

ASyMOB: Algebraic Symbolic Mathematical Operations Benchmark

Michael Shalyt^{†*} Rotem Elimelech* Ido Kaminer

Technion - Israel Institute of Technology, Haifa 3200003, Israel

Abstract

Large language models (LLMs) are rapidly approaching the level of proficiency in university-level symbolic mathematics required for applications in advanced science and technology. However, existing benchmarks fall short in assessing the core skills of LLMs in symbolic mathematics—such as integration, limits, differential equations, and algebraic simplification. To address this gap, we introduce **ASyMOB**, a novel assessment framework focused exclusively on symbolic manipulation, featuring 17,092 unique math challenges, organized by similarity and complexity. **ASyMOB** enables analysis of LLM failure root-causes and generalization capabilities by comparing performance in problems that differ by simple numerical or symbolic ‘perturbations’. Evaluated LLMs exhibit substantial degradation in performance for all perturbation types (up to -70.3%), suggesting reliance on memorized patterns rather than deeper understanding of symbolic math, even among models achieving high baseline accuracy. Comparing LLM performance to computer algebra systems (CAS, e.g. SymPy), we identify examples where CAS fail while LLMs succeed, as well as problems solved only when combining both approaches. Models capable of integrated code execution yielded higher accuracy compared to their performance without code, particularly stabilizing weaker models (up to +33.1% for certain perturbation types). Notably, the most advanced models (o4-mini, Gemini 2.5 Flash) demonstrate not only high symbolic math proficiency (scoring 96.8% and 97.6% on the unperturbed set), but also remarkable robustness against perturbations, (-21.7% and -21.2% vs. average -50.4% for the other models). This may indicate a “phase transition” in the generalization capabilities of frontier LLMs. It remains to be seen whether the path forward lies in deeper integration with specialized external tools, or in developing models so capable that symbolic math systems like CAS become unnecessary.

1 Introduction

In recent years, large language models (LLMs) have shown remarkable capabilities in domains such as mathematical reasoning [1, 2, 3, 4, 5, 6] and code generation [7, 8, 9, 10]. As these models advance, their potential for real-world research and engineering applications grows. A critical requirement for such applications is proficiency in university-level symbolic mathematics, including integration, limit computation, differential equation solving, and algebraic simplification.

However, existing mathematical benchmarks inadequately assess symbolic proficiency. Early benchmarks like GSM8K [11] and MATH [12], while driving progress in arithmetic reasoning, focus on pre-university level questions and have been, for the most part, mastered by frontier LLMs [13]. Furthermore, many popular benchmarks rely on multiple-choice questions [14], which fail to capture the open-ended nature of real-world problem-solving, and artificially lower the difficulty. Word-problem benchmarks mix two fundamentally different challenges, text-to-math conversion and symbolic manipulation, which makes it hard to evaluate the LLM performance in the latter.

*Equal contribution.

[†]Corresponding author: shalyt@technion.ac.il.

Project repository: <https://github.com/RamanujanMachine/ASyMOB>

Conversely, formal proof datasets (e.g., MiniF2F, MathConstruct [15, 16]) address theorem proving but often skip core tasks like integration or solving differential equations.

The broad topic coverage that most benchmarks strive for forces small sample sizes per skill category, hindering robust statistical analysis. For example, only 150 out of 3709 (4%) questions in MathBench [17] address university-level math in English. The 5K test dataset by Lample and Charton [18] targets symbolic integration and differential equations, but due to its creation method, it mainly contains simple problems and was immediately saturated [18]. Recent efforts, such as FrontierMath [13] and Humanity’s Last Exam [19], demand that LLMs exhibit very high proficiency across numerous skills simultaneously, thereby impeding conclusions regarding specific LLM capabilities. Overcoming these limitations can shed light on a fundamental question: do LLMs solve problems through genuine mathematical understanding or merely through advanced pattern recognition [20, 21, 22, 23, 24, 25]. Addressing this question calls for different types of datasets, which can separate sophisticated pattern memorization from true mathematical abilities.

In response, we present ASyMOB: Algebraic Symbolic Mathematical Operations Benchmark (pronounced Asimov, in tribute to the renowned author), for assessing LLM capabilities through systematic perturbations of core symbolic tasks; introducing three key innovations:

1. **Focused Scope:** Targeting pure symbolic manipulation (Figure 1).
2. **Controlled Complexity:** Systematically introduced questions varied by difficulty levels.
3. **High Resolution:** The large scale and fine-grained difficulty steps enable statistically robust measurement of model accuracy, sensitivity to noise types, and impact of tool use.

| Seed Question | Symbolic Perturbation |
|---|--|
| <p><Code / No-Code Prompt></p> <p><i>Solve the following integral.</i></p> $\int_1^2 \frac{e^x(x-1)}{x(x+e^x)} dx$ <hr/> <p>Solution:</p> $\ln\left(\frac{2+e^2}{2+2e}\right)$ | <p><Code / No-Code Prompt></p> <p><i>Solve the following integral.</i> <i>Assume A, B, F, G are real and positive.</i></p> $\int_1^2 \frac{Ae^{Fx}(Fx-1)}{Fx(Be^{Fx}+FGx)} dx$ <hr/> <p>Solution:</p> $\frac{A}{BF} \cdot \ln\left(\frac{e^2B+2G}{2(eB+G)}\right)$ |
| No-Code Prompt | <i>Assume you don’t have access to a computer: do not use code, solve this manually - using your internal reasoning.</i> |
| Code Prompt | <i>Please use Python to solve the following question. Don’t show it, just run it internally.</i> |

Figure 1: **Example ASyMOB question and code-use preambles.** A seed question (left) and its symbolically perturbed variant (right). Proceeding text disallows or encourages code execution (this part is omitted for models without inherent code execution capabilities).

Using ASyMOB, we evaluated the performance of leading open- and closed-weight LLMs, including general and mathematical models. Our results showcase the challenge perturbations pose to LLM symbolic math skills: the success rate on the unperturbed subset is 77% (averaged over all tested models), vs. 33.4% on the full ASyMOB benchmark. The most substantial drop in performance already happens for small perturbations, and is seen across all types (Figure 2).

Following reports on the effects of tool-use in math problem solving [26, 27, 28, 29, 30, 31, 32, 33, 34], we tested code-integrated LLMs with and without code execution (Figure 2 left). Tool use boosts performance in weaker models, but surprisingly has no effect on frontier ones.

Some perturbed variants in ASyMOB proved impossible for the CAS we tested (Mathematica [35], WolframAlpha [36], SymPy [37]), yet certain LLMs managed to solve them (section 3.1). More-

