v2) / (m1 + m2), calculate velocity of One-Dimensional
Completely Inelastic Collision
```

Guided Reasoning (Previous):

```
# Calculate velocities after collision
v1_after = ((mass_particle1 - mass_particle2) * velocity_particle1 + 2 *
mass_particle2 * velocity_particle2) / (mass_particle1 + mass_particle2)
v2_after = (2 * mass_particle1 * velocity_particle1 + (mass_particle2 -
mass_particle1) * velocity_particle2) / (mass_particle1 + mass_particle2)

print(v1_after)
```

Review:
The code has 1 error. The formula used in the code is for an elastic collision, but the problem states that the coefficient of restitution is such that the loss of total kinetic energy is a maximum, which implies an inelastic collision. The formula for an inelastic collision is different from that of an elastic collision.

Guided Reasoning (Refined):

```
# Calculate velocities after inelastic collision
v1_after = (mass_particle1 * velocity_particle1 + mass_particle2 *
velocity_particle2) / (mass_particle1 + mass_particle2)
v2_after = v1_after

print(v1_after)
```

Figure 10: Case study using Physics Reasoner method.