# HARDMATH: A Benchmark Dataset for Challenging Problems in Applied Mathematics

**Jingxuan Fan**[1*], **Sarah Martinson**[1*], **Erik Y. Wang**[1*], **Kaylie Hausknecht**[1*],
**Jonah Brenner**[1], **Danxian Liu**[1], **Nianli Peng**[1], **Corey Wang**[1], **Michael P. Brenner**[1]
[1]School of Engineering and Applied Sciences, Harvard University

## Abstract

Advanced applied mathematics problems are not well-represented in existing benchmark datasets used to evaluate Large Language Models (LLMs). To address this, we introduce **HARDMATH**, a dataset of difficult problems inspired by Harvard University's graduate course on asymptoptic methods. The dataset contains a diverse set of challenging applied mathematics problems with worked solutions that use various analytical approximation methods. Developing such solutions typically requires multiple modes of analysis—including mathematical reasoning, the use of computational tools, and subjective judgment—making this a challenging problem for LLMs. We establish a framework that auto-generates an arbitrarily large number of 'hard' applied mathematics problems with solutions and includes validity checks against numerical ground-truth calculations. Additionally, we report quantitative and qualitative evaluation results on frontier open- and closed-source models using a test set of over 366 problems, as well as a "Word problems in context" test set of 40 problems. Leading closed-source models, such as GPT-4, achieve only around 50% accuracy even with few-shot Chain-of-Thought prompting, which is considerably lower than performance on existing mathematics benchmark datasets. Open-source models demonstrate an even larger performance gap, suggesting room for model fine-tuning. Our findings demonstrate the limitations of current LLMs in solving advanced graduate-level applied mathematics problems and highlight the need for datasets like **HARDMATH** to further improve LLM mathematical capabilities.

## 1 Introduction

The mathematical equations that arise in practical, scientific, and engineering problems can rarely be solved analytically. Traditional mathematics courses typically focus on equations that have exact, analytical solutions and only teach a limited set of techniques for solving them. The mathematical reasoning datasets used to benchmark large language models (LLMs) are predominantly restricted to these types of problems and techniques. However, many practical mathematics problems involve integrals, ordinary differential equations (ODEs), and partial differential equations (PDEs) that do not admit closed-form solutions and must be approached with a different set of techniques. Numerical solutions reveal insights, but do not provide intuition as to *why* solutions behave as they do. A paradigmatic approach seeks *approximate* analytical solutions to complex mathematics problems using tools from applied mathematics, such as asymptotic and applied analysis, which are currently

[*]Equal contribution and co-first authors.

Submitted to the 38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks. Do not distribute.

not well-represented in existing mathematics benchmark datasets for LLMs. To address this gap, we present **HARDMATH**, the Harvard Asymptotic Reasoning Dataset for Mathematics [2]. This dataset captures a fundamentally different type of mathematical reasoning than other benchmarks, and it can be useful for evaluating LLMs' proficiency in making research-relevant approximations.

**HARDMATH** contains 1,516 sample problems modeled after those from Harvard University's graduate course on asymptotic methods, which covers approximation methods for algebraic equations, ODEs, integrals, and PDEs that cannot be solved exactly. The course combines randomly selected coefficients, functional forms, and initial conditions to generate problems; **HARDMATH** follows similar principles. A unifying theme across these problems is careful asymptotic reasoning in which powerful heuristics are used to drastically simplify problems. As such, problems in **HARDMATH** use specialized techniques to tackle these 'messy' mathematical problems that arise in scientific and engineering contexts.

A primary motivation for developing **HARDMATH** is the lack of benchmark datasets targeting the mathematical approximation methods required in many applications. While some recent works have begun to include university-level problems [12], most datasets focus on grade school- to high school-level mathematics problems [3, 11, 7] that also cover "exact" mathematical reasoning (e.g., problems that are either proof-based [24, 26] or only involve direct, clean calculations). However, **HARDMATH** targets applied mathematics problems that require approximate solutions, which is an equally important area of mathematical reasoning. Finding approximate analytical solutions can be challenging even for those with high levels of mathematical proficiency, as it requires advanced techniques from calculus, differential equations, and complex analysis. Computational tools are also needed to explore the behavior of different terms in each equation and to find numerical solutions that can be compared with approximate solutions. Additionally, subjective choices about the regimes of the solution space to consider, the number of terms to include in approximate expressions, and the approximation methods themselves must be made on a case-by-case basis with rigorous mathematical justification.

Rather than the typical procedure of collecting problems from textbooks, standardized tests, or competitions like most existing datasets, we develop algorithms to automatically generate problems and step-by-step solutions. We design a comprehensive testing methodology to assess leading LLMs' mathematical reasoning abilities in this approximation methods domain, presenting evaluation accuracy and error modes analysis. Specifically, we show that the *current* performance of existing LLMs on these problems is poor, leaving much room for future improvement.

## 2 Related work

LLMs have shown promising capabilities in mathematics. However, evaluating and expanding these abilities requires diverse datasets with problems that go beyond basic arithmetic or elementary word problems. Existing benchmarks often focus on these simpler domains, with a gap in addressing graduate-level applied mathematics problems that demand a deeper understanding and diverse, multi-modal analytical skills. Most mathematics datasets for evaluating or training LLMs consist of word problems that either present the problem directly or within a constructed narrative context. Notable examples of these datasets include **MATH** (12,500 high school competition-style problems) [11], **GSM8K** (8,500 multistep grade-school problems) [7], **MATHQA** (37,000 GRE/GMAT-level multiple-choice problems) [3], **ODYSSEY-MATH** (387 hand-curated problems across various difficulty levels) [15], and [25] (a bilingual concept-wise mathematical dataset). While these existing datasets are valuable for assessing LLM math performance in certain areas, most are limited in scope and complexity.

Recent efforts in the field target more advanced problems that are most often manually-sourced. Relevant works for the former include a subset of the **MATHBENCH** dataset and **JEEBENCH**, both of which cover some college-level topics like simple ordinary differential equations and multivari-

[2]The dataset and code used to generate it are available at https://github.com/sarahmart/HARDMath

Table 1: Comparison of **HARDMATH** with related datasets. For each dataset (excluding **MATH** and **GSM8K**), we report the number of relevant problems at a comparable difficulty to our dataset (e.g., **THEORY-KNOWLEDGE-COLLEGE** in **MATHBENCH**, and **GRAD-TEXT** and **HOLES-IN-PROOFS** from **GHOSTS**.) **HARDMATH** is the largest graduate-level dataset.

| Dataset | Size | Data Generation | Difficulty |
|---|---|---|---|
| **MATH** (Hendrycks et al., 2021) | 12.5K | Manual | High School |
| **GSM8K** (Cobbe et al., 2021) | 8.5K | Manual | Grade School |
| **MATHBENCH-T** (Liu et al., 2024) | 632 | Manual, Algorithmic | Undergraduate |
| **JEEBENCH** (Arora et al., 2023) | 236 | Manual | High School |
| **GHOSTS** (Frieder et al., 2023) | 190 | Manual | Graduate |
| **ARB** (Sawada et al., 2023) | 34 | Manual | Graduate |
| **HARDMATH** (Ours) | 1.5K | Algorithmic | Graduate |

able calculus [12] [4]. There have also been more advanced datasets, such as **GHOSTS**, which includes **GRAD-TEXT**—a collection of 130 exercises from graduate-level mathematics textbooks in functional analysis, topology, and probability theory [9]—and **ARB**, which features a small set of university-level formal mathematics problems from prior qualifying examinations in the mathematics departments at Harvard University and the University of California, Berkeley [20]. However, these datasets have several limitations, such as their scalability and size; datasets created by scraping textbooks or similar resources are generally quite small. Some of these challenging datasets also only focus on abstract, formal mathematics and exclude other forms of mathematical reasoning. Finally, textbook problems are often protected by copyright, which can complicate their public use.

One area of LLM research that is of broad and current interest is developing models that are equipped with the ability to use tools. The approximate nature of the solutions to the problems in our dataset makes it unique among other benchmarks. For example, solving these problems cannot be formalized with Lean or other softwares. To achieve high performance on the problems in our dataset, LLMs will need to make sophisticated use of tools and correctly intersperse tool use with other forms of reasoning. As such, one of the greatest strengths of HARDMath is that it is uniquely suited to be a valuable mathematical dataset for benchmarking and developing LLMs capable of effective tool use.

Existing datasets thus lack the scale and specific focus needed to evaluate LLMs on advanced mathematical problems that may be useful for scientific research. They prioritize basic word problems, focus on proving theorems, or manually curate problems from textbooks because it is difficult to generate large sets of new problems and solutions. **HARDMATH** aims to address these limitations by offering a large collection of challenging applied mathematics problems inspired by a graduate-level course on asymptotic methods. It emphasizes problems that require diverse mathematical approaches, numerical calculations, and subjective judgment, mirroring the complexity of problems faced by researchers in a variety of domains. Our code for auto-generating the problems in **HARDMATH** can be used to generate any number of additional problems, which is a unique and powerful feature for scaling LLM benchmarking and training.

# 3 Datasets

## 3.1 HARDMATH design choices

Here, we describe the **HARDMATH** dataset, which contains problems on polynomial nondimensionalization, polynomial root-finding, ODEs, integrals, and word problems that contextualize each of these. **HARDMATH** contains four classes of problem, totaling seven distinct problem types. The main dataset, which can be used for LLM training or fine-tuning, contains 1,050 problems, and the evaluation dataset, which we use in this paper to benchmark LLM performance, contains 437 problems.