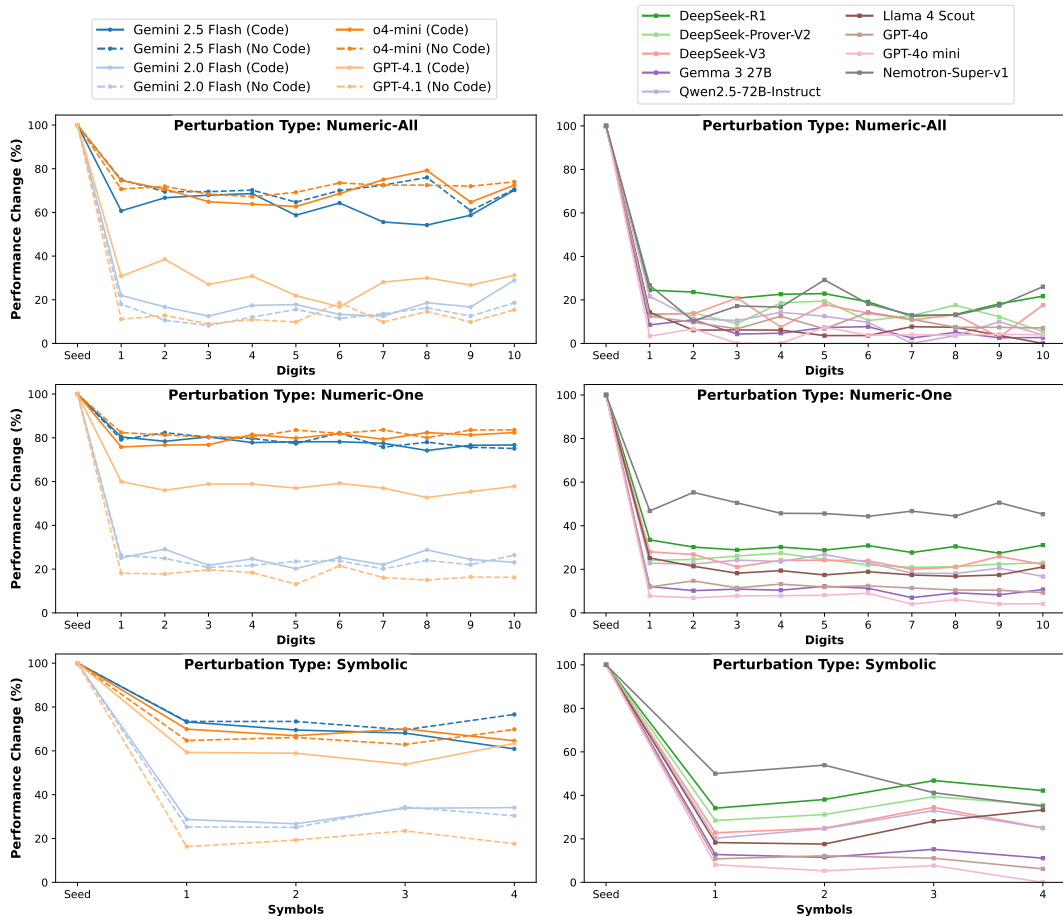


Figure 2: **Degradation of model success rate relative to seed-set performance.** Both code-integrated models (left) and non-code integrated (right) exhibit performance degradation due to numeric and symbolic perturbations, but frontier models are more resilient. Notably, GPT 4.1 is substantially more robust when code-enabled.

over, we present an example where pure CAS and pure LLM approaches fail, yet their combination successfully solves the challenge - leveraging the complementary strengths of each system.

2 Methodology for Symbolic Mathematical Operations Measurement

2.1 Dataset Design and Generation

We begin by curating and creating a set of seed problems that contain only symbolic content - no word-problems or other textual or graphical information beyond the minimal instructions or assumptions needed to define the symbolic task. This restriction excludes almost all olympiad-style problems [38] and separates our dataset from existing benchmarks. Questions were curated from university-level benchmarks (U-Math, MathOdyssey, GHOSTS, UGMathBench [39, 40, 41, 42]) and math olympiads (OMBU, OlympicArena, OlympiadBench [43, 44, 45]). Additionally, some questions were created to cover unrepresented topics.

The seed questions represent university-level symbolic math tasks frequently encountered in engineering and scientific research. These include problems in integration, limits, differential equations, series expansions, and algebraic simplification. The solutions to these seed questions were rigorously validated, primarily using CAS where applicable, and supplemented by manual verification.

Based on these seed questions, we introduce symbolic perturbations to create an overall dataset of 17,092 unique symbolic math challenges (Table 1). The guiding principle was to modify the symbolic structure of the problem - thereby adding a layer of variation - *without substantially altering the core mathematical challenge* or the required solution techniques. To provide a consistent an-

chor for testing and validation, we create each symbolic perturbation such that substituting 1 in its parameters recovers the exact original seed question.

Table 1: **ASyMOB question variants (shown for seed question #6).** For each variant type, the right-most column presents the number of variants for this seed question and the total number of this category in the dataset (e.g. there are 30 ‘Numeric-One-N’ variants and 10 ‘Numeric-All-N’ variants of question #6, totaling 1970 and 610 variants over all questions). XX, YY, and ZZ in ‘Numeric-All-2-S’ represent variants with 2 digit random numbers. Full dataset available at huggingface.co/datasets/Shalyt/ASyMOB-Algebraic_Symbolic_Mathematical_Operations_Benchmark.

| Variant | Example Challenge | Answer | # |
|---|--|-----------------------------|--------------|
| Seed (Original) | $\lim_{x \rightarrow 0} \left(\frac{2 \cdot \tan\left(\frac{x}{2}\right)}{x} \right)^{\frac{3}{x^2}}$ | $e^{\frac{1}{4}}$ | 1 (61) |
| Symbolic-N (Shown for N=3) | $\lim_{x \rightarrow 0} A \cdot \left(\frac{2 \cdot \tan\left(\frac{B \cdot x}{2}\right)}{B \cdot x} \right)^{\frac{C \cdot 3}{(B \cdot x)^2}}$ | $A \cdot e^{\frac{C}{4}}$ | 7 (651) |
| Numeric-All-N (Shown for N=2) | $\lim_{x \rightarrow 0} 17 \cdot \left(\frac{2 \cdot \tan\left(\frac{91 \cdot x}{2}\right)}{91 \cdot x} \right)^{\frac{57 \cdot 3}{(91 \cdot x)^2}}$ | $17 \cdot e^{\frac{57}{4}}$ | 10 (610) |
| Numeric-One-N (Shown for N=6) | $\lim_{x \rightarrow 0} \left(\frac{2 \cdot \tan\left(\frac{x}{2}\right)}{x} \right)^{\frac{838310 \cdot 3}{x^2}}$ | $e^{\frac{838310}{4}}$ | 30 (1970) |
| Numeric-All-2-S | $\lim_{x \rightarrow 0} XX \cdot \left(\frac{2 \cdot \tan\left(\frac{YY \cdot x}{2}\right)}{YY \cdot x} \right)^{\frac{ZZ \cdot 3}{(YY \cdot x)^2}}$ | $XX \cdot e^{\frac{ZZ}{4}}$ | 50 (3050) |
| Equivalence-One-Easy | $\lim_{x \rightarrow 0^+} \left(\frac{2 \cdot \tan\left(\frac{x}{2}\right)}{x} \right)^{\frac{(\sin^2(-Fx) + \cos^2(Fx)) \cdot 3}{x^2}}$ | $e^{\frac{1}{4}}$ | 15 (985) |
| Equivalence-One-Hard | $\lim_{x \rightarrow 0^+} \left(\frac{\sinh(\log(Ax + \sqrt{A^2 x^2 + 1}))}{Ax} \right) \left(\frac{2 \cdot \tan\left(\frac{x}{2}\right)}{x} \right)^{\frac{3}{x^2}}$ | $e^{\frac{1}{4}}$ | 15 (985) |
| Equivalence-All-Easy | $\lim_{x \rightarrow 0^+} (\sin^2(-Ax) + \cos^2(Ax)) \left(\frac{2 \cdot \tan\left(\frac{(-\sinh^2(Bx) + \cosh^2(Bx))x}{(-\sinh^2(Bx) + \cosh^2(Bx))x}\right)}{(-\sinh^2(Bx) + \cosh^2(Bx))x} \right)^{\frac{(\ln(x) \cdot \log_2(Fx)) \cdot 3}{(\ln(Fx) \cdot \log_2(Fx))x}}$ | $e^{\frac{1}{4}}$ | 60 (4390) |
| Equivalence-All-Hard | $\lim_{x \rightarrow 0^+} \left(\frac{\tan(x) \cdot \tan(x(A-1))}{(-\tan(x) \cdot \tan(x(A-1))) + \tan(Ax)} \right)^{\frac{2 \cdot \tan\left(\frac{(\frac{\tan(x) \cdot \tan(x(A-1))}{(-\tan(x) \cdot \tan(x(A-1))) + \tan(Ax)})}{x}\right)}{\frac{(\frac{\tan(x) \cdot \tan(x(A-1))}{(-\tan(x) \cdot \tan(x(A-1))) + \tan(Ax)})}{x}}}{\left(\frac{(\frac{\tan(x) \cdot \tan(x(A-1))}{(-\tan(x) \cdot \tan(x(A-1))) + \tan(Ax)})}{x} \right)^{\frac{(\frac{\tan(x) \cdot \tan(x(A-1))}{(-\tan(x) \cdot \tan(x(A-1))) + \tan(Ax)})}{x}}}$ | $e^{\frac{1}{4}}$ | 60 (4390) |

For instance, consider the elementary integral $\int x^2 e^x dx = e^x (x^2 - 2x + 2)$, typically solved using integration by parts.

- An acceptable perturbation is $\int x^2 e^{Fx} dx = \frac{e^{Fx} (F^2 x^2 - 2Fx + 2)}{F^3}$. Although this variant introduces a substitution step ($t = Fx$), it relies on the same fundamental integration skills as the original problem.
- Conversely, a modification like $\int x^{2B} e^x dx = (-x)^{-2B} x^{2B} \Gamma(2B + 1, -x)$ would *not* be considered a symbolic *perturbation* as it significantly increases the problem’s complexity and demands additional mathematical knowledge compared to the original.

After manually perturbing each seed question with 2-to-5 parameters, we generated additional variants using algorithmic transformations.

One of the key questions we aim to investigate is the effect of the number of symbolic perturbations on model performance. Specifically, we ask whether each additional perturbation further degrades performance, or whether most of the added difficulty for LLMs arises from the introduction of the first symbolic perturbation - transforming a problem to contain non-numeric parameters.

