MEASURING AI PROGRESS TOWARD MATHEMATICAL DISCOVERY WITH AUTOMATIC VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Can AI make progress on important, unsolved mathematical problems? Large language models are now capable of sophisticated mathematical and scientific reasoning, but whether they can perform novel research is still widely debated and underexplored. We introduce **HorizonMath**, a benchmark of 113 predominantly unsolved problems spanning eight domains in mathematics and the mathematical sciences, paired with an open-source evaluation framework for automated verification. Our benchmark targets the generator-verifier gap: problems where discovery is hard and requires meaningful mathematical insight, but verification is computationally straightforward. This contrasts with most existing research-level benchmarks, which instead rely on formal proof verification or manual review, both of which are expensive to scale. Because these solutions are unknown, HorizonMath is resistant to data contamination, and most state-of-the-art models score under 10%. Using this framework, we identify five novel solutions to research problems that either resolve previously open questions or improve on the best-known published results, with GPT-5.4 Pro discovering two and GPT 5.6 Sol discovering three of these solutions. Across seven frontier model families, reasoning efficiency and behavior also vary substantially. We release HorizonMath as an open challenge and a growing community resource, where each verified solution is a candidate contribution to the mathematical literature.[1]

1 INTRODUCTION

Making autonomous mathematical discoveries is a north star in AI research, as mathematics is a key driver of scientific discovery. FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Georgiev et al., 2025) improved several open bounds in combinatorics, geometry, and number theory; frontier LLM-based agents have resolved previously unsolved Erdős problems (Feng et al., 2026; Sothanaphan, 2026); and human-AI collaboration has produced new results across theoretical computer science, physics, and pure mathematics (Knuth, 2026; Guevara et al., 2026; Woodruff et al., 2026). As these capabilities continue to advance, standardized methods are needed to systematically measure AI progress toward autonomous mathematical discovery.

Many mathematics benchmarks exist, but most were designed to test *problem-solving* rather than *discovery*, which requires assessing novel mathematical contributions. This problem-solving framing of applying known techniques to problems with known answers has a natural ceiling—human knowledge—and even the most challenging Olympiad-style and graduate-level benchmarks (Luong et al., 2025; Tsoukalas et al., 2024; Roggeveen et al., 2025; Rein et al., 2024) evaluate problems with known solutions. As such, there exist very few benchmarks that can provide signal on whether AI systems can perform research and produce results not already in the literature.

On the other hand, measuring capabilities on unsolved problems is *naturally difficult* because solutions are unknown, but it is crucial since it requires genuinely novel mathematical contributions. We therefore exploit the generator-verifier gap, a feature in certain classes of problems in which candidate solutions are hard to produce but efficient to check. We identify three such forms that are well-suited to automated evaluation: 1) problems where a closed-form solution is currently unknown, but a candidate expression can be checked against a high-precision numerical reference, 2)

[1]The anonymized benchmark and codebase is available here: `https://anonymous.4open.science/r/HorizonMath-ICLR-2027/README.md`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

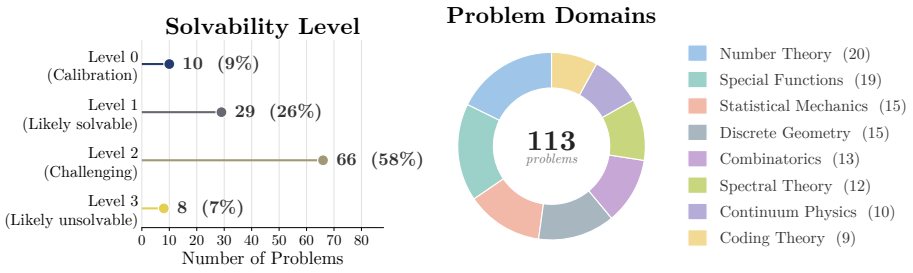


Figure 1: Benchmark composition by solvability level (left) and mathematical domain (right). HorizonMath comprises 113 problems, distributed across several areas of mathematics and physics.

construction and optimization problems seeking objects that improve upon a current baseline not known to be optimal, and 3) existence problems where a target object has not been found but a candidate can be validated by checking that it satisfies all required properties. All three classes can be efficiently verified by deterministic computation, making them natural targets for a scalable benchmark of mathematical discovery (Figure 2).

With these classes of problems, we present **HorizonMath**, a benchmark of 113 predominantly unsolved problems and associated verification scripts that automatically evaluate proposed answers against problem-specific criteria (Figure 1). Our framework is defined by three key contributions:

1. **A contamination-resistant benchmark of open problems:** We present a benchmark of 113 problems across eight subject areas in applied and computational mathematics. Because the open problems in our benchmark do not have publicly known solutions, they are resistant to memorization and any correct solution produced by a model would reflect reasoning ability and autonomous discovery.
2. **Automated verification:** Evaluations of mathematical research capabilities are traditionally bottlenecked by human review. We automate verification using high-precision numeric comparison and deterministic constraint-checkers, using the generator-verifier gap to provide a fast and objective signal supporting correctness.
3. **Open-source and standardized evaluation:** Unlike other benchmarks of unsolved research-level mathematics, our framework includes complete problem definitions, ground-truth computations, and verifier scripts, and is publicly available, with discovered solutions released in encrypted form.