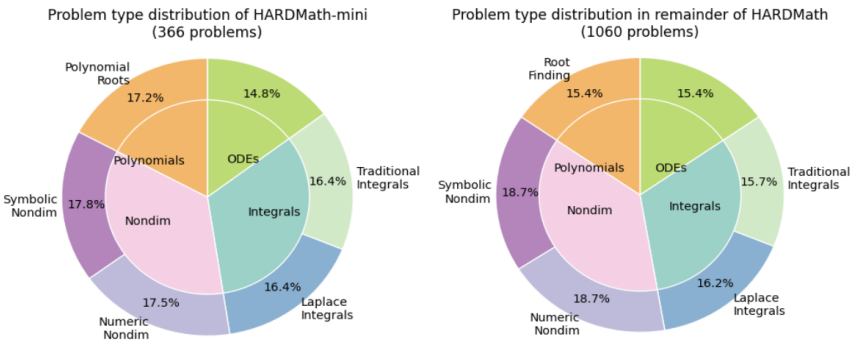



Figure 1: Statistics of the test (left) and train (right) sets.

One key commonality between these problems is the *Method of Dominant Balance*, which reduces an equation to only the terms that "dominate" the behavior of the solution [5]. Our problems also involve other sophisticated mathematical techniques, such as checks for self-consistency and the use of numerical methods. The combination of these tools captures several key aspects of mathematical modeling, including the use of both computational and analytical techniques, and identifying which components to prioritize in the model, which we believe LLMs may struggle with.

### 3.2 Dataset generation and verification

The data generation code uses `SymPy` [14], a library for symbolic mathematics, and `SciPy`, a library for scientific computing [22], to implement the mathematical procedures required for obtaining approximate, analytical solutions. Parameters are randomly sampled according to a sample space described in Appendix A.1, and the solutions are generated by navigating through a set of possible cases during the algorithmic problem-solving strategy. Each mathematical step is embedded in explanatory text so **HARDMATH** solutions match the style and rigor of traditional problem set solutions. The main results for all problems are included in boxed environments in the solution explanations to distinguish them from the rest of the text. This follows the formatting convention used in other mathematics datasets designed for LLM training, such as the **MATH** dataset [11].

For each problem type, the dataset includes: 1) LaTeX-formatted problem statements with prompts, 2) LaTeX-formatted solution steps and final analytical answer(s), 3) Demonstration of the accuracy of the analytical results by comparing with numerical solutions, and 4) Metadata descriptors of the problem and solution types.

For every problem type, we select evaluation points in each solution regime and calculate the relative error between the analytical solution and the numerical solution at these points. Problems were included in **HARDMATH** only if their approximate solutions had less than 10% error from the numerically calculated ground-truths. For the polynomial root correction problems, we also confirm that the corrections improve the original approximation. Manually verifying solutions step-by-step is not feasible for a dataset of this size, but this verification procedure provides a high degree of confidence in the accuracy of the solutions in the dataset. As an example, Figure 2 provides a visual comparison of the numerical and approximate solutions for the Laplace Integral example over a large domain.

### 3.3 Problem types

#### 3.3.1 Nondimensionalization of polynomials

Nondimensionalization is a powerful technique used to simplify equations by reducing the number of its parameters[8]. In **HARDMATH**, the first type of polynomial covered contains symbolic coefficients of the form

$$a_1 x^{n_1} + a_2 x^{n_2} + a_3, \quad n_1 > n_2 > 0 \tag{1}$$

and converts it to the form $\epsilon y^{n_1} + y^{n_2} + 1$. The second type introduces numerical coefficients to create problems with the form

$$\pm a_1 x^{n_1} \pm a_2 x^{n_2} \pm a_3, \;\; n_1 > n_2$$

which can be simplified to $\epsilon y^{n_1} \pm y^{n_2} \pm 1$ given a specific *numerical* value of $\epsilon$.

For all problem types described in this section, we provide more details about the parameters used to generate the problems and the mathematical techniques used to solve the problems in Appendix A.

#### 3.3.2 Polynomial root-finding

Exact formulas exist for quadratic, cubic, and quartic equations, but deriving them for quintic or higher-order polynomials is impossible [21]. **HARDMATH** includes approximate root-finding examples for higher order polynomials of the form $\epsilon x^{n_1} \pm x^{n_2} \pm 1$ (example in Appendix A.1.2). The goal is to solve for roots in terms of $\epsilon$ using the method of dominant balance for small and large positive $\epsilon$ regimes.

#### 3.3.3 Polynomial root correction terms

The previous problem type uses a two-term dominant balance to find an approximate solution; however, this neglects one term in the polynomial root calculations and introduces an error. We can calculate arbitrarily many correction terms to reduce this error. This problem subtype focuses on first- and second-order corrections.

The true roots $x^*$ of a polynomial are given by $x^*(\epsilon) = \overline{x}(\epsilon) + \delta$, where $\overline{x}$ is our approximation to the root and $\delta$ is the error term. Plugging the roots $x^*(\epsilon) = \overline{x}(\epsilon) + \delta$ into the polynomial gives

$$\epsilon(\overline{x} + \delta)^{n_1} \pm (\overline{x} + \delta)^{n_2} \pm 1 = 0. \tag{2}$$

#### 3.3.4 Nonlinear ordinary differential equations

We generate nonlinear third-order ODEs for which there do no exist exact analytical solutions and provide approximate formulae for small and large $x$ regimes, where the small $x$ regime is near $x = 0$ and the large $x$ regime typically involves the solution diverging (example in Appendix A.1.4). The general method is robust for higher-order problems, but for simplicity we include only third-order ODEs.

#### 3.3.5 Traditional integrals

We consider integrals of the form $I(\epsilon) = \int_0^a \frac{1}{\epsilon + P(x)}\, dx$ where $P(x)$ is an arbitrary polynomial. The dataset provides approximations of each integral in three regimes (small, intermediate, and large $\epsilon$).

#### 3.3.6 Laplace integrals

We consider integrals of the form $I(x) = \int_a^b g(t) e^{\pm x f(t)} dt$, which can be approximated using Laplace's Method when $x$ is very large because the integral's value is dominated by the region around $t_0$ [5]. Depending on the where the minimum is, the approximation is either

$$I(x) \approx g(t_0) e^{\pm x f(t_0)} \sqrt{\frac{2\pi}{x|f''(t_0)|}} \quad \text{or} \quad I(x) \approx \frac{g(t_0) e^{\pm x f(t_0)}}{x|f''(t_0)|.}$$

**1. Sample Laplace Integral Problem and Final Analytical Solution**

**Problem:** Consider the integral

$$I(x) = \int_{0.4}^{0.8} (-2.4t^2 - 2.8\tan^{-1}(t)) e^{-x(1.4t^3 - 2.6\cos(t) + 1.3\tan^{-1}(t) + 0.4)}\, dt.$$

Develop an analytical formula for $I(x)$ that is accurate as $x$ becomes large.

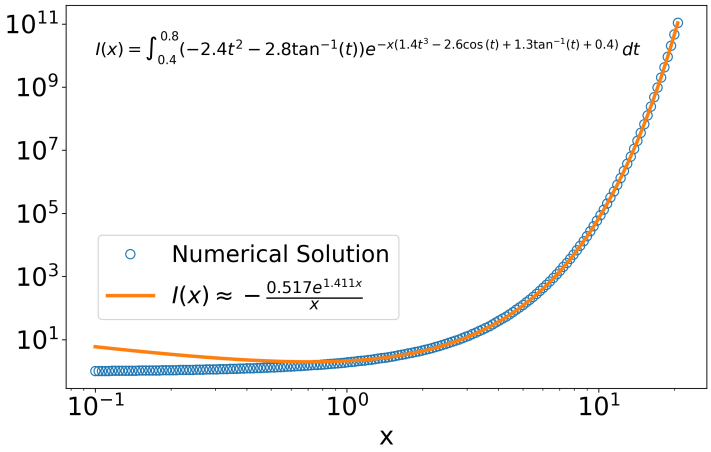



Figure 2: Visual verification by comparing the analytical solution to the numerical solution.

**Final Analytical Solution:**

$$\boxed{I(x) \approx -\frac{0.517e^{1.411x}}{x}}$$

### 3.4 Word problems in context

A motivation for creating **HARDMATH** is to help LLMs recognize and solve problems where approximation techniques are needed. To evaluate how LLMs perform on problems that require asymptotic reasoning in the context of realistic scenarios, we develop a smaller dataset of 40 word problems, written manually (see example in Box 2). Although this dataset is smaller than our hand-verified evaluation set, it is large enough to evaluate the effect of additional context in the problem statement on LLM accuracy.

2. Sample Word Problem with Context

The density of fish at different points along a certain path in a lake can be modeled as $(\epsilon + x^2 + x^5)^{-1}$, where $x$ represents the distance from the shore in kilometers (ranging from 0 to 100 km), and $\epsilon$ represents environmental factors that affect the fish density. To study the total presence of fish along the path, develop an approximate analytical formula for $I(\epsilon)$ given below:

$$I(\epsilon) = \int_0^{100} \frac{1}{\epsilon + x^2 + x^5}\, dx.$$

## 4 Evaluation

### 4.1 Evaluation protocols

We carry out LLMs evaluations on a subset of 437 problems from **HARDMATH**. We separately evaluate four problem types: 1) *Nondim* includes nondimensionalization in symbolic and numerical form; 2) *Roots* includes polynomial root-finding; 3) *ODEs* includes nonlinear ODEs; and 4) *Integrals* includes traditional and Laplace integrals. We leave out the polynomial root-finding with correction terms in the evaluation because neither scoring methods detailed below give consistent and reliable accuracy reports. The input prompt comprises the essential problem setup and a description of the question, supplemented with problem type-specific hints for answer formatting. Few-shot prompting

adds a fixed set of paired problem-solution examples from the corresponding problem types. Example prompts can be found in Appendix A.2.1, Table 6.

