Improving Physics Reasoning in Large Language Models Using Mixture of Refinement Agents

Raj Jaiswal[*1], **Dhruv Jain**[*2], **Harsh Parimal Popat**[1], **Abhishek Dharmadhikari**[1], **Atharva Marathe**[1], **Avinash Anand**[1], **Shin'ichi Satoh**[3], **Rajiv Ratn Shah**[1]

[1]MIDAS Lab, IIIT Delhi
[2]Indian Institute of Technology (BHU) Varanasi
[3]National Institute of Informatics
{jaiswalp, harsh21048, avinasha, rajivratn}@iiitd.ac.in,
dhruv.jain.ece21@itbhu.ac.in, satoh@nii.ac.jp
{abhishekdharmadhikari25, atharvamarathe8}@gmail.com

Abstract

Large Language Models (LLMs) demonstrate remarkable capabilities in various reasoning tasks. However, they encounter significant challenges when it comes to scientific reasoning, particularly in physics, which requires not only mathematical reasoning but also factual and conceptual understanding. When addressing complex physics problems, LLMs typically face three key issues: problem miscomprehension, incorrect concept application, and computational errors. While each of these problems can be addressed individually, there is a need for a generalized approach that can tackle all three issues simultaneously. To address this, we introduce Mixture of Refinement Agents (MoRA), a novel agentic refinement framework that iteratively refines the LLM generated base solution by correcting the aforementioned errors, resulting in a significant performance improvement for open-source LLMs. Our approach aims to bridge the gap between open-source LLMs and GPT-4o by utilizing the latter as error identifier to guide these refinement agents. We evaluate our approach on the SciEval and MMLU subsets along with our own physics dataset (PhysicsQA). MoRA significantly improves the performance of Llama-3-70B and Gemma-2-27B on these datasets, achieving up to a 16% increase in final answer accuracy.

Code — https://github.com/maximus-21/MoRA

PhysicsQA —
https://huggingface.co/datasets/maximus-21/PhysicsQA

Introduction

Scientific reasoning, particularly in field of physics, requires a deep understanding that spans multiple disciplines. It demands not only domain-specific knowledge but also the integration of mathematical reasoning with theoretical concepts, applying abstract principles and formulae across various contexts and scenario. Successfully solving these challenges is a fundamental aspect of human intelligence, as it entails not just recalling information but adapting knowledge to solve diverse complex problems.

Solving complex physics problems still remains a challenge for open source LLMs. The difficulty stems from the need to integrate both mathematical and domain-specific knowledge while engaging in multi-hop, step-by-step reasoning. One approach to address this challenge can be collecting question and solution trajectory annotations and finetune LLMs to enhance these capabilities, similar to recent mathematical reasoning works (Luo et al. 2023; Yuan et al. 2024). However, the process of such annotations and finetuning is time-consuming and costly. On the other hand, solutions generated by LLMs for physics problems using CoT prompting (Wei et al. 2022) often contain errors, such as objective misalignment, incorrect formula application, and computational mistakes, as illustrated in Figure 1. Moreover, solutions to multihop physics problems contain multiple such errors together.

Open source LLMs struggles to accurately directly identify reasoning mistakes in their own solutions (Li et al. 2024; Tyen et al. 2024), making them unreliable for error detection and self-refinement. While objective alignment errors can be corrected once identified, refining computational and conceptual errors requires strong mathematical reasoning and contextual understanding of the specific question. Addressing all these different errors simultaneously remains a significant challenge for open-source LLMs.

This motivated us to develop the Mixture of Refinement Agents (MoRA) framework. MoRA iteratively refines LLM responses through a two-step process in each iteration. First, the framework leverages a advanced model to identify various errors within the solution using appropriate flags and scores. In the next step, based on the identified errors, prioritized agent routing is conducted, in which the appropriate agents are activated to address and mitigate the specific errors. This process results in a progressively refined solution.

In the domain of physics, evaluation benchmarks are essential for assessing the conceptual and mathematical reasoning of LLMs. Benchmarks like MMLU, SciEval (Sun et al. 2024), and ScienceQA (Lu et al. 2022) focus on

[*]These authors contributed equally.

foundational knowledge and general reasoning, while more challenging ones like OlympiadBench (He et al. 2024) and JEEBench (Arora, Singh et al. 2023) require advanced reasoning skills. To bridge the gap, we curated our own dataset PhysicsQA, containing set of diverse, intermediate-level high school physics problems that provide a balanced challenge, allowing a exhaustive evaluation and analysis of open-source LLMs on physics problems.

We perform exhaustive experimentation of MoRA across four datasets including PhysicsQA as shown in Table 3. MoRA improves accuracy on the PhysicsQA benchmark over CoT-generated solutions by 13.38% for Llama-3-70B and by 16.03% for Gemma-2-27B. This significant enhancement highlights MoRA's effectiveness in refining solutions, particularly in complex and diverse physics problems as in PhysicsQA. Our further analysis offers insights into the error distribution across different models and evaluates the effectiveness of individual refinement agents based on their refinement rates.