We note that while matching a high-precision numerical reference does not formally prove that a closed-form expression is exactly correct, our combination of numerical comparison and verification of permitted operations provides significant evidence for correct solutions and can effectively filter inadmissible ones. We identify two novel solutions from GPT-5.4 Pro and three from GPT-5.6 Sol Max that either resolve previously open questions or improve on the best-known published results; these five results are described in Section 6, and two are detailed in Appendix A. **HorizonMath** is an open-source effort that welcomes contributions from the community and will be made available following double-blind peer-review.

2 Related work

We organize related work into three categories that highlight the contrast between solution and discovery, introduced in Section 1. We first review *solution-driven* mathematical benchmarks, which evaluate models on problems with known answers (Section 2.1). We then survey *research-level* benchmarks, which target unsolved problems and help define our design space, and also note the key differences between these works and **HorizonMath** (Section 2.2). Finally, we discuss recent demonstrations of AI-driven mathematical discovery, which establish the capabilities that **HorizonMath** seeks to measure (Section 2.3). Related domains and datasets are discussed in Appendix E.

2.1 SOLUTION-DRIVEN MATHEMATICAL BENCHMARKS

Early benchmarks like GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) are now saturated, and while recent benchmarks targeting higher difficulty levels, such as GPQA (Rein et al., 2024) and the HARDMath series (Fan et al., 2025) for graduate-level problems have emerged, these benchmarks all evaluate problems with *known* solutions and therefore cannot measure novel discovery. Benchmarks based on competitions and Olympiads (e.g., OlympiadBench (He et al., 2024), IMO-Bench (Luong et al., 2025), and MathArena (Balunovic et al., 2025)) have provided new and difficult problems but are typically conducted on an annual basis and do not readily support continuous measurement of research progress, especially as they are all already-solved problems.

Table 1: Comparison of math reasoning benchmarks.

Dataset	Problems	Difficulty	Unsolved	Non-Proof-Centric	Open-evaluation	Auto-verify
FrontierMath (Glazer et al., 2024)	300+	Research	×	✓	×	✓
FrontierMath: Open Problems (Epoch AI, 2026a)	14	Research	✓	✓	×	✓
IMProofBench (Schmitt et al., 2025)	39	Research	×	×	×	×
First Proof (Abouzaid et al., 2026)	10	Research	×	×	×	×
Erdős Problems (Bloom, 2021)	1,000+	Research	✓	×	×	×
IMO-AnswerBench (Luong et al., 2025)	400	Olympiad	×	×	✓	✓
Opt. Constants (Davis et al., 2026)	96	Research	✓	✓	×	×
HorizonMath	113	Research	✓	✓	✓	✓

2.2 RESEARCH-LEVEL BENCHMARKS

In response to the rapidly-advancing capabilities of AI research agents and the saturation of existing benchmarks, several research-level benchmarks have recently appeared. These benchmarks require novel solutions to unsolved problems rather than reproductions of known answers and are typically hand-crafted by experts. FrontierMath (Glazer et al., 2024) comprises several hundred unpublished mathematics problems spanning difficulty tiers from undergraduate through research level. Its most advanced tier targets research-level mathematics, and its "Open Problems" subset contains 14 unsolved problems, some of which have already been solved and removed from the dataset (Epoch AI, 2026a). Moreover, their verifiers are all private; proposed solutions can only be checked by purchasing access to the verifier. IMProofBench (Schmitt et al., 2025) also offers a set of 39 research-level proof problems written by human experts. Both keep their problems and verifiers private or rely heavily on expert human grading, limiting reproducibility.

Similarly, First Proof (Abouzaid et al., 2026) comprises 10 contamination-resistant questions drawn from its authors' unpublished research, but its small scale, manual evaluation, and proof orientation limit its scalability for fast and rigorous model evaluation.

2.3 AI-DRIVEN MATHEMATICAL DISCOVERY

One of the first LLM-driven results on open problems came from FunSearch (Romera-Paredes et al., 2024), which evolved programs to construct large cap sets beating the best known bounds in extremal combinatorics and discover improved heuristics for bin-packing. AlphaEvolve (Novikov et al., 2025) generalized this evolutionary approach and made improvements to complex matrix multiplication algorithms and a kissing-number lower bound. AlphaEvolve's results targeted many topics outside of mathematics, so Georgiev, Tao, Gómez-Serrano, and Wagner (Georgiev et al., 2025) applied AlphaEvolve to 67 problems across analysis, combinatorics, geometry, and number theory, rediscovering most best-known constructions and improving several—such as new bounds for the finite field Kakeya problem and an optimal tile arrangement for a problem from the 2025 International Mathematical Olympiad.

In parallel, human-AI collaboration has resolved further open problems: Woodruff et al. (2026) used Gemini Deep Think on problems in theoretical computer science, information theory, and physics, and Knuth (2026) similarly used Claude Opus 4.6 to obtain a closed-form Hamiltonian-cycle decomposition of a Cayley digraph—a longstanding open problem.