To enable this measurement, we systematically remove added symbols from each manually perturbed question, generating all possible combinations. This approach helps avoid subjective bias regarding which perturbations are harder. Each variant is labeled as ‘Symbolic-N’, where N indicates the number of remaining symbols. For example, a question originally marked as ‘Symbolic-4’ will

yield additional variants: four ‘Symbolic-3’, six ‘Symbolic-2’, and four ‘Symbolic-1’. This process provides a sufficiently large and structured set of symbolic tasks to accurately measure performance degradation.

Another critical axis of evaluation concerns the contrast between symbolic and numerical perturbations. Mathematically, if a model can solve a symbolically perturbed question, it should also be able to solve the same question with numeric perturbations: numeric constants can, in principle, be replaced by symbols, solved symbolically, and substituted back. However, as seen in Figure 2, LLMs do not exhibit this expected behavior - suggesting their mathematical reasoning is still constrained by their token-based architecture.

To test this, for each manually perturbed question, we generate variants by replacing every symbolic parameter with a randomly chosen positive integer of a fixed digit length. We also vary the size of these integers between 1 to 10 digits to test both in- and out-of-training-distribution performance (as symbolic math challenges with large numeric coefficients are very rare unless intentionally created). These are labeled ‘Numeric-All- N ’, where N denotes the number of digits.

Due to the probabilistic nature of LLMs, we measure the stability of mathematical correctness over 50 random variations of ‘Numeric-All-2’ - generating a new set of random 2-digit numbers per variation (see Figure 8 in Appendix C). These variants are marked as ‘Numeric-All-2-S’ in Table 1.

To explore whether the initial introduction of a large number causes a disproportionate performance drop, or whether performance declines progressively with each added numeric coefficient, we also create variants where only one symbolic parameter is replaced by a number (again ranging from 1 to 10 digits), and the remaining symbols are removed. To avoid selection bias, we generate all possible choices of which symbol to retain and replace; these are labeled ‘Numeric-One- N ’.

Numeric perturbations are similar in spirit to previous works like GSM-Symbolic [20], TemplateGSM [46], GSM-Ranges [47], MATH() [48], and MATH-Perturb [23] - which focus on GSM8K [11] or MATH [12] word problems, as well as MathConstruct [16] that focuses on constructive proofs. Differing from these previous benchmarks, here we focus on advanced symbolic math problems, and build a larger-scale dataset.

Finally, we evaluate the impact of equivalent-form perturbations. In this case, we complicate the problem by inserting one or more expressions that mathematically equal to 1. For example, symbol A might be replaced with $\sin^2(-Ax) + \cos^2(Ax)$. While such perturbations introduce extra steps in simplification, the final answer is identical to the original version. We selected five identity types for this transformation - trigonometric, hyperbolic, logarithmic, complex exponential, and series - each with an ‘Easy’ and a ‘Hard’ version, depending on the level of mathematical sophistication required to recognize the identity (see Appendix A for the full list). The ‘Easy’/‘Hard’ classification was done manually, but the results in Figure 3 retroactively validate our assumptions. To implement this transformation at scale, these identities replace the symbols in the symbolic perturbations. For consistency, each variant uses only the easy or the hard forms. Similar to the numeric case, we generate two types of variants: either all symbols are replaced by equivalent forms (‘Equivalence-All-Easy/Hard’), or only one (‘Equivalence-One-Easy/Hard’).

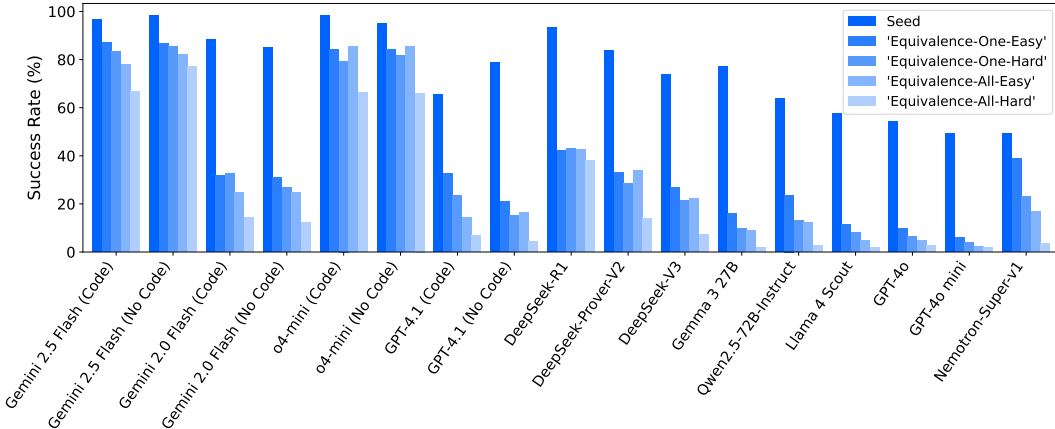


Figure 3: **Effect of equivalence-type perturbations.** Note the substantial drop in success rate for most models, even when performance on the seed set is high.

One of the advantages in ASyMOB is once the seed and manual symbolic perturbations are complete and thoroughly validated, all other tasks are generated algorithmically - removing the risk of errors in specific questions or answers. This is not obvious as existing mathematical benchmarks are known to have mistaken labeling and formatting errors (up to 5-10% [49, 50, 51]).

For example, question 97 from the GHOSTS [41] ‘Symbolic Integration’ subset: “What is the integral of $2x - x^7 \operatorname{atan}(3)$ ”. The output: “...The antiderivative... $\frac{2x^2}{2} - \frac{1}{7}x^8 \operatorname{atan}(3) + C$ ” receives a 5/5 rating, but the $\frac{1}{7}$ should have been $\frac{1}{8}$, potentially creating false positives.

Another example from OlympiadBench [45] (subset ‘OE.TO.maths.en.COMP’ id:2498): “If $\log_2 x - 2 \log_2 y = 2$, determine y , as a function of x ”. The dataset provides both a full solution: “...to obtain $y = \frac{1}{2}\sqrt{x}$ ”, and a final answer: “ $\frac{1}{2}, \sqrt{x}$ ”. The extra comma that appeared in the middle of the final answer prevents deterministic systems from recognizing correct answers. We inserted these questions and corrected answers as two of our seeds.

Additionally, by maintaining consistent question formatting and disallowing substantial textual or graphical information, we prevent potential task ambiguities and missing data [49]. Furthermore, multiple random stages alleviate the risk of data-contamination.

2.2 Testing and Validation

Validating open-ended symbolic problems is harder than closed-form or numerical ones. In the OlympiadBench example above, the reference answer is $\frac{1}{2}\sqrt{x}$. However, solving this using Mathematica yields $e^{\frac{1}{2}(\log(x)-2\log(2))}$. Although structurally different, these expressions are mathematically identical. Our evaluation must accept any correct symbolic form and phrasing without penalizing the LLM (e.g. ‘ $\sqrt{x} \cdot \frac{1}{2}$ ’, ‘ $y = \frac{1}{2}\sqrt{x}$ ’, ‘ $y \rightarrow \frac{1}{2}\sqrt{x}$ ’, etc.). To prevent false negatives, we implement a multi-step validation process with dual verification methods (see Figure 4).

The final mathematical answer is extracted from the LLM’s full textual response using a highly flexible regular expression (see Appendix B). The extracted LaTeX expression is then cleaned (e.g. formatting commands like `\displaystyle` and `\boxed` are removed) and parsed into a SymPy expression using `sympy.parsing.latex.parse_latex`. If the parsing fails, we resort to using an LLM (gemini-2.0-flash [52]) for this translation.

The resulting SymPy expression is validated both **symbolically** (via `SymPy.simplify`) and **numerically** (by generating 5 instances of random values for each symbolic parameter and comparing results of `.evalf()`). See Appendix B for details.

This validation approach avoids the need to employ LLMs as judges during evaluation, as done in several leading benchmarks [39, 40] and introduces inevitable validation errors due to pattern recognition biases of LLMs [53, 39]. ASyMOB employs an LLM only outside the core validation, for translating LaTeX answers into SymPy code, and even then as a fallback when the primary deterministic parser fails (occurred in 18.5% of cases). This reliance on analytical and numerical non-LLM validation is particularly important given that ASyMOB is designed to challenge even frontier models, hence the validation task itself is inherently difficult.

We exclusively use the pass@1 evaluation criterion, reflecting the practical requirement for reliability in real-world applications by engineers or researchers. The inherent LLM randomness is accounted for by evaluating success across the large number of questions within each category.