Related Works

LLM Reasoning LLMs have been successfully applied to address multi-step reasoning tasks by generating intermediate reasoning steps, referred to as Chain-of-Thought (CoT) (Wei et al. 2022), Auto-CoT (Zhang et al. 2022), and Complex-CoT (Fu et al. 2022), among others. Advanced techniques like Iter-CoT (Sun et al. 2023a) and ToT (Yao et al. 2024) extend these capabilities but remain constrained by the knowledge in training data and the specific structures they were designed with. While In-Context Learning (ICL) (Brown et al. 2020) has significantly improved LLM performance, challenges like hallucinations and limitations in reasoning flexibility persist.

LLMs for Scientific Reasoning LLMs face significant limitations in complex knowledge reasoning tasks (Petroni et al. 2020). (Ouyang et al. 2023) introduced a structured reasoning strategy to guide LLMs in solving complex chemistry problems, enabling them to generate high-quality reasoning. Solving these problems requires not only domain knowledge, like formulae and calculations, but also a step-by-step reasoning process. (Ma et al. 2024) proposed a method where agents generate a high-level plan based on the question, retrieve relevant functions from a toolset, and execute low-level actions by integrating natural language and Python code.

Self Verification with LLMs Recent works (Cobbe et al. 2021; Ling et al. 2024) have attempted to address the challenge of error detection in step-by-step reasoning. However, these methods often require additional training data or domain-specific exemplars, making them less practical. (Miao, Teh, and Rainforth 2023) proposes using the LLM itself to verify the conditional correctness of each step in the reasoning chain, similar to how a human reviews their work. Accurate error recognition and correction are crucial for enhancing problem-solving capabilities, as demonstrated by (Li et al. 2024), which defines tasks to assess LLMs' mathematical reasoning abilities in error identification and correction.

LLMs for Mathematical Reasoning LLMs tends to struggle with arithmetic calculations when solving math problems (Cobbe et al. 2021; Gao et al. 2023). However, incorporating code generation and execution has shown promise in enhancing the accuracy of mathematical reasoning. Leveraging these strengths, the GPT-4 Code Interpreter (Zhou et al. 2023) has been integral to frameworks like MathCoder (Wang et al. 2023), which is designed to improve the mathematical reasoning capabilities of open-source models. Findings from (Zhou et al. 2023) indicate that GPT-4 Code's impressive proficiency in solving mathematical problems is largely due to its step-by-step code generation and the dynamic refinement of solutions based on code execution outcomes.

LLM Reasoning with external database (Lewis et al. 2020) proposed RAG framework, which incorporates a retrieval component to fetch relevant information from a given knowledge base. Integrating LLMs with knowledge representation tools, such as knowledge graphs (KGs) (Mruthyunjaya et al. 2023), has further enhanced reasoning capabilities. (Yao et al. 2024) demonstrated that augmenting LLMs with comprehensive external knowledge from KGs can significantly improve their performance and facilitate more robust reasoning processes. A notable example is GraphRAG (Edge et al. 2024), a retrieval enhancement technique that leverages knowledge graphs to map relationships between entities, thereby enhancing the retrieval process using large language models (LLMs).

Dataset: PhysicsQA

Our dataset comprises 370 carefully selected high school physics questions sourced from online resources. These questions are notably complex, often requiring the application of multiple concepts, intricate computations, and multi-hop reasoning. Each question is paired with a comprehensive, step-by-step solution, to support the evaluation and fine-tuning of LLMs for physics reasoning. Table 1 illustrates the topic-wise distribution of the questions, providing a clear overview of the areas covered. PhysicsQA offers a more robust evaluation and analysis of LLM performance by encompassing a diverse range of questions, both in terms of complexity and the topics covered.

Chapter Name	Percentage
Electromagnetism	29.8%
Mechanics and Kinematics	21.8%
Thermodynamics and Heat	15.7%
Waves and Optics	15.4%
Nuclear and Modern Physics	8.9%
Material Properties and Elasticity	8.3%

Table 1: Topic-wise Distribution in PhysicsQA

Mixture of Refinement Agents

This section introduces our mixture of refinement agents (MoRA) framework. We first discuss our motivation behind

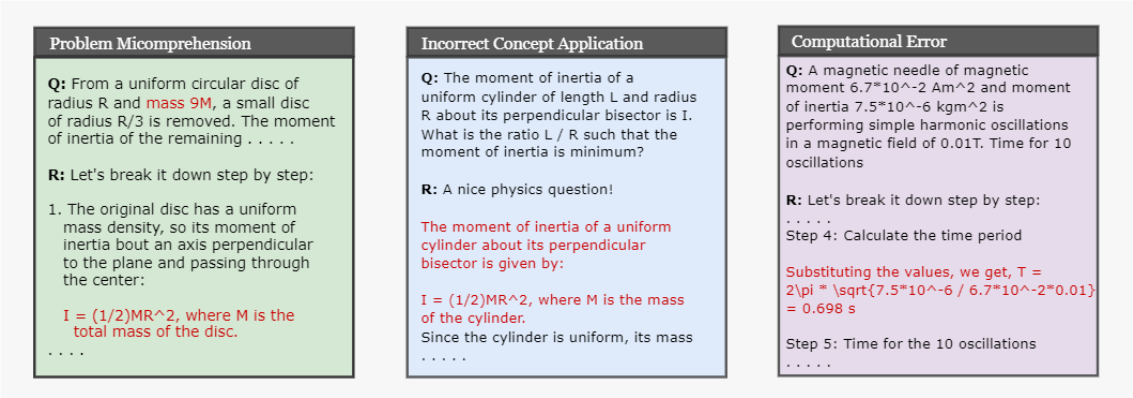

Figure 1: The illustration of three key error observations in the CoT solution of open source LLMs for physics problems. (a) showcases problem miscomprehension, where the LLM response uses the incorrect value of variables given in the question here, M instead of 9M, (b) showcases incorrect concept application in the LLM response, here incorrect moment of inertia formula for uniform cylinder, (c) demonstrate computational error within LLM response here, incorrect calculation of time period.

