EngiBench: A Benchmark for Evaluating Large Language Models on Engineering Problem Solving

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models (LLMs) have shown strong performance on mathematical reasoning under well-posed conditions. However, real-world engineering problems require more than mathematical symbolic computation—they need to deal with uncertainty, context, and open-ended scenarios. Existing benchmarks fail to capture these complexities.We introduce EngiBench, a hierarchical benchmark designed to evaluate LLMs on solving engineering problems. It spans three levels of increasing difficulty (foundational knowledge retrieval, multi-step contextual reasoning, and open-ended modeling) and covers diverse engineering subfields. To facilitate a deeper understanding of model performance, we systematically rewrite each problem into three controlled variants (perturbed, knowledge-enhanced, and math abstraction), enabling us to separately evaluate the model's robustness, domain-specific knowledge, and mathematical reasoning abilities. Experiment results reveal a clear performance gap across levels: models struggle more as tasks get harder, perform worse when problems are slightly changed, and fall far behind human experts on the high-level engineering tasks. These findings reveal that current LLMs still lack the high-level reasoning needed for real-world engineering, highlighting the need for future models with deeper and more reliable problem-solving capabilities. Our source code and data are available at https://github.com/EngiBench/EngiBench.

1 Introduction

Large language models (LLMs) have demonstrated promising capabilities in a range of mathematical reasoning tasks, from foundational skills such as basic computation and structured problem-solving [10], multi-step reasoning [38, 47], to more complex applications like mathematical modeling [19] and the generation or verification of mathematical proofs [49, 27, 36]. However, just using mathematical reasoning is not enough for real-world applications. In practice, many high-impact use cases occur not in abstract mathematical domains, but in engineering contexts, where problems are grounded in physical systems and require balancing uncertainty and constraints inherent to real-world decision making. These characteristics require not only mathematical computation, but also need broader capabilities to understand engineering contexts and solve complex engineering problems.

Engineering problems differ fundamentally from mathematical problems. Mathematical problems aim for abstract theoretical rigor and universality, and are typically characterized by complete information within a clearly defined problem space [20]. In contrast, engineering problems are driven by the need to find "good enough" and feasible solutions for specific objectives, which are often open-ended, highly context-dependent, and must be achieved within real-world constraints [13]. As illustrated in Figure 1, solving real-world engineering problems requires more than retrieving a formula or executing a single calculation. It involves a sequence of interdependent cognitive steps that span from understanding the problem context to formulating robust, feasible solutions. We define this broader

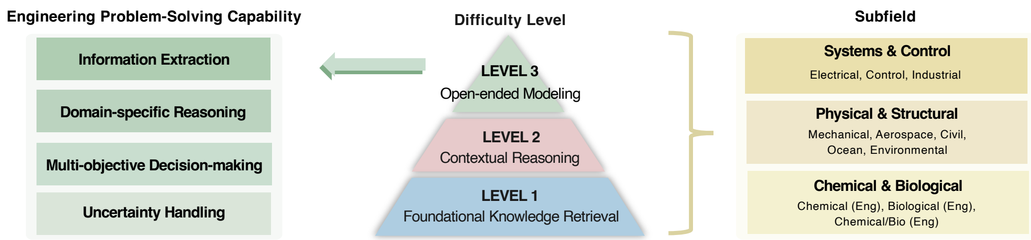


Figure 1: Task taxonomy of EngiBench organized by difficulty, capability, and subfield. Problems are grouped into three difficulty levels, with Level 3 specifically designed to evaluate engineering problem-solving capabilities. All tasks are additionally categorised into three major engineering subfields.

set of competencies as the engineering problem-solving capability, comprising four interconnected dimensions: *information extraction, domain-specific reasoning, multi-objective decision-making, and uncertainty handling*.

Despite the broader requirements of real-world engineering tasks, most existing benchmarks focus narrowly on well-posed mathematical problems. Benchmarks such as GSM8K [10], MATH [20], and Omni-MATH [14] primarily assess symbolic reasoning, calculation, and formal problem-solving under clean and fully specified conditions. While these benchmarks have driven progress in mathematical reasoning, their support for engineering tasks remains unclear. Although some include basic engineering questions, they fail to capture the deeper reasoning required for real-world problem solving [20, 46, 2, 12]. In addition, these benchmarks rely on publicly available datasets without rewriting that may overlap with LLM pretraining corpora, raising concerns about benchmark contamination and overclaimed performance [11, 21, 37]. For example, GSM1k introduces human-written problems in the style of GSM8k to avoid data overlap, revealing up to 8% performance drops and potential overfitting [51]. Without proper safeguards, evaluations may measure memorization rather than true generalization, particularly in engineering contexts requiring practical reasoning. Solely using unmodified public questions thus inadequately assesses true engineering capabilities, limiting insights into real-world model performance.

In this work, we introduce **EngiBench** – an evaluation framework designed to systematically assess LLMs on engineering problem-solving. It spans a wide range of engineering subfields. Meanwhile, **EngiBench** is designed around the broader concept of *engineering problem-solving capability*, evaluating LLMs across multiple dimensions aligned with the demands of practical engineering contexts. As illustrated in Figure 1, it consists of three progressively task levels. We use a structured data construction strategy consisting of three key aspects. First, we systematically rewrite public questions through numerical and semantic perturbations to minimize overlap with pretraining datasets. Second, we introduce controlled problem variations, including knowledge-enhanced and math abstraction versions, to enable fine-grained analysis of model capabilities. Finally, we adopt rubric-based evaluation for open-ended tasks, using expert-designed scoring criteria to assess model performance across key engineering problem-solving capabilities. Together, these measures yield a diverse and high-quality dataset that supports rigorous and contamination-limited evaluation of LLMs' engineering problem-solving abilities.

Experiment results show that our benchmark reveals clear performance stratification across difficulty levels, with higher-level tasks exposing distinct capability gaps. In addition, our perturbed version induce performance drops, even in strong models, revealing that prior evaluations may overestimate true generalization. Most critically, current LLMs consistently underperform on Level 3 tasks involving open-ended, high-level engineering reasoning, falling well short of human expert performance. These results suggest that today's LLMs remain far from reliably solving real-world engineering problems, leaving substantial room for future research.

Our contributions can be summarized as follows: (1) We are among the first to systematically evaluate LLMs on real-world engineering problems; (2) We design a hierarchical benchmark with three difficulty levels and multiple problem variants, enabling fine-grained analysis of model reasoning capabilities and limitations; (3) Unlike prior benchmarks, our benchmark systematically evaluate LLM performance on open-ended engineering tasks; (4) We evaluate a broad set of mainstream LLMs, providing insights that can aid future model development and enhance engineering capabilities.

2 Related Works

LLMs for Engineering Problems. LLMs possess logical-reasoning skills, domain knowledge, and the capacity for multi-step inference that surpass earlier AI paradigms, making them promising tools for tackling complex challenges. Engineering centers on understanding complex problems, building mathematical models, and discovering feasible solutions, making it highly relevant to real-world challenges and a critical domain for evaluating advanced reasoning capabilities. Although LLMs are increasingly applied to simulation, modeling, and system design, their true proficiency in engineering problem solving remains unclear because current benchmarks are inadequate [45, 30, 41, 8]. Some general-purpose benchmarks – like MMLU [20], MMLU-Pro [46], BIG-Math [2], and SuperGPQA [12] – include a few engineering-flavoured questions, but these are mostly fact-recall multiple-choice items that ignore authentic engineering reasoning. Domain-specific benchmarks do exist, such as EEE-Bench [25], ElecBench [53], FEABench [34], TransportBench [40], and JEEBench [6]. However, these benchmarks typically focus on single disciplines and closed-ended tasks, providing limited support for evaluating open-ended and cross-disciplinary engineering reasoning. Moreover, none of these efforts are explicitly designed to evaluate key engineering problem-solving capabilities such as information extraction, domain-specific reasoning, multi-objective decision-making, and uncertainty handling. We introduce a multi-level engineering benchmark spanning multiple subfields that emphasizes not only closed-form tasks but also open-ended problems, enabling a more comprehensive evaluation of the essential skills needed for effective real-world engineering decision-making.

LLM for Mathematical Problems. A closely related area that has been extensively studied is mathematics. Because solving mathematic problems demands strong logical ability, multi-step reasoning, and symbolic manipulation, it has become a primary proving ground for evaluating LLMs. Early benchmarks focus on elementary problems [10, 20, 35, 3] and higher-level symbolic reasoning [20, 2]. Recent efforts like MiniF2F [52], UniMath [26], Omni-MATH [15], and MathVista [29] expand to theorem proving and multimodal tasks. MATH-Vision [44] improves coverage by introducing diverse topics and difficulty levels from real competitions, and SMART-840 [9] benchmarks model performance against human children across grades. While these benchmarks provide rigorous evaluations of mathematical competence, they do not capture engineering-specific reasoning such as modeling, decision-making under constraints, or domain-based assumptions. Our work builds on their methodological insights but shifts the focus toward real-world engineering tasks.

Evaluation Challenges. Evaluating the capability of LLMs to solve engineering problems is challenging due to the inherent complexity involved. Current evaluation methods for LLMs fall into four main categories: reference-based, task-oriented, preference-based, and rubric-based. The first two are effective for problems with clear ground truths or executable outputs – e.g., MathVista [29], CHAMP [31] (reference-based), and EEE-Bench [25], FEABench (task-oriented) [34]. However, the core capabilities of the engineering field we are discussing cannot be effectively evaluated by such closed-form problems. For open-ended tasks, preference-based methods such as MT-Bench-101 [7] use pairwise comparisons, but are often biased by model-specific generation patterns, limiting objectivity and real-world applicability. Rubric-based evaluations aim to improve transparency by scoring along multiple criteria, with general-purpose frameworks like Prometheus [24] focusing on abilities such as context retention and rephrasing.

3 Methodology

3.1 Engineering Problem-Solving Capability

Engineering problems typically require practical, context-aware solutions under real-world constraints [13], fundamentally differing from mathematical problems that emphasize clearly defined, closed-form problem spaces [20]. While both fields value abstraction and logical rigor, engineering problem-solving involves interconnected cognitive steps, from understanding problem context to formulating robust, feasible solutions (see Figure 1 and Table 1). We define this broader set of skills as *engineering problem-solving ability*, comprising four key dimensions: *information extraction, domain-specific reasoning, multi-objective decision-making, and uncertainty handling*.

- *Information extraction* refers to the ability to identify and retrieve critical information from complex or redundant problem descriptions. It involves recognizing relevant variables, constraints, and objectives while distinguishing them from irrelevant or distracting details. This capability reflects the model's proficiency in processing unstructured inputs and converting them into structured representations that facilitate subsequent reasoning. Its significance lies in its capacity to accurately

Table 1: Hierarchical difficulty from mathematics to real-world engineering. This illustrates three levels of increasing complexity. Examples show the progression from closed-form math problems to open-ended engineering scenarios.

