FormulaReasoning: A Dataset for Formula-Based Numerical Reasoning

Xiao Li* **and Bolin Zhu*** **and Kaiwen Shi**
Sichen Liu and Yin Zhu and Yiwei Liu and Gong Cheng†
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{xiaoli.nju,bolinzhu}@smail.nju.edu.cn
{kaiwenshi,sichenliu,yinzhu,ywliu}@smail.nju.edu.cn
gcheng@nju.edu.cn

Abstract

The application of physics formulas is a fundamental human capability in numerical reasoning. While existing datasets often rely on implicit mathematical knowledge, they rarely explicitate the underlying formulas. To address this, we introduce FormulaReasoning, a new benchmark for formula-based numerical reasoning comprising 5,324 questions requiring calculations grounded in external physics principles. We provide high-quality, fine-grained annotations in English and Chinese—including formula structures, parameter names, symbols, values, and units—curated through manual effort and LLM-assisted validation. Additionally, we provide a consolidated formula database as an external knowledge source. To further challenge model performance, we develop an extended version of the dataset by coupling multiple questions. We evaluate various architectural and methodological frameworks, including retrieval-augmented methods, modular reasoning (formula generation, parameter extraction, and calculation), and preference-based optimization. Our analysis identifies critical challenges in formula-based reasoning, highlighting significant opportunities for future methodological advancement.

1 Introduction

Numerical reasoning constitutes one of the significant forms within natural language reasoning (Frieder et al., 2023). The study of numerical reasoning has seen substantial progress in recent years, driven largely by the development of LLMs (OpenAI, 2023; Touvron et al., 2023; Yang et al., 2024) and specialized datasets (Wang et al., 2017; Dua et al., 2019; Amini et al., 2019; Cobbe et al., 2021). Current datasets for numerical reasoning typically include simple commonsense numerical questions that under-reflect the complexity of real-world problems. These datasets either do not provide process supervision information, or the provided reasoning steps are essentially incomplete, as they often rely on implicit commonsense knowledge but not explicit knowledge guiding the reasoning process. This issue becomes particularly evident when LLMs experience hallucination (Frieder et al., 2023; Bang et al., 2023), especially in the absence of clear knowledge guidance. Consequently, one might ask *"What explicit knowledge could be used to guide a numerical reasoning process?"*. Formulas exactly represent such knowledge that has been largely overlooked in previous research but is frequently utilized in real-life applications.

Limitations of existing datasets Take a question from the GSM8K dataset (Cobbe et al., 2021) as an example: "A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?" This question implicitly uses *simple commonsense mathematical knowledge* (e.g., half means dividing by 2) to solve without requiring complex domain-specific formular knowledge for numerical reasoning. Recently, Liu *et al.* (Liu et al., 2023) constructed two formula-based datasets, Math23K-F and MAWPS-F. Although 33% and 38% of the questions in these datasets, respectively, require the use of formulas, they still mainly consist of simple commonsense formulas (e.g., total_amount = unit_amount × total_number). Currently, *there is a lack of datasets where questions require domain-specific formulas to guide complex numerical reasoning*, such as the physics formula used to calculate the heat absorption of an object. Related work is in Appendix A.1.

Our work To fill this gap, we introduce FormulaReasoning, a dataset for numerical reasoning requiring domain-specific (physics) formulas. We provide fine-grained annotations (Figure 1), enabling effective differentiation of model capabilities, especially for smaller models. Table 1

*These authors contributed equally to this work.
†Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 31579–31601
July 2-7, 2026 ©2026 Association for Computational Linguistics

Question

There is a electric water heater, after 50kg of water is loaded into its tank, the water is heated from 20℃ to [X] by electricity. It is known that the specific heat capacity of water is C_water = 4.2×10^3J/(kg·℃). If the total electrical energy consumed during the heating process is 1×10^7J, what is the thermal efficiency of the water heater?
If we already know that the answer to the above question is 84%, then what should the content corresponding to [X] in the question be?

Explanation (Reasoning Steps)

Heat absorbed by water can be obtained from: [Heat absorbed by water] = [Thermal efficiency of the water heater] * [Total electrical energy consumed] = 84% * (1* 10^7 J) = 8400000 J. Heat absorbed by water = 8400000 J.
Degree of water temperature increase is given by: [Degree of water temperature increase] = [Heat absorbed by water] / ([Mass of water] * [Specific heat capacity of water]) = 8400000 J / (50 kg * 4.2 * 10^3 J/(kg·℃)) = 40 ℃. Degree of water temperature increase = 40 ℃.
Calculating the Final temperature: [Final temperature] = [Degree of water temperature increase] + [Initial temperature] = 40℃ + 20℃ = 60℃. Final temperature = 60℃.
Answer = 60℃

Parameter Table

Parameter Name	Symbol	Numerical Value	Unit
Degree of water temperature increase	Δt	40	℃
Final temperature	t_{final}	20	℃
…	…	…	…
Heat absorbed by water	$Q_{absorbed}$	8400000	J
Mass of water	m_{water}	50	kg

Figure 1: An example from FormulaReasoning. Numerical values with units given in the question and obtained from intermediate steps are highlighted in red and purple, respectively. Formulas and their elements are in blue.

Table 1: Statistics of Math23K-F, MAWPS-F, GSM8K, MATH, and our FormulaReasoning.

Dataset	Math23K-F	MAWPS-F	GSM8K	MATH	FormulaReasoning
# questions	23,162	2,373	8,792	12,500	5,324
# questions requiring formulas	7,750	911	N/A	N/A	5,324
# formulas (and variants)	51 (131)	18 (46)	0 (0)	0 (0)	272 (845)
Avg. # reasoning steps	1.16	1.01	3.59	Not Provided	2.36

compares FormulaReasoning with existing datasets. Compared to Math23K-F and MAWPS-F, FormulaReasoning contains *more diverse formulas* (272 vs. 18–51). Moreover, the *higher average number of reasoning steps* (2.36 vs. 1.01–1.16) provides a more comprehensive evaluation framework for understanding how different model architectures handle multi-step formula-based reasoning.

Specifically, we collected questions requiring formula-based numerical reasoning from junior high school physics exams. With combined efforts of manual annotation and LLM assistance, we annotated each question with an explanation text that provides *normalized reasoning steps with relevant formulas* (including formula structures, parameter names, symbols, numerical values, and units) and a final answer. We also built a *consolidated formula database* that functions as an external knowledge base and can be used by retrieval-based/augmented systems. The original questions and annotations are in Chinese. We used an LLM to translate them into high-quality English, forming a *bilingual dataset*.

We evaluated FormulaReasoning using various reasoning paradigms: standard LLMs (4B to >100B parameters), fine-tuned models with CoT supervision, data augmentation, and RAG. We also derived preference data for DPO. Our analysis reveals significant differences in how approaches handle formula-based reasoning.

To further challenge the reasoning capabilities of models, we constructed FormulaReasoning+. This extended version coupled each question with another question and reformulated direct calculation as solving a system of equations, significantly increasing the difficulty of numerical reasoning.

Broader impact It is worth noting that while FormulaReasoning is constructed within the physics domain, the underlying framework is *domain-agnostic*. This pipeline treats formulas fundamentally as operator-argument structures, decoupled from specific physics semantics. By abstracting formulas into an operator-parameter structure, our dataset construction pipeline can be readily extended to other disciplines requiring structured knowledge, such as finance (e.g., compound interest formulas), chemistry (e.g., stoichiometry), and geometry. Furthermore, beyond serving as

a benchmark, FormulaReasoning offers significant research value for *step-wise reasoning*, *structured formula retrieval*, and *preference optimization (DPO)*, providing rich resources for advancing neuro-symbolic reasoning.

Our **contributions** are summarized as follows.

- We construct a formula-based numerical reasoning dataset with fine-grained annotations for each question. It can be applied to evaluate knowledge-guided reasoning capabilities. Its normalized reasoning steps are useful for the emerging study of step-supervised reasoning.

- We perform an evaluation of LLMs of various sizes, RAG, fine-tuned small models, and DPO methods based on derived preference data. Our experimental results establish a solid baseline for future research and also indicate that the performance disparity across model scales presents new research opportunities for efficient formula-based reasoning.

FormulaReasoning is available from HuggingFace https://huggingface.co/datasets/cat-overflow/FormulaReasoning. Our code is available from GitHub https://github.com/nju-websoft/FormulaReasoning.

2 Dataset Construction

We collected questions from Chinese junior high school physics exams. We invited five graduate students to act as annotators, all of whom hold a bachelor's degree in science and engineering. For all questions, we normalized their explanations. Each question was annotated with normalized reasoning steps in natural language and a tabular representation of the steps using formulas, including the numerical values and units for all the parameters in the formulas. This process involved a combination of manual annotation and the assistance of an LLM[1] to improve the efficiency of annotation. We compiled all the formulas and merged those that express the same meaning to create a consolidated formula database. Finally, we altered the questions to avoid label leakage and translated the dataset into English. In the following, we elaborate this process to construct FormulaReasoning.

[1]During dataset construction, we consistently accessed Qwen-max as our LLM via its API.

Table 2: Original explanation and explanation with normalized formulas (highlighted in blue).

Original explanation:
The change in water temperature is 60 - 20 = 40 °C. Therefore, the heat absorbed by the water is Q_{absorbed}=50 kg × 4.2 ×10^3 J/(kg·°C) × 40 °C = 8.4 ×10^6 J. Given that the total electrical energy consumed in the heating process is 1 × 10^7 J, the thermal efficiency of the water heater can be calculated using the formula for the efficiency of a heat engine: η = Q_{absorbed}}/W_{total}×100% = (8.4 × 10^6 J)/(1.0 × 10^7 J)×100% = 84%. Answer: If it is known that the total electrical energy consumed during the heating process is 1 × 10^7, the thermal efficiency of the water heater is 84%.