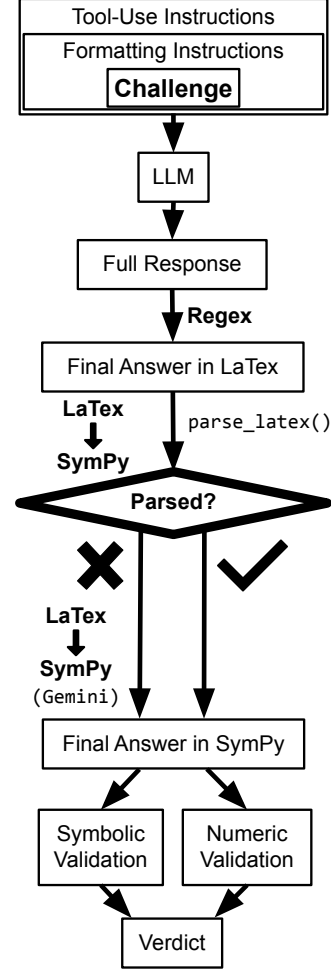


Figure 4: **Result validation pipeline.** The final LaTeX answer is extracted from the full LLM response via a flexible regex. It’s parsed into a computable SymPy expression via a deterministic function or, if it fails, via an LLM. The expression is then validated both symbolically and numerically against the reference answer.

3 Experimental Results

The ASyMOB benchmark evaluates open- and closed-weight LLMs, including general-purpose and mathematical models. Table 2 summarizes performance across the different categories.

Table 2: **Model performance on ASyMOB by perturbation category.** Bold indicates the top performer in each category. Subset titles are color-coded in accordance to Table 1.

| Model | Seed | Symbolic | Numeric | Equivalence | Variance | Total. |
|--------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Closed-Weights Models | | | | | | |
| o4-mini (code) | 98.4 | 67.7 | 76.8 | 76.9 | 67.8 | 75.0 |
| o4-mini (no code) | 95.1 | 64.3 | 77.9 | 77.2 | 67.7 | 75.2 |
| GPT-4.1 (code) | 65.6 | 51.5 | 46.3 | 14.1 | 30.0 | 23.6 |
| GPT-4.1 (no code) | 78.7 | 18.4 | 13.6 | 11.9 | 53.7 | 19.8 |
| GPT-4o | 54.1 | 8.0 | 7.8 | 4.6 | 25.4 | 8.9 |
| GPT-4o-mini | 49.2 | 5.7 | 7.0 | 2.5 | 21.0 | 6.7 |
| Gemini-2.5 Flash (code) | 96.7 | 70.2 | 75.1 | 74.8 | 70.5 | 74.0 |
| Gemini-2.5 Flash (no code) | 98.4 | 73.3 | 76.9 | 80.9 | 73.4 | 78.8 |
| Gemini-2.0 Flash (code) | 88.5 | 29.3 | 23.5 | 22.1 | 64.9 | 30.1 |
| Gemini-2.0 Flash (no code) | 85.2 | 25.1 | 19.0 | 20.5 | 49.8 | 25.8 |
| Open-Weights Models | | | | | | |
| DeepSeek-V3 | 73.8 | 26.1 | 19.4 | 16.6 | 42.2 | 22.1 |
| DeepSeek-R1 | 93.4 | 39.4 | 28.7 | 40.7 | 74.6 | 44.5 |
| DeepSeek-Prover-V2-671B | 83.6 | 31.2 | 19.0 | 25.6 | 52.8 | 28.9 |
| Llama-4-Scout-17B-16E-Instruct | 57.4 | 16.8 | 12.2 | 4.5 | 22.1 | 9.4 |
| Qwen2.5-72B-Instruct | 63.9 | 21.0 | 16.0 | 9.6 | 31.9 | 15.0 |
| Gemma-3-27b-it | 77.0 | 12.2 | 8.2 | 6.7 | 37.7 | 12.7 |
| Nemotron-Super-49B-v1 | 49.2 | 29.3 | 32.4 | 14.0 | 14.4 | 17.5 |

While frontier closed-weight models (o4-mini, Gemini 2.5 Flash [54, 55]) achieve the highest seed accuracy, older (Gemini 2.0 Flash, GPT-4.1, GPT-4o, GPT-4o-mini [52, 56, 57]) and open-weight models (DeepSeek-V3, DeepSeek-R1, DeepSeek-Prover-V2-671B, Llama-4-Scout-17B-16E-Instruct, Qwen2.5-72B-Instruct, Gemma-3-27b-it, Llama-3.3-Nemotron-Super-49B-v1 [58, 59, 60, 61, 62, 63, 64]) also perform reasonably well, all scoring above 49%.

A significant finding is the substantial degradation in performance when models are faced with perturbed versions of the seed questions (Figures 2, 3). Some LLMs struggle more with symbolic perturbations, suggesting gaps in mathematical understanding, while others falter with numeric perturbations, possibly due to longer token chains. Understanding the reasons behind these differences between models may reveal deeper principles of how LLMs process mathematical structures.

Where the top models truly shine is their robustness to perturbations—which is arguably a more critical metric for assessing LLM generalization capabilities—netting a performance gap of 30.6% between o4-mini to the next best model on the total dataset. This robustness persists across perturbation categories and mathematical topics (Figure 5), even when faced with out-of-distribution challenges, which might indicate a recent “phase transition” of frontier LLMs from reliance on memorized patterns to genuine mathematical understanding.

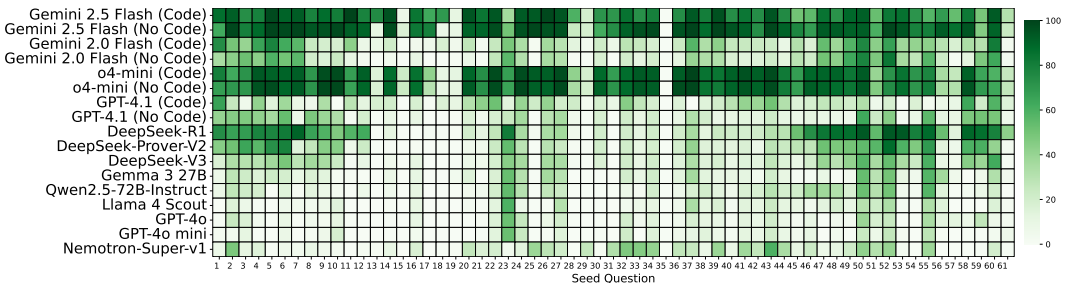


Figure 5: **Heatmap of overall performance per model per seed, averaged over all perturbations.**

Measuring the performance of mathematically fine-tuned models, we notice that DeepSeek-Prover-V2-671B, despite achieving 88.9% pass ratio on the MiniF2F proof benchmark [15, 60], is still outperformed by DeepSeek-R1 (from the same model family), on every category in ASyMOB. This suggests that proficiency in formal proof generation may not directly translate to skill in the broader set of symbolic mathematical operations, where the reasoning capabilities of general models can prove more effective. Nemotron-Super [64], on the other hand, shows fairly high perturbation resilience, despite the low success rate on the seed subset.

The ‘Variance’ subset provides insights into model consistency. The variance of results over all ‘Numeric-All-2-S’ variants was calculated per seed question and per model (Figure 8). An interesting observation is the absence of correlations of variance between models per seed question, indicating that the effect of perturbation is quite similar across the questions.

Enabling code execution for tool-integrated models enhanced their performance by an average improvement of 3.1%. This improvement was particularly notable for GPT 4.1 in the ‘Numeric’ and ‘Symbolic’ subsets (Figure 2), which could be due to its high prowess in coding tasks [56] compensating for its middling symbolic math skills—showcasing the strength of hybrid solution strategies.

3.1 Computer Algebra Systems Limitations

While CAS like SymPy, Mathematica, and WolframAlpha [37, 35, 36] are powerful tools for symbolic mathematics, they are not infallible and have their own limitations. The ASyMOB benchmark includes instances where traditional CAS fail to solve problems that LLMs can manage. Symbolic perturbations, while apparently easier for LLMs to handle than numeric perturbations, seem to have a much larger detrimental effect on CAS, with multiple examples of CAS solving the seed variant and then failing on a ‘Symbolic’ variant.

For example, 2 of the 5 ‘Hard’ equivalence forms (Appendix A) are not recognized by SymPy as identical to 1. Yet, many ‘Equivalence’ variants containing these identities are successfully solved by models in our testing. Another example is the aforementioned ASyMOB question #6 - where WolframAlpha does not simply fail to answer on variant ‘Symbolic-3’, but produces a false result¹. Such examples provide added motivation for developing LLMs skillful at symbolic mathematical manipulations, capable of overcoming CAS shortcomings.