Recent systems push even further toward autonomy. DeepMind's Aletheia (Feng et al., 2026) produced multiple papers with minimal human intervention. OpenAI's GPT-5.2 through 5.6 Pro models

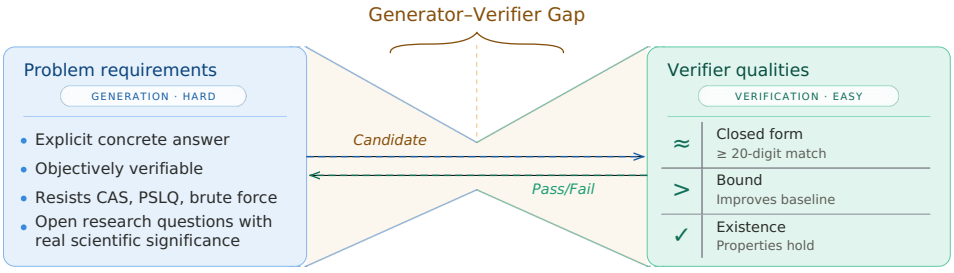


Figure 2: The *generator–verifier gap* of **HorizonMath**. Problems require hard mathematical discovery, while submitted candidates are checked by deterministic verifiers. Formal proofs are not required for any problems in the dataset.

have similarly resolved many Erdős problems (Sothanaphan, 2026; Alexeev et al., 2026). In physics, GPT-5.2 Pro conjectured a closed-form expression for single-minus gluon amplitudes (Guevara et al., 2026) which was then verified by the human authors. Even more recently, Anthropic's models have produced counterexamples to the Jacobian conjecture in dimension three and variations of the Riemann zeta function (Tao; Anthropic, 2026). OpenAI's internal models have produced numerous results resolving or advancing longstanding problems, including the Navier-Stokes problem (OpenAI, 2026). This rapid growth in AI-driven research motivates a standardized and scalable supply of unsolved yet meaningful research problems that can be used to efficiently evaluate such systems.

3 Benchmark Design and Evaluation

3.1 Design Principles

HorizonMath follows four design principles. First, each problem must require an *explicit answer* in the form of a definite mathematical object such as a number, polynomial, set, or graph, rather than a natural-language proof. Second, this answer must be *objectively verifiable* by a simple deterministic computational procedure, whether by numerical comparison against a high-precision reference value, by checking improvement over a published baseline, or by validating that a construction satisfies all required properties. Third, the problem must demand long-horizon *mathematical reasoning* and resist solution by direct application of standard computational algorithms such as computer algebra systems. Fourth, each problem must have *scientific significance*, meaning it is sourced from the mathematical literature or active research rather than artificially constructed for the benchmark.

All problems are sourced from the research literature (such as journal and conference publications, textbooks, and arXiv preprints) across three categories: problems without known closed-form solutions, optimization problems whose current results are not proven optimal, and construction problems where the target object has not been found. AI agents—specifically, Codex and Claude Code—with web search were used to compile a large, initial list of relevant problems, which was iteratively refined to this subset by human experts. Importantly, this made the collection process significantly cheaper and more efficient than problems in comparable benchmarks, which were all compiled by human experts. (E.g., FrontierMath: Open Problems cost roughly $300–$1000 USD per problem due to expert curation, whereas nothing was spent on experts in HorizonMath (Epoch AI, 2026b).) Furthermore, none of the problems in the unsolved categories of the benchmark have a known exact or optimal solution in the required form. The 100+ problems have been carefully selected under these constraints and refined by mathematicians with relevant expertise. Additionally, most problems in the Levels 2 and 3 categories are well-known problems that would constitute substantial mathematical discoveries if solved.

3.2 Dataset Composition

We classify each problem along three axes: the structural output type of the expected solution, the solvability level, and the mathematical domain.

Output types: All problems require concrete solutions formulated in Python code rather than natural-language proofs. The majority are constant discovery problems, where the model must identify an exact closed-form expression for a numerically approximated target value. Construction tasks form the second largest category, requiring the model to produce discrete mathematical objects—such as lattices, point configurations, or adjacency matrices—that improve upon existing baselines. The remaining problems involve generating computable functions with specific properties or discovering general formulas.

Solvability levels: We define four levels to gauge difficulty. Level 0 problems are problems that have already been solved by humans (meaning there exist known closed forms or constructions), included for calibration. This does not imply that all models are capable of solving these problems, however. Levels 1–3 are problems currently unsolved by humans and are assigned the following tiers based on the expertise of the authors. Level 1 problems are judged to be likely solvable with known techniques or near-term capabilities; Level 2 problems require significant new insights and are more challenging than Level 1 problems; and Level 3 problems are problems whose potential solutions would be considered major breakthroughs. Since these assignments were made by authors of the benchmark who have mathematical experience in the relevant subfields, they are not objective and are intended as a guide rather than a precise difficulty score.

Mathematical domains: The benchmark contains problems across eight subject areas: *Number Theory*, *Special Functions*, *Statistical Mechanics*, *Discrete Geometry*, *Combinatorics*, *Spectral Theory*, *Continuum Physics*, and *Coding Theory*. Each problem is assigned to its corresponding mathematical subject. The distribution in Figure 1 reflects the current snapshot, but since it is public and welcomes new contributions from the community, its composition will evolve.