Model-generated responses are evaluated and scored for accuracy using a protocol combining automatic assessment of final answers and procedural LLM-based grading. Automatic assessment follows[11], prompting models to wrap the final answer with the LaTeX `\boxed{}` command (Table 6). Evaluation then compares the model's output within `\boxed{}` command to **HARDMATH** solution. To handle different mathematical expression formats, we implement both `SymPy`-based [14] equivalence checks, as well as numerical evaluations.

In addition to the standard automatic assessment of final answers, we developed a novel procedural grading approach leveraging LLMs, tailored to the unique nature of our dataset. Our dataset presents two unique challenges compared to existing mathematics datasets: 1) Some problem types require complex, multi-step solution procedures where a single cut-off criterion at the final answer cannot capture the full spectrum of model performance. Thus, intermediate checkpoints in the solving procedure are necessary for comprehensive assessment (Appendix A.2.2 Table 4). 2) **HARDMATH** targets the model's ability to make human-like abstraction and approximation judgments. Some problem types allow a narrow range of solutions rather than a single exact one, as long as the reasoning is self-consistent and the final result agrees with the numerical ground truth.

Inspired by LLMs' ability to generate consistent ratings for response content and style [10], we use GPT-4o as a procedural grader, prompted with the ground truth answer key and clear grading rubrics for each problem type (example grading prompts in Appendix A.2.2 Table 4). We hand-verified a subset of grading responses and found that LLM-based grading is closely aligned with human grading used in the graduate course. We implemented this procedural grading alongside automatic final answer assessment for the problem types *Roots*, *ODEs*, and *Integrals*.

### 4.2 Model choice

We compare the performance of several closed- and open-source models with different prompting methods on **HARDMATH**: 1) Closed-source LLMs include GPT-3.5 [18, 19, 17] and GPT-4 [1] in zero- and few-shot settings with the Chain-of-Thought (CoT) [23] prompting, and 2) Open-source LLMs include Llama 3 [2] and CodeLlama [13] in zero- and few-shot settings with CoT prompting. We provide the prompts and hyper-parameters for LLMs evaluations in Appendix A.2.4 Table 6.

### 4.3 Quantitative results

We present the accuracy of the models and prompting settings for each problem type and the combined test set (Table 2, Figure 4). Few-shot CoT prompting significantly boosts performance for all models, with GPT-4 showing the greatest improvement, consistent with [23] (Figure 4a). The varying performance increases among different problem types may be due to different error modes in model answers, which we discuss in the following section. Among closed-source models, GPT-4 with 5-shot CoT prompting achieves the highest overall accuracy of 46.3%. Among open-source models, Llama3-8b with 5-shot CoT prompting achieves the highest overall accuracy of 21.1%. We discuss the performance of these representative models—GPT-4 and Llama3—on **HARDMATH** in comparison with established datasets, including **GSM-8K** [7], **MATH** [11], and more advanced mathematics datasets like **GHOSTS** [9].

Llama3-8b achieves a test accuracy of 30.0% on the **MATH** dataset with 4-shot CoT and 79.6% on the **GSM-8K** dataset with 8-shot CoT prompting [2]. Testing Llama3-8b on **HARDMATH** resulted in an overall accuracy of 21.1% with 5-shot CoT prompting. GPT-4 (gpt-4-turbo-2024-04-09) is reported to achieve 72.2% accuracy on the **MATH** dataset with 0-shot CoT prompting [16] and 92.0% on the **GSM-8K** dataset with 5-shot CoT prompting [1]. On a recently released dataset **MINIGHOSTS**, which measures the advanced mathematical abilities of LLMs, GPT-4 reaches an average score of 4.15 out of 5. We tested GPT-4 on our **HARDMATH** dataset and obtained an overall accuracy of 46.3% with 5-shot CoT prompting. Even with few-shot CoT prompting, the test accuracy of both models on all **HARDMATH** problem types is considerably lower than on existing

benchmark mathematics datasets. This suggests that our problems are more challenging particularly in the domains of mathematical reasoning required by our dataset. Notably, GPT-4 also performs poorly on a subset of **MINIGHOSTS** taken from graduate mathematics textbooks, scoring only 3.5 out of 5. These results highlight the value of **HARDMath** as a challenging benchmark for evaluating the mathematical capabilities of LLMs.

#### 4.3.1 Extensions to word problems

We test the best-performing model, GPT-4, on a set of word problems that included a mixture of *Nondim*, *Roots*, *ODEs*, and *Integrals*. Within *Integrals*, we did not generate Laplace integral word problems. To assess the model's ability to solve problems in relevant research contexts, we avoided additional prompt engineering and used only the questions as prompts, omitting the problem-specific hints listed in Table 3. This evaluation resulted in an overall accuracy of 28.1%.

Table 2: Evaluation Accuracy (percentage) on the **HARDMATH** evaluation set.

| Model | ALL | Nondim | Roots | ODEs | Integrals |
|---|---|---|---|---|---|
| **Closed-source models** | | | | | |
| GPT-3.5 (0 shot) | 4.54 | 5.05 | 17.2 | 7.41 | 3.33 |
| GPT-3.5 (1 shot CoT) | 14.8 | 6.11 | 28.9 | 22.6 | 18.2 |
| GPT-3.5 (5 shot CoT) | 25.6 | 24.3 | 29.3 | 32.1 | 23.1 |
| GPT-4 (0 shot) | 13.4 | 6.04 | 32.1 | 15.7 | 14.9 |
| GPT-4 (1 shot CoT) | 37.6 | 36.5 | 50.4 | 20.3 | 40.5 |
| GPT-4 (5 shot CoT) | 46.3 | 48.6 | 58.1 | 34.7 | 41.4 |
| **Open-source models** | | | | | |
| Llama3-8b (0 shot) | 3.98 | 0.50 | 11.3 | 11.6 | 2.52 |
| Llama3-8b (5 shot CoT) | 21.1 | 17.9 | 17.5 | 21.9 | 28.1 |
| CodeLlama-13b (0 shot) | 3.13 | 0.00 | 8.70 | 14.1 | 0.50 |
| CodeLlama-13b (5 shot CoT) | 10.3 | 8.41 | 13.1 | 15.3 | 9.57 |

### 4.4 Qualitative analysis

In addition to reporting the summarized test accuracy, we study the specific error modes of LLMs solving these challenging applied mathematics questions. This analysis helps us compare the performance nuances and understands reasoning paths from axes of model, prompting technique and question type.

We first break down model performance by percentage of correct, partial and incorrect responses (Figure 3). This analysis reveals how few-shot prompting enhances model performance across varying problem types. For complex problems like ODEs and integrals, full correctness is rare. Here, models tend to increase partial credit responses with CoT prompting, as they struggle to solve the problems entirely but manage to partially address them. In contrast, for simpler problems like polynomial root finding, more advanced models like GPT-3.5 and GPT-4 achieve full correctness more often.

Consistent with the parameter-size dependent scaling law reported by [6], GPT-4 shows the best performance increase upon few-shot prompting across all problems. Figure 5 illustrates how 5-shot CoT prompting significantly alters the error structure compared to 0-shot. The most common error mode—incorrectly setting up dominant balance by considering only the leading term—diminishes substantially. Instead, errors shift to more nuanced issues: 1) setting up correct dominant balances but missing certain cases, or 2) failing to calculate complex roots. This shift indicates that CoT prompting improves the model's understanding and application of dominant balance techniques, enabling it to move beyond intuitive yet incorrect simplifications. More broadly, our unique dataset provides rich insights into the strengths and weaknesses of LLM reasoning in mathematical contexts that demand approximation skills. This analysis underscores the importance of our benchmark in evaluating and improving LLMs' ability to tackle graduate-level applied mathematics problems.

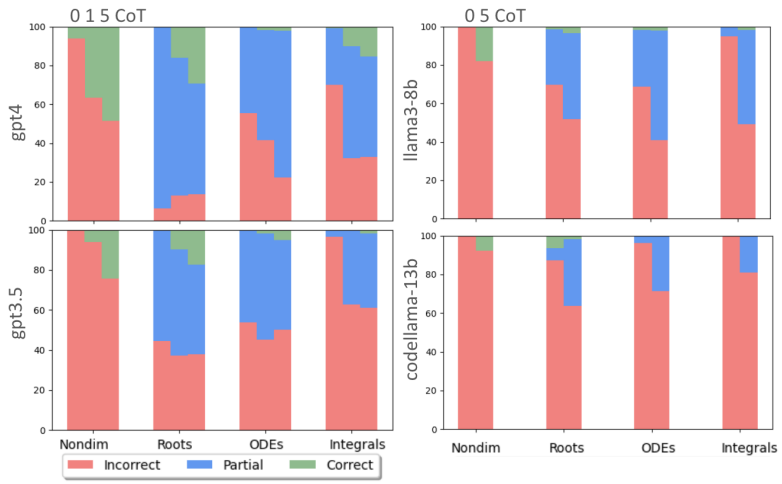



Figure 3: Percentage of correct, partial and incorrect responses for all models, prompting conditions and problem types.