Level	Definition	Example
Mathematics	Mathematical tasks are typically well-posed and self-contained, with complete information and clearly defined solution spaces.	A machine produces 45 parts per minute. If it operates continuously for 2 hours, how many parts will it produce in total? *This task requires only basic multiplication and does not involve any domain knowledge. It represents a typical closed-form numerical computation problem.*
Upgrading Condition: Incorporating domain-specific engineering knowledge		
Engineering Level 1: Foundational Knowledge Retrieval	Apply basic engineering concepts or formulas to structured problems via single-step computation.	A drone operates at a constant power of 200W for 30 minutes. Calculate the total energy consumption in joules. *This task requires applying the basic physical formula $E = P \times t$, with unit conversion from minutes to seconds. It tests the model's ability to retrieve and apply foundational engineering knowledge in a single-step calculation.*
Upgrading Condition: Multi-step reasoning and contextual integration		
Engineering Level 2: Contextual Reasoning	Perform multi-step reasoning under well-defined constraints by integrating conditions and domain knowledge.	A drone needs to fly 6 km. The first half is uphill, increasing power usage by 20%, while the second half is flat at 180W. The drone flies at 30 km/h and uses a battery rated at 8000mAh, 11.1V. Can the battery support the trip? *This task requires multi-step reasoning: estimating flight time, adjusting power consumption, and comparing with battery capacity.*
Upgrading Condition: Solving open-ended, under-specified problems		
Engineering Level 3: Open-ended Modeling	Solve open-ended, real-world problems through information extraction, trade-off reasoning, and uncertainty handling.	Design a drone system for urban delivery that balances multiple factors, including flight range, payload capacity, and cost control. Propose a feasible solution and justify your design decisions. *This is an open-ended problem with incomplete constraints and potentially conflicting objectives, requiring information extraction, trade-off analysis, and robustness under uncertainty.*
Information Extraction	Identify and extract relevant information from complex or redundant problem descriptions.	Identify critical variables—such as payload weight, wind speed, flight duration, and battery margin—from complex or verbose task descriptions.
Domain-specific Reasoning	Apply specialized engineering principles and structured knowledge to guide logical inference and solution formulation.	Apply specialized engineering knowledge—such as flight mechanics and battery discharge principles—to formulate models and perform technical analysis.
Multi-objective Decision-making	Make justified trade-offs between competing in the absence of a single optimal solution.	Justify trade-offs among competing objectives like range, cost, safety, and operational efficiency when no single optimal solution exists.
Uncertainty Handling	Ensure solution robustness by reasoning under incomplete, variable, or ambiguous real-world conditions.	Account for unpredictable factors such as weather, task variation, and battery aging, and design robust strategies (e.g., adding 20% battery reserve) to ensure reliable performance.

capture the essential elements of a problem, thereby minimizing errors in subsequent reasoning processes and ultimately enabling the precise formulation of effective and implementable solutions.

- *Domain-specific reasoning* refers to the model's ability to apply specialized engineering knowledge such as physical principles, empirical rules, and practical engineering conventions to interpret a given scenario and formulate appropriate solutions. This includes understanding when certain approximations are valid, recognizing implicit assumptions commonly made in specific domains, and selecting solution strategies that align with real-world engineering practices. Such reasoning requires both conceptual understanding and practical judgment, distinguishing engineering tasks from purely mathematical problem solving.
- *Multi-objective decision-making* denotes the capability to evaluate and balance competing objectives in situations where no single optimal solution exists. Engineering problems commonly involve trade-offs among factors such as cost, performance, and safety. This dimension reflects the model's ability to navigate such trade-off spaces and justify rational decisions within given constraints. Consequently, it is this inherent requirement for trade-offs that imparts engineering problems with their distinctive characteristics of multiplicity, openness, and flexibility compared to traditionally studied problem domains.
- *Uncertainty handling* characterizes the capability to reason under conditions of incomplete or variable information. Real-world engineering scenarios frequently involve missing data, noisy inputs, or dynamic conditions. This dimension evaluates whether a model can anticipate such uncertainties, incorporate safety margins or adaptive strategies, and consistently deliver robust and reliable solutions despite these challenges. Effectively managing uncertain and ambiguous information, including making informed assumptions or estimations, is thus a critical yet complex challenge that LLMs must address to successfully solve practical engineering problems.

3.2 Problem Hierarchical Difficulty Design

As discussed above, the capabilities involved in solving engineering problems are multifaceted and complex, making it challenging to evaluate them comprehensively through any single task. Each capability emphasizes distinct cognitive demands and cannot be adequately represented within a single hierarchical dimension. Without a clear taxonomy, it is difficult to pinpoint the specific skills in which a model may be deficient. To address this issue, we introduce a structured evaluation framework. Unlike previous benchmarks that merely aggregate tasks, our framework classifies tasks according to the core capabilities required by engineering scenarios. As illustrated in Table 1, engineering problem-solving spans three levels: foundational knowledge retrieval, contextual reasoning, and open-ended modeling. This hierarchy mirrors the cognitive progression in engineering problem-solving—from applying basic formulas to reasoning under uncertainty and conflicting objectives. EngiBench reflects this hierarchical organization through a three-level difficulty framework.

1. **Level 1:** This level focuses on foundational knowledge retrieval. Tasks at this stage are well-structured and self-contained, requiring the model to *directly* apply fundamental engineering formulas or principles. Such tasks typically involve a single computational step, minimal contextual reasoning, and emphasize factual recall and precise formula application. The purpose of this level is to evaluate whether the model possesses the essential knowledge base required for engineering problem-solving, as well as its ability to accurately retrieve relevant information and perform computations reliably.
2. **Level 2:** This level emphasizes contextual reasoning within explicitly defined scenarios. Tasks at this level extend beyond the direct application of formulas, requiring the model to interpret structured problem descriptions, recognize implicit relationships among variables, and reason through *multiple* computational or logical steps. Although these problems remain situated within clearly specified contexts and have unique correct answers, solving them demands a deeper conceptual understanding of engineering conditions and the capacity to integrate knowledge across multiple domains. Crucially, straightforward, single-step application of knowledge is no longer sufficient; instead, multi-step reasoning processes become necessary.
3. **Level 3:** This level targets open-ended modeling scenarios that mirror the complexity of real-world challenges. These problems are often under-specified, with implicit constraints, ambiguous inputs, and potentially conflicting objectives. Unlike tasks in the previous two levels, they typically lack well-defined solutions. Solving them requires advanced reasoning across four key capabilities: information extraction, domain-specific reasoning, multi-objective decision-making, and uncertainty handling. These dimensions are summarized in the lower portion of Table 1, which illustrates how each contributes to robust decision-making in open-ended settings.

3.3 Dataset Construction

We collect data from three primary sources: problems selected from existing public benchmarks, university educational materials, and modeling competitions. These problems reflect the intended hierarchy of difficulty described above and address the lack of open-ended engineering modeling problems with expert-defined evaluation criteria in existing datasets.

For Level 1 and Level 2, the questions primarily consist of structured questions sourced from public benchmarks and university educational resources. We found that valuable entry-level engineering problems were often buried within previously published benchmarks, which used these tasks merely to roughly evaluate some general capabilities of the models. We systematically extracted relevant tasks from established benchmarks, including MMLU [20], MATH [20], GSM8k [10], Orca-Math [33], HARP [50], Omni-MATH [15], Big-Math [2], and competition datasets such as cn_k12, Olympiads, AOPS forum, and AMC-AIME [22]. Additionally, we incorporated problems collected from public platforms and university educational materials authorized by instructors. These questions have standard answers that can be used to determine whether they are right or wrong. All data were processed through standardization, re-labeling, and validation through a consistent and rigorous pipeline (see Appendix).

For Level 3, EngiBench is the first systematic attempt to curate open-ended engineering problems with established expert scoring criteria. This level comprises 43 problems gathered exclusively from modeling competitions held between 2010 and 2024. Non-engineering and multimodal content were manually filtered out, and some visual elements were converted to text to enhance accessibility. Unlike Level 1 and Level 2 questions, Level 3 questions do not have well-defined standard answers, only detailed scoring criteria. This reflects the open-ended nature of Level 3 questions. The dataset construction was supervised by doctoral-level professionals with deep expertise in engineering and mathematical modeling to ensure technical rigor and domain alignment. Consequently, the resulting dataset provides a robust evaluation framework for advanced reasoning and decision-making under uncertain conditions, filling a critical gap in current benchmarking efforts.

All problems in EngiBench cover three main engineering subfields: Systems & Control with 112 problems, Physical & Structural with 120 problems, and Chemical & Biological 455 problems. This classification follows standard engineering disciplines and reflects differences in problem focus, required knowledge, and reasoning approach.

3.4 Controlled Problem Variations for Fine-grained Capability Analysis

Multiple factors may influence the performance of LLMs on individual problems. Potential reasons include insufficient domain-specific knowledge, errors in performing calculations, or difficulties in accurately interpreting the engineering context. Merely collecting problems to test overall accuracy

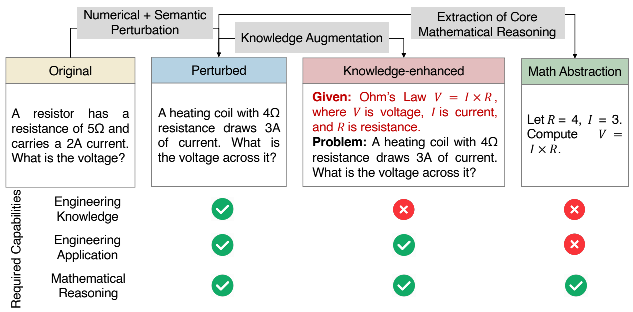


Figure 2: We create variants of the original problem to test different reasoning skills. *Perturbed* changes context and numbers to assess robustness. *Knowledge-enhanced* adds domain knowledge to focus on reasoning. *Math Abstraction* isolates engineering knowledge to test math ability. Each version targets specific capabilities.

provides only a broad indication of comprehensive performance. We propose to systematically rewrite problems to conduct controlled experiments, enabling more fine-grained analyses of LLM performance on our benchmark. This approach allows us to isolate particular challenges, evaluate the robustness of model, and detect potential data leakage issues [21, 51, 32, 39, 18]. By comparing model performance across different problem variations, we gain deeper insights into the specific capabilities required for realistic engineering tasks.

Motivated by this, we construct three distinct versions for each problem, each targeting a different aspect of the reasoning process, as illustrated in Figure 2. The detailed construction procedure is provided in the Appendix. Starting from an *original version* of each problem, we construct the following versions: (1) The *perturbed version* introduces numerical and semantic perturbations to the original problem, thereby reducing overlap with pretraining datasets. (2) The *knowledge-enhanced version*, built upon the perturbed version, explicitly provides relevant engineering knowledge, such as formulas, physical constants, and domain-specific definitions. This version helps to diagnose whether model errors stem specifically from a lack of critical knowledge. (3) The *math abstraction version* removes all contextual and domain-specific elements, reformulating the problem purely as a symbolic computation task. This isolates the model from the engineering context, reverting the evaluation to well-established mathematical reasoning and computational capabilities. Consequently, this version explicitly illustrates the impact of the engineering context on model performance.