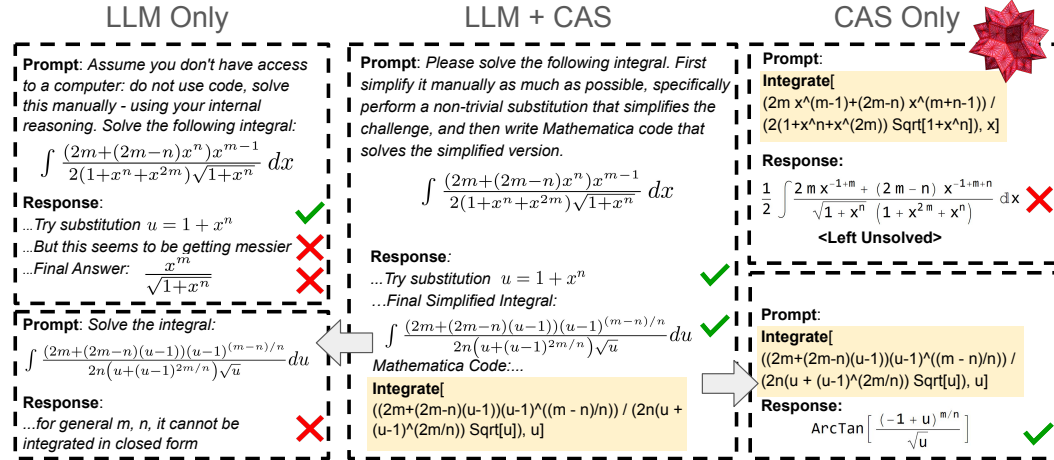


Figure 6: Example question solved exclusively by a hybrid LLM+CAS approach. ASyMOB’s question #122 was solved incorrectly (left) by standard LLMs, shown for GPT-4o, despite the model “considering” a correct substitution. Standard CAS systems (SymPy, Mathematica, and WolframAlpha) also failed to solve the question (right). However, a hybrid strategy succeeded: GPT-4o was prompt to first simplify the problem via substitution and then use CAS code on the simplified expression. This process produced correct code enabling Mathematica to solve the question.

¹Tested on Wolfram Language version 14.2.1: <https://www.wolframalpha.com/input?i=Limit%5BA+%28Tan%5B%28B+x%29%2F%5D%2F%28%28B+x%29%2F%29%29%5E%28%28C+3%29%2F%28B+x%29%5E2%29%2C+x+-%3E+0%2C+Direction+-%3E+%22FromAbove%22%5D>

Perhaps the most teaching example is ASyMOB question #61 on GPT-4o (Figure 6). Pure CAS (SymPy, Mathematica, WolframAlpha) and pure LLM approaches both failed. However, when instructed to simplify the integral first and then solve using CAS, the model succeeded, demonstrating the power of combining LLM strategic ability with CAS rigor.

4 Discussion and Outlook

We introduced ASyMOB, a high-resolution symbolic mathematics benchmark that isolates core symbolic reasoning skills, containing 17,092 challenges. Assessment of leading models shows:

- LLMs’ symbolic math performance substantially degrades under perturbations, suggesting reliance on pattern memorization and lack of “true understanding”.
- Frontier LLMs (o4-mini, Gemini 2.5 Flash) show a leap in robustness against perturbations of various kinds, suggesting strong symbolic math generalization capabilities.
- Correct tool-use (code execution) can meaningfully improve performance.
- Hybrid LLM+CAS strategies can solve challenges beyond the reach of either system alone.

Benchmarks aspire to have truly “new”, uncontaminated questions. ASyMOB bypassed this challenge via the perturbations we added. Even with potential contamination in the seed questions, important conclusions can still be drawn from the results. This approach is also relevant for future LLM advances, as it will gradually become harder to rely on truly new questions for large-scale LLM training datasets.

Instead, we should provide other methods of teaching LLMs to generalize, such as training on the best use of tools. Another path for teaching LLMs to generalize is by training on question variations/perturbations, even using data contamination as a feature. Any LLM capable of using its prior knowledge of the seed question to apply similar mathematical steps to solve the perturbed problems shows generalization capabilities.

One of our perturbations is inspired by GSM-Symbolic [20]—which showed that even “trivial” complications in textual math questions can substantially reduce success rates (up to 65%). Similarly, in our work, trivial symbolic complications also led to substantial performance drops (up to 70.3%). This test generalizes the finding of GSM-Symbolic that “current LLMs are not capable of genuine logical reasoning”, now shown in the domain of symbolic manipulations and not just in text-to-math conversion.

Importantly, our results also points toward a possible solution: Once the LLM learns *when* and *where* to use tools, it can mitigate the above-mentioned pitfalls, using code execution as a form of grounding. This behavior can already be encouraged through prompting strategies like “simplify-then-code” (Figure 6).

Until recently, the hybrid LLM+CAS approach appeared to be the most promising path forward. However, the surprising finding that frontier models *no longer benefit* from CAS use for symbolic math triggers deeper and more fascinating possibilities. Looking ahead, we see three possible trajectories for future developments in AI for math and AI for science:

1. **Intrinsic mastery:** Frontier models may continue to improve in their inherent abilities, eventually surpassing the need for external symbolic math tools, as in the frontier model behavior observed in this work.
2. **Deeper integration:** Tool use may remain essential, but will demand increasingly sophisticated CAS capabilities that co-evolve with LLMs, complementing their inherent abilities and motivating the next generation of CAS infrastructure.
3. **Autonomous tool creation:** LLMs may internalize symbolic computation itself—leveraging their reasoning and coding capacities to build internal, CAS-like mechanisms that blur the boundary between model and tool.

Acknowledgments

This research received support through Schmidt Sciences, LLC.

5 References

References

- [1] A. Lewkowycz et al. “Solving Quantitative Reasoning Problems with Language Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh et al. 2022. URL: <https://openreview.net/forum?id=IFXTZERXdm7>.
- [2] T. Kojima et al. “Large language models are zero-shot reasoners”. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS ’22*. New Orleans, LA, USA: Curran Associates Inc., 2022.
- [3] X. Wang et al. “Self-Consistency Improves Chain of Thought Reasoning in Language Models”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=1PL1NIMMrw>.
- [4] T. H. Trinh et al. “Solving olympiad geometry without human demonstrations”. In: *Nature* 625.7995 (2024).
- [5] H. Luo et al. “WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=mMPMHWOdOy>.
- [6] A. Davies et al. “Advancing mathematics by guiding human intuition with AI”. In: *Nature* 600 (2021). URL: <https://api.semanticscholar.org/CorpusID:244837059>.
- [7] B. Rozière et al. *Code Llama: Open Foundation Models for Code*. 2024. arXiv: 2308.12950 [cs.CL]. URL: <https://arxiv.org/abs/2308.12950>.
- [8] T. Ridnik et al. “Code Generation with AlphaCodium: From Prompt Engineering to Flow Engineering”. In: *arXiv preprint arXiv:2401.08500* (2024).
- [9] D. Zan et al. “Large Language Models Meet NL2Code: A Survey”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers et al. Toronto, Canada: Association for Computational Linguistics, July 2023. URL: <https://aclanthology.org/2023.acl-long.411/>.
- [10] X. Hou et al. “Large Language Models for Software Engineering: A Systematic Literature Review”. In: *ACM Trans. Softw. Eng. Methodol.* 33.8 (Dec. 2024). ISSN: 1049-331X. URL: <https://doi.org/10.1145/3695988>.
- [11] K. Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: <https://arxiv.org/abs/2110.14168>.
- [12] D. Hendrycks et al. “Measuring Mathematical Problem Solving With the MATH Dataset”. In: *NeurIPS* (2021).
- [13] E. Glazer et al. *FrontierMath: A Benchmark for Evaluating Advanced Mathematical Reasoning in AI*. 2024. arXiv: 2411.04872 [cs.AI]. URL: <https://arxiv.org/abs/2411.04872>.
- [14] D. Rein et al. “GPQA: A Graduate-Level Google-Proof Q&A Benchmark”. In: *First Conference on Language Modeling*. 2024. URL: <https://openreview.net/forum?id=Ti67584b98>.
- [15] K. Zheng et al. “miniF2F: a cross-system benchmark for formal Olympiad-level mathematics”. In: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=9ZPegFuFTFv>.
- [16] M. Balunović et al. “MathConstruct: Challenging LLM Reasoning with Constructive Proofs”. In: (2025).
- [17] H. Liu et al. “MathBench: Evaluating the Theory and Application Proficiency of LLMs with a Hierarchical Mathematics Benchmark”. In: *Findings of the Association for Computational Linguistics: ACL 2024*. Ed. by L.-W. Ku et al. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024. URL: <https://aclanthology.org/2024.findings-acl.411/>.