Explanation with normalized formulas:
1. Calculating the temperature increase in water: [Degree of water temperature increase] = [Final temperature] - [Initial temperature] = 60 °C - 20 °C = 40 °C. The degree of water temperature increase = 40 °C.
2. Calculating the heat absorbed by water: [Heat absorbed by water] = [Mass of water] × [Specific heat capacity of water] × [Degree of water temperature increase] = 50 kg × 4.2 × 10^3 J/(kg·°C) × 40 °C = 8400000 J. The heat absorbed by water = 8400000 J.
3. The thermal efficiency of the water heater can be obtained from: [Thermal efficiency of the water heater] = [Heat absorbed by water] / [Total electrical energy consumed] × 100% = 8400000 J / (1 × 10^7 J) * 100% = 84%. The thermal efficiency of the water heater = 84%.
Answer = 84%

2.1 Preprocessing

We crawled 18,433 junior high school physics exam questions in China from 2015 to 2024 from public sources, including only those with free-text answers and excluding multiple-choice and true/false questions. Each raw question contains a *question text* and an *explanation text that includes reasoning steps*. We eliminated questions requiring diagrams.

We filtered the questions by identifying the presence of numerical values within the explanation and confirming that the final answer was numerical. Using regular expressions, we extracted the *final numerical answer* including its unit from the explanation. We found that for 487 questions, our regular expressions did not return results, so we manually extracted their answers from the explanation text. After this preprocessing, we compiled an initial dataset comprising 6,306 questions.

2.2 Formula Normalization

The reasoning steps in the raw explanation text lacked a normalized format and were expressed casually. Some formulas mixed parameter names (e.g., "mass of water") with symbols (e.g., "m_{water}"), while others simply provided calculations over numerical values without parameter

names or symbols. To ensure that all explanations adopted a common form of formulas, we performed normalization, as illustrated in Table 2. We aimed to *identify and normalize the formulas in the original explanations* and then *verify and unify their formats*. Manually carrying out such tasks would require a significant effort. However, since the process is not open-ended, but rather structured and verifiable, we could automatically, e.g. *using an LLM*, extract formulas from a normalized explanation, symbolicly calculate each step, and compare the result with the given answer to ensure the accuracy of our normalization.

Specifically, to improve the efficiency of annotation, we adopted a *coarse-to-fine annotation process* with the help of an LLM. We first prompted the LLM to revise the reasoning steps (in particular, the formulas) in the explanation text into a normalized form. Then, we prompted the LLM to correct minor errors within the normalized explanations, including formatting issues in formula annotations and inaccuracies in the parameters used during calculation. In the following, we elaborate this two-stage process. More details, including the prompts, are provided in Appendix A.2.1.

Coarse-grained annotation We introduced each question with its original explanation and answer to guide the LLM through a few-shot prompt to normalize the explanation. We observed that the quality of the normalized explanations was generally satisfactory. We also required the LLM to present the symbol, numerical value, and unit of each parameter in formulas in the form of a table, as illustrated in Figure 1.

Fine-grained annotation We checked the correctness of the formula format in the explanations by rules, including whether there were omissions in parameter names, symbols, or units, and these minor issues were correctable. To assess the correctness of each normalized explanation, we extracted formulas from the explanation and calculated an answer using the Numbat calculator[2]. We achieved a formula normalization accuracy of 85.95% through this programmatic verification. We used few-shot prompts to correct LLM errors and removed poor-quality questions, such as those missing reasoning steps, as illustrated in Appendix A.2.2. After that, our dataset contains 5,420 questions remaining. To further improve data quality, we performed a rigorous human audit on a critical subset of the data. Specifically, we manually reviewed and corrected 363 questions (approximately 45.4% of the test set; see Section 3.1 for dataset split), which primarily consisted of questions where the model predictions deviated from the ground truth.

[2]https://numbat.dev. Numbat is designed for scientific computation with support for physical units.

Table 3: Changes in the number of formulas after each merging step.

Step	# Formulas
Before merging	12,906
After symbolic rules based merging	1,163
After semantics based merging	439
After manual review and error correction	272

2.3 Formula Database Construction

Our next step was to *construct a consolidated formula database for the entire dataset*. Parameters in the same formula can be expressed differently in various problem contexts. For example, the two formulas "[weight of water] = [mass of water] * [gravitational acceleration]" and "[weight] = [mass] * [gravitational acceleration]" both calculate the weight of an object. It would be helpful to standardize these formulas for use with LLMs, particularly in methods such as RAG (Tran et al., 2025), step supervision (Zhang et al., 2024), and tree-based search (Zhao et al., 2024), to avoid long-tail distribution issues caused by treating different formula expressions as different formulas.

We divided the construction process of a consolidated formula database into three steps: 1) Merge formulas through symbolic rules. 2) Merge formulas through a semantics-based method. 3) Manually review and correct errors. In Table 3, we present the initial number of formulas and the remaining number of formulas after each step.

Symbolic rules based merging This was achieved by *comparing formula structures and symbols*. Consider the following as an example of judging whether two formulas have the same structure. The three formulas "$f_1: a_1 = (b_1 + c_1)/d_1$", "$f_2: a_2 = (b_2 + c_2)/d_2$", and "$f_3: b_1 = a_1 * d_1 - c_1$" have the same structure because f_2 can be derived from f_1 by renaming parameters, and f_3 can be obtained from f_1 by transformation. Moreover, in physics, physical quantities are conventionally represented by specific symbols. For example, the mass of an object is often denoted by "m" and the density of an object is frequently represented by

"ρ". Subscripts are then used to specify which specific object a physical quantity refers to, such as "ρ_{water}" for the density of water. Therefore, for each pair of formulas, we first computed all possible transformations of each formula to obtain a set of all its variants. Then, we compared the formula structures in the two sets to determine if the two formulas shared a structure. If so, we checked whether their symbols, with subscripts removed, were identical. If so, we considered these two formulas to be mergeable. When merging, we retained the parameter with the shorter name of the two. After merging based on such symbolic rules, we reduced the number of formulas in the formula database from 12,906 to 1,163.

Semantics based merging In the previous step, the semantic information in the parameter names was not used. This led us to *perform merges grounded on the semantics of parameter names*. For example, two formulas that were not merged during the symbolic merging stage, "[density] = [mass] / [volume]" and "[density of water] = [mass of water] / [volume of water]", should actually be merged. We identified such mergeable formulas based on the semantic information in the parameter names, e.g., "density" and "density of water" are semantically similar. Specifically, for formulas with identical structures, we tokenized[3] each pair of their corresponding parameter names into two sets of words. When the two sets overlapped, the two parameters were considered to have a semantic connection and the two formulas became candidates for merging. Using this approach, we identified a set of pairs of potentially mergeable formulas and then consulted the LLM for a detailed evaluation of each pair. The prompt is provided in Appendix A.2.3. After this step, the number of formulas in the formula database was reduced from 1,163 to 439.

Manual review and error correction Upon completing the aforementioned merging process, we manually inspected the accuracy of the results, rectified the instances where errors occurred during merging, and manually merged formulas that were overlooked by the LLM. In this process, two human volunteers cross-validated the results of manual review and correction. Finally, we obtained a consolidated formula database consisting of 272 distinct formulas.

[3]We used jieba: https://github.com/fxsjy/jieba.

2.4 Question Alteration

The original questions in FormulaReasoning were collected from publicly accessible online sources. Considering that LLMs were typically trained on massive-scale corpora that potentially encompassed these publicly available resources, there existed a non-negligible risk of data contamination and label leakage. To mitigate this potential bias and construct a more rigorous benchmark, we implemented systematic strategies for question alteration. Specifically, we employed *inverse transformation* (You et al., 2024) and *parameter noise injection* (Wei and Zou, 2019) to generate a modified version for each original question while preserving its fundamental reasoning nature.

Inverse transformation We replaced a random parameter in the original question with a placeholder "[X]", and added the original answer to the question. The new question then asked for the value of the masked parameter "[X]". Due to the various expressions of parameters, some could not be easily found and replaced in the question by string matching, e.g., "$4.2 \times 10^6 J/h$" and "$1 \times 10^{-3} dm^3$" could be written as "4.2×10^6 J per hour" and "1L", respectively. We had to exclude such questions.

Parameter noise injection We randomly selected a parameter from the entire set of parameters in all questions and appended both its parameter name and its numerical value to the question as a distractor. To ensure that the added parameter would not affect the solvability of the question (e.g., not introducing two conflicting values for the same physical quantity), we ensured that the added parameter was different from all the parameters involved in answering the original question.

To verify that the modified questions were solvable, we used SymPy[4] to solve the equations. During the verification process, some equations could not be solved (e.g., SymPy cannot handle "%" and "1000kg/t"), so we removed these questions. The final number of the remaining questions is 5,324.

2.5 English Version of FormulaReasoning

LLMs translated the Chinese questions, explanations, and formulas into English using prompts from Appendix A.2.4. All components, including parameter names and reasoning steps, were translated.

[4]https://www.sympy.org/

To assess translation quality, we compared the perplexity of LLM and manual translations for 50 sampled questions using the Qwen2.5-14B-Instruct model. The LLM scored 6.28 compared to 7.26 for human versions, demonstrating reasonable fluency. Additionally, a manual review of these 50 questions found no translation errors that would influence consistency or question-answering accuracy.

2.6 Question Coupling

To further elevate the reasoning complexity, we introduced an extended version FormulaReasoning+ that tightly coupled the original questions with a synthesized auxiliary question and reformulated direct calculation as solving a system of equations. First, we selected a mutable parameter in the original question to mask as $[X]$ and denoted the original answer as $[Y]$. Second, we constructed a linear expression $E(X, Y)$ (e.g., $c_1Y + c_2X$) ensuring a positive result and magnitude alignment between X and Y by adjusting coefficients. Third, we calculated the value $v = E(X, Y)$ and prompted an LLM to generate an independent physics question (Q_{aux}) whose answer is v. Finally, we formulated a hybrid question that requires first solving Q_{aux} to obtain v and then solving a system of equation in X and Y where one equation represents the original question and the other is $v = E(X, Y)$. After removing the questions with failed coupling, the final FormulaReasoning+ contains 4,392 extended questions. It is worth noting that Question Coupling and Question Alteration (Section 2.4) are parallel processes, both derived from the same original questions. For the detailed coupling process, please refer to Appendix A.2.5.