These three variations, along with the original versions, are constructed systematically for all tasks in Level 1 and Level 2. For Level 3 tasks, however, the open-ended and inherent complexity typically render knowledge enhancement and mathematical abstraction impractical. Hence, only the original and perturbed versions are provided for this level. Moreover, we provide a detailed scoring criteria for Level 3, based on the official evaluation criteria disclosed by the competition organizers. This rubric enables assessment of an LLM's response to the specific requirements of each capability dimension.

4 Experiments

4.1 Experiment Setup

Evaluated LLMs. As the first batch, 16 LLMs were evaluated under the zero-shot setting, covering a representative range of model types. Specifically, we include: (1) closed-source models such as GPT-4.1, GPT-4.1 Mini, and GPT-4.1 Nano from OpenAI [1]; Claude 3.7 Sonnet and Claude 3.5 Sonnet from Anthropic [5, 4]; and Gemini 2.5 Flash and Gemini 2.0 Flash from Google DeepMind [42, 43]; (2) open-source models, including GLM-4-32B and GLM-4-9B from THUDM [16], Qwen2.5-72B and Qwen2.5-7B from Alibaba [48], Llama 4 and Llama 3.3 from Meta [17], and DeepSeek-v3 and DeepSeek-R1-Distill-Qwen-1.5B (referred to as DeepSeek-R1 7B) from DeepSeek [28, 19], Mixtral-8x7B-Instruct-v0.1 (referred to as Mixtral 8x7B) from Mistral AI [23]. This selection spans a diverse range of model sizes, training paradigms, and accessibility levels. We ensured consistent formatting and output parsing across all models.

Evaluation protocols. A key challenge in evaluating engineering problem-solving lies in determining not whether a solution is correct, but whether it is good enough given practical constraints. Unlike mathematical problems with definitive answers, real-world engineering tasks often involve uncertainty, redundant information, and competing objectives. These characteristics make binary judgments insufficient for capturing the quality and completeness of a solution.

For Level 1 and Level 2, which consist of well-structured problems with clear solutions, we adopt binary scoring. A response is marked correct only if it exactly matches the reference answer, and overall performance is reported as accuracy. Level 3 tasks are open-ended and under-specified, lacking a single correct answer, which makes it essential to evaluate not just correctness but the

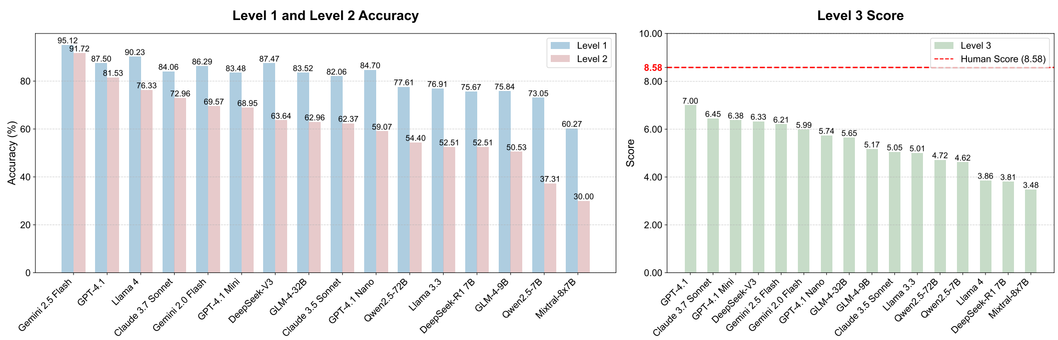


Figure 3: Overview of model performance across engineering reasoning tasks. The left subfigure shows model accuracy on Level 1 and Level 2 tasks, while the right subfigure presents expert-assigned scores on Level 3 open-ended tasks, with the human expert score indicated by the red line.

quality and completeness of a model's reasoning. Their evaluation depends on how well a model extracts relevant information, applies domain knowledge, balances competing objectives, and reasons under uncertainty. We therefore employ a rubric-based scoring framework, constructed from officially released and expert-designed criteria and refined with LLM assistance. For each question, we extract the rubric points relevant to our target competencies and convert them into concrete scoring items. To ensure scoring quality, all results were reviewed by PhD-level professionals with expertise in mathematical modeling.

Also, we introduce human scores for Level 3 tasks for comparison with LLMs' performance. We obtain human scores from two sources: award-winning competition submissions (original version) and manual solutions by top-performing students for the perturbed version. All responses are evaluated using the same rubric as LLM outputs to ensure consistency and fairness.

4.2 Results

4.2.1 Overall

Model stratification and design validation. Model performance exhibits a clear downward trend from Level 1 to Level 3, demonstrating the effectiveness of our hierarchical difficulty design. As shown in Figure 3, most models achieve high accuracy on Level 1, perform moderately on Level 2, and struggle significantly on Level 3. This progression indicates that our hierarchical framework successfully separates problems by cognitive difficulty, with each level revealing distinct capability thresholds. The results validate that a multi-level design is necessary to capture the full spectrum of engineering problem-solving capabilities.

Evaluating high-level engineering reasoning. Level 3 is designed to assess high-level engineering reasoning that goes beyond formulaic computation. Unlike Level 1 and Level 2, which focus on structured problem solving, Level 3 features open-ended and underspecified tasks that better reflect real-world engineering challenges. The sharp performance drop at this level reveals the current limitations of LLMs in handling such complex scenarios. Besides, the gap between LLMs and human experts at Level 3 also reveals a key deficiency in high-level engineering capabilities. All evaluated models score well below the human expert, who achieves an average of 8.58, indicating that current LLMs are still far from reliably handling complex engineering problems. This underscores the need for further research to bridge this gap.

Smaller-scale LLMs struggle with complex tasks. While all LLMs show room for improvement on complex, open-ended engineering tasks, smaller-scale LLMs exhibit significantly greater limitations. As task complexity increases, performance disparities widen: on Level 1, most models score similarly (70–90%), but on Level 2, top models like GPT-4.1 and Gemini 2.5 Flash exceed 80% accuracy, while smaller-scale models fall below 40%. This trend continues at Level 3, where leading models approach scores of 7.0, yet smaller-scale models remain under 4.0.

Robustness and contamination risk. Some LLMs may achieve high scores not through model-internal reasoning, but due to overlap with pretraining data. To reveal this, we introduce perturbed variants that modify surface details but keep the core problem structure unchanged. As shown in Figure 4, performance remains relatively stable on Level 1 but drops sharply on Level 2—e.g., 9.3% for GPT-4.1 Nano, 11.4% for Qwen2.5-7B, and 8.3% for Mixtral-8x7B. These declines reveal that

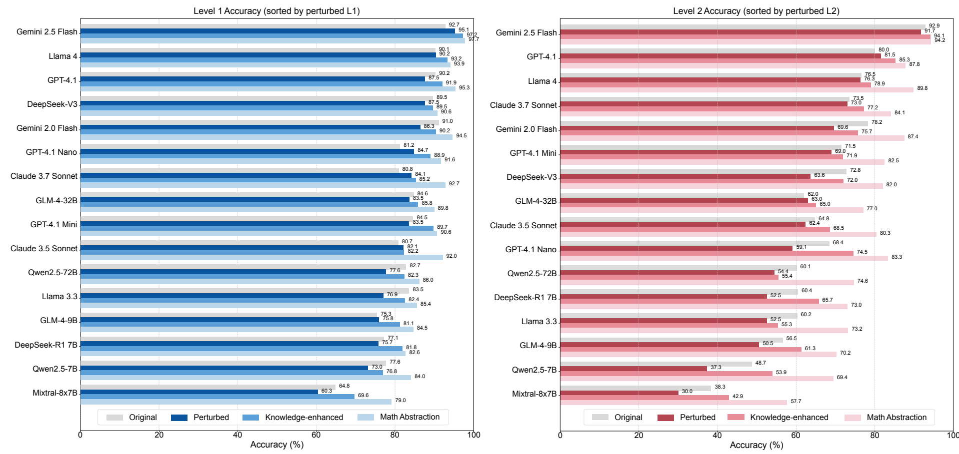


Figure 4: Accuracy of LLMs on Level 1 (left) and Level 2 (right) tasks across four variants: Original, Perturbed, Knowledge-enhanced, and Math Abstraction. Drops in the Perturbed version indicate sensitivity to input changes, while gains in the latter two show that current LLMs require external knowledge or reformulation to improve accuracy—highlighting their lack of these abilities.

many models rely on superficial pattern matching rather than robust reasoning. This underscores the value of perturbation-based evaluation in exposing overestimated capabilities and assessing true generalization.

4.2.2 Performance for Level 1 & Level 2 Tasks

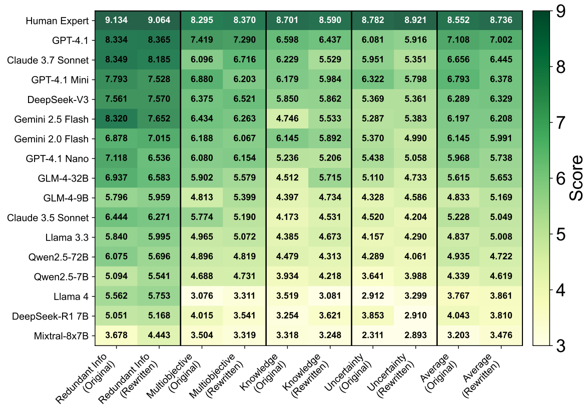
Our results show that adding explicit domain knowledge significantly improves model accuracy across all levels, especially for weaker models. As shown in Figure 4, models perform consistently better on knowledge-enhanced variants than on perturbed inputs. These gains may reflect two common failure modes: either the model lacks sufficient domain knowledge, or it fails to recognize when and how to apply it during multi-step reasoning. The use of explicit knowledge prompts thus provides a useful diagnostic signal for distinguishing between knowledge gaps and reasoning failures—an important capability dimension for engineering benchmarks.

In addition, LLMs' performance further improves when problems are abstracted into symbolic mathematical form, eliminating engineering context. As shown in Figure 4, most models achieve their highest accuracy under this variant, particularly smaller-scale LLMs that struggle with contextual interpretation. This trend reveals that the primary difficulty in engineering problem-solving lies not in the computation itself, but in the upstream reasoning required to structure the problem from natural input. This affirms the necessity of assessing reasoning steps that precede formula application—steps often overlooked by traditional math benchmarks.

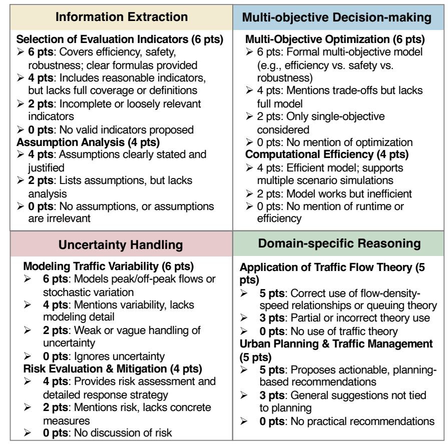
Smaller-scale LLMs exhibit significantly greater performance variation across different input versions, revealing their limited generalization and unstable reasoning processes. As shown in Figure 4, Qwen2.5-7B drops by 11.4% under the perturbed version, but gains 16.6% when explicit domain knowledge is added and a further 15.5% under math abstraction. In contrast, Gemini 2.5 Flash—a top-performing model—remains largely stable, with only minimal changes relative to its perturbed performance (-1.2%, +2.4%, and +0.1% respectively). This contrast highlights that smaller-scale models are sensitive to input formulation and often rely on surface patterns rather than consistent, context-aware reasoning.