3. Grading Response: *Roots*

**Question:** Consider the polynomial $P(x) = \epsilon x^8 + x^4 - 1$. Find approximate expressions for all roots of the polynomials in the limit of small positive $\epsilon$ and large positive $\epsilon$. Only a single term approximation to the root is required.
**Grading for Small Positive $\epsilon$**:
**Model Response:** For small $\epsilon$ : $[1, -1, i, -i]$
**Ground Truth**:

$$\text{For small positive } \epsilon: \left[-\sqrt[4]{-\frac{1}{\epsilon}},\ \sqrt[4]{-\frac{1}{\epsilon}},\ -i\sqrt[4]{-\frac{1}{\epsilon}},\ i\sqrt[4]{-\frac{1}{\epsilon}},\ -1,\ 1,\ -i,\ i\right]$$

The response only includes the roots from the balance $B + C = 0$ and completely misses the roots from the balance $A + B = 0$. Therefore, score for small positive $\epsilon$ is $\boxed{0.5}$

## 5 Conclusion

We introduce **HARDMATH**, a new dataset that covers several problem types from an advanced applied mathematics course that can be used to train LLMs and evaluate their mathematical capabilities. The dataset consists of 1060 training examples and 366 examples for testing, and 40 verified 'problems in context.' **HARDMATH** is unique in several ways: 1) there do not exist large-scale mathematical datasets covering problems of similar difficulty or from applied mathematics, and 2) the dataset's problems and solutions are algorithmically generated, meaning that one could produce datasets of arbitrary size using our framework. This feature of **HARDMATH** is especially unique, since most existing mathematical datasets require manual problem-setting or curation from other sources (many of which are not publicly accessible).

**HARDMATH** is intended for use as a comprehensive evaluation set, but future work will involve fine-tuning open-source LLMs on a larger training set generated by our framework. If a fine-tuned LLM could achieve mastery on **HARDMATH**, it could replace or aid human teaching assistants. This type of "LLM-assistant" could help students determine which approximation methods to use, how to apply them, and even grade homework using our procedural evaluation framework. The course also covered several other topics that are not included in this initial dataset, such as boundary value problems and PDEs, which can also be algorithmically generated following our framework.

## 6 Acknowledgments

We thank the students who participated in the initial stages of the Fall 2023 AM 201 final project. Thanks also to the Harvard Medical School Research Computing Consultant Group for their consulting services, which facilitated the computational analyses detailed in this paper.

# 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] Meta AI. Meta llama 3, 2024. URL `https://ai.meta.com/blog/meta-llama-3/`. Accessed: 2024-06-03.

[3] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of NAACL-HLT*, pages 2357–2367, 2019.

[4] 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.

[5] Carl M Bender and Steven A Orszag. *Advanced mathematical methods for scientists and engineers I: Asymptotic methods and perturbation theory*. Springer Science & Business Media, 2013.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL `https://arxiv.org/abs/2005.14165`.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL `https://arxiv.org/pdf/2110.14168v1`.

[8] John H Evans. Dimensional analysis and the buckingham pi theorem. *American Journal of Physics*, 40(12):1815–1822, 1972.

[9] Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. Mathematical capabilities of chatgpt. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Veronika Hackl, Alexandra Elena Müller, Michael Granitzer, and Maximilian Sailer. Is gpt-4 a reliable rater? evaluating consistency in gpt-4's text ratings. *Frontiers in Education*, 8, December 2023. ISSN 2504-284X. doi: 10.3389/feduc.2023.1272229. URL `http://dx.doi.org/10.3389/feduc.2023.1272229`.

[11] 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. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks*. NeurIPS, 2021.

[12] Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. *arXiv preprint arXiv:2405.12209*, 2024.

[13] Meta. Code llama: Ai for coding, 2023. URL `https://about.fb.com/news/2023/08/code-llama-ai-for-coding/`. Accessed: 2024-06-03.

[14] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL `https://doi.org/10.7717/peerj-cs.103`.

[15] Netmind.AI. Odyssey-math. `https://github.com/protagolabs/odyssey-math/tree/main`, 2024. Accessed: April 22, 2024.

[16] OpenAI. Simple evals. `https://github.com/openai/simple-evals?tab=readme-ov-file#user-content-fn-1-43aa11412dfb93b343474c8d56f8882f`, 2024. Accessed: 2024-06-03.

[17] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL `https://api.semanticscholar.org/CorpusID:160025533`.

[20] Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J Nay, Kshitij Gupta, and Aran Komatsuzaki. Arb: Advanced reasoning benchmark for large language models. *arXiv preprint arXiv:2307.13692*, 2023.

[21] Ian Stewart. *Galois Theory*. CRC Press, Taylor & Francis Group, Boca Raton, FL, 4th edition, 2015. ISBN 978-1-4822-4583-7. Version Date: 20150112.

[22] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[24] Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[25] Yanan Wu, Jie Liu, Xingyuan Bu, Jiaheng Liu, Zhanhui Zhou, Yuanxing Zhang, Chenchen Zhang, Zhiqi Bai, Haibin Chen, Tiezheng Ge, et al. Conceptmath: A bilingual concept-wise benchmark for measuring mathematical reasoning of large language models. *arXiv preprint arXiv:2402.14660*, 2024.

[26] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2021.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 3 and Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We discuss potential societal impacts but do not consider them to be negative.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The data, code used for data generation, evaluation, and corresponding documentation are available at at this GitHub repository, which is also linked in the paper text.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix and GitHub repository for all code generation details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section A.2.5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Section 3 cites relevant code assets and Section 4.2 cites LLMs used.
   (b) Did you mention the license of the assets? [Yes] Llama3 and CodeLlama models and weights are released under the Meta AI license and used for both researchers and commercial entities. For detailed licensing information, refer to `https://github.com/meta-llama`.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The dataset and codebase are available on GitHub. They are also described in the Supplementary Material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Appendix

## A.1 Implementation and method details for data generation

The following subsections detail the process used to generate the problems and solutions for each problem type.

### A.1.1 Nondimensionalization of polynomials

The first nondimensionalization sub-type is generalized by varying the integer values for the degrees $n_1$ and $n_2$ within the range $0 < n_2 < n_1 < 10$, while keeping $a_1, a_2, a_3 > 0$ symbolic. Solutions to these problems express the dimensionless parameter $\epsilon$ in terms of these three coefficients.

**Sample Symbolic Nondimensionalization Problem and Full Solution**

**Problem:** Nondimensionalize the polynomial

$$a_1 x^{10} + a_2 x^9 + a_3$$

into one of the form $\epsilon y^{10} + y^9 + 1$. Express $\epsilon$ as a function of $a_1$, $a_2$, and $a_3$.

---

**Solution:** We begin with the substitution

$$x = y \sqrt[9]{\frac{a_3}{a_2}}$$

This gives the expression

$$a_1 y^{10} \left(\frac{a_3}{a_2}\right)^{\frac{10}{9}} + a_3 y^9 + a_3$$

Divide by the coefficient remaining in front of the constant, leaving us with the nondimensionalized polynomial with coefficients in terms of $a_1$, $a_2$, and $a_3$:

$$\boxed{\frac{a_1 y^{10} \left(\frac{a_3}{a_2}\right)^{\frac{10}{9}}}{a_3} + y^9 + 1.}$$

By inspection, we can see that

$$\boxed{\epsilon = \frac{a_1 \left(\frac{a_3}{a_2}\right)^{\frac{10}{9}}}{a_3}.}$$

The second subtype implements integer numerical values for the coefficients $a_1$, $a_2$, $a_3$ that are are randomly chosen from $[-10, 10]$.

**Sample Numeric Nondimensionalization Problem and Full Solution**

**Problem:** Nondimensionalize the polynomial

$$P(x) = 2x^7 + 8x^2 + 5$$

into a polynomial of the form $\epsilon y^7 \pm y^2 \pm 1$. Solve for $\epsilon$.

---

**Solution:** For now, we ignore the numeric values of the coefficients and instead call them $a_1, a_2, a_3$. Our polynomial is then:

$$a_1 x^7 + a_2 x^2 + a_3.$$

Use the substitution

$$x = y \sqrt{\frac{a_3}{a_2}},$$

which gives the expression

$$a_1 y^7 \left(\frac{a_3}{a_2}\right)^{\frac{7}{2}} + a_3 y^2 + a_3.$$

Divide all terms by the coefficient remaining in front of the constant term, giving us the nondimensionalized polynomial with coefficients in terms of $a_1, a_2, a_3$:

$$\frac{a_1 y^7 \left(\frac{a_3}{a_2}\right)^{\frac{7}{2}}}{a_3} + y^2 + 1$$

Substituting in the known numeric values for $a_1, a_2, a_3$ (using their absolute values as we have already accounted for sign), we get:

$$\frac{25\sqrt{10}y^7}{1024} + y^2 + 1$$

From inspection of this nondimensionalized equation, we can now identify $\epsilon$:

$$\epsilon = \frac{25\sqrt{10}}{1024} \implies \boxed{\epsilon \approx 0.08.}$$

### A.1.2 Polynomial root-finding

As with the nondimensionalization problems, degrees in the polynomial are randomly generated with maximum order ten and $0 < n_2 < n_1$. See a full problem and solution below.

**Sample Polynomial Root-finding Problem and Full Solution**

**Problem:** Consider the polynomial

$$P(x) = \epsilon x^6 - x^5 + 1.$$

Find first order approximations for all roots of the polynomials in the limit of small positive $\epsilon$ and large positive $\epsilon$.

---

**Solution:** We begin by equating the polynomial to zero to solve for the roots: $P(x) = 0$. This problem can be rewritten in the form $A + B + C = 0$, where: $A = \epsilon x^6$; $B = -x^5$; $C = 1$.
This problem has no analytical solutions, so we find approximate solutions to the roots by considering the three possible dominant balances. For each dominant balance, we find the roots of the resulting equation and evaluate whether each balance is self-consistent for small or large positive $\epsilon$.

We start with the balance $A + B = 0$, assuming that $|C|$ is negligible when compared to $|A|$ and $|B|$. Solving this for $x$ in terms of $\epsilon$ then gives us 1 non-zero root:

$$\epsilon x^6 - x^5 = 0$$
$$\implies \boxed{x = \left[\frac{1}{\epsilon}\right].}$$

To verify that these roots are consistent with the assumption that $|A|, |B| \gg |C|$, we substitute these found roots back into the terms $A$, $B$, and $C$ and compare their magnitudes. Using this method, we find that it is true that these roots are valid for small $\epsilon$, while validity for large $\epsilon$ is false.
Therefore, these roots are valid in the limit of small positive $\epsilon$ only.