3 Experimental Setup

To study how methodological choices affect formula-based reasoning in FormulaReasoning, we analyzed prompting, fine-tuning, and retrieval-augmented methods. We developed two specific approaches: one decomposing reasoning into formula generation and calculation steps, and another utilizing data augmentation. We also explored preference learning for refinement. This setup enables a comprehensive comparison of architectural and methodological impacts on performance.

3.1 Dataset Split

We divided FormulaReasoning into three subsets: training, *HoF* (Homologous Formulas) test, and *HeF* (Heterologous Formulas) test, comprising 4,524, 413, and 387 questions, respectively. Inheriting the same split from FormulaReasoning, FormulaReasoning+ contains 3,608 training questions, with the *HoF* test and *HeF* test sets containing 406 and 378 questions, respectively. All formulas in the HoF test set appeared in the training set, while in the HeF test set, each question required at least one formula not seen in the training set. This division was to evaluate the generalizability of fine-tuned models on new formulas.

3.2 Evaluated Methods

Human Performance We recruited 108 students from a high school, each student being assigned 7–8 questions. Each student was given 40 minutes to complete these questions. These questions were used as part of their in-class exercises and, at the end, each student received a gift. The final statistics were collected to assess human performance, which was consented to by all students. To ensure rigorous grading, the correctness of each student's response was checked manually. A student's response was marked as correct only if both the final answer and the detailed solution process were correct. The reported human accuracy represents the aggregated performance across all participants based on this strict criterion.

The human performance for FormulaReasoning+ was established in a similar manner, with graduate students serving as participants assigned 100 questions.

LLMs Following (Kojima et al., 2022), we incorporated "Let's think step by step" (i.e. **CoT**) into a zero-shot prompt to guide LLMs in generating reasoning steps. The prompt is in Appendix A.3.2.

We conducted experiments across a diverse spectrum of models including both cost-friendly small-scale models and high-cost LLMs. The list of evaluated models is provided in Appendix A.3.1. This comprehensive evaluation allowed us to assess capabilities across different model scales while identifying persistent challenges in formula reasoning.

We also compared CoT with **Program of Thought (PoT)** (Chen et al., 2023). In PoT, we used a Python interpreter to execute the code and obtain an answer.

Formula Retriever We trained a formula retriever on the training set. Specifically, we encoded each question using the Chinese-BERT-wwm-base model (Devlin et al., 2019; Cui et al., 2021) to

obtain the CLS vector of the question. The formulas in the formula database were not encoded in this way, as they contained structured information so that two textually similar formulas could have entirely different meanings (e.g., $z_1 = x \times y$ and $z_2 = x/y$). Therefore, each formula was instead represented by a randomly initialized vector that was updated during training. We calculated the cosine score between the question vector and the formula vector. The retriever was then trained with in-batch negatives and contrastive learning loss (Gao et al., 2021). During inference, for each question in the HoF test, we retrieved the top five formulas with the highest scores and included them in the prompt to augment LLM generation. More details are provided in Appendix A.3.3.

Supervised Fine-Tuned Models We found that directly prompting models having fewer than 8B parameters did not produce satisfactory results, so we performed supervised fine-tuning of small models (Qwen2.5-Math-7B-Instruct and Qwen2.5-Math-1.5B-Instruct). Unlike large models, small models struggled with numerical extraction and calculation. To enhance their reasoning capabilities, we developed: (1) **CoT-Supervised Fine-Tuning (CoT-SFT)**, a three-step process where the model generates relevant formulas, extracts parameter values and units from questions, and uses Numbat for final calculation; and (2) **Data Augmentation (DA)**, using Qwen-max to generate and verify new training examples with correct reasoning steps (details in Appendix A.3.4).

Direct Preference Optimization (DPO) We used the preference data derived in Appendix A.3.5 to experiment on DPO. In the generation process, we used DeepSeek-R1-Distill-Qwen-7B as the generation model, and used Qwen2.5-Math-PRM-7B as the process reward model (PRM).

The implementation details are provided in Appendix A.3.6.

3.3 Evaluation Metrics

We utilized Numbat to evaluate the prediction generated by each model against the gold-standard answer. A prediction was deemed correct if the relative error, (prediction − gold) / gold, was less than 1%. We used **accuracy**, which is the proportion of questions correctly answered, as our metric. Crucially, our evaluation leveraged Numbat for **dimensional analysis**, allowing us to automate the unit conversion (e.g., $1000\,\mathrm{J} = 1 \times 10^3\,\mathrm{N \cdot m}$) beyond simple numerical matching.

To evaluate the quality of the complete reasoning steps, we also used **PRM score** to evaluate the reasoning steps generated by LLMs. There were two evaluation settings: **one-step** and **multi-step**. One-step evaluation scored the overall output using PRM, whereas multi-step evaluation divided the output into multiple steps, scored each individually, and took the average as the final score. Following the setting of DPO mentioned above, here we again used Qwen2.5-Math-PRM-7B as the PRM. We adopted PRM scoring instead of strict formula matching because valid reasoning paths in formula-based problems are often non-unique; different formula combinations can lead to the correct answer.

4 Experimental Results

We present the evaluation results on the Chinese version of FormulaReasoning. The results on FormulaReasoning+ are in Table 5. The results on the English version are in Appendix A.4.1, which are consistent with those on the Chinese version. The results of PoT are in A.4.2, not better than CoT.

Table 4: Accuracy of LLMs with CoT prompts.

Model	HoF	HeF	Avg.
ERNIE-4.5-21B-A3B	66.34	67.96	67.12
Gemma-3n-E4B-it	14.04	14.47	14.25
GPT-oss-20B	85.96	82.95	84.50
GPT-4o	69.49	65.37	67.50
GLM-4-plus	61.74	63.57	62.63
Qwen3-235B-A22B	80.63	83.98	82.25
DeepSeek-R1	88.62	91.47	90.00
o3-mini	92.01	89.66	90.87
GPT-5	92.01	92.76	92.37
Human	93.49	90.47	92.03

Table 5: Accuracy of LLMs with CoT prompts on FormulaReasoning+.

Model	HoF	HeF	Avg.
ERNIE-4.5-21B-A3B	6.89	6.08	6.48
Gemma-3n-E4B-it	0.99	1.32	1.16
GPT-oss-20B	59.36	57.41	58.38
GPT-4o	15.27	17.72	16.50
GLM-4-plus	17.98	16.67	17.32
Qwen3-235B-A22B	78.08	77.78	77.93
DeepSeek-R1	83.99	80.42	82.20
o3-mini	73.89	73.96	73.93
GPT-5	76.11	73.28	74.69
Human	90.00	88.00	89.00

4.1 Results of LLMs with CoT Prompts

We evaluated a diverse range of LLMs, as shown in Table 4 and Table 5. Our analysis reveals that model performance correlates with both parameter scale and architectural design. As shown in Table 4, large-scale models (e.g., DeepSeek-R1 and o3-mini) achieved competitive results with accuracies exceeding 88% on both HoF and HeF metrics. However, substantial performance gaps existed for smaller-scale models, where architectures like Gemma-3N-E4B-it exhibit accuracies below 15%. Notably, despite their impressive capabilities, even advanced large-scale models such as GPT-4o (67.50%), GLM-4-plus (62.63%), and Qwen3-235B-A22B (82.25%) still trailed behind human performance (92.03%). This significant variance suggests that *while parameter scaling yields notable improvements, there remains considerable optimization potential for resource-efficient models*. Future work should investigate architectural adaptations and training strategies specifically tailored for compact models, which are critical for practical deployment scenarios where computational resources are constrained. The observed performance spectrum underscores the importance of scale-aware evaluation in formula-based reasoning research.

The results on FormulaReasoning+, as shown in Table 5, demonstrate the increased difficulty of the extended dataset. All models experienced a drop in performance. Notably, the accuracy of GPT-4o and GLM-4-plus declined sharply to 16.50% and 17.32% respectively. Even GPT-5 showed a decrease of 17.68%, highlighting the challenges posed by the more complex reasoning tasks. Furthermore, none of the evaluated models managed to surpass the human baseline on this extended dataset, suggesting that the task still preserved a meaningful level of difficulty for LLMs.

We provide an error analysis in Appendix A.5.

PRM scores to assess the reasoning steps generated by LLMs are included in Appendix A.4.3.

4.2 Results of LLMs with Formula Retriever

We found that *incorporating retrieved formulas into the prompts improved the performance of LLMs*. For example, for ERNIE-4.5-21B-A3B, the accuracy increased by 0.72 from 66.34 to 67.06 on the HoF. Note that it would be meaningless to carry out this experiment on the HeF test set where the required formulas were not included when training the retriever.

To further understand the impact of the number of retrieved formulas, we conducted a k-ablation study with $k \in \{0, 1, 3, 5\}$. Compared to the baseline without retrieval ($k = 0$), setting $k = 1$ and $k = 5$ yielded improvements of $+1.50\%$ and $+1.20\%$, respectively, while the accuracy peaked at $k = 3$ with an improvement of $+2.18\%$. This suggests an optimal balance between providing sufficient relevant formulas and minimizing the introduction of irrelevant, noisy information.

Indeed, our investigation found that the top five formulas retrieved often included irrelevant ones, as the number of formulas required varied for different questions. The presence of extraneous formulas might introduce noise, suggesting considerable room for improvement in structured formula retrieval.

Table 6: Accuracy of supervised fine-tuned models.