4.2.3 Performance for Level 3 Tasks

Dimension-wise and model-wise performance. As shown in Figure 5a, human experts lead across all four dimensions with a balanced capability profile. In contrast, LLMs show uneven performances: they perform best on redundant information extraction, moderately on multi-objective decision-making, and poorly on domain-specific reasoning and uncertainty handling—highlighting a lack of deep, context-aware reasoning. Results also demonstrate that model performance also correlates with scale and accessibility. Larger, closed-source models like GPT-4.1 and Gemini 2.5 Flash consistently score above 6, demonstrating broader coverage though limited in-depth analysis. In contrast, smaller open-source models (e.g., Qwen2.5-7B, Mixtral-8x7B) average below 4, often omitting key factors such as trade-offs or uncertainty handling.



(a) Level 3 Model Evaluation.



(b) Scoring rubric example.

Figure 5: Level 3 Model Evaluation and Scoring Rubric. This figure summarizes Level 3 evaluation results and scoring standards. Subfigure (a) reports average model scores across four capabilities under both original and rewritten inputs. Subfigure (b) shows an example rubric outlining scoring criteria across capability dimensions.

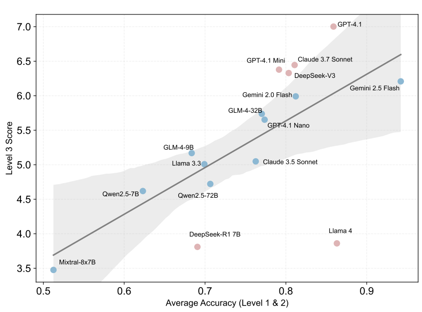


Figure 6: Correlation between structured tasks (Level 1&2) and open-ended tasks (Level 3).

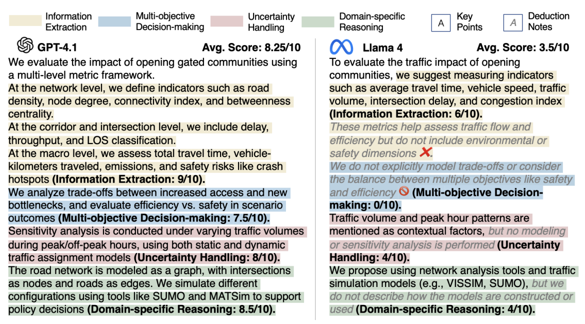


Figure 7: Case study showing why Llama 4 received low Level 3 scores.

Correlation analysis. To quantify this trend, Figure 6 illustrates the relationship between model performance on structured tasks (Levels 1 & 2) and open-ended tasks (Level 3). Overall, we observe a clear positive correlation: models that achieve higher accuracy on structured tasks tend to also perform well on open-ended tasks, suggesting a general consistency across task types.

However, few models deviate from the general trend. For example, GPT-4.1, Claude 3.7 Sonnet, and DeepSeek-V3 outperforming expectations on Level 3 tasks—showing not just factual recall but stronger reasoning and modeling abilities. In contrast, models like Llama 4 perform pretty well on structured tasks but falter on open-ended ones, revealing weak high-level reasoning. Figure 7 illustrates this gap: Llama 4 scores 0 in multi-objective decision-making due to missing trade-off analysis, while GPT-4.1 provides a structured evaluation and scores 7.5. A similar shortfall also appears in uncertainty handling. These examples show that Llama 4 can recall facts but struggles to apply them in complex, judgment-based scenarios.

5 Conclusion

We introduce **EngiBench**, a benchmark for evaluating LLMs on engineering problem solving across increasing levels of complexity. Our results show that while current models perform well on foundational knowledge retrieval, their performance declines significantly in multi-step contextual reasoning tasks, due to both domain knowledge gaps and limited mathematical reasoning. On open-ended modeling tasks, even the strongest models fall short of human-level performance, revealing persistent limitations in high-level reasoning, trade-off analysis, and uncertainty handling. These findings underscore the need for LLMs to move beyond pattern matching and toward deeper reasoning capabilities for real-world engineering applications.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, et al. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*, 2025.

[3] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

[4] Anthropic. Claude-3 family: Opus, sonnet, haiku, 2024. Available at: https://assets.anthropic.com/m/61e7d27f8c8f5919/original/Claude-3-Model-Card.pdf.

[5] Anthropic. Claude-3.5 sonnet, 2024. Available at: https://www.anthropic.com/news/claude-3-5-sonnet.

[6] Daman Arora, Himanshu Singh, et al. Have llms advanced enough? a challenging problem solving benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7527–7543, 2023.

[7] Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv preprint arXiv:2402.14762*, 2024.

[8] Yuheng Cheng, Huan Zhao, Xiyuan Zhou, Junhua Zhao, Yuji Cao, Chao Yang, and Xinlei Cai. A large language model for advanced power dispatch. *Scientific Reports*, 15(1):8925, 2025.

[9] Anoop Cherian, Kuan-Chuan Peng, Suhas Lohit, Joanna Matthiesen, Kevin Smith, and Josh Tenenbaum. Evaluating large vision-and-language models on children's mathematical olympiads. *Advances in Neural Information Processing Systems*, 37:15779–15800, 2024.

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[11] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8706–8719, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[12] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.

[13] Clive L Dym, Alice M Agogino, Ozgur Eris, Daniel D Frey, and Larry J Leifer. Engineering design thinking, teaching, and learning. *Journal of engineering education*, 94(1):103–120, 2005.

[14] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

[15] Lei Gao, Dongkai Wang, Yanjun Cui, Jun Zhao, Yi Gu, Ge Zhang, Yuxiao Dong, and Jie Tang. OmniMath: An open-source and reproducible large benchmark for llm mathematical reasoning ability evaluation, 2024.

[16] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

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

[18] Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno de Moraes Dumont, and Sanmi Koyejo. Putnam-axiom: A functional and static benchmark for measuring higher level mathematical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

[19] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

[21] Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, et al. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *arXiv preprint arXiv:2502.06453*, 2025.

[22] Yiming Huang, Zhenghao Lin, Xiao Liu, Yeyun Gong, Shuai Lu, Fangyu Lei, Yaobo Liang, Yelong Shen, Chen Lin, Nan Duan, et al. Competition-level problems are effective llm evaluators. *arXiv preprint arXiv:2312.02143*, 2023.

[23] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[24] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[25] Ming Li, Jike Zhong, Tianle Chen, Yuxiang Lai, and Konstantinos Psounis. Eee-bench: A comprehensive multimodal electrical and electronics engineering benchmark. *arXiv preprint arXiv:2411.01492*, 2024.

[26] Zhenwen Liang, Tianyu Yang, Jipeng Zhang, and Xiangliang Zhang. Unimath: A foundational and multimodal mathematical reasoner. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7126–7133, 2023.

[27] Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025.

[28] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[29] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*, 2024.

[30] Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, Joshua B Tenenbaum, Daniela Rus, Chuang Gan, and Wojciech Matusik. Llm and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. *arXiv preprint arXiv:2405.09783*, 2024.

[31] Yujun Mao, Yoon Kim, and Yilun Zhou. CHAMP: A competition-level dataset for fine-grained analyses of LLMs' mathematical reasoning capabilities. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13256–13274, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[32] Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[33] Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

[34] Nayantara Mudur, Hao Cui, Subhashini Venugopalan, Paul Raccuglia, Michael P Brenner, and Peter Norgaard. Feabench: Evaluating language models on multiphysics reasoning ability. *arXiv preprint arXiv:2504.06260*, 2025.

[35] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 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, 2021.

[36] ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

[37] Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[39] Saurabh Srivastava, Anto PV, Shashank Menon, Ajay Sukumar, Alan Philipose, Stevin Prince, Sooraj Thomas, et al. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *arXiv preprint arXiv:2402.19450*, 2024.

[40] Usman Syed, Ethan Light, Xingang Guo, Huan Zhang, Lianhui Qin, Yanfeng Ouyang, and Bin Hu. Benchmarking the capabilities of large language models in transportation system engineering: Accuracy, consistency, and reasoning behaviors. *arXiv preprint arXiv:2408.08302*, 2024.

[41] Zhengyang Tang, Chenyu Huang, Xin Zheng, Shixi Hu, Zizhuo Wang, Dongdong Ge, and Benyou Wang. Orlm: Training large language models for optimization modeling. *arXiv preprint arXiv:2405.17743*, 2024.

[42] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[43] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[44] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.

[45] Xinlei Wang, Maike Feng, Jing Qiu, Jinjin Gu, and Junhua Zhao. From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection. *Advances in Neural Information Processing Systems*, 37:58118–58153, 2024.

[46] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[48] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[49] Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36:21573–21612, 2023.

[50] Albert S Yue, Lovish Madaan, Ted Moskovitz, DJ Strouse, and Aaditya K Singh. Harp: A challenging human-annotated math reasoning benchmark. *arXiv preprint arXiv:2412.08819*, 2024.

[51] Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, et al. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37:46819–46836, 2024.

[52] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022.

[53] Xiyuan Zhou, Huan Zhao, Yuheng Cheng, Yuji Cao, Gaoqi Liang, Guolong Liu, Wenxuan Liu, Yan Xu, and Junhua Zhao. Elecbench: a power dispatch evaluation benchmark for large language models. *arXiv preprint arXiv:2407.05365*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract and Section 1

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of our work are discussed in a separate "Limitations" section in the supplementary material.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: See `https://huggingface.co/datasets/EngiBench/EngiBench` and `https://github.com/EngiBench/EngiBench`

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See https://huggingface.co/datasets/EngiBench/EngiBench and https://github.com/EngiBench/EngiBench

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See https://huggingface.co/datasets/EngiBench/EngiBench and https://github.com/EngiBench/EngiBench

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Experiments were run with deterministic hyperparameters. Re-running with randomness would incur significant costs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Reproducing experimental results does not rely on this information

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Anonymous and meets the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section 1

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper does not release any models or datasets that pose a risk of misuse

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Section 3.3

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See `https://huggingface.co/datasets/EngiBench/EngiBench` and `https://github.com/EngiBench/EngiBench`

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: See Section 3.3 and Section 4.1

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: See Section 3.3 and Section 4.1

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: This paper evaluates the capabilities of LLMs on engineering-related tasks using a structured benchmark. The use of LLMs is central to the research and forms the basis of the core methodology and experimental design.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

Contents

A Future Works

While EngiBench establishes a strong foundation for evaluating LLMs on engineering problem-solving, several avenues remain for further development and expansion:

Scalability Across Engineering Domains. EngiBench currently covers three core engineering subfields—Systems & Control, Physical & Structural, and Chemical & Biological—which together span a wide range of disciplines such as Mechanical, Electrical, and Chemical/Biological Engineering. The benchmark framework is designed to be broadly applicable and adaptable across domains. In future work, we plan to expand the dataset by incorporating problems from additional engineering disciplines to further enhance data volume and subject diversity.