MoRA; then, we introduce the error identification stage and refinement agents. Finally, we discuss how these agents are routed iteratively to correct different errors in the solutions generated by the LLM.

Motivation

While analyzing physics problems and their CoT solutions generated with LLMs (Llama-3-70B & Gemma-2-27B), we observed three key errors made by them:

Observation 1: *LLMs in few cases struggle to fully grasp the objective of the question, along with misinterpreting the values of variables and constants provided in the question.*

Although this issue has been identified in only a few cases, it is significant one because it leads to solutions that fails to address the correct interpretation of a given question resulting in *problem miscomprehension*.

Observation 2: *LLMs struggle to apply the correct concepts or formulae with respect to the context of the given problem.*

This issue is a more recurring one in LLMs, especially for problems requiring considering a specific case rather than relying on a generic formula. For instance, the formula for calculating the *moment of inertia* varies depending on the distribution of mass.

Observation 3: *Many physics problems involve mathematical reasoning and algebraic computation, areas where LLMs tend to struggle.*

Computational errors account for the majority of errors in solutions generated by LLMs. LLMs struggles with accurate algebraic and arithmetic computations resulting in errors within the reasoning and final answer.

While these three issues can be addressed individually, solutions often exhibit multiple errors together. Therefore, a single framework is required to rectify all three issues effectively, which motivated us to develop the MoRA. We first perform error identification on a given solution; then these errors are mitigated iteratively using specialized refinement agents, resulting in accurate solutions.

Error Identification

The errors in the solutions are classified into three categories: 1) *problem miscomprehension*, 2) *incorrect concept application*, and 3) *computational errors* as showed in Figure 1.

For error identification, we choose to rely on GPT-4o. Our experiments and analysis shows that GPT-4o showcases superior performance compared to other models, particularly in problem comprehension and correct physics concept application required. Thus, it is adequate for locating errors within solutions generated by other models. Given a question and it's LLM response, we prompt GPT-4o to identify and locate different errors in the solution using the combination of following flags and scores:

Problem Comprehension Flags: We prompt GPT-4o to check for the problem miscomprehension using the following two flags: (i) **Objective Alignment Flag**, F_{obj}, verifies whether the solution is focused on solving the correct objective of the given question. (ii) **Variables Application Flag**, F_{val}, verifies whether the solution uses the correct values for all variables and constants provided in the question, ensuring their correct values are applied in formulae and reasoning.

Concept Verification Score: We instruct GPT-4o to check the given solution against the relevant concept and formulae required to solve the given problem, based on its own understanding of the question. A score ($Score_{\text{concept}}$) is assigned to each solution to quantify the correctness of the applied physics concepts and formulae. The score is designed to identify the stage at which any conceptual or formulae error first occurs, if at all. $Score_{\text{concept}}$ ranges from 0 to 1, where a lower score indicates an earlier-stage error and a higher score indicates a later-stage error in the solution

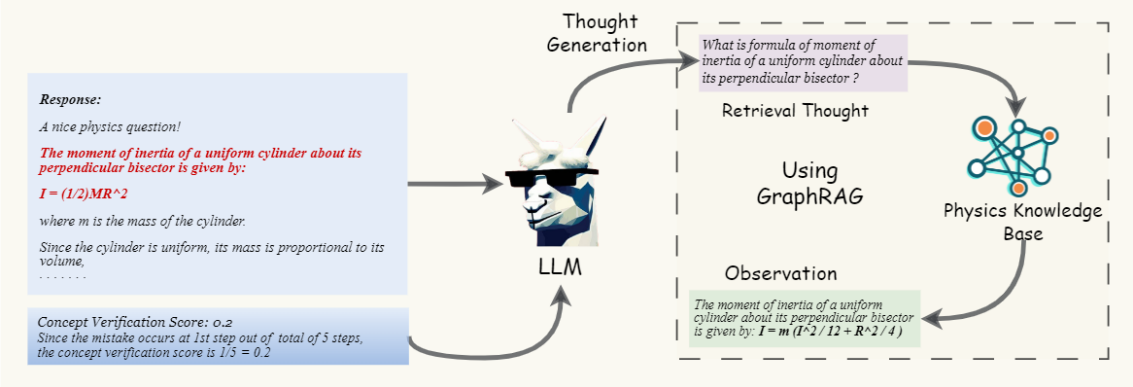

Figure 2: The illustration of thought generation and concept retrieval for conceptual error refinement in LLM response. Given the response and concept verification score, LLM generates a retrieval thought, which acts as a query to retrieve the correct conceptual context from an physics knowledge base using GraphRAG.