- [18] G. Lample et al. “Deep learning for symbolic mathematics”. In: *8th International Conference on Learning Representations, ICLR 2020*. <https://openreview.net/forum?id=S1eZYeHFDS>. 2020. URL: https://iclr.cc/virtual_2020/poster_S1eZYeHFDS.html.
- [19] L. Phan et al. *Humanity’s Last Exam*. 2025. arXiv: 2501.14249 [cs.LG]. URL: <https://arxiv.org/abs/2501.14249>.
- [20] S. I. Mirzadeh et al. “GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=AjXkRZIVjB>.
- [21] J. Boye et al. *Large Language Models and Mathematical Reasoning Failures*. 2025. arXiv: 2502.11574 [cs.AI]. URL: <https://arxiv.org/abs/2502.11574>.
- [22] Z. Zhou et al. “MathAttack: attacking large language models towards math solving ability”. In: *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI’24/IAAI’24/EAAI’24. AAAI Press, 2024. URL: <https://doi.org/10.1609/aaai.v38i17.29949>.
- [23] K. Huang et al. “MATH-Perturb: Benchmarking LLMs’ Math Reasoning Abilities against Hard Perturbations”. In: *Workshop on Reasoning and Planning for Large Language Models*. 2025. URL: <https://openreview.net/forum?id=M80LGgYK7e>.
- [24] Z. Zhou et al. “Is Your Model Really A Good Math Reasoner? Evaluating Mathematical Reasoning with Checklist”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=nDvgHIBRxQ>.
- [25] B. Jiang et al. “A Peek into Token Bias: Large Language Models Are Not Yet Genuine Reasoners”. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Y. Al-Onaizan et al. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024. URL: <https://aclanthology.org/2024.emnlp-main.272/>.
- [26] A. Novikov et al. *AlphaEvolve: A coding agent for scientific and algorithmic discovery*. Tech. rep. Google DeepMind, May 2025. URL: <https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/alphaevolve-a-gemini-powered-coding-agent-for-designing-advanced-algorithms/AlphaEvolve.pdf>.
- [27] X. Yue et al. “MAMmoTH: Building Math Generalist Models through Hybrid Instruction Tuning”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: <https://openreview.net/forum?id=yLC1Gs770I>.
- [28] A. Zhou et al. “Solving Challenging Math Word Problems Using GPT-4 Code Interpreter with Code-based Self-Verification”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: <https://openreview.net/forum?id=c8McWs4Av0>.
- [29] OpenAI. *Introducing o3 and o4-mini*. Apr. 2025. URL: <https://openai.com/index/introducing-o3-and-o4-mini/>.
- [30] M. Liao et al. “MARIO: MATH Reasoning with code Interpreter Output - A Reproducible Pipeline”. In: *Findings of the Association for Computational Linguistics: ACL 2024*. Ed. by L.-W. Ku et al. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024. URL: <https://aclanthology.org/2024.findings-acl.53/>.
- [31] Z. Gou et al. “ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving”. In: *ICLR*. 2024. URL: <https://openreview.net/forum?id=Ep0TtjVoap>.
- [32] S. Imani et al. “MathPrompter: Mathematical Reasoning using Large Language Models”. In: *Annual Meeting of the Association for Computational Linguistics*. 2023. URL: <https://api.semanticscholar.org/CorpusID:257427208>.
- [33] B. Romera-Paredes et al. “Mathematical discoveries from program search with large language models”. In: *Nature* 625 (2023). URL: <https://api.semanticscholar.org/CorpusID:266223700>.
- [34] O. Dugan et al. “OccamLLM: Fast and Exact Language Model Arithmetic in a Single Step”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/3eceb70f47690051d6769739fbf6294b-Paper-Conference.pdf.
- [35] W. R. Inc. *Mathematica, Version 14.2*. Champaign, IL, 2024. URL: <https://www.wolfram.com/mathematica>.

- [36] Wolfram Alpha LLC. *Wolfram—Alpha: Computational Intelligence*. <https://www.wolframalpha.com/>. 2025.
- [37] A. Meurer et al. “SymPy: symbolic computing in Python”. In: *PeerJ Computer Science* 3 (Jan. 2017). ISSN: 2376-5992. URL: <https://doi.org/10.7717/peerj-cs.103>.
- [38] B. Gao et al. “Omni-MATH: A Universal Olympiad Level Mathematic Benchmark for Large Language Models”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=yaqPf0KA1N>.
- [39] K. Chernyshev et al. *U-MATH: A University-Level Benchmark for Evaluating Mathematical Skills in LLMs*. 2025. arXiv: 2412.03205 [cs.CL]. URL: <https://arxiv.org/abs/2412.03205>.
- [40] M. Fang et al. *MathOdyssey: Benchmarking Mathematical Problem-Solving Skills in Large Language Models Using Odyssey Math Data*. 2024. arXiv: 2406.18321 [cs.CL]. URL: <https://arxiv.org/abs/2406.18321>.
- [41] S. Frieder et al. “Mathematical capabilities of chatgpt”. In: *Advances in neural information processing systems* 36 (2023).
- [42] X. Xu et al. “UGMathBench: A Diverse and Dynamic Benchmark for Undergraduate-Level Mathematical Reasoning with Large Language Models”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=fovPyqPcKY>.
- [43] U. L. Brazilian Mathematical Olympiad. *OBMU 2019 Mathematics Competition*. Accessed: 2025-04-08. 2019. URL: <https://www.obm.org.br>.
- [44] Z. Huang et al. “OlympicArena: Benchmarking Multi-discipline Cognitive Reasoning for Superintelligent AI”. In: *arXiv preprint arXiv:2406.12753* (2024). URL: <https://arxiv.org/abs/2406.12753>.
- [45] C. He et al. “OlympiadBench: A Challenging Benchmark for Promoting AGI with Olympiad-Level Bilingual Multimodal Scientific Problems”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by L.-W. Ku et al. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024. URL: <https://aclanthology.org/2024.acl-long.211/>.
- [46] Y. Zhang et al. “Training and Evaluating Language Models with Template-based Data Generation”. In: *arXiv preprint arXiv:2411.18104* (2024).
- [47] S. Shrestha et al. *Mathematical Reasoning in Large Language Models: Assessing Logical and Arithmetic Errors across Wide Numerical Ranges*. 2025. arXiv: 2502.08680 [cs.LG]. URL: <https://arxiv.org/abs/2502.08680>.
- [48] S. Srivastava et al. *Functional Benchmarks for Robust Evaluation of Reasoning Performance, and the Reasoning Gap*. 2024. arXiv: 2402.19450 [cs.AI]. URL: <https://arxiv.org/abs/2402.19450>.
- [49] J. Vendrow et al. “Large Language Model Benchmarks Do Not Test Reliability”. In: *Neurips Safe Generative AI Workshop 2024*. 2024. URL: <https://openreview.net/forum?id=XSeN6xZtZ9>.
- [50] W. Zhang et al. *Beyond the Singular: The Essential Role of Multiple Generations in Effective Benchmark Evaluation and Analysis*. 2025. arXiv: 2502.08943 [cs.CL]. URL: <https://arxiv.org/abs/2502.08943>.
- [51] A. Patel et al. “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*. Online: Association for Computational Linguistics, June 2021. URL: <https://aclanthology.org/2021.naacl-main.168>.
- [52] S. Pichai et al. *Introducing Gemini 2.0: our new AI model for the agentic era*. Dec. 2024. URL: <https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/#ceo-message>.
- [53] Y. Mao et al. *CHAMP: A Competition-level Dataset for Fine-Grained Analyses of LLMs’ Mathematical Reasoning Capabilities*. 2024. arXiv: 2401.06961 [cs.CL]. URL: <https://arxiv.org/abs/2401.06961>.
- [54] OpenAI. URL: <https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf>.

- [55] K. Kavukcuoglu. *Gemini 2.5: Our most intelligent AI model*. Mar. 2025. URL: <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/>.
- [56] OpenAI. *Introducing GPT-4.1 in the API*. Apr. 2025. URL: <https://openai.com/index/gpt-4-1/>.
- [57] OpenAI. *GPT-4o System Card*. 2024. arXiv: 2410.21276 [cs.CL]. URL: <https://arxiv.org/abs/2410.21276>.
- [58] DeepSeek-AI. *DeepSeek-V3 Technical Report*. 2025. arXiv: 2412.19437 [cs.CL]. URL: <https://arxiv.org/abs/2412.19437>.
- [59] DeepSeek-AI. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. URL: <https://arxiv.org/abs/2501.12948>.
- [60] Z. Z. Ren et al. *DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition*. 2025. arXiv: 2504.21801 [cs.CL]. URL: <https://arxiv.org/abs/2504.21801>.
- [61] Meta. *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation*. Apr. 2025. URL: <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>.
- [62] A. Yang et al. “Qwen2.5 Technical Report”. In: *arXiv preprint arXiv:2412.15115* (2024).
- [63] C. Farabet et al. *Introducing Gemma 3: The most capable model you can run on a single GPU or TPU*. Mar. 2025. URL: <https://blog.google/technology/developers/gemma-3/>.
- [64] A. Bercovich et al. *Llama-Nemotron: Efficient Reasoning Models*. 2025. arXiv: 2505.00949 [cs.CL]. URL: <https://arxiv.org/abs/2505.00949>.