Model	HoF	HeF	Avg.
Qwen2.5-Math-7B-Instruct	65.62	58.91	62.27
+ CoT-SFT	72.64 +7.02	65.63 +6.72	69.14 +6.87
+ DA	67.55 +1.93	62.53 +3.62	65.04 +2.77
Qwen2.5-Math-1.5B-Instruct	64.41	55.56	59.99
+ CoT-SFT	66.34 +1.93	53.75 −1.81	60.05 +0.06
+ DA	66.59 +2.18	59.17 +3.61	62.88 +2.89

4.3 Results of Supervised Fine-Tuned Models

Table 6 shows the results for the supervised fine-tuned models, with and without CoT-SFT or DA. In most settings, both models achieved higher scores on the HoF test set than on HeF, yet they still exhibited considerable performance on the latter. This indicated that the unseen formulas influenced the performance of the models, though they still demonstrated a level of generalizability.

In particular, Qwen2.5-Math-7B-Instruct with CoT-SFT outperformed GPT-4o on FormulaReasoning, *demonstrating the effectiveness of our fine-tuning method: delegating numerical calculations to a calculator and focusing on CoT reasoning*, although such a comparison might not be entirely fair. Data augmentation also improved the reasoning capacity of small models.

4.4 Results of DPO (including Results on Other Numerical Reasoning Datasets)

As shown in Appendix A.4.4, DPO with the preference data derived from the training set of FormulaReasoning generally improved the accuracy of 1.5B–7B models on both FormulaReasoning and other numerical reasoning datasets:

GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), and GaoKao2023-en (Chen et al., 2025). These models exhibited much lower accuracy on FormulaReasoning than those datasets. *The results underscore the distinct characteristics of our dataset compared to other numerical reasoning benchmarks and its additional value as a potential DPO resource.*

5 Conclusion

We introduced FormulaReasoning, a bilingual dataset for formula-based numerical reasoning. For each question, we annotated its complex reasoning steps with normalized physics formulas. We constructed a consolidated formula database after merging equivalent formulas, serving as an external knowledge base to be used in RAG. We used our dataset to evaluate LLMs of various sizes, RAG, fine-tuned small models, and DPO methods, revealing significant performance variations across model scales. Our findings highlight that while larger models show promising performance, smaller-scale LLMs still face significant challenges in formula-based numerical reasoning, indicating opportunities for further advancements in multi-step reasoning guided by domain knowledge.

Future work will use FormulaReasoning's formula knowledge to enhance LLM numerical reasoning via knowledge-driven or reinforcement learning methods. Moreover, given the domain-agnostic nature of our construction framework, we aim to extend our implementation to other scientific and engineering domains, such as chemistry and finance, to build a more comprehensive multi-disciplinary formula reasoning benchmark.

Limitations

One limitation of our work is that although FormulaReasoning provides step-level supervision information (i.e., formulas), due to the diversity of reasoning paths and formula expressions, we only used PRM to perform step-level process evaluation. Another limitation is that our dataset is focused on physics. We chose junior high school physics because it is not too hard to be understood by ordinary people, which benefited our annotation and evaluation efforts. It is possible to explore formula-based numerical reasoning in other domains such as chemistry and engineering.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Baidu-ERNIE-Team. 2025. Ernie 4.5 technical report.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–718, Nusa Dua, Bali. Association for Computational Linguistics.

Jiayi Chen, Xiao-Yu Guo, Yuan-Fang Li, and Gholamreza Haffari. 2022. Teaching neural module networks to do arithmetic. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1502–1510, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*

Zui Chen, Tianqiao Liu, Mi Tian, Weiqi Luo, Zitao Liu, and 1 others. 2025. Advancing mathematical reasoning in language models: The impact of problem-solving data, data synthesis methods, and training stages. In *ICLR 2025*.

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. *arXiv preprint arXiv:2110.14168*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert. *IEEE Transactions on Audio, Speech and Language Processing*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Luu Anh Tuan, and Shafiq Joty. 2024. Data augmentation using llms: Data perspectives, learning paradigms and challenges. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1679–1705.

Iddo Drori, Sarah Zhang, Zad Chin, Reece Shuttleworth, Albert Lu, Linda Chen, Bereket Birbo, Michele He, Pedro Lantigua, Sunny Tran, and 1 others. 2023. A dataset for learning university stem courses at scale and generating questions at a human level. In *AAAI 2023*, pages 15921–15929.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, Chevalier Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2023. Mathematical capabilities of chatgpt. *Advances in neural information processing systems*, 36:27699–27744.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gemma. 2025. Gemma 3n.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2019. Neural module networks for reasoning over text. In *ICLR 2019*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *NeurIPS Systems Datasets and Benchmarks Track 2021*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Zixian Huang, Yulin Shen, Xiao Li, Gong Cheng, Lin Zhou, Xinyu Dai, Yuzhong Qu, and 1 others. 2019. Geosqa: A benchmark for scenario-based question answering in the geography domain at high school level. In *EMNLP 2019*, pages 5866–5871.

Jeonghwan Kim, Junmo Kang, Kyung-min Kim, Giwon Hong, and Sung-Hyon Myaeng. 2022. Exploiting numerical-contextual knowledge to improve numerical reasoning in question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1811–1821, Seattle, United States. Association for Computational Linguistics.

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

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Xiao Li, Yawei Sun, and Gong Cheng. 2021. Tsqa: tabular scenario based question answering. In *AAAI 2021*, pages 13297–13305.

Xiao Li, Yin Zhu, Sichen Liu, Jiangzhou Ju, Yuzhong Qu, and Gong Cheng. 2023a. Dyrren: A dynamic retriever-reranker-generator model for numerical reasoning over tabular and textual data. In *AAAI 2023*, pages 13139–13147.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Jia-Yin Liu, Zhenya Huang, Zhiyuan Ma, Qi Liu, Enhong Chen, Tianhuang Su, and Haifeng Liu. 2023. Guiding mathematical reasoning via mastering commonsense formula knowledge. *KDD 2023*.

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 *NeurIPS 2022*, pages 2507–2521.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. GPT-4o.

OpenAI. 2025a. GPT-5.

OpenAI. 2025b. o3-mini.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, Lisbon, Portugal. Association for Computational Linguistics.

Kashun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In *Findings of EMNLP 2023*, pages 12113–12139.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hieu Tran, Zonghai Yao, Zhichao Yang, Junda Wang, Yifan Zhang, Shuo Han, Feiyun Ouyang, and Hong Yu. 2025. RARE: Retrieval-augmented reasoning enhancement for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18305–18330, Vienna, Austria. Association for Computational Linguistics.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *AAAI 2019*, pages 7144–7151.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *ICLR 2022*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS 2022*, 35:24824–24837.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Chenxi Whitehouse, Monojit Choudhury, and Alham Fikri Aji. 2023. LLM-powered data augmentation for enhanced cross-lingual performance. In *EMNLP 2023*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang

Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Weihao You, Shuo Yin, Xudong Zhao, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. 2024. MuMath: Multi-perspective data augmentation for mathematical reasoning in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2932–2958, Mexico City, Mexico. Association for Computational Linguistics.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, and 1 others. 2024. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*.

Chujie Zheng, Sahand Sabour, Jiaxin Wen, Zheng Zhang, and Minlie Huang. 2023. AugESC: Dialogue augmentation with large language models for emotional support conversation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1552–1568, Toronto, Canada. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. In *ICLR 2022*.

A Appendix

A.1 Related Work

A.1.1 Numerical Reasoning Datasets

The study of numerical reasoning in natural language has existed for years. Numerous datasets, such as GSM8K (Cobbe et al., 2021), TSQA (Li et al., 2021), and MATH (Hendrycks et al., 2021), have addressed natural language numerical reasoning. Another line of research that focuses on numerical reasoning in natural language is math word problem (MWP). MWP tasks typically provide a short passage (i.e., a question) and require the generation of an arithmetic expression that can compute an answer. Representative datasets include MAWPS (Koncel-Kedziorski et al., 2016), Math23K (Wang et al., 2017), MathQA (Amini et al., 2019), etc. Several works focus on numerical reasoning in specialized domains. Examples include GeoSQA (Huang et al., 2019), which focuses on the geography domain, STEM (Drori et al., 2023) and ScienceQA (Lu et al., 2022) which cover multiple disciplines in science and technology. *Distinguished from these datasets, the numerical reasoning questions in our FormulaReasoning have explicitly labeled formulas*.

The recently introduced datasets Math23K-F and MAWPS-F (Liu et al., 2023) require formulas for 33% and 38% of the questions, respectively, but their formulas are all simple commonsense formulas (e.g., total_cost = unit_cost $\times$ total_number). *By contrast, our FormulaReasoning adapts questions from physics exams, with every question accompanied by a fine-grained annotation of domain-specific formulas. In addition, we provide a consolidated formula database that can serve as an external knowledge base to assess RAG solutions.*

A.1.2 Numerical Reasoning Methods

Methods for numerical reasoning have evolved from statistical approaches (Hosseini et al., 2014; Kushman et al., 2014) to those based on rules and templates (Shi et al., 2015; Wang et al., 2019) and further to deep learning models (Gupta et al., 2019; Chen et al., 2022; Kim et al., 2022; Li et al., 2023a). In recent years, with the rapid development of LLMs, they have demonstrated strong capabilities to resolve numerical reasoning questions. Consequently, several methods have been proposed to improve the reasoning abilities of LLMs, including the notable CoT method (Wei et al., 2022), along with many subsequent variant approaches (Kojima et al., 2022; Wang et al., 2022; Zhou et al., 2022; Li et al., 2023b). Preference learning methods have also emerged (Rafailov et al., 2023).

Our experiments implemented and evaluated representative existing methods as baselines for FormulaReasoning, including zero-shot CoT prompting for LLMs ranging from 4B to more than 100B parameters. We trained a specialized formula retriever to be used in RAG. We divided the reasoning process into formula generation, parameter extraction, and numerical calculation, and used data augmentation to enhance fine-tuned small models. We derived preference data and performed DPO.