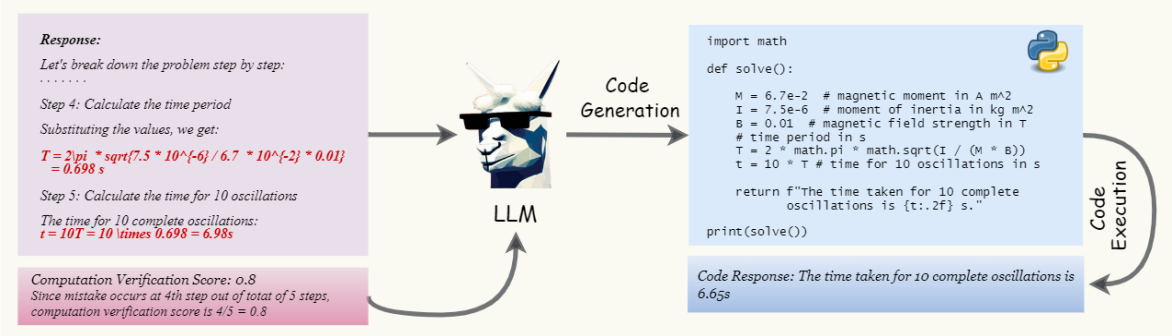

Figure 3: The illustration of code generation and execution for computation error refinement in LLM response. Given the response and computation verification score, LLM generates a code to perform the correct required computation; the code is then executed to obtain the response.

process. The score is calculated as follows:

$$Score_{\text{concept}} = \begin{cases} \frac{n}{N} & \text{if } 1 \leq n < N \text{ (error at step } n) \\ \frac{n}{N+1} & \text{if } n = N \text{ (error at last step)} \\ 1 & \text{if no errors occur} \end{cases}$$

where n is the step at which the first error occurs, and N is the total number of steps in the solution process.

Computation Verification Score: We employ GPT-4o with OpenAI Code Interpreter for generation and execution of python code to evaluate the correctness of all arithmetic and algebraic operations in the given solution. Similar to the $Score_{\text{concept}}$, we assign $Score_{\text{comp}}$ to each solution. This score quantifies the accuracy of the mathematical computation performed, ranges from 0 to 1, and is calculated similar to $Score_{\text{concept}}$. All the computations are evaluated with an error tolerance of 0.1. Using Code Interpreter enables us to leverage the code generation capabilities of GPT-4o rather than solely relying on its mathematical reasoning, which sometimes can lead to erroneous scores. Recent works, such as (Zhou et al. 2023) and (Wang et al. 2023), highlights the remarkable capability of OpenAI Code Interpreter in solving challenging math problems and self-verification.

We utilize GPT-4o solely for error identification, which guides the routing to the appropriate refinement agent. The scores are used as the feedback to help refinement agents understand the first stage of the mistake from where the refinement needs to be initiated.

Refinement Agents

To address the three key errors in LLM generated solutions, we introduce a set of specialized refinement agents. Each agent is designed to rectify a specific type of error within the solution, ensuring targeted and effective corrections. The refinement agents use the same LLM with which the original solutions are generated.

Miscomprehension Refinement Although there are very few cases of problem miscomprehension in LLM generated solutions, once identified these mistakes can easily be corrected with simple instruction prompting:

You are tasked with solving a physics problem. Here is the question: [question], The following is your generated solution: [solution], In the generated solution, the correct objective of the question is not being addressed. The solutions contains mistakes which leads to misalignment with the objective of the question. Please carefully review the question

& understand the objective in detail and regenerate the solution accordingly.

The above prompt assists in refining the solution to align with the correct objective of the given question. This may involve regenerating the entire solution or correcting an intermediate mistake to ensure the solution addressed the correct objective. Similarly, any incorrect variable values used within the solution is corrected using instruction prompting.

Concept Refinement To address the incorrect concepts and formulae applied in the LLM's solutions, we utilize an external physics knowledge base. This is necessary because LLMs may not always have access to or accurately retrieve the correct formulae, as this information may not be embedded in their internal knowledge. The conceptual refinement occurs in two steps:

1. **Error Identification & Thought Generation:** Given a solution S_{orig} and a concept score $Score_{\text{concept}}$, the LLM systematically reviews the solution to identify the earliest stage where an incorrect concept or formula has been applied. $Score_{\text{concept}}$ pinpoints this stage of error within the solution. LLM then generates a retrieval thought T_{R} for the the concept or formula required at the failure stage. The thought is structured to be simple and sequential query.

2. **Concept Retrieval & Solution Refinement:** Given the retrieval thought T_{R} and physics knowledge base K_{P}, we use GraphRAG (Edge et al., 2024) to query the K_{P} to retrieve an observation O_{T}, based on T_{R} as demonstrated in Figure 2. O_{T} contains the correct context for concept and formulae required at the stage of failure. The LLM then initiates the refinement from this stage using the information present in O_{T}, resulting in the refined solution S_{refined} with corrected physics concepts and reasoning.