A Additional Details About the Dataset

A.1 List of Equivalence Perturbations

The complete list of equivalence perturbations, discussed in section 2.1, is provided below.

Easy:

- **Trigonometric:** $\sin^2(-Qx) + \cos^2(Qx)$
- **Hyperbolic:** $\cosh^2(Qx) - \sinh^2(Qx)$
- **Logarithmic:** $\frac{\ln(x) \cdot \log_x(Q)}{\ln(Q)}$
- **Complex exponential:** $\frac{Q \sum_{N=1}^{\infty} \frac{2^{-N} x}{Q}}{x}$
- **Series:** $-\frac{i(e^{iQx} - e^{-iQx})}{2 \sin(Qx)}$

Hard:

- **Trigonometric:** $\frac{\tan(x) + \tan(x(Q-1))}{(-\tan(x) \tan(x(Q-1)) + 1) \tan(Qx)}$
- **Hyperbolic:** $\frac{\sinh\left(\log\left(Qx + \sqrt{Q^2 x^2 + 1}\right)\right)}{Qx}$
- **Logarithmic:** $\frac{\log_Q\left(\frac{x}{e}\right) + \log_Q(e)}{\log_Q(x)}$
- **Complex exponential:** $-\frac{2i(e^{4iQx} + 1) \tan(Qx)}{(1 - e^{4iQx})(1 - \tan^2(Qx))}$
- **Series:** $\frac{Q \sum_{N=1}^{\infty} \frac{6x}{\pi^2 N^2 Q}}{x}$

These perturbations were selected to avoid significantly altering the mathematical complexity of the challenge compared to the original questions. We confirmed that the tested LLMs could correctly simplify each expression when presented individually, indicating that the ‘Equivalence’ variants’ difficulty arises mostly from the original question and its combination with the equivalence perturbations.

Notably, SymPy [37] was unable to simplify the more difficult trigonometric and hyperbolic identities to 1, providing another example for CAS limitations in university-level symbolic math challenges.

Figure 3 shows that for most LLMs the challenge level of a single ‘Hard’ perturbation is lower than multiple ‘Easy’ perturbations - but not for all LLMs. The reasons behind this difference are a topic for future investigation.

A.2 ‘Symbolic-N’ Subsets Analysis

Due to the requirement that substituting all symbols with 1 reverts the question to its original seed form, the total number of ‘Symbolic-N’ variations depends on N. For instance, ASyMOB contains only 7 ‘Symbolic-5’ questions. This small sample size is the reason ‘Symbolic-5’ is not represented in Figure 2, as it is insufficient for robust statistical analysis. This variability also means that the baseline difficulty of ‘Symbolic-N’ questions changes with different values of N. The 7 seed questions with a maximal perturbation of 5 symbols have an average success rate across all models of 86.6%. In contrast, the 13 seed questions with a maximal perturbation of 4 symbols have a 74.7% success rate, and the overall success rate across all seeds is 73.9%. The ‘Symbolic-4’ subset includes 13 questions with maximal symbolic perturbation (derived from the 13 seeds mentioned above) and

35 permutations based on the 7 maximally perturbed ‘Symbolic-5’ questions. It is likely that the lower initial difficulty of the seeds influences the difficulty of their derived variations to some extent. Therefore, the difficulty of each ‘Symbolic’ subset should not be assumed to be identical. This effect can account for the slight increase in success rate observed across most models in the bottom graphs of Figure 2 for 3 and 4 symbols.

B Testing Details

As noted in section 2.2, a core principle of the test process is to rely on deterministic and predictable tools whenever possible. Figure 4 shows a “Formatting Instructions” wrap around the challenge text. Specifically, these instructions state:

“Finish your answer by writing ”The final answer is:” and then the answer in LaTeX in a new line. Write the answer as a single expression. Do not split your answer to different terms. Use \$\$ to wrap the LaTeX text. Do not write anything after the LaTeX answer.”

The primary goal is to encourage the LLM to produce a clear LaTeX expression, labeled with “The final answer is:”. We opt against using forced structured outputs, even when available, to ensure a fair comparison with models lacking this capability and to avoid introducing requirements beyond symbolic math skills. In essence, we aim to minimize the impact of specific phrasing and structural choices in both language and mathematical presentation.

Once the full answer is received, a series of regexes are used to extract the final answer:

Pattern 1 (as instructed):
`r'**[Tt]he final answer is:?*\s*'`
`r'(?:(?:\\\) |(?:\\\[|(?:\$+)))'`
`r'(.*)'`
`r'(?:(?:\\\) |(?:\\\[|(?:\$+)))'`

Pattern 2 (last boxed expression):
`r'\\boxed\{(.*)\}' + '(?:\n|$|")'`

Pattern 3 (last display expression):
`r"\$+ (.*) \$+ "`

Pattern 4 (output=' ' case):
`r"output='(.*)'"`

Pattern 5 (output=" " case):
`r'output="(.*)"'`

While the first pattern represents the given formatting instructions - other output formats were accepted as well. It’s important to note that responses claiming, for example, the challenge is impossible or asking for specific values to substitute into the symbols, will frequently lack fitting LaTeX expressions. Therefore, the absence of relevant LaTeX usually indicates a missing or incoherent answer, not a parsing issue. Overall, this stage was successful in 98% of cases.

The extracted LaTeX expression is then cleaned and parsed into a SymPy expression using `sympy.parsing.latex.parse_latex`. If the parsing fails, we resort to using an LLM (gemini-2.0-flash) for this translation. It’s important to note that not all “final answer” expressions extracted by our permissive regexes are valid LaTeX or even mathematical expressions. Therefore, a failure to produce a working SymPy expression usually indicates a broken or irrelevant answer, rather than a translation issue. Overall, this stage was successful in 96.1% of cases.

The resulting SymPy expression undergoes two distinct validation checks against the reference answer (also represented as a SymPy object):

Symbolic validation. The difference between the extracted expression and the correct answer is simplified via `SymPy.simplify`. If the simplification reduces this difference to zero (or a constant, in the case of indefinite integrals), the answer is deemed correct.

Numeric validation. We randomly generate numerical values for each variable (e.g., x and any symbolic perturbation parameters) and substitute them into both the LLM’s result and the correct answer. If the relative difference between the two evaluations is less than 0.002%, the answers are considered matching. This process is repeated five times to mitigate the risk of coincidental matches. To allow the detection of numeric equivalence between indefinite integrals, we require that all 5 repetitions produce the same difference (not necessarily zero), concluding that the expressions are equivalent up to a constant factor.

Due to the limitations of SymPy (imperfections in `SymPy.simplify`, handling of very large numbers in `.evalf()`, etc.), if either validation method confirms an answer, it is treated as correct (false positives are highly unlikely). Out of all the valid SymPy expressions created on the previous stage, 97.6% were successfully tested. Responses that could not be verified by either method due to SymPy’s technical limitations were excluded from the data analysis and omitted from the reported statistics.

In terms of resources required for this work, by far the biggest compute consumer was querying the LLMs. The total number of successful queries for all the tests is $17092 \cdot 17 = 290,564$ (5-10% additional calls were made during development, and due to provider rate limits and network issues). These were done via cloud API calls, for a total expense of: OpenAI $\sim 1400\$$ (for o4-mini, GPT-4.1, GPT-4o, GPT-4o-mini), Google $\sim 150\$$ (for Gemini 2.5 Flash, Gemini 2.0 Flash, Gemma-3-27b-it), Hugging Face $\sim 600\$$ (for all other models; interface providers used: Novita, Together AI, Nebius AI Studio). Temperature was set to the default value (1 for OpenAI and Google, 0.5 for Hugging Face).

Dataset generation compute was negligible (less than 5 minutes on a single workstation), while the validation stage was more resource-intensive (~ 10 hours on 3 workstations). Note that the validation process is trivially parallelizable.

C Data Analysis

Figure 7 presents each model’s (with and without code execution) success on each seed question - showing a mix of easier and harder challenges.