Next we examine the balance $B + C = 0$, assuming that $|A|$ is negligible when compared to $|B|$ and $|C|$. Solving this for $x$ in terms of $\epsilon$ gives us 5 non-zero roots:

$$1 - x^5 = 0$$

$$\implies \boxed{\begin{array}{c} x = 1,\ -\frac{1}{4} + \frac{\sqrt{5}}{4} - \frac{i\sqrt{2\sqrt{5}+10}}{4},\ -\frac{1}{4} + \frac{\sqrt{5}}{4} + \frac{\sqrt{-10-2\sqrt{5}}}{4}, \\ -\frac{\sqrt{5}}{4} - \frac{1}{4} - \frac{i\sqrt{10-2\sqrt{5}}}{4},\ -\frac{\sqrt{5}}{4} - \frac{1}{4} + \frac{i\sqrt{10-2\sqrt{5}}}{4} \end{array}}.$$

To verify that these roots are consistent with the assumption that $|B|, |C| \gg |A|$, we substitute these found roots back into $A$, $B$, and $C$ and compare their magnitudes. Using this method, we find that it is true that these roots are valid for small $\epsilon$, while validity for large $\epsilon$ is false. Therefore, these roots are valid in the limit of small positive $\epsilon$ only.

Finally, we examine the balance $A + C = 0$, assuming that $|B|$ is negligible when compared to $|A|$ and $|C|$. Solving this for $x$ in terms of $\epsilon$ gives us 6 non-zero roots:

$$\epsilon x^6 + 1 = 0$$

$$\implies \boxed{\begin{array}{c} x = \left[ -\sqrt[6]{-\frac{1}{\epsilon}},\ \sqrt[6]{-\frac{1}{\epsilon}},\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(-1-\sqrt{3}i\right)}{2}, \right. \\ \left. \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(-1+\sqrt{3}i\right)}{2},\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(1-\sqrt{3}i\right)}{2},\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(1+\sqrt{3}i\right)}{2} \right] \end{array}}.$$

To verify that these roots are consistent with the assumption that $|A|, |C| \gg |B|$, we substitute these found roots back into $A$, $B$, and $C$ and compare their magnitudes. Using this method, we find that it is false that these roots are valid for small $\epsilon$, while validity for large $\epsilon$ is true. Therefore, these roots are valid in the limit of large positive $\epsilon$ only.

By the Fundamental Theorem of Algebra, a polynomial of degree 6.0 has exactly 6.0 roots.We have found 6.0 roots that are valid in the limit of small positive $\epsilon$ and 6.0 roots valid in the limit of large positive $\epsilon$. Our method therefore provides a complete solution to the problem, finding the correct number of roots in each $\epsilon$ regime.
The roots of $P(x)$ for large positive $\epsilon$ are

$$\boxed{\begin{array}{c} -\sqrt[6]{-\frac{1}{\epsilon}},\ \sqrt[6]{-\frac{1}{\epsilon}},\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(-1-\sqrt{3}i\right)}{2}, \\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(-1+\sqrt{3}i\right)}{2},\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(1-\sqrt{3}i\right)}{2},\ \frac{\sqrt[6]{-\frac{1}{\epsilon}}\left(1+\sqrt{3}i\right)}{2} \end{array}}$$

and the roots of $P(x)$ for small positive $\epsilon$ are

$$\boxed{\begin{array}{c} \frac{1}{\epsilon},\ 1,\ -\frac{1}{4} + \frac{\sqrt{5}}{4} - \frac{i\sqrt{2\sqrt{5}+10}}{4},\ -\frac{1}{4} + \frac{\sqrt{5}}{4} + \frac{\sqrt{-10-2\sqrt{5}}}{4}, \\ -\frac{\sqrt{5}}{4} - \frac{1}{4} - \frac{i\sqrt{10-2\sqrt{5}}}{4},\ -\frac{\sqrt{5}}{4} - \frac{1}{4} + \frac{i\sqrt{10-2\sqrt{5}}}{4} \end{array}}$$

### A.1.3 Polynomial root correction terms

The true roots $x^*$ of a polynomial are given by $x^*(\epsilon) = \overline{x}(\epsilon) + \delta$, where $\overline{x}$ is our existing approximation to the root as found in Appendix A.3 and $\delta$ is the error term. This requires us to solve

$$\epsilon(\overline{x} + \delta)^{n_1} \pm (\overline{x} + \delta)^{n_2} \pm 1 = 0$$

for $\delta$ by equating coefficients of $\epsilon$ terms of the same order, as detailed in the worked solution below.

Sample Numeric Nondimensionalization Problem and Full Solution

**Problem:** Consider the polynomial

$$P(x) = \epsilon x^3 - x + 1.$$

Find approximate expressions for all roots of the polynomial in the limit of small positive $\epsilon$ and large positive $\epsilon$. Use a series expansion to calculate improved formulae for these roots to order 1 i.e. calculate $\mathcal{O}(1)$ corrections for each root.

---

**Solution:** Note: The root calculation in this problem follow the same method as those demonstrated in the A.3, so they has been omitted here. We include only correction term calculations for the sake of brevity.

We now need to calculate correction terms for these roots to give us better approximations. We consider the ansatz that the root is given by $\overline{x} + \delta$, where the correction term $\delta$ is the sum of higher order terms of $\epsilon$ that we initially neglected in our approximation $\overline{x}$. By definition, $\delta < \overline{x}$. We plug this ansatz into the polynomial and perform a series expansion in $\delta$. We keep terms only up to $\mathcal{O}(1)$ in $\delta$. Then, we set the expression equal to 0 and solve for $\delta$.

Regime 1: valid for small $\epsilon$

Root 1: $-\sqrt{\frac{1}{\epsilon}}$

$$\overline{x} + \delta = -\sqrt{\frac{1}{\epsilon}} + \delta$$

Substitute this into $P(x)$ for $x$ and equate to 0:

$$-\delta + \epsilon \left(\delta - \sqrt{\frac{1}{\epsilon}}\right)^3 + \sqrt{\frac{1}{\epsilon}} + 1 = 0.$$

We then expand this expression to get

$$\delta^3\epsilon - 3\delta^2\epsilon\sqrt{\frac{1}{\epsilon}} + 2\delta - \epsilon\left(\frac{1}{\epsilon}\right)^{\frac{3}{2}} + \sqrt{\frac{1}{\epsilon}} + 1 = 0$$

and represent it as a series of $\mathcal{O}(1)$ in $\delta$, discarding higher order $\delta$ terms

$$2\delta - \epsilon\left(\frac{1}{\epsilon}\right)^{\frac{3}{2}} + \sqrt{\frac{1}{\epsilon}} + 1 \approx 0.$$

We can then solve the expression for the correction $\delta$ to $\mathcal{O}(1)$, and get

$$\boxed{\delta \approx \frac{\epsilon\left(\frac{1}{\epsilon}\right)^{\frac{3}{2}}}{2} - \frac{\sqrt{\frac{1}{\epsilon}}}{2} - \frac{1}{2}.}$$

Root 2: $\sqrt{\frac{1}{\epsilon}}$

$$\overline{x} + \delta = \sqrt{\frac{1}{\epsilon}} + \delta$$

Substitute this into $P(x)$ for $x$ and equate to 0:

$$-\delta + \epsilon \left(\delta + \sqrt{\frac{1}{\epsilon}}\right)^3 - \sqrt{\frac{1}{\epsilon}} + 1 = 0.$$

We then expand this expression to get

$$\delta^3\epsilon + 3\delta^2\epsilon\sqrt{\frac{1}{\epsilon}} + 2\delta + \epsilon\left(\frac{1}{\epsilon}\right)^{\frac{3}{2}} - \sqrt{\frac{1}{\epsilon}} + 1 = 0$$

and represent it as a series of $\mathcal{O}(1)$ in $\delta$, discarding higher order $\delta$ terms

$$2\delta + \epsilon \left(\frac{1}{\epsilon}\right)^{\frac{3}{2}} - \sqrt{\frac{1}{\epsilon}} + 1 \approx 0.$$

We can then solve the expression for the correction $\delta$ to $\mathcal{O}(1)$, and get

$$\boxed{\delta \approx -\frac{\epsilon \left(\frac{1}{\epsilon}\right)^{\frac{3}{2}}}{2} + \frac{\sqrt{\frac{1}{\epsilon}}}{2} - \frac{1}{2}.}$$

Regime 2: valid for small $\epsilon$

Root 1: 1

$$\overline{x} + \delta = 1 + \delta$$

Substitute this into $P(x)$ for $x$ and equate to 0:

$$-\delta + \epsilon \left(\delta + 1\right)^3 = 0.$$

We then expand this expression to get

$$\delta^3\epsilon + 3\delta^2\epsilon + 3\delta\epsilon - \delta + \epsilon = 0$$

and represent it as a series of $\mathcal{O}(1)$ in $\delta$, discarding higher order $\delta$ terms

$$\delta\left(3\epsilon - 1\right) + \epsilon \approx 0.$$

We can then solve the expression for the correction $\delta$ to $\mathcal{O}(1)$, and get

$$\boxed{\delta \approx -\frac{\epsilon}{3\epsilon - 1}.}$$

Regime 3: valid for large $\epsilon$

Root 1: $\sqrt[3]{-\frac{1}{\epsilon}}$

$$\overline{x} + \delta = \sqrt[3]{-\frac{1}{\epsilon}} + \delta$$

Substitute this into $P(x)$ for $x$ and equate to 0:

$$-\delta + \epsilon \left(\delta + \sqrt[3]{-\frac{1}{\epsilon}}\right)^3 - \sqrt[3]{-\frac{1}{\epsilon}} + 1 = 0.$$