Multimodal Evaluation Extensions. Future versions of EngiBench will introduce a dedicated multimodal subset to evaluate models on tasks involving vision-language reasoning. This will enable systematic assessment of model performance in scenarios that demand visual interpretation alongside textual understanding.

Support for Long-Context Reasoning. We plan to extend the benchmark to include long-context engineering tasks by leveraging models with expanded context windows or hierarchical processing capabilities. This will allow for evaluation of more complex, information-rich tasks currently excluded due to input length limitations.

B Limitations

While EngiBench provides the first systematic evaluation of LLMs on real-world engineering problems—including multi-level tasks, variant-based reasoning diagnostics, and open-ended modeling—several limitations remain that we plan to address in future work.

Multimodal Support. Many real-world engineering problems require interpreting visual elements such as diagrams, schematics, or structured tables. However, the current version of EngiBench excludes such tasks due to the lack of multimodal input capabilities in most existing LLMs. To ensure consistency across evaluations, we restrict all inputs to text-only formats.

Long-Context Support. Some engineering tasks involve long problem descriptions or extensive tabular data that exceed the input length limits of current LLMs. To avoid unfair model truncation effects and ensure uniform evaluation settings, such problems are not included in this version of the benchmark.

Human-in-the-loop Curation. Building the dataset involves substantial human effort, including problem collection, answer generation, and variant validation. This ensures data quality and alignment with engineering standards, but also reflects the significant manual effort behind the benchmark.

C Dataset Curation- Additional Details

C.1 Level 1 & Level 2 Extraction Process

To construct a high-quality and diverse dataset for Level 1 and Level 2, we systematically extract relevant tasks from a range of established public benchmarks, including MMLU [20], MATH [20], GSM8k [10], Orca-Math [33], HARP [50], Omni-MATH [15], Big-MATH [2], and competition datasets such as cn_k12, Olympiads, AOPS forum, and AMC-AIME [22]. In addition to these public sources, we also incorporate university-level engineering educational materials, including assignments, examinations, and instructor-provided teaching content, to further increase task diversity and real-world relevance.

To transform mathematical and logic-oriented problems into engineering-relevant evaluation tasks, we design a structured data processing pipeline that combines LLM-based analysis with human verification to ensure engineering relevance and classification accuracy. This pipeline ensures that all included problems align with real-world engineering semantics and reasoning demands, forming the basis for Level 1 and Level 2 in EngiBench.

The processing pipeline consists of the following steps:

1. **Engineering Relevance Filtering:** Each problem is evaluated for its applicability to engineering scenarios. Problems lacking domain relevance are excluded to maintain the technical integrity of the benchmark. The prompt used to determine whether a problem pertains to engineering is as follows:

```
"""Determine if ORIGINAL problem can be solved with ONLY mathematical knowledge (NO engineering background):
- False if requires any domain-specific knowledge
- True if solvable through pure mathematical calculations"""

```

2. **Discipline and Subfield Classification:** Relevant problems are first assigned to a specific engineering discipline (e.g., Electrical, Civil, Mechanical), and then grouped into one of EngiBench's three high-level analytical subfields: Systems & Control, Physical & Structural, or Chemical & Biological. The prompt used for assigning a problem to a specific engineering discipline is as follows:

```
"""If yes, which engineering category? (Chemical/Bioengineering/Geotechnical/Energy/Nuclear/Aerospace/Automotive/Biomedical/Civil/Control/Electrical/Industrial/Mechanical/Ocean/Environmental/Other) (Please try to avoid Other)
If not an engineering problem, return "N/A"."""

```

3. **Difficulty Level Assignment:** Based on the complexity of the required reasoning process, tasks are categorized into Level 1 or Level 2. Level 1 includes basic knowledge recall and single-step computation, while Level 2 involves multi-step inference, contextual understanding, and integration of structured constraints. The prompt used for classifying the difficulty level of a problem is as follows:

```
"""Difficulty level? (Level 1/Level 2) (Please try to avoid unknown):
```

```
- Level 1: The problem can be solved by a direct retrieval of information or by directly substituting values into a known formulai.e., the shortest possible solution path. No chaining of intermediate steps is required. (Example: Using Ohm's Law, V = IR, to directly compute voltage when given current and resistance.)
- Level 2: The problem requires multi-step reasoningmeaning that it involves chaining together several logical deductions, intermediate calculations, or systematic strategies beyond a single direct formula application. (Example: Analyzing a circuit to compute total resistance by first calculating individual branch resistances and then combining them.)"""

```

C.2 Level 3 Data Collection and Processing

To construct the Level 3 dataset in **EngiBench**, we focus on real-world, open-ended engineering tasks sourced from major mathematical modeling competitions. Specifically, we collect problems from publicly accessible archives of contests such as the China Undergraduate Mathematical Contest in Modeling (CUMCM), the Mathematical Contest in Modeling / Interdisciplinary Contest in Modeling (MCM/ICM), and the Asia and Pacific Mathematical Contest in Modeling (APMCM), covering the years 2010 to 2024.

To ensure domain relevance and evaluation consistency, we apply strict filtering criteria. We retain only problems with clear engineering context and official scoring rubrics, and exclude those that depend heavily on complex diagrams or large external tables requiring multimodal input.

We standardize the selected problems using a structured pipeline that combines LLM-based processing with human oversight. This ensures language clarity, formatting consistency, and reduced risk of data contamination. The pipeline includes the following steps:

1. **Language Normalization:** Non-English problems are translated into fluent English using machine translation, while preserving the original engineering semantics.
2. **Expression Rewriting:** To minimize potential overlap with pretraining data, each problem is paraphrased by the LLM using diverse sentence structures and reasoning styles. While surface expressions are significantly altered, the core logic, numerical values, and solution paths remain unchanged. This step produces the *perturbed version* of each task, which is used to evaluate model robustness to superficial input variations.
3. **Multimodal Simplification:** For problems containing simple figures or tables, we extract and describe the essential information using plain text or LaTeX-formatted representations to support uniform text-based evaluation.

LLM Prompt Template: The following instruction prompt is used to guide the LLM in modifying each problem:

```
"""Assuming you are a question expert, please translate this question into English. And while ensuring that the meaning of the question remains unchanged (preserving all logic, values, and the type of reasoning required), change the way the question is expressed by rewriting it in a way that is radically different from your regular logical structure, simulating the randomness of manual rewriting by human experts, and using as many sentence variations as possible. If there is a table, please convert it into a table form using LaTeX. For simple pictures, please describe them directly. The question is required to be converted into is in str format."""

```

To ensure the technical rigor and domain consistency of the Level 3 dataset, the entire generation and transformation process was closely supervised and iteratively revised by doctoral-level professionals with extensive expertise in engineering and mathematical modeling. These experts reviewed both the selection of source problems and the outputs produced by the language model, verifying that each task preserved the original problem's intent, accurately reflected real-world engineering reasoning, and met the standards expected in academic and professional modeling contexts.

The details of how the original contest scoring standards were mapped into EngiBench's formal scoring rubrics are described in the later subsection (see Section E.2).

C.3 Version Variant Generation

To assess model robustness and isolate specific reasoning limitations, we generate three structured variants for each Level 1 and Level 2 problem: *Perturbed*, *Knowledge-Enhanced*, and *Math Abstraction*. These variants are created through LLM prompting, with manually verified outputs to ensure alignment with the original problem logic and correctness. Below, we describe the purpose and generation criteria for each variant, accompanied by illustrative prompts.

- **Perturbed Version.** This variant alters the surface form of the original problem—either through numerical or linguistic changes—while preserving its core logic and computational requirements. The purpose is to test whether model performance stems from true reasoning ability or superficial pattern matching. A rewriting suitability code (0–3) guides the type of modification to apply. The prompt used to generate the perturbed version and related content is as follows:

```
"""
1. Rewriting Suitability: Determine the type (0-3):
   - 0: Non-rewritable (use only when necessary)
   - 1: Modify expressions only
   - 2: Modify numerical values only
   - 3: Modify both expressions and numerical values
   // Note: All rewrites must maintain the original problem logic, engineering context, and reasoning/computational requirements

2. Rewritten Problem: Rewrite the problem according to the type of rewriting suitability above. Make the answer as difficult as possible while ensuring that the answer is correct. (Please rewrite the problem in a way that is radically different from your regular logical structure by: (1) avoiding common reasoning patterns in your model, (2) simulating human expert manual rewriting randomness, and (3) using maximum sentence variation.)
   - If 0, return original problem unchanged
   - If 1, modify expressions only
   - If 2, modify numerical values only
   - If 3, modify both expressions and values

3. Rewritten Solution Process: Provide step-by-step explanation including all reasoning, calculations and logic. Clearly state if answer can be obtained directly through formula substitution (shortest solution path without intermediate steps).

4. Rewritten Answer: Provide correct answer for rewritten problem (only types 2/3 may change)"""
```

- **Knowledge-enhanced Version.** In this version, relevant domain knowledge—such as formulas, constants, and conversions—is explicitly provided before the original question. This allows us to evaluate whether performance deficits are due to missing knowledge or failures in application. The question itself is unchanged to isolate the impact of added context. The prompt used to generate the knowledge-enhanced version is as follows:

```
"""Knowledge-Enhanced Version:
WARNING: Make sure the final numerical answer to the converted mathematical problem is exactly the same as the original problem.
```

```

Given:
- List all relevant formulas or principles (e.g., Ohm's Law: V
    = I * R)
- Include physical constants with values if they are involved (
    e.g., g = 9.8 m/s^2)
- Specify unit conversions if applicable (e.g., 1 kWh = 3.6 *
    10^6 J)
- State any assumptions or ideal conditions if necessary (e.g.,
    assume no heat loss)

Problem:
Repeat the original question exactly as stated

Example:
Original: "Calculate voltage across 5 Ohm resistor with 2 A
    current"
Enhanced:
"Given:
- Ohm's Law: V = I * R
- Problem: Calculate voltage across 5 Ohm resistor with 2 A
    current" """

```

- **Math Abstraction Version.** This version reformulates the original engineering problem into a purely mathematical format by removing all domain-specific context. Variables and operations are explicitly defined to preserve the exact calculation logic. This allows us to isolate whether reasoning failure arises from contextual understanding or mathematical ability. The prompt used to generate the math abstraction version is as follows:

```
"""Rewrite the given problem into a purely mathematical version
    by:

a. Remove all domain-specific context (e.g., chemistry, physics
    , economics).
b. Keep only numbers, variables, and math operations.
c. If domain-specific knowledge is required (e.g., reaction
    ratio, atomic mass), extract only the final numerical ratio
    or constant and include it directly.
d. Maintain the exact calculation logic and final answer.
e. Use structured symbolic language in a compact form:
- Introduce variables explicitly (e.g., "Let x = 2 and y = 3.")
- Define the calculation clearly (e.g., "Total z = min(x, y) *
    2.")
- End with "Find the result."

WARNING: Make sure the final numerical answer to the converted
    mathematical problem is exactly the same as the original
    problem.

Examples:

Original: "In the reaction: Cl2 + H2 -> 2HCl, 1 mole of Cl2
    reacts with 2 moles of H2. How many moles of HCl can be
    formed?"
converted_problem: "Let x = 1 and y = 2. They react in the
    ratio x : y : z = 1 : 1 : 2. Total product z = min(x, y) *
    2. Find the result."

Original: "A 2m wide platform sinks 0.01m under 60kg. Estimate
    its length assuming water density = 1000 kg/m^3."
converted_problem: "Let x = 60 / (2 * 0.01 * 1000). Find the
    result." """

```

D Dataset URLs, License, and Hosting Plan

The dataset is uploaded for public download under the CC-BY-NC-4.0 license: https://huggingface.co/datasets/EngiBench/EngiBench

Our dataset is also hosted on https://github.com/EngiBench/EngiBench which will provide long-term support for hosting the dataset and maintaining version control.