3.3 Admissible solutions for closed forms

For closed-form discovery problems, we enforce strict structural requirements. While the definition of a closed-form is somewhat field-dependent, we adopt a fairly open-ended convention: it must be a finite symbolic expression built exclusively from the following set of operations: rationals, algebraic numbers, standard transcendentals (π, e, Euler's γ, Catalan's G), elementary functions ($\exp$, $\log$, trigonometric, and hyperbolic functions), and special functions as long as they are defined in `mpmath` (Borwein & Crandall, 2013; Johansson et al., 2013). We choose this flexible definition given the lack of a universal agreement on what "closed-form" means. These conventions define admissibility for our benchmark. For problems classified as unsolved, our review of the literature and assessment from mathematicians on our team suggest that a correct solution in the required form would constitute a new mathematical result.

We explicitly forbid solutions that include unevaluated integrals, infinite series, limits, implicit definitions, and numerical approximations presented as exact values—except for those that are special functions. We also do not permit solutions that use computational tools such as numerical integration, truncated infinite series, numerical root-finding, and computation of resultants by evaluating one polynomial at the roots of another. We also exclude closed-form problem classes that can be directly solved via standard deterministic algorithms, such as the Remez algorithm, PSLQ, or symbolic differentiation and integration. Finally, we show at most five significant figures of precision in the prompt to limit direct fitting to our full-precision approximation.

4 Automated verification

A core feature of our benchmark is a fully automated and reproducible evaluation pipeline. Unlike theorem-proving benchmarks, which require complex formalization in proof assistants or expert grading, and unlike other verifiable benchmarks that do not release their evaluation infrastructure, our framework enables fast, automatic verification.

4.1 Execution pipeline and admissibility

Frontier models are prompted to generate a self-contained Python function, `def proposed_solution()`, returning the solution as either an `mpmath` symbolic expres-

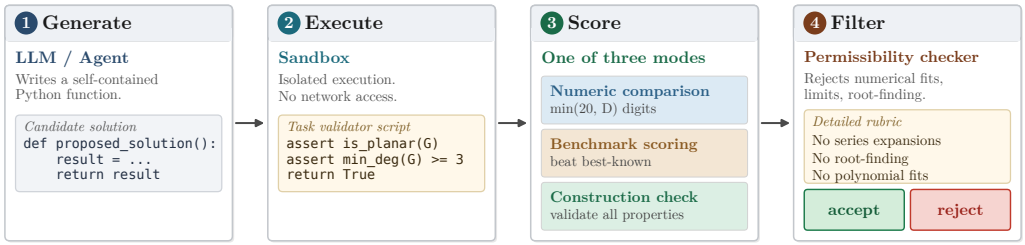


Figure 3: **HorizonMath**'s automated evaluation pipeline. A model generates a proposed solution, which is screened by a compliance checker that rejects forbidden operations. Admissible solutions are routed to one of three evaluation modes: numeric comparison against a ground-truth value, benchmark scoring for improvement over the best-known result, and construction checking to validate all required structural properties.

sion or a JSON-serializable dictionary representing a discrete construction. Strict admissibility criteria are enforced on the proposed solutions in the system prompt as well as via the compliance checker. Closed-form expressions must be finite symbolic combinations of permissible operations (rational numbers, elementary functions, standard transcendentals at algebraic arguments); infinite series, limits, implicit definitions, and numerical approximations are forbidden. Explicit constructions must follow the output type defined in the function template. These constraints are enforced via an LLM-based compliance checker, which inspects the proposed solution for forbidden operations such as numerical root-finding, quadrature, and hard-coded constants. Appendix B.6 provides further details, and Appendix B.7 gives a detailed analysis highlighting the checker's robustness and consistency, especially with stronger models. We additionally manually review all numerically passing candidates counted as successful and those rejected to further verify its results; we found no incorrect rejections. We do not restrict tool use (e.g., code execution or web search) in the APIs to make the problem-solving environment as realistic and capable as possible. Every returned solution script received 110-digit `mpmath` precision and a five-minute execution limit.

4.2 Evaluation modes and scoring mechanics

Closed-form problems require the model to produce a symbolic closed-form expression, evaluated to high precision via `mpmath` and compared against a reference value typically computed via integration or series expansion. A candidate is accepted if it matches the reference to $\min(20, D)$ decimal digits, where D is the number of verified digits available for that ground truth. **Bound-improvement** problems target optimization tasks where the global optimum is unknown but a published baseline exists. The model's output is executed and scored by problem-specific validators on the relevant metric (e.g., packing density, asymptotic bounds). A solution passes if it strictly improves upon the best-known result, and solutions are further ranked by relative improvement over the baseline. **Existence** problems concern construction questions, such as finding a Hadamard matrix of an unresolved order or a set of mutually orthogonal Latin squares of a given size. Scoring is pass/fail: deterministic, problem-specific validators exhaustively check whether the proposed object satisfies all required properties, with no baseline comparison.[2]

We acknowledge that while matching a high-precision reference value does not completely prove the correctness of a closed-form expression, *such solutions that pass the permissibility check can be regarded as high-confidence conjectures until proven*. While a formal proof would be needed to guarantee the correctness, new theorems often first begin with highly-accurate conjectures, making this numerical framework essential. Constructions that beat a published bound or satisfy the corresponding validator, by contrast, are verified deterministically by checking all required properties.