We then expand this expression to get

$$\delta^3\epsilon + 3\delta^2\epsilon\sqrt[3]{-\frac{1}{\epsilon}} + 3\delta\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}} - \delta - \sqrt[3]{-\frac{1}{\epsilon}} = 0$$

and represent it as a series of $\mathcal{O}(1)$ in $\delta$, discarding higher order $\delta$ terms

$$\delta\left(3\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}} - 1\right) - \sqrt[3]{-\frac{1}{\epsilon}} \approx 0.$$

We can then solve the expression for the correction $\delta$ to $\mathcal{O}(1)$, and get

$$\boxed{\delta \approx \frac{\sqrt[3]{-\frac{1}{\epsilon}}}{3\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}} - 1}.}$$

Root 2: $\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1-\sqrt{3}i\right)}{2}$

$$\overline{x} + \delta = \frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1 - \sqrt{3}i\right)}{2} + \delta$$

Substitute this into $P(x)$ for $x$ and equate to 0:

$$-\delta+\epsilon\left(\delta+\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1-\sqrt{3}i\right)}{2}\right)^3-\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1-\sqrt{3}i\right)}{2}+1=0.$$

We then expand this expression to get

$$\delta^3\epsilon-\frac{3\delta^2\epsilon\sqrt[3]{-\frac{1}{\epsilon}}}{2}-\frac{3\sqrt{3}i\delta^2\epsilon\sqrt[3]{-\frac{1}{\epsilon}}}{2}-\frac{3\delta\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}$$
$$+\frac{3\sqrt{3}i\delta\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}-\delta+\frac{\sqrt[3]{-\frac{1}{\epsilon}}}{2}+\frac{\sqrt{3}i\sqrt[3]{-\frac{1}{\epsilon}}}{2}=0$$

and represent it as a series of $\mathcal{O}(1)$ in $\delta$, discarding higher order $\delta$ terms

$$\delta\left(-\frac{3\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}+\frac{3\sqrt{3}i\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}-1\right)+\frac{\sqrt[3]{-\frac{1}{\epsilon}}}{2}+\frac{\sqrt{3}i\sqrt[3]{-\frac{1}{\epsilon}}}{2}\approx 0.$$

We can then solve the expression for the correction $\delta$ to $\mathcal{O}(1)$, and get

$$\boxed{\delta\approx\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(1+\sqrt{3}i\right)}{3\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}-3\sqrt{3}i\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}+2}.}$$

Root 3: $\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1+\sqrt{3}i\right)}{2}$

$$\overline{x}+\delta=\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1+\sqrt{3}i\right)}{2}+\delta$$

Substitute this into $P(x)$ for $x$ and equate to 0:

$$-\delta+\epsilon\left(\delta+\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1+\sqrt{3}i\right)}{2}\right)^3-\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(-1+\sqrt{3}i\right)}{2}+1=0.$$

We then expand this expression to get

$$\delta^3\epsilon-\frac{3\delta^2\epsilon\sqrt[3]{-\frac{1}{\epsilon}}}{2}+\frac{3\sqrt{3}i\delta^2\epsilon\sqrt[3]{-\frac{1}{\epsilon}}}{2}-\frac{3\delta\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}$$
$$-\frac{3\sqrt{3}i\delta\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}-\delta+\frac{\sqrt[3]{-\frac{1}{\epsilon}}}{2}-\frac{\sqrt{3}i\sqrt[3]{-\frac{1}{\epsilon}}}{2}=0$$

and represent it as a series of $\mathcal{O}(1)$ in $\delta$, discarding higher order $\delta$ terms

$$\delta\left(-\frac{3\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}-\frac{3\sqrt{3}i\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}}{2}-1\right)+\frac{\sqrt[3]{-\frac{1}{\epsilon}}}{2}-\frac{\sqrt{3}i\sqrt[3]{-\frac{1}{\epsilon}}}{2}\approx 0.$$

We can then solve the expression for the correction $\delta$ to $\mathcal{O}(1)$, and get

$$\boxed{\delta\approx\frac{\sqrt[3]{-\frac{1}{\epsilon}}\left(1-\sqrt{3}i\right)}{3\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}+3\sqrt{3}i\epsilon\left(-\frac{1}{\epsilon}\right)^{\frac{2}{3}}+2}.}$$

### A.1.4 ODEs

We generate third-order ordinary differential equations of the form

$$y''' = f_1(x)(y'')^a + f_2(x)(y')^b + f_3(x)y^c + f_4(x),$$

where $f_1(x), f_2(x), f_3(x), f_4(x)$ are rational functions with integer coefficients. The initial conditions are randomly selected integers from $[0, 3]$. The dataset excludes problems with a function of $x$ as a dominant term because of the difficulty of deriving power law expressions in these cases.

Approximate solutions at small $x$ can be derived using a Taylor series expansion (up to the third order) around $x = 0$. Solving ODEs in the large $x$ regime involves determining the two largest terms, assuming a divergence at some large $x^*$, and solving the dominant balance between these terms to create a power law approximation of the form

$$y(x) = A(x^* - x)^p.$$

**ODE Problem and Solution**

Problem: Consider the following third-order ordinary differential equation:

$$y''' = -\frac{y}{24x^4 + 6x^2 + 3} + y'^2 - \frac{y''}{5x^3 - 2x^2 - x + 2} - \frac{1}{12x^2 - \cos(x) + 11}$$

with initial conditions at x = 0:

$$y(0) = 1.00$$
$$y'(0) = 0.00$$
$$y''(0) = 0.00$$

Find analytical expressions that approximate the solution of y(x) at small and large $x$.

---

**Solution:**
The dominant balance in the large x regime is given by

$$\frac{d^3}{dx^3}y = \left(\frac{d}{dx}y\right)^2.$$

We recognize that the solution of this ODE will diverge at finite $x$ and that divergences typically follow a power law of the form

$$y = \alpha(x - x^*)^p,$$

where $x^*$ is the divergence point. The divergence point can be determined by estimated by examining the numerical solution generated by code.
Plugging in the dominant terms we found previously yields the following equation:

$$\alpha p\,(p-2)\,(p-1)\,(x - 11.45)^{p-3} = \alpha^2 p^2\,(x - 11.45)^{2p-2}.$$

After substituting the derivatives, the equation is reorganized to collect terms with respect to $(x - x^*)$. This leads to an equation where the coefficients and powers of $(x - x^*)$ are equated on both sides. Simplifying the equation gives us two separate equations, one for the coefficients and another for the powers of $(x - x^*)$. There is now a system of equations, where the coefficients' equation is

$$\alpha p\,(p-2)\,(p-1) = \alpha^2 p^2$$

and the powers' equation is:

$$p - 3 = 2p - 2.$$

Solving this system of equations provides the values of $\alpha$ and $p$. A valid solution is identified if $\alpha$ and $p$ are both nonzero. Here, the solution for $\alpha$ and $p$ is found to be:

$$\alpha = -6, \quad p = -1$$

With these values, the analytical approximation for the solution at large $x$ (near the divergence point) is given by

$$y = -6(x - 11.45)^{-1}.$$

The approximate solution at small $x$ can also be solved used dominant balance, but one can take advantage of the initial conditions and form a Taylor series instead around $x = 0$, which is given by

$$y(x) \approx y(0) + y'(0)x + \frac{y''(0)}{2!}x^2 + \frac{y'''(0)}{3!}x^3.$$

Plugging in the initial conditions, we get the following expression at small $x$:

$$y(x) = 1 - \frac{13}{180}x^3$$

Thus, with rounding for clarity, the solution is given by

$$\boxed{y(x) = 1 - \frac{13}{180}x^3,\ y = -6(x - 11.45)^{-1}.}$$

### A.1.5 Integrals

The polynomial $P(x)$ is randomly generated to consist of up to ten terms, where each term is a power function of $x$ with an integer power randomly sampled from 1 and 20 and an integer coefficient sampled from 1 to 10. The integration bound $a \in [0, 100]$ is also randomly selected. This form ensures that the integral does not oscillate.

The height is approximated as the maximum value of the integrand, which is $\frac{1}{\epsilon}$, and the width can be estimated as the distance over which the integrand decreases from its maximum value by a factor of 2, which implies that the width $x$ obeys the equation

$$\frac{1}{\epsilon + P(x)} = \frac{1}{2\epsilon} \Rightarrow P(x) = \epsilon.$$

In the regime of small $\epsilon$, the term with the smallest degree and $\epsilon$ are the dominant terms, and in the regime of intermediate $\epsilon$, the term with the largest degree and $\epsilon$ are dominant. There exists one more solution regime when the width of the integral exceeds the limits of integration, or when $\epsilon$ is "very large." In this case, the integral is approximated by $L/\epsilon$, where $L$ is the integration range.

**Sample Integral Problem and Full Solution**

**Problem:**
Consider the integral $I(\epsilon) = \int_0^{56.00} \frac{1}{\epsilon + 2.0x^{6.0} + 2.0x^{9.0} + 5.0x^{11.0} + 5.0x^{13.0}} dx$. Develop analytical formulas that approximate $I(\epsilon)$ for different regimes of $\epsilon$.