A.2 Appendix for Dataset Construction

A.2.1 Prompts for Formula Normalization

The formula normalization process followed a coarse-to-fine annotation process as described in Section 2.2. In the coarse-grained annotation phase, the natural language explanations were normalized and the associated parameters were extracted from these explanations. These prompts are provided in Figures 2 and 3. The fine-grained annotation phase focused on error correction, which was divided into three specific categories: input errors, where the parameters mentioned in the explanation were absent from the question; calculation errors, which occurred when the Numbat calculator reported an error during the computation process; and output errors, where the final calculated answer was incorrect. We provide a prompt for correcting the calculation errors as an example in Figure 4; the prompts for the other two error types are provided in our GitHub repository. The entire normalization procedure used a few-shot prompt with six examples.

A.2.2 Examples of Removed Questions

The questions that remained incorrect despite multiple attempts by the LLM were of notably poor quality, e.g. missing important reasoning steps or having an incorrect reference answer. We provide an example of these questions in Figure 5.

A.2.3 Prompt for Semantics Based Merging

Semantics based merging primarily employed the LLM to comprehend formulas, determined if two formulas were semantically equivalent, and subsequently determined whether they could be merged into a single formula. The prompt for this procedure is illustrated in Figure 6. This approach ensured that nuanced meanings embedded within

Prompt actually used	English translation
我需要你修改问题原有的解析，给出规范格式的新解析，要求如下： 1.请逐步地进行思考,如果有公式组合的部分需要一步步地拆分成基本公式进行求解 2.公式中的计算符号，如“+”、“-”、“×”、“/”和“^”不能省略 3.公式需要同时给出符号和有具体含义的两种形式，然后代入数值计算得出答案 4.涉及到单位换算的部分需要展示出来具体过程 5.使用"[]"标注公式中的变量，其中科学计数法形式的数字"a×10^b"以及复杂的单位，需要使用"()"标注 6.如果有latex格式的公式，比如“\frac{Q_吸}{Q_放}”需要改成正常算式的形式：“Q_吸/Q_放” 7.解析开始如果给出“（解析可能存在错误）”表明给出的解析中可能存在错误 下面是一些例子： input: 问题：在某次测试中该车匀速行驶时受到的阻力是车重的 0.02 倍，在 0.5h 内行驶了 28km 的路程，求：行驶的平均速度为多少？ 解析：解：车辆在0.5h内行驶了 28km 的路程，所以行驶的平均速度为：V=s/t=28km/0.5h=56km/h。 output: 解： 原始解析比较清晰，下面将其转换为规范格式： 第一步，先计算平均速度: v=s/t [平均速度]=[路程]/[时间] 算式=28km/0.5h=56km/h 平均速度=56km/h 答案=56km/h ***...omitted...*** 下面是一个新的问题： 问题：**{{question}}** 解析：**{{explanation}}**	I need you to modify the original explanation of the question and provide a new explanation with the following requirements: 1. Please think step by step. If there has formula combination, you need to decompose the combination into basic formulas step by step. 2. Calculation symbols such as "+", "-", "×", "/" and "^" in formulas cannot be omitted. 3. The formula needs to be given in both symbolic and concrete forms. After that, you need to substitute into the numerical calculation to obtain the answer. 4. The part related to unit conversion needs to show the specific process. 5. Use "[]" to label variables in formula, "()" for numbers like "a×10^b" in scientific notation and for complex units. 6. If you have a latex formula such as \frac{Q_in}{Q_out}, change it to a normal formula: "Q_in /Q_out". 7. Start with "(Explanation may be wrong)" indicates that there may be an error in the given explanation. Here are some examples: input: Question: In a test, the resistance of the car at a constant speed is 0.02 times the weight of the car, and the car traveled a distance of 28km in 0.5h. What is the average speed? Explanation: The car has traveled a distance of 28km in 0.5h.The average speed of the vehicle is V=s/t=28km/0.5h=56km/h. output: Solution: The raw parsing is clear, so let's convert it to a canonical format: 1.we compute the average speed: v=s/t [Average speed]=[distance]/[time] expression=28km/0.5h=56km/h Average speed =56km/h Answer=56km/h ***...omitted...*** There's a new question: Question: **{{question}}** Explanation: **{{explanation}}**

Figure 2: Prompt for explanation normalization.

formulas were accurately captured and evaluated for potential merging, thereby enhancing the quality of the formula database.

A.2.4 Prompt for Dataset Translation

The prompts for translating the dataset are provided in Figure 7.

A.2.5 Question Coupling

In this section, we provide a detailed walkthrough of the *Question Coupling* method using a concrete example. This method increased reasoning difficulty by coupling the original question with a generated auxiliary question through a mathematical expression. All the questions had been verified by Numbat to meet the constraints and be solvable.

Original Question Consider the following original question from FormulaReasoning:

> **Question:** A block of aluminum with a mass of 50 g releases 880 J of heat, and its temperature decreases to 12°C. What was the original temperature of the aluminum block? [$C_{aluminum} = 0.88 \times 10^3$ J/(kg·°C)]
> **Answer:** 32°C

Step 1: Parameter Masking We first identified mutable parameters in the question. In this example, we randomly selected the heat released (880 J) as the parameter to mask, denoted as $[X]$. We also designated the original answer (32°C) as the second unknown parameter $[Y]$.

- $[X]$: Heat released (880)
- $[Y]$: Original temperature (32)

Step 2: Expression Construction We constructed a linear expression involving $[X]$ and $[Y]$. To ensure numerical stability and reasonable magnitude, we adjusted the coefficients based on the order of magnitude of the parameters. In this example, we aligned the magnitudes and constructed the expression:

$$E(X, Y) = 50 \cdot Y - 1 \cdot X$$

Substituting the values:

$$50 \times 32 - 1 \times 880 = 1600 - 880 = 720$$

Prompt actually used	English translation
我需要你帮我从给定的问题和解析中提取出解决问题使用到的公式，以及公式对应的参数表： 1.请逐步地进行思考,先对解析进行分析，然后生成提取结果，如果有公式组合的部分需要一步步地拆分成基本公式进行求解 2.公式中所有的计算符号，如“+”、“-”、“×”、“/”和“^”不能省略 3.公式中的每个变量需要使用"[]"标注出来，而且变量需要使用有意义的文字标识，尽量避免直接使用数值 4.如果有latex格式的公式，比如“\frac{Q_吸}{Q_放}”需要改成正常算式的形式：“[Q_吸] / [Q_放]”，算式中的单位换算部分不属于公式，不需要被提取 5.参数表中的参数是公式中使用到的参数（参数名称要与公式中的参数一致），表格包括：概念、符号、数值、单位,使用"|"分割单元格 6.参数表中的数值和单位来自于问题本身以及解析计算的中间结果，如果参数进行了单位换算，参数表要给出原始的参数形式（没有进行单位换算） 7.参数表中的参数如果没有单位或者是表示某个物体的数量，比如“3盏灯”、“转了8圈”、“4个人”等等，那么单位写入“null” 8.提取结果中给出的公式和参数表中不需要任何额外的解释信息 下面是一些例子： input: 问题：在某次测试中该车匀速行驶时受到的阻力是车重的 0.02 倍，在 0.5h 内行驶了 28km 的路程，求：行驶的平均速度为多少？ 解析：解： 原始解析比较清晰，下面将其转换为规范格式： 第一步，先计算平均速度: v=s/t [平均速度]=[路程]/[时间] 算式=28km/0.5h=56km/h 平均速度=56km/h 答案=56km/h output: 提取分析： 1.解析中的公式格式符合要求，计算符号没有省略，变量使用"[]"标注，而且变量使用有意义的文字标识 2.计算过程中没有进行单位换算，参数可以直接提取 下面给出提取结果（单位换算部分不属于公式，不需要被提取；参数表中不需要额外解释信息） 提取结果： 1.先计算平均速度，根据路程和时间的关系： [平均速度]=[路程]/[时间] 参数表: 概念| 符号| 数值| 单位 路程| s| 28| km 时间| t| 0.5| h 平均速度| v| 56| km/h ***...omitted...*** 下面是一个新的问题： 问题：**{{question}}** 解析：**{{explanation}}**	I need you to help me extract the formula and parameters table for the formula from given question and explanation: 1. Think step by step, analyze the explanation first, and then generate the extracted results. If there is a combination of formulas, the combination needs to be split into basic formulas step by step. 2. All calculation symbols such as "+", "-", "×", "/" and "^" in the formula cannot be omitted. 3. Each variable in the formula needs to be labeled with "[]", and the variable needs to be identified with meaningful text instead of numbers. 4. If a latex formula such as \frac{Q_in}{Q_out} needs to be changed to a normal formula: [Q_in]/[Q_out]. The unit conversion does not need to be extracted. 5. The parameters table come from the parameters in formula (the parameter name should be consistent with the parameters in the formula), the table include: concept, symbol, numeric, unit, using cell division “|”. 6. The numeric and unit in the parameter table come from the problem itself and the intermediate results of analytical calculation. If the parameters are converted into different units, the parameter table should give the original parameter form (without unit conversion). 7. If the parameter in the parameters table has no units or represents the amount of an object, such as "3 lights", "8 revolutions", "4 people", etc., then the units are written as "null". 8. Apart from formula and parameter table, no additional information is required in the extraction results. Here are some examples: input: Question: In a test, the resistance of the car at a constant speed is 0.02 times the weight of the car, and the car traveled a distance of 28km in 0.5h. What is the average speed? Explanation: 1.we compute the average speed: v=s/t [Average speed]=[distance]/[time] expression=28km/0.5h=56km/h Average speed =56km/h Answer=56km/h output: Extraction analysis: 1. The formula format in the analysis meets the requirements. The calculation symbols are not omitted. Variables are labeled with "[]", and variables are expressed with meaningful text. 2. No unit conversion was performed during the computation, and parameters can be directly extracted. Below is the extraction result (the unit conversion part does not belong to the formula and does not need to be extracted; no additional explanatory information is required in the parameter table). Extraction result: 1. First calculate the average speed, based on the relationship between distance and time: [average speed]=[distance]/[time] Parameter table: Concept | Symbol | Numeric | Unit distance | s | 28 | km time | t | 0.5 | h average speed | v | 56 | km/h ***...omitted...*** There’s a new question: Question: **{{question}}** Explanation: **{{explanation}}**

Figure 3: Prompt for parameter extraction.