Solution release: Because **HorizonMath** is designed so that new solutions continue to be discovered, publicly publishing all solutions would expose them to web search during evaluation and future

[2]In the released dataset these modes respectively correspond to the identifiers `ground_truth_computable`, `benchmark_best_known`, and `new_construction`.

training data, making results difficult to compare across models released at different times. As such, we provide encrypted solutions on the GitHub—groups that wish to inspect a solution can request a key that we would provide. In the anonymized codebase linked above, we provide a password that can be used to inspect the solutions purely for the review process. Importantly, new submissions can be validated with the benchmark's public verifiers without access to earlier solutions.

5 REPRESENTATIVE PROBLEM EXAMPLES

We illustrate the benchmark with three representative problems covering the `closed-form`, `bound-improvement`, and `new-construction` categories (the last in Appendix C); the examples below show the core task and output specification given to the model.

1. Spinor Norm Elliptic Integral (I_0)

Special Functions | *Constant* | `Closed-Form` | *Solvability 1*

Task. Let $K(m)$ and $E(m)$ denote the complete elliptic integrals of the first and second kinds in the parameter convention. Define

$$I_0 = \frac{4\sqrt{2}}{\pi} \int_0^1 \frac{1+\sqrt{z}}{(1+z)^3} \left(2E(z) - (1-z)K(z)\right)\, dz.$$

This elliptic-integral constant arises in a spinor norm calculation and has numerical value $I_0 \approx 1.77425\ldots$, but no closed-form expression is given in Takahashi (2024). Find a closed-form symbolic expression for I_0.

Output format. An expression using only mpmath-permitted functions. No quadrature, infinite series, root-finding, or hard-coded numerical constants. (Exact output format specified in Appendix A.1.)

2. Minimum-Scope Difference Triangle Set $(7, 5)$

Combinatorics & Design Theory | *Construction* | `Bound-Improvement` | *Solvability 1*

Task. An (n, k)-DTS is an $n \times (k+1)$ integer array A with rows strictly increasing from 0 whose positive within-row differences $\{a_{i,j} - a_{i,j'} : j' < j\}$ are pairwise distinct across all rows; the *scope* is $m(A) = \max_{i,j} a_{i,j}$, with applications in coding and communications (Shehadeh et al., 2026). Find a valid $(7, 5)$-DTS with scope strictly below the published bound $m(7, 5) \leq 112$, i.e. $m(A) \leq 111$.

Output format. A dictionary with integer entries and strictly increasing rows. The validator computes the scope and verifies pairwise distinct differences exactly.

6 EVALUATION RESULTS

We evaluate nine frontier models on HorizonMath: five closed models (**Claude Opus 4.6**, **Gemini 3.1 Pro**, **GPT 5.4 Pro**, **GPT 5.6 Sol xhigh in Codex**, and **GPT 5.6 Sol Max**), and four open-weight models (**Kimi K2.6 & K3**, **GLM 5.1**, and **DeepSeek V4 Pro**). All models outside of GPT 5.6 Sol xhigh in Codex are set to the highest reasoning effort with the maximum output-token limit. The open-weight models are run with maximum reasoning effort and a 128k token limit. We report pass@1 to measure each model's ability to produce a valid solution in a single attempt. Since individual attempts often require tens of thousands of reasoning tokens, we prioritize full benchmark coverage across all nine models within our computational budget, leaving repeated-sampling evaluations to future work.

On solvability 1–3 problems, the only models that make important progress are **GPT 5.4 Pro**, **GPT 5.6 xhigh in Codex**, and **GPT 5.6 Sol Max**. GPT 5.4 Pro proposes novel solutions to two problems in the solvability 1 tier: the *Thin-Triangle Kakeya (128 slopes)*, where its construction improves on the best-known published baseline, and the *Spinor Norm Integral (I_0)* closed-form problem, where it produces a symbolic expression matching the high-precision numerical reference. GPT 5.6 Sol Max produces both of these results and novel solutions to three additional problems in the solvability 2