Computational Refinement Inspired by recent works such as PAL (Gao et al. 2023), PoT (Chen et al. 2022), CSV (Zhou et al. 2023), MathCoder (Wang et al. 2023), we use code generation for the refinement of computational and mathematical errors within a solution. The computation score $Score_{\text{comp}}$ allows the LLM to locate the first step of error and then initiate the refinement of the failure stage and subsequent computations. The process occurs in two steps:

1. **Code Generation & Execution:** Given the original solution S_{orig} and computation score $Score_{\text{comp}}$, the LLM first locates the error step and then generates a Python code C_{p} designed to accurately perform the necessary computation at the identified failure stage and produce the correct result. The generated code C_{p} is then executed to obtain the response R_{c} as shown in Figure 3.

2. **Solution Refinement:** The LLM is then instructed to refine S_{orig} using the correct response R_{c} generated by the code. This involves pinpointing the exact step where the error occurred, guided by the computation score $Score_{\text{comp}}$, and integrating the correct computation from R_{c} into the solution. The refined solution S_{refined} is then presented with the corrected computations.

Agent Routing and Iterative Refinement

After the error identification, the respective refinement agents are activated to mitigate these errors. The agent routing follows a prioritized sequence: 1.) miscomprehension refinement, 2.) concept refinement, 3.) computational refinement. This prioritization mirrors the human approach to solving physics problems: first, understanding the objectives and variables; next, identifying relevant concepts and formulae; and finally, applying them to perform the necessary computations.

The activated refinement agent then acts upon the solution to mitigate the error. The solution undergoes iterative cycles of error identification and refinement until all flags and scores are resolved or a maximum iteration limit is reached. This process ensures that all errors are corrected without introducing new ones in final refined solution. The complete process is illustrated in Algorithm 1.

Algorithm 1: Error Identification and Iterative Refinement

Require: Question Q, Initial Solution S_0, GPT-4o $\mathcal{L}$, Refinement Agents $\mathcal{R}$, Maximum Iterations N, Threshold ϵ
Ensure: Final refined solution to Q

$i = 0$, $S_i = S_0$
while $i < N$ **do**
  $(F^i_{\text{obj}}, F^i_{\text{val}}, Score^i_{\text{concept}}, Score^i_{\text{comp}}) \leftarrow \mathcal{L}(Q, S_i)$
  if $F^i_{\text{obj}} == -1$ **or** $F^i_{\text{val}} == -1$ **then**
    $S_{i+1} \leftarrow \mathcal{R}_{\text{miscomprehension}}(Q, S_i)$
  else if $Score^i_{\text{concept}} < 1 - \epsilon$ **then**
    $S_{i+1} \leftarrow \mathcal{R}_{\text{concept}}(Q, S_i)$
  else if $Score^i_{\text{comp}} < 1 - \epsilon$ **then**
    $S_{i+1} \leftarrow \mathcal{R}_{\text{computation}}(Q, S_i)$
  else
    return S_i
  end if
  $i \leftarrow i + 1$
end while
return S_N

Experiments

Setup

Datasets In our experiments, we use four datasets: SciEval-Static, PhysicsQA, MMLU High School and MMLU College. SciEval-Static is a subset of SciEVal (Sun et al. 2023b), consisting 164 questions from physics divided into multiple sub-topics. MMLU (Hendrycks et al. 2021), consists of a 118 College level and 173 high school multiple-choice questions from various disciplines.

LLMs We utilize the API of a range of models with varying parameters and capabilities including LLaMa-3-70B, LLaMa 3.1-405B, Gemma-2-27B, Gemini-1.5-Flash, GPT-3.5 Turbo and GPT-4 as our LLMs for the evaluation. We use same prompts for all the datasets and LLMs during our evaluation.

Baselines We employ an Answer-only approach (AO), where the model is given a question with four options and asked to select the correct answer without any explanation relying solely on its pre-existing knowledge . In contrast, few-shot prompting (Xu et al. 2023; Yasunaga et al. 2023) uses a few examples to help the model learn and apply that knowledge to similar tasks. Chain-of-Thought (CoT)

Model	SciEval-Static			PhysicsQA			MMLU - High			MMLU - College		
	AO	CoT	3-Shot	AO	CoT	3-Shot	AO	CoT	3-Shot	AO	CoT	3-Shot
LLaMa-3-70B	70.07%	82.23%	63.41%	38.37%	56.76%	59.29%	60.16%	72.88%	73.66%	59.41%	71.76%	71.76%
LLaMa 3.1 405B	**79.87%**	89.63%	**82.92%**	**50.81%**	76.75%	**78.37%**	**72%**	91.52%	**88.98%**	**75.29%**	**88.23%**	**85.29%**
Gemma-2-27B	60.36%	79.26%	53.04%	39.18%	54.59%	59.45%	55.93%	77.11%	74.45%	51.11%	73.52%	67.64%
Gemini 1.5 Flash	68.29%	85.97%	81.70%	44.86%	62.97%	69.72%	58.47%	79.66%	80.05%	60.58%	72.35%	72.94%
GPT 3.5 Turbo	41.46%	66.46%	48.78%	28.10%	42.70%	42.70%	47.45%	58.47%	33.89%	35.29%	50.58%	42.35%
GPT4o	64.02%	**92.68%**	81.09%	49.45%	**79.45%**	**78.37%**	62.71%	**94.06%**	87.28%	70%	84.70%	84.17%