Contact information for the authors is available on our project website. We also welcome questions, feedback, and issue reports via GitHub or email. We will use the GitHub Issues page to manage any errata or community-suggested updates to the benchmark.

D.1 Dataset Instance Metadata

For the EngiBench dataset, each instance corresponds to an engineering task and is stored in a structured format. Instances are categorized according to task difficulty (Level 1, 2, or 3) and are constructed with multiple versions to enable fine-grained evaluation of different capabilities. The metadata fields for each level are described below:

Level 1 and Level 2 Each row in the Level 1 & 2 dataset corresponds to a closed-form or structured engineering problem, and includes the following fields:

- **problem** – Original natural language problem statement.
- **answer** – Ground truth answer to the original problem.
- **subfield** – Engineering subfield to which the problem belongs (e.g., Systems & Control).
- **category** – Topic-specific classification within the subfield (e.g., Thermodynamics).
- **difficulty** – Either `Level 1` (Foundational Knowledge Retrieval) or `Level 2` (Contextual Reasoning).
- **converted_problem** – Abstract mathematical formulation of the problem.
- **converted_problem_llm_answer** – LLM-generated response to the converted problem.
- **knowledge_enhanced_problem** – Problem reformulated with explicit formulas and domain definitions.
- **rewritten_problem** – Semantically or numerically perturbed variant of the original problem.
- **rewritten_answer** – Answer to the rewritten problem.
- **rewritten_converted_problem** – Mathematical abstraction of the rewritten problem.
- **rewritten_converted_problem_llm_answer** – LLM response to the rewritten converted problem.
- **rewritten_knowledge_enhanced_problem** – Knowledge-enhanced version of the rewritten problem.

Level 3 Each Level 3 instance represents an open-ended modeling task and includes both the problem prompt and a rubric-based evaluation across multiple capability dimensions:

- **question_original_language** – Native language version of the open-ended task (typically Chinese).
- **question** – English translation of the open-ended modeling task.
- **question_modified** – Semantically perturbed variant of the task.
- **subquestion_original_language** – Rubric sub-criteria in the original language.
- **subquestion** – English translation of the rubric sub-criteria.
- **subquestion_modified** – Semantically perturbed variant of the sub-criteria.
- **source_detail** – Source of the modeling task (e.g., MCM, coursework).
- **official_scoring_standard_original_language** – Original rubric definition.

- **official_scoring_standard** – English translation of rubric criteria.
- **subfield** – Engineering subfield of the task.
- **category** – Domain or topic under which the task is categorized.
- **information_extraction_score** – Score for identifying relevant variables and constraints.
- **multi_objective_decision_score** – Score for resolving trade-offs across objectives.
- **uncertainty_handling_score** – Score for reasoning under ambiguity or variable inputs.
- **domain_specific_reasoning_score** – Score for applying engineering-specific logic and formulas.

E Evaluation Details

E.1 Level 1 & Level 2 Evaluation Details

Level 1 and Level 2 tasks consist of well-structured problems with clearly defined solutions. Therefore, we adopt a **binary scoring** method. Each model-generated answer is compared against a reference answer and marked as either correct (1) or incorrect (0). Final performance is reported as overall accuracy.

To improve evaluation robustness, we introduce an automated comparison prompt executed by a large language model. This prompt is carefully designed to evaluate whether the generated answer matches the reference answer based on mathematical correctness, unit validity, and reasoning soundness. For numerical questions, a tolerance of $\pm 2\%$ is allowed to account for rounding differences in complex calculations. The model is instructed to output only a Boolean result ("True" or "False") to ensure consistent scoring across all instances. The evaluation prompt used for this process is as follows:

```
"""Please analyze these two answers carefully:
Generated Answer: {generated_answer}
Standard Answer: {correct_answer}

Follow these rules for comparison:
1. For calculation-focused problems:
   - If the numerical values match, consider it correct even if units
     are missing
   - Focus on the mathematical reasoning and final numerical result
   - Check if the core calculation steps are correct
   - For complex calculations, allow  2 % tolerance in the final
     numerical result

2. For conceptual or unit-specific problems:
   - Units and their consistency must be considered
   - The complete answer including units is required

3. Consider the answer correct if:
   - The mathematical reasoning is sound
   - The final numerical value matches (within  2 % tolerance for
     complex calculations)
   - For calculation-focused problems, matching units are not
     mandatory

Reply only with "True" or "False". """
```

E.2 Level 3 Evaluation Details

To convert open-ended modeling problems into evaluable tasks suitable for benchmarking LLMs, we systematically transform official scoring standards into structured rubrics aligned with the four key capabilities identified in Section 3.1: **information extraction**, **domain-specific reasoning**, **multi-objective decision-making**, and **uncertainty handling**. These capabilities form the foundation of Level 3 evaluation.

To construct these scoring rubrics from the official contest-provided evaluation standards, we used an LLM to transform raw scoring descriptions into a capability-oriented rubric aligned with our four target assessment dimensions. Each contest problem was paired with its corresponding official scoring criteria, and this combined input was passed to the LLM using a carefully designed instruction prompt. The goal was to generate specific, well-structured rubrics that are detailed enough to capture subtle distinctions in model outputs, while remaining concise and practical for use in benchmark-scale evaluations. The overall scoring workflow is illustrated in Figure 8.

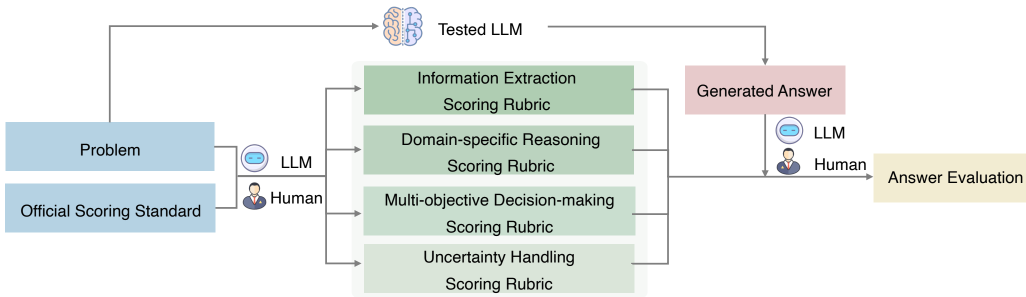


Figure 8: Workflow for generating the modified scoring rubrics. The official scoring standard and contest problem are first provided to a LLM to generate draft rubrics aligned with four core capabilities: Information Extraction, Domain-Specific Reasoning, Multi-Objective Decision-Making, and Uncertainty Handling. The resulting rubrics are then reviewed and refined by domain experts to ensure technical accuracy and alignment with modeling principles.

The prompt used for rubric generation is provided below:

```
"""Assume you are an expert in problem design and grading, with deep familiarity with mathematical modeling. Please help me design an evaluation rubric for assessing large language models' engineering capabilities. Specifically, I will provide a problem and its scoring criteria, and you will tell me which of the following capabilities are assessed by this rubric: redundant_information_filtering_score, multi_objective_tradeoff_score, uncertainty_handling_score, and deep_knowledge_integration_score. In particular, please identify how each capability is assessed through specific aspects of the problem or rubric.

For each capability that is covered, provide a scoring rubric in the following format:

Problem [(Problem ID)]:
redundant_information_filtering_score: (1)(2)...
multi_objective_tradeoff_score: (1)(2)...
uncertainty_handling_score: (1)(2)...
deep_knowledge_integration_score: (1)(2)...

Notes: Each capability has a total possible score of 10 points. In other words, the total score for each listed capability should sum to 10 points. Capabilities that are not covered in this problem receive 0 points. The rubric should further specify, under each capability, the different score levels (e.g., 1 point, 2 points, 3 points, etc.) and the corresponding specific behaviors or response characteristics associated with each level.

Please read the problem and rubric carefully and provide a capability-based evaluation rubric for how this problem assesses the output of large language models."""
```

The prompt used to evaluate the generated answer against the rubric is as follows:

```
f"""
    You are a professional modeling competition judge with extensive experience in evaluating mathematical and engineering models. Please conduct a rigorous evaluation of the following answer based on the provided criteria.

    Answer to evaluate:
    {answer}

    Evaluation Criteria:
    {score_criteria}

    Please evaluate strictly according to the criteria and provide your assessment in the following JSON format:
    {{
        "score": <score between 0-10, can use decimal points for precision>,
        "reason": "Detailed evaluation breakdown:\n
                  1. [Specific criterion] - [sub-score] points: [justification]\n
                  2. [Specific criterion] - [sub-score] points: [justification]\n
                  3. [Specific criterion] - [sub-score] points: [justification]\n
                  Final score: [total] points"
    }}

    Note:
    - Break down your scoring into specific components
    - Provide clear justification for each sub-score
    - Be objective and consistent in your evaluation
    - Consider both the technical accuracy and the methodology
    """
```

E.3 Level 3 Scoring Examples

As results shown in section 4.2.3, the answers of LLMs to open-ended tasks show significant differences in four dimensions of information extraction, multi-objective decision making, uncertainty handling and domain-specific reasoning. Figure 7 preliminarily presents two scoring segments, 3 points and 8 points, for the evaluation of models' answers. To demonstrate the response performance of different segments more clearly and intuitively, we provide the following examples with more Level 3 scoring details:

1. **Full Mark (Avg. Score: 9.475):** The problem requires optimizing Hu sheep farm pen utilization under stochastic conditions (conception rates, gestation periods, litter sizes) while adhering to strict capacity constraints and cohabitation rules. The solution must minimize expected losses from idle pens (1 unit/day) or shortages (3 units/day) through dynamic scheduling and statistical validation.