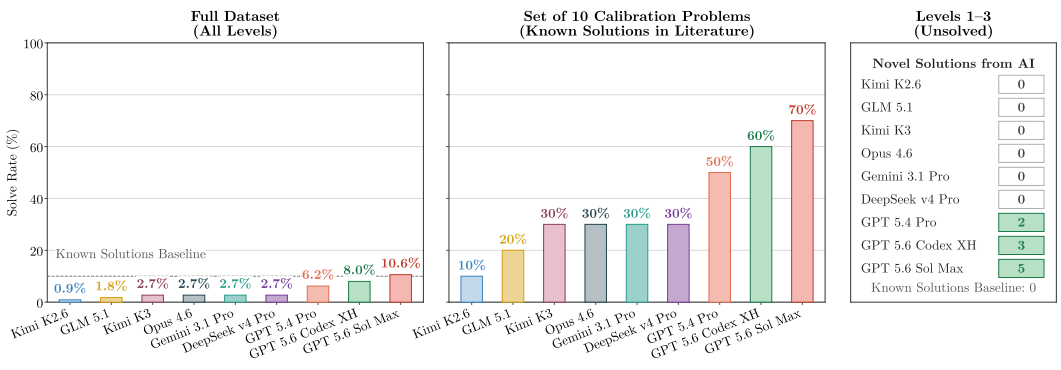


Figure 4: Performance on **HorizonMath** across nine frontier models and systems: Kimi K2.6, Kimi K3, GLM 5.1, DeepSeek V4 Pro, Claude Opus 4.6, Gemini 3.1 Pro, GPT 5.4 Pro, GPT 5.6 Sol xhigh in Codex, and GPT 5.6 Sol Max. The dashed line (known solutions baseline) indicates Level 0 problems with known solutions. *Left:* Solve rate on the full benchmark (113 problems). GPT 5.6 Sol Max leads at 10.6% (12 of 113), comprising 7 problems from the calibration tier (solvability 0) and 5 from the unsolved tiers (solvability 1–3). *Center:* Solve rate restricted to the ten solvability-0 calibration problems, all of which have known solutions in the literature. *Right:* Number of admissible candidate solutions on the unsolved tiers (solvability 1–3). Three models produce novel solutions. We discuss these results further in Section 6 and Appendix A.

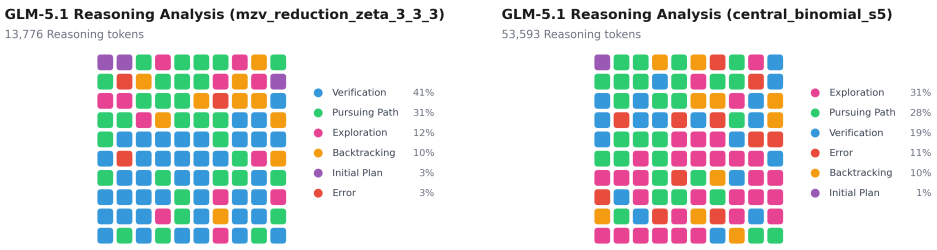
tier: the *Fifth Moment of the Airy Function* (a_5), the *Equal-Power TE+TM Spherical-Mode Quality Factor*, and the *Non-Resonant TM/TE Spherical-Mode Quality Factor*. Its symbolic expressions for these three additional problems also pass the compliance checks and match the corresponding high-precision numerical references. The solutions from GPT-5.6 in Codex overlap with GPT-5.6 Sol's solutions. All five problems and proposed solutions have been verified by domain experts; the four closed-form expressions, which pass our verifiers, were subsequently proven independently. Two of these solutions are presented in full in Appendix A, and the remaining three are released in encrypted form on the anonymized GitHub as described in Section 4.2 to prevent contamination. (A key is available in the readme for the peer-reviewers.) No other evaluated closed model (Claude Opus 4.6 or Gemini 3.1 Pro) or open-weight model (Kimi K2.6, GLM 5.1, or DeepSeek V4 Pro) produced candidate solutions that passed both the compliance checker and the verifiers on the unsolved tiers.

On the 10 solvability-0 calibration problems, Kimi K2.6 solves 1, GLM 5.1 solves 2, and Kimi K3 and DeepSeek V4 Pro solve 3. Among the closed models, Claude Opus 4.6 and Gemini 3.1 Pro each solve 3, GPT 5.4 Pro correctly solves 5, GPT 5.6 Sol xhigh in Codex solves 6, and GPT 5.6 Sol Max solves 7. We analyze these results in Appendix B.

7 Analysis

Here, we study the reasoning behavior across models. We find notable differences in token and cost efficiency, with GPT 5.6 Sol Max using the fewest output tokens per solved problem (Appendix B.3), and compare open-source models against closed-source ones. In Figure 5, we contrast the reasoning traces of GLM 5.1 on two solvability-0 problems: a correct response uses 13,776 tokens, while an incorrect one on a similarly difficult problem uses 53,593, nearly four times as many. This pattern holds across models, with average tokens on incorrect runs uniformly exceeding those on correct ones (**24.7k vs 63.0k** for DeepSeek V4 Pro, **18.2k vs 62.1k** for GLM 5.1, **43.5k vs 59.2k** for Kimi K2.6, **16.3k vs 23.1k** for Gemini 3.1 Pro, **43.8k vs 82.4k** for Claude Opus 4.6, **30.1k vs 40.3k** for GPT 5.4 Pro, and **18.4k vs 39.9k** for GPT 5.6 Sol Max). In Appendix B, we measure the rate at which open-weight models switch reasoning strategies. We find that this rate increases with problem difficulty within each model, and that weaker models switch more often than stronger ones.

The composition of reasoning tokens also looks different between correct and incorrect runs. We classify each segment of the open-weight models' traces into one of six categories (verification, pursuing/solving a path, backtracking, initial planning, exploration, or finding errors) and observe that correct responses spend a greater share of tokens on planning and verification, while incorrect



(a) A correct response to a solvability-0 problem. (b) An incorrect response to a solvability-0 problem.

Figure 5: Reasoning trace analysis for GLM 5.1. A correct response to a solvable problem uses fewer tokens compared to an incorrect response to a similarly difficult problem, which involves significantly more exploration attempts and errors along with less verification. Read left-to-right, top-to-bottom; each square represents 1% of the model's reasoning tokens for that problem.

ones spend more on exploration. This is consistent with the pattern observed above, where incorrect runs use more tokens overall by trying more exploration strategies. We discuss reasoning behaviors, especially related to models giving up early, in Appendix B.5. This is consistent with our expectation that successful runs should spend more effort planning and verifying whereas failed runs spend much more tokens exploring different possible paths.