Table 2: Experimentation of Answer-Only (AO) , COT and Few-Shot (3-shot) on different Datasets

Model	Dataset	AO	COT	3-Shot	MORA
Gemma 2 27B	MMLU College	51.11%	73.52%	67.64%	**82.20%**
	MMLU High School	55.93%	77.11%	74.45%	**75.88%**
	PhysicsQA	39.18%	54.59%	59.45%	**70.62%**
	SciEval-Static	60.36%	79.26%	53.04%	**88.76%**
LLaMa 3 70B	MMLU College	59.41%	71.76%	71.76%	**78.82%**
	MMLU High School	60.16%	72.88%	73.66%	**78.81%**
	PhysicsQA	38.37%	56.76%	59.29%	**70.14%**
	SciEval-Static	70.07%	82.23%	63.41%	**86.58%**

Table 3: Comparison of baseline approaches with MoRA across four datasets: SciEval-Static, PhysicsQA, MMLU High School and College based on final answer accuracy.

prompting (Wei et al. 2022) guides the model to generate intermediate reasoning steps, improving its performance on complex tasks by breaking them down into smaller, more manageable parts. These three approaches form our primary baselines.

Evaluation Most of the existing works (Luo et al. 2023) , (Chern et al. 2023) , (Yu et al. 2023) measure the mathematical reasoning quality of LLMs by directly comparing the final answer and calculating the overall accuracy on a given dataset. We choose to follow the same evaluation for physics reasoning as well.

Results

In Table 2 we present results from our experiments reveal compelling insights into the strengths and challenges of various models across diverse benchmarks. In the SciEval-Static benchmark, LLaMa-3-70B and Gemini 1.5 Flash stand out, with LLaMa-3-70B achieving an accuracy of 82.23% using (CoT), and Gemini 1.5 Flash not far behind at 85.97%. In the PhysicsQA domain, which demands intricate reasoning skills, the models face more significant challenges. LLaMa-3-70B and Gemma 2-27B both show improved performance with CoT, reaching 56.76% and 54.59%, respectively. On the MMLU-High benchmark, LLaMa-3-70B continues to perform solidly, achieving 72.88% with CoT, while Gemini 1.5 Flash pushes ahead to 79.66%. Interestingly, in MMLU-College, a benchmark with a mix of academic and reasoning tasks, Gemma 2-27B shows a significant leap in performance with CoT, reaching 73.52%, which surpasses its base score by over 22%, indicating the effectiveness of CoT in enhancing reasoning in academic settings.

As shown in Table 3, MoRA framework delivers marked improvements across all benchmarks for both LLaMA-3-70B and Gemma-2-27B models. In SciEval-Static, the introduction of MoRA enhances LLaMA-3-70B's accuracy from 82.23% (CoT) to 86.58%, and Gemma-2-27B sees a boost from 79.26% (CoT) to 88.76%. In the PhysicsQA dataset, MoRA significantly improves LLaMA-3-70B's performance from 59.29% (3-shot) to 70.14%, and Gemma-2-27B's from 59.45% (3-shot)to 70.62%. In MMLU High School, LLaMA-3-70B accuracy reaches from 73.66% (3-Shot) to 78.81% and in Gemma-2-27B, accuracy reaches from 77.11% (CoT) to 75.88%. In MMLU College, LLaMA-3-70B accuracy reaches from 71.76% (3-Shot) to 78.82% and in Gemma-2-27B, accuracy reaches from 73.52% (CoT) to 82.20%. The results show that even without extensive fine-tuning, these models can achieve competitive performance. These improvements demonstrate MoRA's ability to elevate smaller models to compete effectively with much larger ones across a range of complex tasks.

Analysis

In this section, we first conduct an in-depth error analysis of physics CoT solutions across various models and datasets, highlighting the error distribution that inspired the development of MoRA. We then present ablation studies, analyzing the effectiveness of each refinement agent in our framework.

Error Analysis

We perform manual analysis of the incorrect CoT solutions of GPT-4o, Llama-3-70B, and Gemma-2-27B on the following datasets: SciEval-Static, PhysicsQA, MMLU High School and College as shown in Table 4. Based on this analysis here are our observations:

(i) **LLMs demonstrate good problem comprehension ability for physics question.** All models demonstrate strong problem comprehension across the datasets. GPT-4o excels, achieving near-perfect accuracy on SciEval-Static, MMLU College, and High School, with only minor errors in PhysicsQA. This suggests a deep understanding of physics problem structure. Llama-3-70B and Gemma-2-27B also perform well but show slightly higher error rates, particularly in PhysicsQA and SciEval-Static, indicating occasional missed details that need attention.

(ii) **Open source LLMs sometimes struggles to retrieve correct physics concept and formulae while reasoning.** On average, 18.11% of questions in the PhysicsQA dataset are answered with conceptual errors by Gemma-2-27B and Llama-3-70B, highlighting the difficulty open-source LLMs face in applying correct concepts to physics problems. In contrast, GPT-4o excels with an average accuracy of 96.4% across all four datasets. Notably, Gemma-2-27B outperforms Llama-3-70B on SciEval-Static, PhysicsQA, and MMLU High School. The high error rates of both Llama-3-70B and Gemma-2-27B on PhysicsQA suggest that medium-parameter open-source LLMs may still need external knowledge bases for complex physics problem-solving.