Prompt actually used	English translation
我需要你帮助我纠正解析中的错误，我会给出问题和错误信息，下面是错误纠正的要求： 1.你需要先进行错误分析，分析如何修改来纠正错误，然后给出错误纠正部分，纠正解析中的错误 2.错误纠正部分不需要任何额外解释信息，错误纠正部分的格式为:"内容：修改前的内容->修改后的内容"，增加内容时"修改前的内容"为null，删除内容时"修改后的内容"为null 3.问题缺失参数：如果问题中没有缺失的参数，那么向题目中增加缺失的参数;如果问题中的参数与缺失参数的含义相同但格式不同，修改题目中的参数与缺失参数相同 4.算式错误：算式存在错误需要对公式和错误的参数进行修改，如果算式中存在“[参数]”或“null”，需要补齐缺失的参数；如果参数没有问题可能需要对公式进行修改 5.公式的格式为“[待求解参数]=[参数1](+|-|×|/)[参数2]...”；参数表的格式为:"概念|符号|数值|单位"，比如"水的沸点是100℃"，表示为"水的沸点| t_沸| 100| ℃" 下面是一些例子： input: 问题：假设13.0t烟煤在煤炉中完全燃烧，放出的热量部分被水吸收，可以使4×10^5kg的水从20℃升高到100℃，求水吸收的热量为多少J [c_水=4.2×10^3J /（kg·℃）] 错误信息： 算式错误: 1.计算水升高的温度差: 公式: [水升高的温度差]=[末温]-[初温] 算式=[末温]-[初温] 问题缺失参数: 水升高的温度差=80 ℃; output: 错误分析： 1.根据错误信息：算式存在错误，而且算式中存在"[参数]"的情况："[末温]"、"[初温]"，需要对参数表增加缺失的参数 根据错误错误信息，"[末温]-[初温]"，从题目中可以找到相关文本"从20℃升高到100℃"，按照要求的参数格式表示为： 初温| t_0| 20| ℃ 末温| t| 100| ℃ 这样参数表增加缺失的参数后，代入1. 计算水升高的温度差的公式可以得到： 算式=((100) ℃)-((20) ℃)=80 ℃ 水升高的温度差=80 ℃ 2.根据错误信息，问题缺失参数，由于分析1中纠正算式后计算得到了"水升高的温度差=80 ℃"，所以问题不再缺失参数，不需要进行修改 错误纠正： 参数表：null->初温| t_0| 20| ℃ 参数表：null->末温| t| 100| ℃ ***...omitted...*** 下面是一个新的问题： 问题：**{{question}}** 错误：**{{error}}**	I need your help to correct the error in the explanation. I will provide the question and error information. The following are the requirements for error correction: 1. You need to first conduct error analysis, analyze how to modify to correct the error, and then provide the error correction to correct the error in the explanation. 2. The error correction section does not require any additional explanatory information. The format of the error correction section is: "Content: Pre modified Content ->Modified Content". When adding content, "Pre modified Content" is null, and when deleting content, "Modified Content" is null. 3. Missing parameters in the question: If there are no missing parameters in the question, add the missing parameters to the question; If the parameters in the question have the same meaning as the missing parameters but different formats, modify the parameters in the question to be the same as the missing parameters. 4. Expression error: The formula and incorrect parameters need to be modified. If there is "[parameter]" or "null" in the expression, the missing parameters need to be filled in; If there are no issues with the parameters, it may be necessary to modify the formula. 5. The format of the formula is "[parameter to be solved]=[parameter 1] (+|-| ×|/) [Parameter 2]… "; The format of the parameter table is: “concept | symbol | numeric | unit", for example, "The boiling point of water is 100 ℃", which is represented as “the boiling point of water | t_boiling | 100 | ℃" Here are some examples: input: question: Assuming that 13.0 tons of bituminous coal is completely burned in a coal furnace, the heat released is partially absorbed by water, which can make 4×10^5kg of water is raised from 20 ℃ to 100 ℃, how much heat is absorbed by the water ? [c_water=4.2×10^3J/(kg · ℃)] Error message: Expression error: 1. Calculate the temperature difference of water rise: Formula: [temperature difference of water rise]=[final temperature] - [initial temperature] expression=[final temperature] - [initial temperature] Question missing parameter: Temperature difference of water rise=80 ℃; output: Error analysis: 1. According to the error message: there is an error in the formula, and there are cases of "[parameter]" in the formula: "[final temperature]" and "[initial temperature]". It is necessary to add them to the parameter table. According to the error message, "[Last Temperature] - [Initial Temperature]", the relevant text "Increase from 20 ℃ to 100 ℃" can be found in the title. According to the required parameter format, it is represented as: Initial temperature | t_ 0 | 20 | ℃ Final temperature | t | 100 | ℃ After adding missing parameters to the parameter table, substitute them into the formula for calculating the temperature difference can be obtained as follows: expression=((100) ℃)-((20) ℃)=80 ℃ temperature difference of water rise=80 ℃ 2. According to the error message, the question is missing parameters. After analyzing the correction equation in 1 step, it was calculated that "the temperature difference of water rise=80 ℃", so the question is no longer missing parameters and does not need to be modified. Error correction: Parameter table: null ->Initial temperature | t_ 0 | 20 | ℃ Parameter table: null ->final temperature | t | 100 | ℃ ***...omitted...*** There’s a new question: Question: **{{question}}** Error: **{{error}}**

Figure 4: Prompt for correcting calculation errors.

Question:
As shown in the figure, the Xuelong 2 scientific research icebreaker designed in China. ***...omitted...*** When traveling at a constant speed of 3.6km/h in thick ice covered waters, the resistance experienced by the icebreaker is approximately 2×10°N. Calculate the propulsion power of the icebreaker at this time.
Reference answer: 2×10^7 W

Formula:
[thrust]=[resistance]
[propulsion power]=[thrust]×[constant speed]

Parameter table:

Parameter	symbol	value	unit
resistance	f	2×10^7	N
ship speed	v	1	m/s

Explanation:
1.Calculate thrust:
thrust=resistance=2×10^7N
2.Calculate propulsion power:
propulsion power=thrust×constant speed=2×10^7N×constant speed*(cannot find value)*

Error:
1. *The parameter "resistance" in the question is in the incorrect format.*
2. *"constant speed" could not be located in the parameter table.*

Figure 5: An example of removed question.

Prompt actually used	English translation
下面我会给出两个公式，每个公式由参数和运算符号构成，[]中的表示参数。 你需要判断我给出的两个公式中对应参数表达含义是否相同，是否是同一个公式： 如果含义不相同，不是同一个公式，只需要回答不是； 如果各个参数含义相同，是同一个公式，则需要给出最终的公式，并且给出一个三行的表格来表示参数的对应关系，每个单元格内容是一个参数，前两行填写两个公式的参数，第三行填写统一后的公式参数。 下面是公式1： {**公式 1**} 下面是公式2： {**公式 2**} 通过表达含义判断，是否是同一个公式：	I will give two formulas below. Each formula consists of parameters and operation symbols. The text in [] represent parameter. You need to judge whether the corresponding parameters in the two formulas I gave have the same meaning and whether they are the same formula: If the meaning is different, and they are not the same formula, just answer no; If each pair of parameters have the same meaning, and they are the same formula, the final formula needs to be given, and a three-row table needs to be given to indicate the corresponding relationship between the parameters. The content of each cell is a parameter, and the first two rows are filled with two formulas. Parameters, fill in the unified formula parameters in the third row. Here is formula 1: {**formula 1**} Here is formula 2: {**formula 2**} Judge whether they are the same formula by their meanings:

Figure 6: Prompt for semantics based merging.

The result of the expression is 720.

Step 3: Auxiliary Question Generation We then generated an independent physics question whose answer corresponds to the calculated result (720). We randomly assigned a unit (e.g., Newtons) to this value. Using GPT-4o, we generated the following auxiliary question:

> **Auxiliary Question:** Ming is conducting a physics experiment. He needs to use a crane to lift a mass of 80 kg vertically upwards at a constant speed. Given that the local gravitational acceleration is 9.0 m/s^2, what force does the crane need to exert to make the mass rise at a constant speed?

The answer to this auxiliary question is $F = mg = 80 \times 9.0 = 720$ N, which matches our target value.