    - **Information Extraction (10/10):**
      Exclusion of Deterministic Assumptions (5/5): Section 1 (System Overview) clarifies all critical parameters modeled as random variables (e.g., "$X_c \sim \text{Binomial}(N_m, 0.85)$: Number of successful conceptions; $G \sim U[147, 150]$: Gestation days; $L_s \sim \text{Poisson}(\lambda = 2.2)$: Liveborn lambs per ewe, with 3% mortality ($L_a = L_s \cdot 0.97$); $L_d \sim U[35, 45]$: Lactation days"). Section 3A (Scenario Generation) replaces fixed values with dynamic sampling (e.g., "For each scenario, sample: - Which ewes conceive (Bernoulli, 85%) - Their gestation (G) - Number of lambs (L_s), apply mortality - Lactation length (L_d)"). Section 6B (Robust Planning) makes flexible scheduling responsive to stochastic outcomes (e.g., "Adjust mating/rest period within allowed windows to shift animal flows.").
      Identification of Valid Uncertainty Parameters (5/5): Section 1 clarifies explicit distributions for all uncertainties (e.g., "$X_c \sim \text{Binomial}(N_m, 0.85)$... $G \sim U[147, 150]$...

$L_s \sim$ Poisson(2.2)... $L_d \sim U[35, 45]$"). Section 3A ensures consistent application in scenario generation (e.g., "Sample conception (Bernoulli), gestation (G), litter size (L_s), lactation (L_d)."). Section 5 (Loss Function) offers loss calculation integrating stochastic inputs (e.g., "$\mathbb{E}_{scenario}\left[\sum_t [I_t + 3S_t]\right]$").

- **Multi-objective Decision making (9.2/10):**
  Minimized Expected Loss & Output Maximization (4.5/5): Section 5 (Loss Function) contains rigorous mathematical formulation balancing idle (1 unit) vs. shortage (3 unit) costs (e.g., "Objective: $\min \mathbb{E}_{scenario}\left[\sum_t [I_t + 3S_t]\right]$ $I_t =$ Idle pens, $S_t =$ Shortages"). Section 7B (Robust Planning) includes statistical validation of tradeoffs (e.g., "Monte Carlo over Scenarios: Simulate losses across all scenarios for each candidate policy.") Section 8 (Results Table) applies quantitative comparison of policies.
  Lactation Flexibility & Fattening Tradeoffs (4.7/5): Section 1 (System Overview) makes explicit dynamic linkage between lactation and fattening (e.g., "$L_d \sim U[35, 45]$: Lactation days $\rightarrow F_d = 210 + 2 \cdot (40 - L_d)$: Fattening days"). Section 6B (Robust Planning) considers operational use of flexibility to smooth demand (e.g., "Adjust rest periods to align cohorts, minimizing 'loner pens'."). Section 3A (Scenario Generation) has stochastic integration of tradeoff (e.g., "Sample lactation length (L_d), impact on fattening (F_d).").
- **Uncertainty Handling (9.2/10):**
  Stochastic Process Models (4/4): Section 1 (System Overview) specifies explicit distributions for all stochastic parameters (e.g., "$X_c \sim$ Binomial$(N_m, 0.85)$, $G \sim U[147, 150]$, $L_s \sim$ Poisson(2.2), $L_d \sim U[35, 45]$"). Section 3A (Scenario Generation) implements full Monte Carlo (e.g., "Generate 1000 scenarios... sample conception (Bernoulli), gestation (G), litter size (L_s), lactation (L_d)."). Section 7B (Robust Planning) includes statistical validation of stochastic outcomes (e.g., "For each candidate policy, simulate losses across all scenarios.").
  Dynamic Adjustment Strategies (2.7/3): Section 1 (Fattening Calculation) establishes mechanistic linkage of lactation-fattening tradeoff (e.g., "$F_d = 210 + 2 \cdot (40 - L_d)$: Fattening days adjusted by lactation."). Section 6B (Robust Planning) makes adaptive scheduling but lacks two-way feedback (e.g., "Adjust rest periods to align cohorts... weekly rolling re-optimization.").
  Contingency Sets (2.5/3): Section 2 (Cohabitation Rules) contains hard-coded tolerance for uncertainty (e.g., "Group into largest feasible penfuls within 7-day windows."). Section 8 (Statistical Assessment) analyzes multi-scenario sensitivity (e.g., "Tabulate average loss, shortage probability, and max pen use.").
- **Domain-specific Reasoning (9.5/10):**
  Integration of Empirical Rules (4/4): Section 2 (Cohabitation Rules) adds hard-codes industry constraints into algorithms (e.g., "7-day tolerance window for nursing ewes, lambs, and resting ewes... Group into largest feasible penfuls (14 fattening lambs/pen, 6 nursing ewes/pen)."). Section 1 (System Overview) uses embeds empirical flexibility ranges as distributions (e.g., "$L_d \sim U[35, 45]$: Lactation days... $R \sim U[18, 22]$: Adjustable rest period.") Section 6B (Robust Planning) operationalizes flexible rest rules (e.g., "Extend rest periods to align cohorts if pens would otherwise idle.").
  Expected Loss Functions (3/3): Section 5 (Loss Function) has rigorous probabilistic loss aggregation (e.g., "$\min \mathbb{E}_{scenario}\left[\sum_t [I_t + 3S_t]\right]$ $I_t = \max(P_{avail} - P_{req}(t), 0)$, $S_t = \max(P_{req}(t) - P_{avail}, 0)$."). Section 8 (Results Table) quantifies loss distribution across scenarios. Section 3B (State Evolution) links stochastic occupancy to loss calculation (e.g., "For each day t: Compute $P_{req}(t)$ from sampled cohorts.").
  Stochastic Optimization Algorithms (2.5/3): Section 7B (Robust Planning) applies sample average approximation (SAA) method (e.g., "Monte Carlo simulation over 1000 scenarios to evaluate policies."). Section 6A (Rolling Horizon) uses heuristic dynamic programming (e.g., "Re-optimize mating batches weekly to maximize cohabitation.").

2. **5 points (Avg. Score: 5.375):** The problem involves modeling a team coordination exercise ("Unity Drum") where 8 members control a drum's tilt by pulling ropes to bounce a ball. Key tasks include: 1. Calculating the drum's tilt angle at t=0.1s based on force/timing inputs (Table 1), accounting for initial 11cm displacement. 2. Ensuring physics-based accuracy in torque, angular acceleration, and geometric relationships.
   - **Information Extraction (7.5/10):**

Error Source Analysis (5/6): Explicit Recognition: Timing errors-"Some members may apply force slightly before others" (Algorithm section); strength variation-"Members likely have different strengths" (Considerations). Partial Implementation: Timing logic in code (if timing$[i] \leq 0.1$) is noted but lacks vector-time coupling; force scaling (effective_force $= \frac{\text{force(member\_id}-1)}{10}$) is arbitrary.

Physical Model Simplification (2.5/4): Justified Simplifications: "Ignores damping for short-duration calculation" (Considerations); Drum as uniform cylinder ($I = 0.5 \cdot \text{drum\_mass} \cdot r^2$). Over-Simplifications: Fixed torque angle ($\sin\left(\frac{\pi}{2}\right)$) ignores vector geometry; rope tautness assumption ("If the drum tilts too far, ropes could slack") not modeled.

- **Multi-objective Decision making (6.5/10):**
  Tilt Angle and Force Relationship (4.5/6): Physics Foundation: Correctly derives torque ($\tau = r \cdot F \cdot \sin(\theta)$), inertia ($I = 0.5 \cdot m \cdot r^2$), and angular kinematics ($\theta = \theta_0 + \frac{1}{2}\alpha t^2$); maps rope geometry (angle_radians $= (\text{member\_id} - 1) \cdot \left(\frac{2\pi}{8}\right)$). Implementation Gaps: Timing logic (if timing$[i] \leq 0.1$) is crude; forces are binary (on/off) rather than time-interpolated; no optimization for tilt minimization (e.g., predictive control or force balancing).
  Computational Efficiency (2/4): Basic Looping-iterates over 8 members with O(1) operations per member (e.g., torque $= \text{drum\_radius} \cdot \text{force} \cdot \sin\left(\frac{\pi}{2}\right)$). No Advanced Techniques-lacks vectorization, memoization, or scalability for larger teams.
- **Uncertainty Handling (2/10):**
  Error Propagation Analysis (2/4): Acknowledgment Only: Mentions "members likely have different strengths and reaction times" (Considerations); suggests "extended to simulate more realistic distributions" but provides no math or implementation. No Quantification: Lacks sensitivity analysis or error bounds on tilt angle.
  Numerical Simulation Estimation (0/4): No Monte Carlo: Code calculates tilt for fixed inputs only (force_data); no randomization of force/timing or statistical output (mean/variance).
  Methodological Clarity (N/A): Physics steps are clear but irrelevant to uncertainty scoring.
- **Domain-specific Reasoning(5.5/10):**
  3D Mechanics Modeling (2.5/6): 2D Limitation: Explicitly states "our coordinate system will be planar (X and Y only)" (Key Equations); torque calculation ($\tau = r \cdot F \cdot \sin(\theta)$) ignores out-of-plane forces. Partial Physics: Correctly models drum as cylinder ($I = 0.5 \cdot m \cdot r^2$) but lacks 3D rotation dynamics.
  Model-Based Optimization Strategy (3/4): Suggestions Without Implementation: Proposes "damping term proportional to angular velocity" (Considerations); mentions "member variation" but no adaptive control (e.g., PID for tilt correction).

3. **1 point (Avg. Score: 1.25):** The problem involves coordinating multiple meteorological units (each with 1 primary and 2 secondary stations) to ensure reliable hourly weather data collection and full data sharing under strict communication constraints. Key challenges include managing transmission reliability (80% for secondaries, 100% for primaries), message capacity limits, and achieving 97% success probability within 8 minutes for primary data exchange. The goal is to determine the maximum number of units (Nmax), design transmission schemes, and compute performance metrics.
   - **Information Extraction (2/10):** High-Probability Constraint Processing (0/5): Failure to Address Probabilistic Guarantee: The answer calculates secondary transmission success as "expected number of reports received... is $4 \times 0.8 = 3.2$" (Step 4) but never models retransmissions or redundancy to achieve 97% success. The assumption of direct success ignores the problem's explicit probability requirement. Missing Critical Logic: No discussion of how to compensate for the 20% failure rate (e.g., retrying failed transmissions, acknowledgments, or error correction).
     Time Window Isolation (2/5): Interleaved Logs Without Justification: The primary and secondary transmission logs (Tables 1 2) are interleaved in the solution ("Round 1: Primary 1→2; Round 1: Secondary 1→1a"), but no protocol ensures collision avoidance (e.g., TDMA, priority scheduling). Unverified Simultaneity Assumption: The answer states "Simultaneous reception allowed during transmission" (Step 1) but

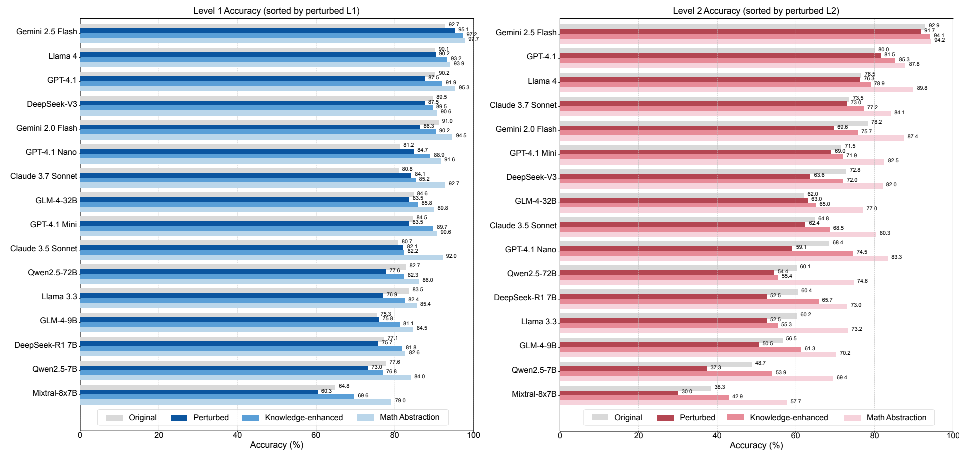


Figure 9: Accuracy of LLMs on Level 1 (left) and Level 2 (right) tasks across four variants: Original, Perturbed, Knowledge-enhanced, and Math Abstraction. Drops in the Perturbed version indicate sensitivity to input changes, while gains in the latter two show that current LLMs require external knowledge or reformulation to improve accuracy—highlighting their lack of these abilities.

doesn't prove this suffices for concurrent primary/secondary transmissions under the 8-minute constraint.