(iii) **Open-source LLMs struggles with algebraic and arithmetic computation required while solving physics questions.** On average, 21.62% of questions in the PhysicsQA dataset are answered with computational mistakes by Gemma-2-27B and Llama-3-70B, highlighting challenges in executing correct calculations. GPT-4o excels with an average accuracy of 95.64% across four datasets. Gemma-2-27B outperforms Llama-3-70B on SciEval Static and MMLU (High School and College), with both performing similarly on PhysicsQA. The accuracy gap between GPT-4o and Llama-3-70B (12.16%) and GPT-4o and Gemma-2-27B (13.24%) on PhysicsQA suggests that open-source LLMs could benefit from further refinement in handling complex calculations.

Error Type	Dataset	GPT-4o	Gemma 2-27B	LLaMa 3-70B
Computational Error	MMLU College	2.54%	5.08%	9.32%
	MMLU High School	2.35%	3.53%	6.47%
	PhysicsQA	8.92%	22.16%	21.08%
	SciEval-Static	3.06%	10.37%	10.98%
Problem Miscomprehension	MMLU College	0.00%	0.85%	1.69%
	MMLU High School	0.59%	0.59%	1.18%
	PhysicsQA	0.54%	2.16%	1.62%
	SciEval-Static	0.00%	1.22%	1.83%
Wrong Concept	MMLU College	0.85%	12.71%	10.17%
	MMLU High School	3.53%	8.24%	11.76%
	PhysicsQA	7.57%	17.11%	18.92%
	SciEval-Static	1.22%	7.36%	9.15%

Table 4: Error Analysis of incorrect physics CoT solutions of different models across four datasets.

Ablation

To understand the effectiveness of each refinement agent, we conduct ablation of each of the refinement agents with Llama-3-70B and Gemma-2-70B in terms of their refinement rate across different datasets as shown in Table 5. Here are our observations:

(i) **Problem miscomprhension errors are mitigated with simple instruction prompting and error feedback.** Llama-3-70B and Gemma-2-27B demonstrate good miscomprehension error refinement with instruction prompting,

Error Type	Dataset	Gemma 2-27B	LLaMa 3-70B
Computational Refinement	MMLU College	100%	81.8%
	MMLU High School	33.3%	75.0%
	PhysicsQA	73.3%	72.6%
	SciEval-Static	57.1%	60.0%
Miscomprehension Refinement	MMLU College	37.5%	33.3%
	MMLU High School	16.7%	37.5%
	PhysicsQA	48.7%	46.9%
	SciEval-Static	62.5%	57.1%
Concept Refinement	MMLU College	100%	100%
	MMLU High School	100%	100%
	PhysicsQA	62.5%	66.7%
	SciEval-Static	100%	66.7%

Table 5: Ablation studies for different refinement agent in MoRA using Gemma-2-27B and Llama-3-70B across four datasets, evaluated by refinement rate.

particularly in the MMLU datasets (High School and College), where both models achieve a perfect 100% refinement rate. Llama-3-70B outperforms Gemma-2-27B slightly on PhysicsQA, with a refinement rate of 66.7% compared to Gemma-2-27B's 62.5%. However, Gemma-2-27B excels in SciEval Static and MMLU (College and High School). The decent accuracy on PhysicsQA suggests that open-source LLMs sometimes fail to rectify their misinterpretations in complex physics problems.

(ii) **Open-source LLM performers moderately in identifying the conceptual mistake and retrieval thought generation.** Llama-3-70B shows a 46.9% refinement rate in PhysicsQA and slightly improves in SciEval Static at 57.1%, but struggles with MMLU datasets, achieving 37.5% and 33.3% refinement in High School and College, respectively. Gemma-2-27B has a similar 48.7% refinement rate in PhysicsQA and performs better in SciEval Static at 62.5%, but underperforms significantly on MMLU High School with a 16.7% refinement rate, improving modestly to 37.5% on MMLU College. These results suggest that open-source LLMs have difficulty generating relevant retrieval thoughts at the initial stage of failure.

(iii) **Using code-driven refinement significantly corrects the computational errors.** Llama-3-70B and Gemma-2-27B excel in refining computational errors, demonstrating the effectiveness of code generation and execution. Llama-3-70B shows consistent performance with a 72.6% refinement rate on PhysicsQA and strong results across SciEval Static (60%), MMLU High School (75%), and MMLU College (81.8%). Gemma-2-27B slightly outperforms in PhysicsQA at 73.3% and achieves a 100% refinement rate in MMLU College. However, Gemma-2-27B's performance is more variable, particularly in MMLU High School (33.3%), indicating potential challenges in specific code generation scenarios.

Conclusion

In this work, we introduce MoRA, a novel agentic refinement framework designed to mitigate three critical errors commonly made by LLMs when solving complex physics problems. MoRA first leverages GPT-4o for error identification and score assignment, which are then subsequently used to guide the refinement agents. This process is done iteratively until all the errors in the solution are mitigated successfully. To ensure a comprehensive evaluation, we also curate our own dataset, PhysicsQA, which includes a diverse set of high school-level physics problems. Our experiments and in-depth analysis across multiple datasets demonstrate that MoRA significantly enhances the performance of Llama-3-70B and Gemma-2-27B across multiple datasets.