Step 4: Final Compilation Finally, we assembled a new hybrid question. We injected a distractor parameter (e.g., "calorific value of gas") to further increase difficulty. The final question requires the model to solve the auxiliary question to find the value of the expression (i.e., 720 here), and then solve a system of equations in X and Y

Prompt actually used	English translation
你是一个物理问题的英文翻译专家，我需要你将中文的问题和对应的参数翻译为英文形式，不需要给出任何解释信息，下面是一些例子： Input: 问题: {{**样例1中文问题**}} 参数: {{**样例1中文参数**}} Output: Question: {{**样例1英文问题**}} Parameters: {{**样例1英文参数**}} ***…omitted…*** 下面是一个新的问题： Question: {{**问题**}} Parameters: {{**参数**}}	You are an English translation expert for physics problems. I need you to translate the Chinese questions and corresponding parameters into English. No extra information is required. Below are some examples: Input: Question: {{**Chinese question of example 1**}} Parameters: {{**Chinese parameters of example 1**}} Output: Question: {{**English question of example 1**}} Parameters: {{**English parameters of example 1**}} ***…omitted…*** Below is a new question: Question: {{**question**}} Parameters: {{**Parameters**}}

Prompt actually used	English translation
你是一个物理问题的英文翻译专家，我需要你将中文的内容翻译为英文形式，不需要给出任何解释信息，下面是具体的要求： 1.给出的英文形式的内容中应该只有英文，不应出现中文，翻译时不要遗漏内容中的任何信息 2.不需要给出其他任何解释信息 以下是你要翻译的内容：{{**公式**}}	You are an English translation expert for physics problems, and I need you to translate the Chinese content into English. No extra information is required. Here are the specific requirements: 1. The English content provided should only contain English, without any Chinese characters. Do not omit any information from the content during translation. 2. No other extra information is required. Here is the content you need to translate: {{**formula**}}

Figure 7: Prompts for translating questions (top) and formulas (bottom).

where one equation represents their relation in the original question and the other is, in this example, $720 = E(X, Y)$.

Final Question: A block of aluminum with a mass of 50 g releases $[X]$ amount of heat and its temperature decreases to 12°C. What was the original temperature of the aluminum block? [$C_{aluminum}$ = 880 J/(kg·°C)] The answer to the above question is $[Y]$. Under the International System of Units (with the exception that temperature is expressed in degrees Celsius), the value of the numerical expression $50Y - X$ of the physical quantities corresponding to $[X]$ and $[Y]$, is the same as the result of the following independent question: Ming is conducting a physics experiment. He needs to use a crane to lift a mass of 80 kg vertically upwards at a constant speed. Given that the local gravitational acceleration is 9.0 m/s^2, what force does the crane need to exert to make the mass rise at a constant speed?

A.3 Appendix for Experimental Setup

A.3.1 Evaluated LLMs

We conducted experiments across a diverse spectrum of models including cost friendly small-scale models including ERNIE-4.5-21B-A3B (Baidu-ERNIE-Team, 2025), Gemma-3n-E4B-it (Gemma, 2025), GPT-oss-20B (Agarwal et al., 2025), alongside high-cost LLMs such as GPT-4o (OpenAI, 2024), GLM-4-plus (GLM et al., 2024), Qwen3-235B-A22B (Qwen3-235B-A22B-Thinking-2507) (Yang et al., 2025), DeepSeek-R1 (DeepSeek-AI et al., 2025), o3-mini (OpenAI, 2025b) and GPT-5 (OpenAI, 2025a).

A.3.2 Prompt for Evaluated LLMs

The prompt is provided in Figure 8.

A.3.3 Formula Retriever

Implementation Let the number of formulas in the formula database be N. During training, we randomly initialized a matrix $\mathbf{F} \in \mathbb{R}^{N \times d}$, where d is the hidden size and the i-th row in $\mathbf{F}$ represented the initial representation of the i-th formula in formula database. We denoted a batch of questions with a batch size of B as $Q = \{q_1, q_2, ..., q_B\}$. The indices of the gold-standard formulas corresponding to these B questions were denoted as $L = \{l_1, l_2, \cdots, l_B\}$ (i.e. the label of q_i is l_i,

Prompt actually used	English translation
这是一个初中物理题目，根据问题给出计算的过程，让我们一步一步地地思考，在最后用“###”作为开始给出最终答案（一个数字）和答案的单位。 Question: {{**问题**}} Answer:	This is a junior high school physics question. Based on the given question, provide the calculation process and let's think step by step. Finally, use "###" to start giving the final answer (a number) and the unit of the answer. Question: {{**question**}} Answer:

Figure 8: Prompt for evaluated LLMs.

where $1 \leq i \leq B$).

BERT was utilized to encode each question,

$$\mathbf{h}_{cls}^{i}, \mathbf{h}_{1}^{i}, \cdots = \texttt{BERT}(q_i), 1 \leq i \leq B. \quad (1)$$

Subsequently, we took the CLS vector $\mathbf{h}_{cls}^{i}$ as the representation for the i-th question.

We utilized in-batch negatives and contrastive learning loss,

$$\mathcal{L} = -\frac{1}{B} \sum_{1 \leq i \leq B} \log \frac{\exp(\cos(\mathbf{h}_{cls}^{i}, \mathbf{F}_{l_i}))}{\sum_{1 \leq j \leq B} \exp(\cos(\mathbf{h}_{cls}^{i}, \mathbf{F}_{l_j}))}. \quad (2)$$

Each question might correspond to multiple correct formulas, and we ensured that the same question did not appear twice in the same batch when loading the data. Based on the implementation of Chinese-BERT-wwm-base, we tested the retrieval performance on the HoF test set and found that Recall@5 reached 82.46%.

Prompt for Evaluated LLMs with Formula Retriever For each question, we included the top-5 retrieved formulas in the prompt provided in Figure 9.

A.3.4 Data Augmentation for Supervised Fine-Tuned Models

Several studies (Ding et al., 2024; Zheng et al., 2023; Whitehouse et al., 2023) use LLM for data augmentation, which mainly focus on daily conversations or sentiment analysis and do not require rigorous numerical calculations. Some research (Shum et al., 2023) on data augmentation that involves numerical calculations employs LLM to generate solutions for existing questions to aid training, rather than to generate new questions. In contrast to these approaches, we generated complete questions that involve numerical calculations (particularly focusing on formulas), along with automatic improvement and selection to ensure data quality.

We divided the data generation process into the following steps. First, we randomly generated 17,000 prompts. Each prompt was obtained by stacking five question-explanation pairs sampled from the training set. At the end of the prompt, the LLM was required to generate the sixth question-explanation pair. Second, we normalized the formulas generated. Except for the absence of manual review, the remaining steps were consistent with those in Section 2.2. Finally, we utilized Numbat to check whether the calculation process in the data generated by the LLM was correct, and discarded the data with incorrect calculations. After the above steps, we finally retained about 2,500 questions.

We found that mixing the newly generated data into the original training set did not always bring positive improvement, perhaps because the newly generated data did not undergo manual review. We found that randomly selecting a small portion of the newly generated data could improve the performance of the model. We set the mixing ratio to 10% which is an experiential proportion. We fine-tuned each model using the augmented dataset. After training for a fixed number of epochs (50 epochs), we selected the final checkpoints as DA models.

A.3.5 Preference Data Generation

We applied the Monte Carlo Tree Search (MCTS) method to the unaltered questions in the training set of FormulaReasoning to generate preference data for DPO. It used the reasoning steps as nodes, and used the generation model to expand nodes. Each node had one more reasoning step than its parent node. The root node had no reasoning step, only containing the original question. The termination state was reached when the termination token was generated or the current question was solved. We used the PRM to guide the generation. The PRM score was used as a reward to update the nodes.

After generation, a tree was created for each sample. Each node contained the reasoning steps generated up to that node and the correctness score of the generated steps. We identified the best and worst paths in the tree. Each path was scored by the

Prompt actually used	English translation
这是一个初中物理题目，根据问题给出计算的过程，用公式表示。	This is a junior high school physics question. Based on the given question, provide the calculation process.
可能用到的公式有: {{**top 5检索到的公式**}} Question: {{**问题**}} Answer:	The formulas that may be used include: {{**top 5 retrieved formulas**}} Question: {{**question**}} Answer:

Figure 9: Prompt for evaluated LLMs with formula retriever.

average score from the root node to the terminated leaf node. Limited by the number of iterations, some leaf nodes might not represent termination and their corresponding paths were not used.

To consider the inclusion of formula knowledge, we further screened the generated data. For each sample, we examined its best and worst paths. If there was no significant formula knowledge on these paths, we would discard this sample. We used a rule-based pattern matching method to identify formula knowledge. This pattern was consistent with the example given in the generation prompt used in MCTS, which was an expression consisting of "[{parameter}]".

A.3.6 Implementation Details

We accessed DeepSeek-R1 through the DeepSeek API[5], Qwen series models through the Alibaba Bailian platform[6], GLM-4-plus through ZHIPU–AI API[7], and other models through OpenRouter[8] with their default hyper-parameters. We set temperature=0 for direct answer generation and set the maximum output length to 1,024. In the case of CoT-SFT, which directly outputted formulas along with numerical values and units of parameters, we set temperature=0 to obtain the response with the highest probability and set the maximum output length to 512 to obtain formulas and parameters. Training Qwen2.5-Math-7B-Instruct(lora) and Qwen2.5-Math-1.5B-Instruct(full) in CoT-SFT used 3 and 8 hours, respectively.

A.4 Appendix for Experimental Results

A.4.1 English Version of FormulaReasoning

The evaluation results for LLMs with CoT prompts on the English FormulaReasoning dataset are presented in Table 7. Large-scale models (e.g., GPT-5 and DeepSeek-R1) achieved competitive results with accuracies exceeding 87% on both HoF and HeF metrics. However, substantial performance gaps existed for smaller-scale models, where architectures like Gemma-3n-E4B-it exhibited poor accuracies. We found that Gemma-3n-E4B-it and Qwen3-235B-A22B showed performance drops compared to their results on the Chinese version due to instruction-following issues in the English version, where these models failed to output required units as specified in the prompt.

Notably, despite their impressive capabilities, even the best model GPT-5 (88.63%) still trailed behind human performance (92.03% as reported in the main text), falling short by 3.4 percentage points. Other advanced models such as GPT-4o (75.38%) and GLM-4-plus (72.37%) showed even larger gaps with human performance.

There was a general trend of performance decline on FormulaReasoning+. Notable drops were seen in models like GPT-4o (from 75.38% to 13.48%), o3-mini (from 83.13% to 68.38%), and GPT-5 (from 88.63% to 70.51%). This confirmed that the the extended FormulaReasoning+ successfully introduced greater complexity, providing a more rigorous benchmark for evaluating the reasoning capabilities of LLMs.