- **Multi-objective Decision making (2/10):**

  3D Parameter Optimization (0/6): Single-Parameter Focus: The answer only optimizes for **N_max** ("$N(N-1)/28 \rightarrow N_{\text{max}} = 4$", Step 2) but ignores joint optimization of capability (no analysis of 158-character message limits or segment splitting efficiency), reliability (no adjustment for secondary station 80% success rate such as no retransmission strategy) and time (assumes 8 minutes suffice without validating secondary transmission overhead). Missed Pareto Frontier: Fails to explore tradeoffs (e.g., "Could N=5 work if secondary transmissions are reduced?").

  Resource Allocation Strategy (2/4): Equal Bandwidth Only: Primary stations follow a round-robin schedule ("1→2, 1→3, 1→4, 2→3, ...", Table 1), and secondaries transmit uniformly ("1→1a, 1→1b, 2→2a, ...", Table 2). No Prioritization: Critical objectives (e.g., ensuring 97% success) aren't prioritized in scheduling.

- **Uncertainty Handling (0/10):**

  High-Order Probability Events (0/6): No Threshold Calculation: The answer states secondary stations have an "80% transmission/reception success rate" (Step 1) but never computes the probability of achieving 97% success (e.g., via binomial distribution for multiple retries). Misleading Metric: The "mean secondary reports received per primary station (3.2)" (Step 4) is irrelevant to the cumulative success probability requirement.

  Asymmetric Loss (0/4): No Cost Analysis: The solution ignores idle time cost (unused transmission slots due to failures) and rental loss (penalties for delayed data delivery implied by "critical rescue operations").

- **Domain-specific Reasoning (1/10):**

  Mixed-Integer Programming (0/5): No Optimization Model: The answer derives $N_{\text{max}} = 4$ via a simple inequality ("$\frac{N(N-1)}{2} \leq 8$", Step 2) but lacks an objective function (e.g., "maximize N while meeting time/reliability constraints"), and omits integer constraints (N must be discrete) or linear relaxation techniques. Ad-Hoc Calculation: No use of MINLP (Mixed-Integer Nonlinear Programming) to jointly optimize N, transmission scheduling, and reliability.

  Fault-Tolerant Protocol Design (1/5): Basic Segmentation: Mentions "reports can split into two 50-character segments" (Step 1) but no dual verification (never states if segments are sent redundantly to different primaries) and no formal protocol (assumes secondary stations report to all primaries without fault recovery like checksums, ACKs).

F Additional Analysis

F.1 Level 1 Analysis

Minor perturbations cause performance drops, revealing shallow generalization. Figure 9 (left) presents model accuracy on Level 1 tasks across four input variants: Original, Perturbed, Knowledge-enhanced, and Math Abstraction. When problems are perturbed through minor changes in wording or numerical values, average model accuracy drops from 82.9% to 81.5%. Notably, Llama 3.3 and Qwen2.5-72B decline by 6.6% and 5.1%, respectively. This indicates that some models exhibit limited robustness and often rely on memorized phrasing or surface patterns rather than generalizable reasoning.

Explicit knowledge prompts mitigate reasoning failures in weaker models. When explicit domain knowledge—such as formulas, constants, or unit conversions—is added to the input, accuracy improves to 85.5% on average. Weaker models benefit the most: GPT-4.1 Mini gains 6.2% and Mixtral-8x7B improves by 9.3%. This pattern suggests that many errors are not caused by a complete lack of knowledge, but rather by the inability to retrieve and apply relevant concepts without targeted prompting. Explicitly embedding domain knowledge thus serves as an effective intervention for enhancing reasoning activation.

Removing contextual language highlights semantic limitations. Performance further increases to 89.4% when problems are rewritten into abstract mathematical form, removing all contextual language. For example, Qwen2.5-7B and Mixtral-8x7B improve by 10.9% and 18.8%, respectively. This reveals that most Level 1 failures are not due to weak computational ability, but rather arise during semantic interpretation and variable binding. Once language ambiguity is removed, models can more reliably execute the required calculations, underscoring a gap between symbolic proficiency and contextual understanding.

F.2 Level 2 Analysis

Level 2 tasks emphasize multi-step reasoning under structured constraints, making them more sensitive to input variability. As shown in Figure 9 (right), the average model accuracy declines from 66.6% on the Original version to 61.6% on the Perturbed variant. This 5.0% drop indicates that even minor changes to semantic phrasing or numerical values can significantly disrupt reasoning chains. For instance, GPT-4.1 Nano drops by 9.3% and Qwen2.5-7B by 11.4%, revealing their limited robustness when facing contextual and structural perturbations in problem inputs.

Incorporating explicit domain knowledge helps reduce ambiguity and recover performance. With knowledge-enhanced inputs, the average accuracy rises to 68.6%, a 7.0% improvement over the perturbed baseline. Larger gains are observed for models such as GPT-4.1 Nano (+15.4%) and Qwen2.5-7B (+16.6%), suggesting that knowledge prompts assist in constraint interpretation and formula selection. However, some models such as DeepSeek-V3 show minimal improvement, implying that knowledge access alone may not compensate for limitations in multi-step reasoning capabilities.

Symbolic abstraction of Level 2 tasks into pure mathematical form results in the largest performance gains. The average accuracy increases to 79.2%, with many models gaining over 15%. This trend is especially prominent for weaker models like Qwen2.5-7B (from 37.3% to 69.4%) and Mixtral-8x7B (from 30.0% to 57.7%). These improvements confirm that many model failures stem not from computational weakness, but from difficulties parsing, organizing, and executing the reasoning steps embedded in natural language problem statements. This underscores the importance of assessing upstream cognitive processes that precede symbolic computation—dimensions often underexamined in traditional mathematical benchmarks.

F.3 Level 3 Analysis

Figure 10 presents the performance of various models across four key capabilities: Redundant Information, Multi-Objective Decision, Domain Knowledge, and Uncertainty Handling. The results are further separated into *original* and *rewritten* problem formulations. Overall, human experts substantially outperform all models across all dimensions, with average scores of 8.552 (original) and 8.736 (rewritten). In contrast, LLMs demonstrate significantly lower scores, revealing a persistent gap between current LLMs' capabilities and human-level reasoning. The average model scores before

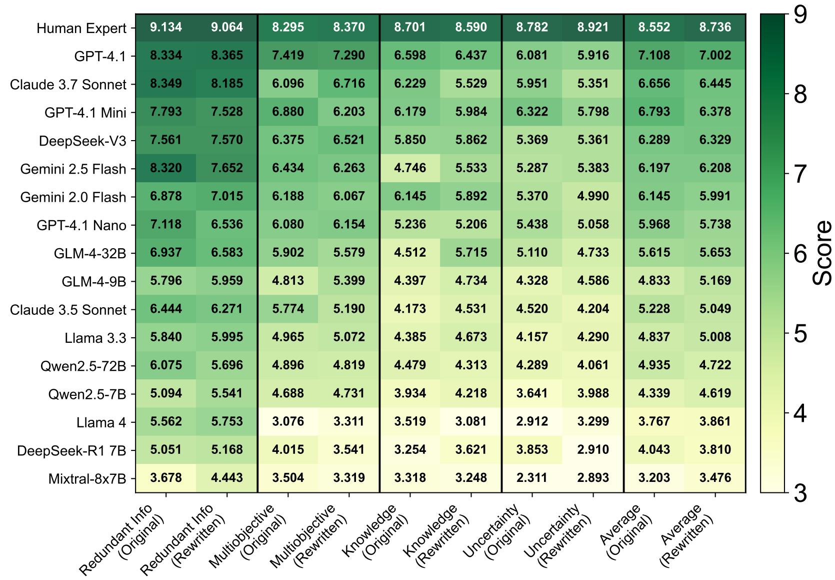


Figure 10: Level 3 Model Evaluation. The figure presents average model performance on Level 3 tasks across four capability dimensions—information extraction, domain-specific reasoning, multi-objective decision-making, and uncertainty handling—under both original and rewritten problem formulations.

and after rewriting are 5.663 and 5.617, respectively—a marginal difference of only 0.81%. This indicates that most models possess a reasonable degree of generalization, and the benchmark shows no signs of data contamination across reformulated prompts, preserving task consistency.

Based on the overall average scores, we categorize model performance into three tiers:

Tier 1 (Average Score > 6.5) This tier includes GPT-4.1, Claude 3.7 Sonnet, and GPT-4.1 Mini. These models demonstrate strong performance across all four evaluated capabilities. In particular, their scores in Information Extraction and Multi-Objective Decision often exceed 7, approaching human expert levels. Their performance in Domain Knowledge and Uncertainty Handling also remains consistently above 6, indicating robust reasoning capabilities and broad task adaptability.

Tier 2 (Average Score $\approx$ 5.7–6.5) This tier consists of DeepSeek-v3, Gemini 2.5 Flash, Gemini 2.0 Flash, GPT-4.1 Nano, and GLM-4-32B. These models achieve reasonable performance in Information Extraction and Multi-Objective Decision, but exhibit noticeable weaknesses in Domain Knowledge and Uncertainty Handling, where scores commonly fall below 6. Some models approach the 5-point threshold in these dimensions, reflecting limitations in complex reasoning and knowledge integration.

Tier 3 (Average Score < 5.7) This tier includes GLM-4-9B, Claude 3.5 Sonnet, Llama 3.3, Qwen2.5-72B, Qwen2.5-7B, Llama4, DeepSeek-R1 7B, and Mixtral-8x7B. These models consistently underperform across all four capabilities, typically scoring between 3 and 5. Their weakest areas are Domain Knowledge and Uncertainty Handling, where some models fall below 4. These results indicate substantial deficiencies in background reasoning and generalization to ambiguous or underspecified tasks.