References

Arora, D.; Singh, H. G.; et al. 2023. Have llms advanced enough? a challenging problem solving benchmark for large language models. *arXiv preprint arXiv:2305.15074*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, W.; Ma, X.; Wang, X.; and Cohen, W. W. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Chern, E.; Zou, H.; Li, X.; Hu, J.; Feng, K.; Li, J.; and Liu, P. 2023. Generative ai for math: Abel. *URL https://github.com/GAIR-NLP/abel*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; and Larson, J. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Fu, Y.; Peng, H.; Sabharwal, A.; Clark, P.; and Khot, T. 2022. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; and Neubig, G. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, 10764–10799. PMLR.

He, C.; Luo, R.; Bai, Y.; Hu, S.; Thai, Z. L.; Shen, J.; Hu, J.; Han, X.; Huang, Y.; Zhang, Y.; et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, X.; Wang, W.; Li, M.; Guo, J.; Zhang, Y.; and Feng, F. 2024. Evaluating Mathematical Reasoning of Large Language Models: A Focus on Error Identification and Correction. *arXiv preprint arXiv:2406.00755*.

Ling, Z.; Fang, Y.; Li, X.; Huang, Z.; Lee, M.; Memisevic, R.; and Su, H. 2024. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36.

Lu, P.; Mishra, S.; Xia, T.; Qiu, L.; Chang, K.-W.; Zhu, S.-C.; Tafjord, O.; Clark, P.; and Kalyan, A. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35: 2507–2521.

Luo, H.; Sun, Q.; Xu, C.; Zhao, P.; Lou, J.; Tao, C.; Geng, X.; Lin, Q.; Chen, S.; and Zhang, D. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ma, Y.; Gou, Z.; Hao, J.; Xu, R.; Wang, S.; Pan, L.; Yang, Y.; Cao, Y.; and Sun, A. 2024. SciAgent: Tool-augmented Language Models for Scientific Reasoning. *arXiv preprint arXiv:2402.11451*.

Miao, N.; Teh, Y. W.; and Rainforth, T. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.

Mruthyunjaya, V.; Pezeshkpour, P.; Hruschka, E.; and Bhutani, N. 2023. Rethinking language models as symbolic knowledge graphs. *arXiv preprint arXiv:2308.13676*.

Ouyang, S.; Zhang, Z.; Yan, B.; Liu, X.; Han, J.; and Qin, L. 2023. Structured chemistry reasoning with large language models. *arXiv preprint arXiv:2311.09656*.

Petroni, F.; Piktus, A.; Fan, A.; Lewis, P.; Yazdani, M.; De Cao, N.; Thorne, J.; Jernite, Y.; Karpukhin, V.; Maillard, J.; et al. 2020. KILT: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

Sun, J.; Luo, Y.; Gong, Y.; Lin, C.; Shen, Y.; Guo, J.; and Duan, N. 2023a. Enhancing chain-of-thoughts prompting with iterative bootstrapping in large language models. *arXiv preprint arXiv:2304.11657*.

Sun, L.; Han, Y.; Zhao, Z.; Ma, D.; Shen, Z.; Chen, B.; Chen, L.; and Yu, K. 2023b. SciEval: A Multi-Level Large Language Model Evaluation Benchmark for Scientific Research. *arXiv preprint arXiv:2308.13149*.

Sun, L.; Han, Y.; Zhao, Z.; Ma, D.; Shen, Z.; Chen, B.; Chen, L.; and Yu, K. 2024. Scieval: A multi-level large language model evaluation benchmark for scientific research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19053–19061.

Tyen, G.; Mansoor, H.; Carbune, V.; Chen, P.; and Mak, T. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. In Ku, L.-W.; Martins, A.; and

Srikumar, V., eds., *Findings of the Association for Computational Linguistics ACL 2024*, 13894–13908. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics.

Wang, K.; Ren, H.; Zhou, A.; Lu, Z.; Luo, S.; Shi, W.; Zhang, R.; Song, L.; Zhan, M.; and Li, H. 2023. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xu, B.; Yang, A.; Lin, J.; Wang, Q.; Zhou, C.; Zhang, Y.; and Mao, Z. 2023. Expertprompting: Instructing large language models to be distinguished experts. *arXiv preprint arXiv:2305.14688*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Yasunaga, M.; Chen, X.; Li, Y.; Pasupat, P.; Leskovec, J.; Liang, P.; Chi, E. H.; and Zhou, D. 2023. Large language models as analogical reasoners. *arXiv preprint arXiv:2310.01714*.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Yuan, Z.; Yuan, H.; Li, C.; Dong, G.; Lu, K.; Tan, C.; Zhou, C.; and Zhou, J. 2024. Scaling Relationship on Learning Mathematical Reasoning with Large Language Models.

Zhang, Z.; Zhang, A.; Li, M.; and Smola, A. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Zhou, A.; Wang, K.; Lu, Z.; Shi, W.; Luo, S.; Qin, Z.; Lu, S.; Jia, A.; Song, L.; Zhan, M.; et al. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921*.