Figure 8 illustrates the variance within each 50-question subset of variant ‘Numeric-All-2-S’ (per seed). Each cell is marked with a ‘V’ if the model correctly solved at least half of the ‘Numeric-All-2-S’ variants from that seed question, and an ‘X’ otherwise.

It is important to note that while correct answers are unique (aside from presentation differences), incorrect answers can vary significantly, including instances where no answer is provided. Consequently, low consistency might result in lower variance for questions with a low success rate compared to those with a high success rate. Indeed, the average variance for all ‘V’ questions is 0.11, whereas for ‘X’ questions, it is 0.07.

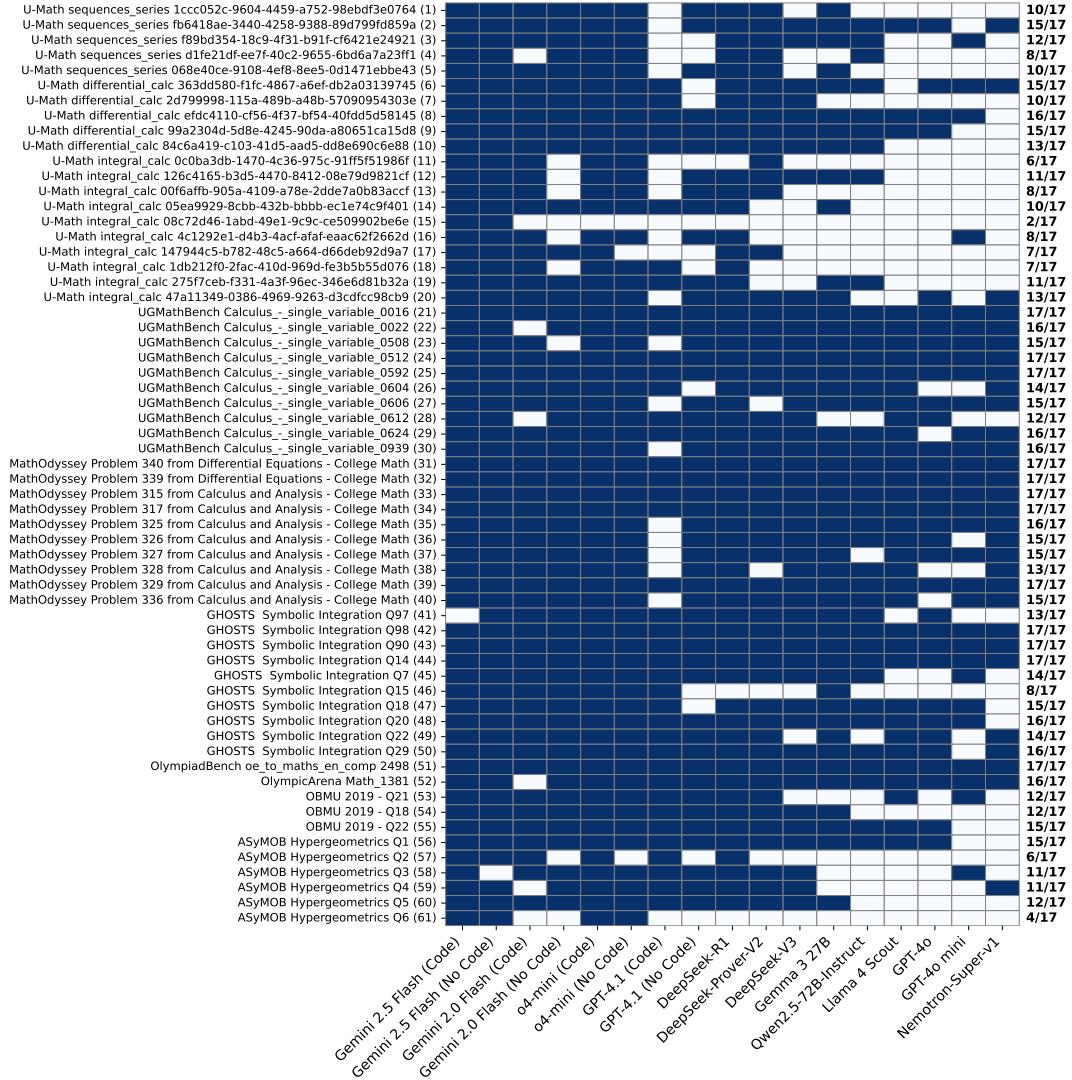


Figure 7: **Model success (blue) / failure (white) per seed question.** Seeds are marked by their source and index in the dataset. Note the difference in challenge level between seeds with different sources. ‘ASyMOB’ source indicates original questions that were created for the purpose of this work.

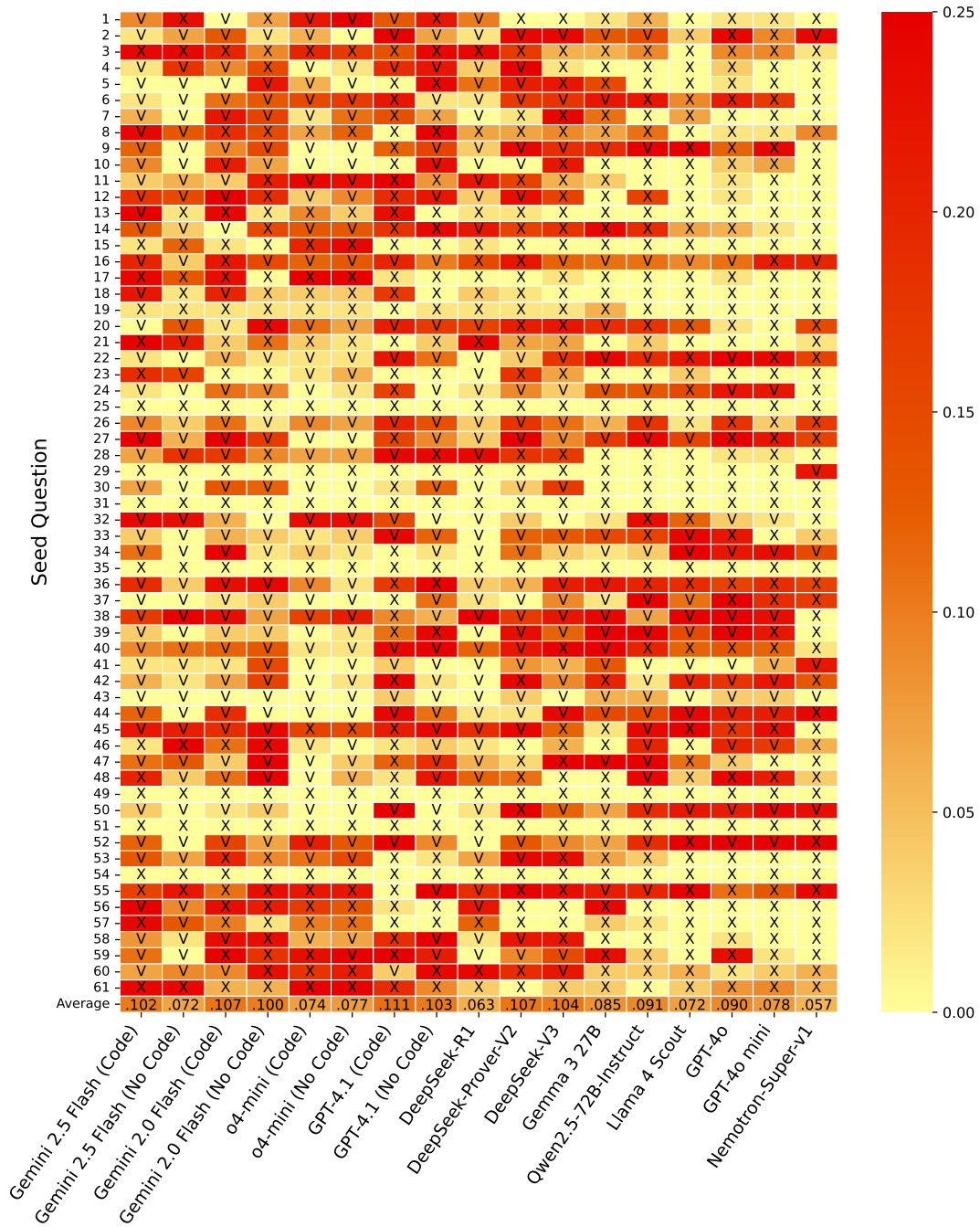


Figure 8: **Variance per model per seed question.** ‘V’/ ‘X’ marks a majority of success/failure in the corresponding 50-variant subset. The bottom row shows the average variance of each model across all questions.