This substantial performance disparity indicates that although scaling parameters leads to marked improvements, there is still significant room for optimizing resource-efficient models. Future research should prioritize architectural and training optimizations for compact models, which are essential for resource-constrained environments. The performance variation observed across models highlights the critical need for scale-aware evaluation in formula-based reasoning.

A.4.2 PoT Prompts

The results of GPT-4o using CoT compared to PoT are shown in Table 8. CoT consistently outperformed PoT. One possible reason was that the effectiveness of PoT could be influenced by a hybrid of the language and coding capabilities of LLMs, but that of CoT was only related to the

[5]https://api-docs.deepseek.com/
[6]https://bailian.console.aliyun.com/
[7]https://open.bigmodel.cn/dev/api/normal-model/glm-4
[8]https://openrouter.ai/

Table 7: Accuracy of LLMs with CoT prompts on English FormulaReasoning.

	FormulaReasoning			FormulaReasoning+		
Model	HoF	HeF	Avg.	HoF	HeF	Avg.
ERNIE-4.5-21B-A3B	63.20	56.85	60.13	7.39	9.26	8.33
Gemma-3n-E4B-it	2.66	2.84	2.75	2.71	2.65	2.68
GPT-oss-20B	86.20	84.75	85.48	53.45	57.67	55.56
GPT-4o	75.79	74.94	75.38	14.53	12.43	13.48
GLM-4-plus	75.79	68.73	72.37	17.73	20.11	18.92
Qwen3-235B-A22B	62.47	66.67	64.50	77.34	77.51	77.43
DeepSeek-R1	87.17	87.60	87.38	76.85	79.37	78.11
o3-mini	84.02	82.17	83.13	68.23	68.52	68.38
GPT-5	88.38	88.89	88.63	71.43	69.58	70.51

language capabilities. Moreover, for LLMs with strong calculation capabilities, the advantage of using Python to perform numerical calculations might be eroded. The observed underperformance of PoT compared to CoT appears to also stem from the models' faced challenge in accurate application of formulas within a programmatic context. Therefore, on FormulaReasoning, the bottleneck for current LLMs was mainly in their ability to apply formulas, rather than in calculation.

Table 8: Accuracy of LLMs with CoT and PoT prompts.

Model	HoF test	HeF test	Avg.
GPT-4o (CoT)	69.49	65.37	67.50
GPT-4o (PoT)	54.72	41.34	48.25

A.4.3 PRM Scores

Tables 9 and 10 show the PRM scores given by Qwen2.5-Math-PRM-7B in one-step and multi-step settings, respectively.

Additionally, we analyzed whether PRM scores can distinguish correct from incorrect answers across different model families to explore potential model-family bias. For GPT-4o (non-Qwen), PRM scores for correct answers (mean=0.90) were significantly higher than incorrect answers (mean=0.73). For Qwen3-235B-A22B (Qwen-family), PRM scores showed smaller separation (correct: mean=0.82 vs incorrect: mean=0.77). If the PRM favored the Qwen-family models, we would expect the opposite pattern. This suggests the PRM does not exhibit systematic bias toward its own model family.

We acknowledge that PRM scores alone cannot fully capture reasoning quality. Therefore, we primarily relied on Accuracy as the primary metric, and used PRM scores as a supplementary indicator for step-level analysis rather than definitive judgments.

Table 9: PRM score (one-step) of LLMs with CoT prompts.

Model	HoF test	HeF test	Avg.
ERNIE-4.5-21B-A3B	0.7753	0.7582	0.7668
Gemma-3n-E4B-it	0.3695	0.3660	0.3678
GPT-oss-20B	0.5704	0.5203	0.5454
GPT-4o	0.6150	0.5583	0.5867
GLM-4-plus	0.6728	0.6591	0.6660
Qwen3-235B-A22B	0.6502	0.6476	0.6489
DeepSeek-R1	0.7395	0.7393	0.7394
o3-mini	0.7537	0.7556	0.7547
GPT-5	0.8196	0.8249	0.8223

Table 10: PRM score (multi-step) of LLMs with CoT prompts.

Model	HoF test	HeF test	Avg.
ERNIE-4.5-21B-A3B	0.8695	0.8711	0.8703
Gemma-3n-E4B-it	0.7366	0.7360	0.7363
GPT-oss-20B	0.5738	0.5266	0.5502
GPT-4o	0.8655	0.8444	0.8550
GLM-4-plus	0.8726	0.8738	0.8732
Qwen3-235B-A22B	0.7946	0.7996	0.7971
DeepSeek-R1	0.7603	0.7535	0.7569
o3-mini	0.8104	0.8032	0.8068
GPT-5	0.8196	0.8248	0.8222

A.4.4 DPO

We employed the preference data derived from the training set of FormulaReasoning to perform DPO with several small models on not only the test sets of FormulaReasoning but also other numerical reasoning datasets. Table 11 shows the results. DPO improved accuracy on FormulaReasoning and also improved accuracy in most cases on other datasets, showing an additional value of the preference data derived from our dataset. Notably, although these models exhibited good performance on other datasets, their low accuracy on FormulaReasoning indicated that they did not truly master formula-based reasoning required in the physics domain, characterizing FormulaReasoning as a valuable dataset presenting unique challenges for LLMs.

A.5 Error Analysis

FormulaReasoning posed challenges to existing models in terms of formula selection and numerical calculation, including unit calculation and arithmetic calculation. As illustrated in Figure 10a, the incorrect use of formulas was the main type of error. Another type of error was parameter noise, as illustrated in Figure 10b. In terms of the performance of the models, we found that the models

Table 11: Accuracy of DPO on FormulaReasoning and other datasets.

Model	GSM8K	MATH	SVAMP	Gaokao2023-en	FormulaReasoning
DeepSeek-R1-Distill-Qwen-7B	87.7	58.8	87.7	55.3	34.0
+DPO	85.8 −1.9	59.7 +0.9	88.1 +0.4	55.8 +0.5	37.0 +3.0
DeepSeek-R1-Distill-Qwen-1.5B	79.8	68.2	85.7	55.3	7.23
+DPO	81.3 +1.5	68.0 −0.2	86.3 +0.6	55.1 −0.2	10.47 +3.24
Qwen2.5-Math-1.5B-Instruct	85.6	76.1	93.3	66.0	4.94
+DPO	85.8 +0.2	76.5 +0.4	93.7 +0.4	65.2 −0.8	7.52 +2.58
Qwen2.5-1.5B-Instruct	72.9	55.0	85.5	47.5	5.94
+DPO	73.1 +0.2	56.2 +1.2	86.6 +1.1	47.8 +0.3	6.57 +0.63

that achieved higher accuracy were less affected by the parameter noise injected into the questions (Section 2.4). For example, among the questions to which DeepSeek-R1 did not provide a correct answer, approximately 14% were influenced by the parameter noise. This indicated that our question alteration imposed further requirements on the model's ability to correctly apply formulas and perform accurate reasoning.

Question

There is a hand-held roller brush snow blower with a mass of 150 kg (g = 10 N/kg). Its traveling power is 200 W when not clearing snow and 3 kW when clearing snow. The maximum traveling speed is 1 m/s. When the snow blower travels at the maximum speed to clear snow for 1 km, it consumes 0.3 kg of fuel. What is the heat released by the snow blower? (Supplement: The calorific value of the fuel is 4×10^7 J/kg; The work done by the snow blower is [X];) The contact area between the snow blower and the ground is 0.1 m^2. If we already know that the answer to the above question is 9×10^6 J, what should the content of [X] be?

Output of Qwen-Max (Incorrect use of physical formulas)

Solution: 1. Calculate the total heat released by the fuel [Total Q] = [fuel quality m] x [Calorific value] Calculation = 0.3 x 4 x 10^7 = 1.2x10^7 J.
2. Calculate the work done by the snow blower: [Total work W] = [Release heat Q] = 9×10^6 J. … *omitted* …

Gold Explanation (Correct use of physical formulas)

Solution: 1.Calculate the time required for the snow removal process: [Total time T] = [Total Distance S] / [Speed V] Calculation = 1000 m / 1 m/s = 1000 s.
2. Calculate the total work done by the snow blower: [Total work W] = [Total power during snow removal P] x [Total time T] = 3000 W x 1000 s = 3x10^6 J.
3. Checkout: The total energy provided by fuel: [Total Q] = 0.3 x 4 x 10^7 = 1.2x10^7 J; [Total Q] = [Total work W] + [Release Q] … *omitted* …

(a) An error case caused by formula error.

Question

Given that the heat of fusion of ice is [X], and its melting point is 0℃. The specific heat capacity of ice is 2.1×10^3 J/(kg•℃). How much heat must 1 kg of ice at -1℃ absorb to completely melt into water at 0℃? The value of the parameter electrical conversion efficiency is 21%. If we already know that the answer to the above question is 3.3621×10^6 J, what should the content corresponding to [X] be in the question?

Output of Qwen-Max (Parameter noise highlighted in red)

Solution: 1. Calculate the actual absorbed heat (Total Q): [Total Q] = [Input Q] x [Electrical conversion efficiency] Calculation = 3.3621 × 10^6 × 0.21 = 7.06041 × 10^5 J.
2. Calculate the heat of melting: [Melting Q] = [Total Q] – [Rising Q] Calculation = 7.06041×10^5 −2.1×10^3 = 7.05831×10^5 J. … *omitted* …

Gold Explanation (Correct without using parameter noise)

Solution: 1. The total absorbed heat is: [Total Q] = 3.3621×10 6 J.
2. Calculate the heat of melting: [Melting Q] = [Total Q] – [Rising Q] Calculation = 3.3621×10^6 −2.1×10^3 ≈ 3.36×10^6 J.
3. Calculate the heat of melting X: … *omitted* …

(b) An error case caused by parameter noise.

Figure 10: Examples of errors.