---

**Solution:** The integral is of the form $I(\epsilon) = \int_0^{56} \frac{1}{\epsilon + P(x)} dx$ where $P(x)$ is a polynomial. Thus, its value can be estimated as the product between a height and a width.
Since the integrand is maximized at $x = 0$, the height can be set to $\frac{1}{\epsilon}$.
For small $\epsilon$, we define the width as the point where the integrand becomes half of its maximum height. This corresponds to solving for $x$ given $P(x) = \epsilon$. Applying dominant balance, considering the term in $P(x)$ with the smallest degree, the width is approximated as $\left(\frac{1}{2.0 * \epsilon}\right)^{1/6.0}$. Therefore, the analytical approximation of the integral for small $\epsilon$ is $I(\epsilon) = \frac{0.8909}{\epsilon^{0.8333}}$.
For an intermediate regime where $\epsilon$ is large, we also define the width based on the term with the largest degree. The width is approximated as $\left(\frac{1}{5.0 * \epsilon}\right)^{1/13.0}$. Therefore, the analytical approximation of the integral for large $\epsilon$ is $I(\epsilon) = \frac{0.7647}{\epsilon^{0.8333}}$.
If the width of the integral exceeds the range of integration, we consider one more regime for very large $\epsilon$. The width is then just the range of integration, so in this regime, the integral can

be approximated as $\frac{L}{\epsilon}$. Therefore, the analytical approximation of the integral for very large $\epsilon$ is $I(\epsilon) = \frac{56}{\epsilon}$.

Altogether, the solutions at small, large, and very large $\epsilon$ are $\boxed{\frac{0.89}{\epsilon^{0.83}}, \frac{0.76}{\epsilon^{0.83}}, \frac{56}{\epsilon}}$.

### A.1.6 Laplace integrals

Laplace integrals of the form $I(x) = \int_a^b g(t)e^{\pm xf(t)}dt$ assume that $f(t) > 0$, is never a constant, and has an absolute minimum at a point $t_0$ either in the interior of or on the bounds of the interval $[a, b]$.

The set of possible Laplace integrals $I(x)$ in our dataset are parameterized by four parameters: the bounds $[a, b]$, $g(t)$, $f(t)$, and the sign in front of $x$. To generate the dataset, the bounds for each problem were randomly sampled from the $[-1, -0.9, \dots 0.9, 1]$, and the sign was uniformly sampled from $\{-1, 1\}$. The functions $f(t)$ and $g(t)$ were generated by randomly selecting a linear combination of polynomials up to fifth order and basic trigonometric functions.

Our solution uses SymPy under the hood to find the minima of $f(t)$ (or the dual annealing algorithm if SymPy fails to return the minima).

**Laplace Integral Problem and Solution**

Problem: Consider the integral

$$I(x) = \int_{-0.9}^{0.3} (-1.6t^2 - 0.5\sin(t) - 1.9)e^{+x(-2.5t^4 - 0.8t^3 + 1.4t^2)}\, dt \tag{3}$$

Develop an analytical formula for $I(x)$ that is accurate as $x \to \infty$.

**Solution:**
The integral is of the form

$$I(x) = \int_a^b g(t)e^{+xf(t)}\, dt \tag{4}$$

where $a = -0.9$, $b = 0.3$, $g(t) = -1.6t^2 - 0.5\sin(t) - 1.9$, and $f(t) = -2.5t^4 - 0.8t^3 + 1.4t^2$. This means we can use Laplace's method to develop an analytical approximation in the limit that $x \to \infty$. In this limit, the integral will be dominated by the integrand near the maximum of $f(t)$ within the bounds $[-0.9, 0.3]$. So, to simplify the integral, we will expand the integrand around this maximum. In this case, we can find the maximum of $f(t) = -2.5t^4 - 0.8t^3 + 1.4t^2$ on the interval analytically. We begin by looking for critical point(s) $t_{crit}$ of $f(t)$ by solving $f'(t) = -10.0t^3 - 2.4t^2 + 2.8t = 0$ for $t$. This gives us that $t_{crit} = [-0.66, 0]$. To find the maximum on this interval, we evaluate $f(t)$ at the critical point(s) $t_{crit}$ and the bounds $-0.9$ and $0.3$. We take the $t$ that gives the largest value. Here, this maximum $t_0 = [-0.66]$. Since the integral is dominated by the value of the integrand near -0.66, we Taylor expand the integrand around this point.

$$I(x) = \int_a^b (g(-0.66) + (t + 0.66)g'(-0.66) + ...) \\ * e^{+x(f(-0.66) + (t+0.66)f'(-0.66) + \frac{(t+0.66)^2}{2}f''(-0.66) + ...)}dt \tag{5}$$

But $f'(-0.66) = 0$ by definition, so we can remove this term from the exponent. We can then approximate

$$I(x) \approx \int_a^b g(-0.66)e^{+x(f(-0.66) + \frac{(t+0.66)^2}{2}f''(-0.66))}\, dt, \tag{6}$$

which equals

$$g(-0.66)e^{+xf(-0.66)} \int_a^b e^{+x(\frac{(t+0.66)^2}{2}f''(-0.66))}\, dt \tag{7}$$

We perform the change of variables $u = \sqrt{x\frac{|f''(-0.66)|}{2}}(t + 0.66)$, rewriting the integral as

$$g(-0.66)e^{+xf(-0.66)} \int_{\sqrt{x\frac{|f''(-0.66)|}{2}}(a+0.66)}^{\sqrt{x\frac{|f''(-0.66)|}{2}}(b+0.66)} \sqrt{\frac{2}{x|f''(-0.66)|}} e^{-u^2}\, dt \tag{8}$$

Since $x \to \infty$, we approximate this as

$$g(-0.66)e^{+xf(-0.66)} \sqrt{\frac{2}{x|f''(-0.66)|}} \int_{-\infty}^{\infty} e^{-u^2}\, dt \tag{9}$$

Solving the integral and evaluating, we find that

$$\boxed{I(x) \approx -1.21\sqrt{\frac{\pi}{x}}e^{0.37x}} \tag{10}$$

## A.2 Evaluation setup

### A.2.1 Prompts for response generation

Table 3: Problem type specific hints by Question and Answer Type

| **Question Type** | **Answer Type** | **Task instruction** |
|---|---|---|
| Nondim-symbolic | SymPy | Please answer the question requiring an answer in a SymPy convertible formula containing variables and math operation expressions and provide the final answer, e.g., $x^3$, $\frac{x}{y}$ inside a Latex boxed format `\boxed{}`. |
| Nondim-numerical | Float (2) | Please answer the question requiring a floating-point number with two decimal places and provide the final value, e.g., 0.80, 3.12, inside a Latex box `\boxed{}`. |
| Polynomial Roots | SymPy List | Please answer the question requiring a Python list containing SymPy convertible formulas of variable $\epsilon$ and math operation expressions and provide the final list, e.g., [$\epsilon^3$, $\frac{1}{\epsilon}$] inside a Latex boxed format `\boxed{}`. |
| ODEs | SymPy List | Please answer the question requiring a Python list containing SymPy convertible formula of $y = f(x)$ and provide the final list, e.g., $[y = 1 - x^3, y = -6/(x - 5)]$, inside a Latex boxed format `\boxed{}`. |
| Integrals | SymPy | Please answer the question requiring an answer in a SymPy convertible formula containing formulas of variable $x$ and math operation expressions and provide the final answer, e.g., $x^3$ inside a Latex boxed format `\boxed{}`. |

### A.2.2 Prompts for grading

Table 4: LLM-based grading prompts by Question and Answer Type

| Question type | Answer type | Task instruction |
|---|---|---|
| Polynomial Roots | SymPy List | Please take this response `{response}` and this answer key `{answer key}` and grade the response based on the following criteria: 1) Check both the small and large $\epsilon$ solutions. 2) For each solution, give full credit if it completely matches the elements in the answer key; give partial credit proportional to the number of matching roots between the response and the answer key; give no credit if it is completely wrong. 3) For both partial and no credit briefly state the error reason. 4) Average the scores for the small and large epsilon solutions to obtain a final score between 0 and 1. 5) Give the final grading as a float in Latex boxed format `\boxed{}`. |
| ODEs | SymPy List | Please take this response `{response}` and this solution `{answer key}` and grade the response based on the following criteria: 1) Check both the small and large $\epsilon$ solutions. 2) For each solution, give full credit if it matches the formula in the answer key; give no credit if it is completely wrong and briefly state the reason for the error. 3) Average the scores for the small and large epsilon solutions to obtain a final score between 0 and 1. 4) Give the final grading as a float in Latex boxed format `\boxed{}`. |
| Integrals (traditional) | SymPy List | Please take this response `{response}` and this solution `{answer key}` and grade the response based on the following criteria: 1) Check both the small and large $\epsilon$ solutions. 2) For each solution, give full credit if it matches the formula in the answer key; give no credit if it is completely wrong and briefly state the reason for the error. 3) Average the scores for the small and large epsilon solutions to obtain a final score between 0 and 1. 4) Give the final grading as a float in Latex boxed format `\boxed{}`. |
| Integrals (Laplace) | SymPy | Please take this response `{response}` and this solution `{answer key}` and grade the response based on the following criteria: 1) Check the large $x$ final solution. 2) Give full credit if it matches the formula in the answer key; give half credit if the `{response}` get to the checkpoint where it correctly identifies $t_0$ where $f$ attains its maximum and attempt performing Taylor's expansion around it but the final answer is wrong; give no credit if it is completely wrong. 3) For both partial and no credit briefly state the error reason. 4) Give the final grading as a float in Latex boxed format `\boxed{}`. |

### A.2.3 GPT grading human verification

| Model | Roots | ODEs | Integrals |
|---|---|---|---|
| GPT3.5 (0) | 0 | 0 | 0 |
| GPT3.5 (1) | 0 | -0.09 | -0.02 |
| GPT3.5 (5) | +0.02 | +0.07 | +0.02 |
| GPT4 (0) | 0 | -0.02 | 0 |
| GPT4 (1) | 0 | -0.04 | -0.02 |
| GPT4 (5) | +0.07 | -0.07 | -0.15 |
| Llama3-8b (0) | 0 | 0 | -0.02 |
| Llama3-8b (5) | -0.07 | -0.02 | -0.02 |
| Codellama3-14b (0) | 0 | -0.02 | 0 |
| Codellama3-14b (5) | 0 | -0.02 | 0 |

Table 5: Average adjusted points using human judgment from GPT-based grading. Rows with score adjustments of 0.1 or more are highlighted in pink.

### A.2.4 Model hyper-parameters

Table 6: Generating parameters for various LLMs.

| Model | Generation Setup |
|---|---|
| GPT-3.5 | `model = gpt-3.5-turbo, temperature = 0, max_tokens = 4000` |
| GPT-4 | `model = gpt-4-turbo, temperature = 0, max_tokens = 4000` |
| Llama3 | `model = llama3:8b, temperature = 0` |
| CodeLlama | `model = codellama:13b, temperature = 0` |

### A.2.5 Computing resource

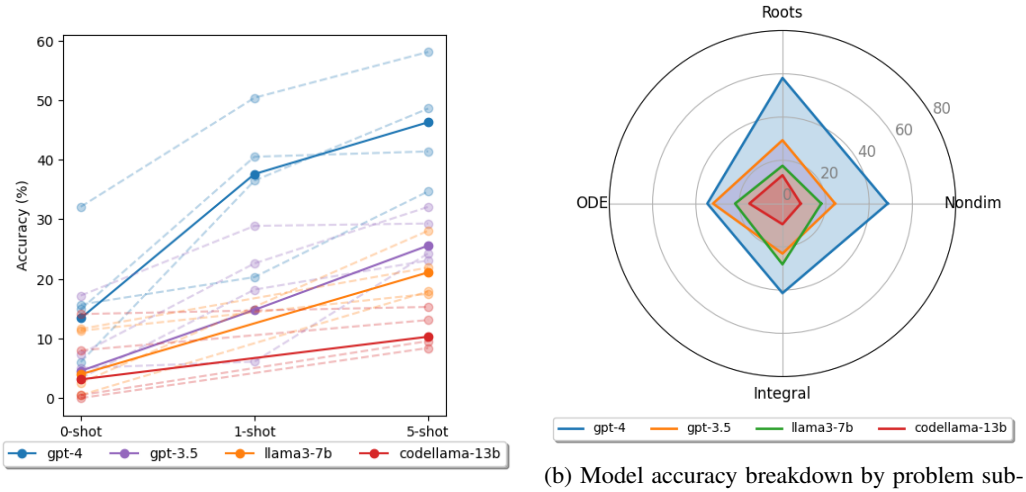
Evaluations of open-source models on **HARDMATH** are conducted on the O2 High Performance Compute Cluster, supported by the Research Computing Group, at Harvard Medical School. See `https://it.hms.harvard.edu/our-services/research-computing` for more information. We would like to thank the HMS Research Computing Consultant Group for their consulting services, which facilitated the computational analyses detailed in this paper.

### A.3 Extended experimental results

### A.4 Student results

| Problem Type | Types of Errors (Ranked from most to least common) |
|---|---|
| **Polynomial roots** | 1. Forgot $x = 0$ as a solution<br>2. Made algebra mistakes |
| **ODEs** | 1. Did not explore all solution regimes<br>2. Made algebra mistakes<br>3. Did not solve for numerical solutions correctly |
| **Integrals** | 1. Did not explore all solution regimes<br>2. Made algebra mistakes |

#### A.4.1 Extended evaluation results




(a) Model accuracy increases with shot numbers.

(b) Model accuracy breakdown by problem sub-types for all models with 5-shot CoT prompting.

Figure 4: Model evaluation accuracy breakdown by shot number and problem sub-types. (a) evaluation accuracy for all models increases with shot numbers for CoT prompting with GPT-4 having the largest improvement; (b) evaluation accuracy breakdown for all models on all problem sub-types under the 5-shot CoT condition.

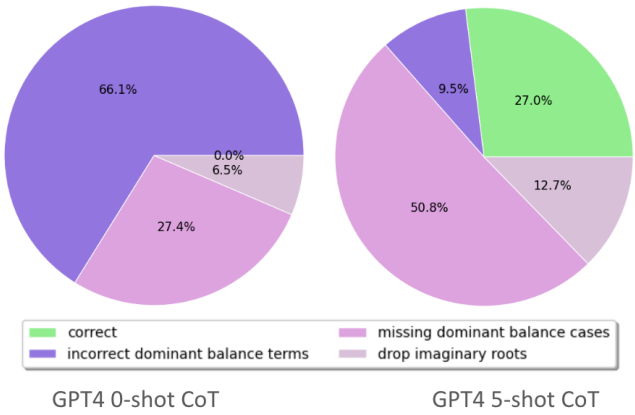



Figure 5: GPT-4 error modes for problem type *Roots* at 0 vs. 5 shot CoT prompting

### A.4.2 Extended example GPT-4o grading responses

Grading Response: *ODEs* (incorrect)

To grade the response, we will compare the provided solutions in the response with the ground truth solutions in both the small $x$ and large $x$ regimes.

## Small $x$ Regime

Response: $y(x) = 1 - 5e^{-x}$
Ground Truth: $y(x) = 1 - \frac{5x^3}{6}$
The response and the ground truth solution for small $x$ are significantly different. The ground truth solution is a polynomial in $x$, while the response includes an exponential term. They do not match at all, so the solution for the small $x$ regime is completely wrong.

## Large $x$ Regime

Response: $y(x) =$ "Assumed rapid growth form due to $y'^4$ term."
Ground Truth: $y = \sqrt[3]{30}(x - 2.68)^{\frac{1}{3}} + (1.84)$
The response does not provide a specific analytical expression for the large $x$ regime, while the ground truth does provide a specific form. The response mentions rapid growth due to $y'^4$, but without a clear analytical form, this response doesn't match the ground truth solution. This indicates that the solution for the large $x$ regime is also completely wrong.
Given the evaluation criteria, both solutions do not match or are incomplete compared to the ground truth. Therefore, each solution gets no credit.
The final grading is $\boxed{0.0}$.

Box 3: Example grading response for a *ODEs* problem that gets zero point.

Grading Response: *Integrals* (correct)

The model response approximates the integral for small $\epsilon$ as $\boxed{\frac{1}{6^{1/4}}\epsilon^{-3/4}}$ and for large $\epsilon$ as $\boxed{\frac{1}{8^{1/18}}\epsilon^{-17/18}}$.
Now, let's compare these results with the ground truth solutions:

**For small $\epsilon$:** - The ground truth solution is $\boxed{I(\epsilon) = \frac{0.64}{\epsilon^{0.75}}}$. - The model's result is $\boxed{\frac{1}{6^{1/4}}\epsilon^{-3/4}}$.
Both results for small $\epsilon$ are essentially the same, with:

$$\frac{1}{6^{1/4}} \approx 0.64$$

**For large $\epsilon$:** - The ground truth solution is $\boxed{I(\epsilon) = \frac{0.89}{\epsilon^{0.94}}}$. - The model's result is $\boxed{\frac{1}{8^{1/18}}\epsilon^{-17/18}}$.
Both results for large $\epsilon$ are essentially the same, with:

$$\frac{1}{8^{1/18}} \approx 0.89$$

Since the model's response closely matches the ground truth solutions in both regimes (small and large $\epsilon$) and provides essentially the same approximations, I would give full credit $\boxed{1.0}$

Box 4: Example grading response for a *Integrals* sub-type simple integral problem that gets full points.

Grading Response: *Integrals* (partial)

**Answer Key**
1. Setup and Formulation: it 1) correctly identifies the integral form; 2) uses Laplace's method; 3) identifies critical points and uses Taylor expansion around the point of interest; 4) performs a change of variables to simplify the integral.
2. Key Steps and Approximations: it 1) expands the integrand around the minimum of $f(t)$; 2) approximates the integral by focusing on the dominant exponential term; 3) provides a final analytical approximation for $I(x)$ as $x \to \infty$.
3. Final Result:

$$\boxed{I(x) \approx -\frac{0.004e^{0.89x}}{x}}$$

**Model Answer**
1. Setup and Formulation: it 1) correctly identifies the integral form; 2) recognizes that the exponential term will dominate for large $x$; 3) focuses on the region where the integrand is most significant (near $t = 0.7$).
2. Key Steps and Approximations: it 1) uses the method of steepest descents heuristically without detailed Taylor expansion; 2) approximates by evaluating the polynomial part at $t = 0.7$ and considering the exponential decay; 3) recognizes the dominant behavior of the exponential term but does not perform a detailed variable change or expansion.
3. Final Result:

$$\boxed{I(x) \approx -0.017 \cdot e^{x \cdot 2.6 \cdot 0.7^3}}$$

**Comparison**

1. Similarities: 1) understanding of dominance: both the answer key and the student correctly identify that the behavior of the integral for large $x$ is dominated by the exponential term; 2) identifying significant points: both recognize that $t = 0.7$ is the critical point where the integrand contributes significantly; 3) approximate behavior: Bbth solutions focus on approximating the behavior of the integral for large $x$, emphasizing the exponential term's dominance.
2. Differences: 1) mathematical rigor: the answer key employs a more rigorous mathematical approach, including Taylor expansion and change of variables, providing a more precise and detailed solution; 2) heuristic approach: the student takes a more heuristic approach, focusing on the dominant exponential term without detailed mathematical expansions or variable changes; 3) accuracy of final expression: the answer key provides a more accurate final expression with specific constants derived from the detailed process, while the student's result, though on the right track, has a different coefficient due to the heuristic method. I would give partial credit $\boxed{0.5}$

Box 5: Example grading response for a *Integrals* sub-type Laplace integral problem highlighting the error of failing to develop Taylor's expansion.