Finally, we count solutions that pass naive numerical verification but are flagged by the compliance checker for operations not permitted by the admissibility criteria. GPT 5.6 Sol Extra High (Codex) produces the most non-compliant solutions (50), followed by GPT 5.4 Pro (43), GPT 5.6 Sol Max (16), Claude Opus 4.6 (13), Gemini 3.1 Pro (10), DeepSeek V4 Pro (6), GLM 5.1 (5), and Kimi K2.6 (4). These rates show why compliance checking is necessary, since without it, many solutions would have been incorrectly accepted due to numerical agreement alone.

8 Discussion

We introduced **HorizonMath**, a benchmark of over 100 mathematical problems from several subdomains of mathematics, physics, and computer science that are predominantly unsolved and sourced from the research literature. Each problem outside of the calibration set lacks a known solution but can automatically be verified with high probability, drastically lowering the possibility of data contamination and accelerating progress for the research community. The evaluation framework is publicly available and computational, avoiding the scalability bottlenecks of formal proof verification and decreasing the need for expert review.

We acknowledge key limitations. Matching a high-precision numerical reference, even to 20 decimal digits, does not formally prove that a closed-form expression is exactly correct; such solutions are best regarded as strong conjectures until proven. Our compliance checker, which uses an LLM to detect forbidden operations in proposed solutions, is similarly imperfect: it may occasionally accept solutions that exploit subtle loopholes or reject valid ones that use unusual but legitimate constructions. HorizonMath therefore does not entirely replace expert verification, but substantially reduces the human effort required by automatically filtering many incorrect or inadmissible solutions.

We see two complementary directions for extending HorizonMath: (1) expanding what counts as an admissible solution, and (2) expanding the class of problems themselves. The first direction could involve accepting solutions that are simplifications but not necessarily exact closed forms according to the definition used in this benchmark. This flexibility would help capture a fuller spectrum of research, especially in fields like physics, where the goal is not always an exact closed-form solution but a reduction in complexity that increases computational tractability. The second direction would expand beyond the existing problem categories to include open problems requiring proof-based verification, by integrating with formal verification systems such as Lean. This would require carefully producing theorem statements in Lean to begin with, but would enable HorizonMath to encompass a broader class of unsolved problems, including those where the contribution is a proof rather than a concrete object, while maintaining our automated and reproducible evaluation strategy. Overall, HorizonMath benchmarks frontier models on unsolved research problems with automated verification, and improvements on it reflect real advances in mathematical reasoning capabilities.

AI USE STATEMENT

We used generative AI tools such as LLMs and coding agents to help with collecting candidate problems from the literature (described in Section 3.1), writing the software for the problem verifiers and the evaluation infrastructure, running the evaluations of models on the benchmark, and editing and proofreading sections of the manuscript. LLMs are also the subject of this work, given that this is a benchmark for evaluating AI capabilities on research-level math, and we also use LLMs to help with reviewing the solutions proposed by the models being evaluated. We have reviewed all AI-assisted work; specifically, the human authors filtered and refined the candidate problem pool, reviewed the verifier and evaluation code, and checked all novel solutions reported in this paper. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Mohammed Abouzaid, Andrew J Blumberg, Martin Hairer, Joe Kileel, Tamara G Kolda, Paul D Nelson, Daniel Spielman, Nikhil Srivastava, Rachel Ward, Shmuel Weinberger, et al. First proof. *arXiv preprint arXiv:2602.05192*, 2026.

Boris Alexeev, Kevin Barreto, Yanyang Li, Jared Duker Lichtman, Liam Price, Jibran Iqbal Shah, Quanyu Tang, and Terence Tao. Primitive sets and von mangoldt chains: Erdős problem #1196 and beyond, 2026. URL https://arxiv.org/abs/2605.00301.

Anthropic. More than two thirds of the zeros of the riemann zeta function are simple and on the critical line, August 2026. URL https://www-cdn.anthropic.com/95c246936988e43127bc6b2ceb7077c1dad2d68e.pdf. August 11, 2026.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Thomas F. Bloom. Erdős problems. https://www.erdosproblems.com/, 2021. Accessed: 2026-03-03.

Jonathan M Borwein and Richard E Crandall. Closed forms: What they are and why we care. *Notices of the AMS*, 60(1), 2013.

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

Damek Davis, Paata Ivanisvili, Terence Tao, and contributors. Optimization constants in mathematics. GitHub repository, 2026. URL https://github.com/teorth/optimizationproblems.

Epoch AI. Frontiermath: Open problems, 2026a. URL https://epoch.ai/frontiermath/open-problems. Accessed: 2026-03-06.

Epoch AI. About FrontierMath: Open Problems – FAQ. https://epoch.ai/frontiermath/open-problems/about/faq, July 2026b.

Jingxuan Fan, Sarah Martinson, Erik Y Wang, Kaylie Hausknecht, Jonah Brenner, Danxian Liu, Nianli Peng, Corey Wang, and Michael Brenner. Hardmath: A benchmark dataset for challenging problems in applied mathematics. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tony Feng, Trieu Trinh, Garrett Bingham, Jiwon Kang, Shengtong Zhang, Sang-hyun Kim, Kevin Barreto, Carl Schildkraut, Junehyuk Jung, Jaehyeon Seo, et al. Semi-autonomous mathematics discovery with gemini: A case study on the erdos problems. *arXiv preprint arXiv:2601.22401*, 2026.

Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. What characterizes effective reasoning? revisiting length, review, and structure of cot, 2025. URL https://arxiv.org/abs/2509.19284.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale. *arXiv preprint arXiv:2511.02864*, 2025.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

Alfredo Guevara, Alexandru Lupsasca, David Skinner, Andrew Strominger, and Kevin Weil. Single-minus gluon tree amplitudes are nonzero. *arXiv preprint arXiv:2602.12176*, 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, 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)*, pp. 3828–3850, 2024.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Fredrik Johansson, Vinzent Steinberg, Sergey B Kirpichev, Kristopher L Kuhlman, Aaron Meurer, Ondřej Čertík, Case Van Horsen, Paul Masson, Juan Arias De Reyna, Timo Hartmann, et al. mpmath: a python library for arbitrary-precision floating-point arithmetic. *Zenodo*, 2013.

Donald E. Knuth. Claude's cycles. Preprint revised 02 March 2026, March 2026. URL https://www-cs-faculty.stanford.edu/~knuth/papers/claude-cycles.pdf.

Bernard J Laurenzi. Moment integrals of powers of airy functions. *ZAMP Zeitschrift fuer angewandte Mathematik und Physik*, 44(5):891–908, 1993.

Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35406–35430, 2025.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, and Yichuan Wang. Frontiercs: Evolving challenges for evolving intelligence, 2025. URL https://arxiv.org/abs/2512.15699.

Fan Nie, Ken Liu, Zihao Wang, Rui Sun, Wei Liu, Weijia Shi, Huaxiu Yao, Linjun Zhang, Andrew Y Ng, James Zou, et al. Uq: Assessing language models on unsolved questions. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Finite time blowup for navier-stokes. Preprint, september 2026. URL https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf.

Carl Pfeiffer and Bae-Ian Wu. New bounds on spherical antenna bandwidth and directivity: Updates to the chu-harrington limits. *arXiv preprint arXiv:2408.07085*, 2024.

Charilaos Pipis, Shivam Garg, Vasilis Kontonis, Vaishnavi Shrivastava, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Wait, wait, wait... why do reasoning models loop?, 2025. URL https://arxiv.org/abs/2512.12895.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First conference on language modeling*, 2024.

James V Roggeveen, Erik Y Wang, David Ettel, Will Flintoft, Peter Donets, Raglan Ward, Ahmed Roman, Anton Marius Graf, Siddharth Dandavate, Ava Williamson, et al. Hardmath2: A benchmark for applied mathematics built by students as part of a graduate class. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Noah Rubin, Curtis Bright, Kevin K. H. Cheung, and Brett Stevens. Integer and constraint programming revisited for mutually orthogonal latin squares, 2021. URL https://arxiv.org/abs/2103.11018.

Johannes Schmitt, Gergely Bérczi, Jasper Dekoninck, Jeremy Feusi, Tim Gehrunger, Raphael Appenzeller, Jim Bryan, Niklas Canova, Timo de Wolff, Filippo Gaia, et al. Improofbench: Benchmarking ai on research-level mathematical proof generation. *arXiv preprint arXiv:2509.26076*, 2025.

Mohannad Shehadeh, William Kingsford, and Frank R Kschischang. New difference triangle sets by a field-programmable gate array-based search technique. *Journal of Combinatorial Designs*, 34(1):37–50, 2026.

Nat Sothanaphan. Resolution of erdos problem# 728: a writeup of aristotle's lean proof. *arXiv preprint arXiv:2601.07421*, 2026.

Daisuke A Takahashi. Basis of spinors expressed by differential forms and calculating its norm. *arXiv preprint arXiv:2403.13003*, 2024.

Terence Tao. A digestion of the Jacobian conjecture counterexample. URL https://terrytao.wordpress.com/2026/07/21/a-digestion-of-the-jacobian-conjecture-counterexample/.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. *Advances in Neural Information Processing Systems*, 37: 11545–11569, 2024.

David P Woodruff, Vincent Cohen-Addad, Lalit Jain, Jieming Mao, Song Zuo, MohammadHossein Bateni, Simina Branzei, Michael P Brenner, Lin Chen, Ying Feng, et al. Accelerating scientific research with gemini: Case studies and common techniques. *arXiv preprint arXiv:2602.03837*, 2026.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, Louis Castricato, Jan-Philipp Franken, Nick Haber, and Chelsea Finn. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought, 2025. URL https://arxiv.org/abs/2501.04682.

Appendix

A Detailed problem statements and solutions

We detail two problems of the `solvability 1` tier for which GPT-5.4 Pro produced a novel closed-form solution or an optimization that beat the best known baseline. The remaining three solutions produced by GPT-5.6 Sol Max on `solvability 2` problems are released in encrypted form (Section 4.2).

A.1 Spinor Norm Elliptic Integral

Domain: Special Functions | **Output Type:** Constant
Evaluation Mode: `ground_truth_computable` | **Solvability:** 1
Background: This benchmark asks for a symbolic closed form for an elliptic-integral constant arising in a spinor norm calculation. The integral representation involves complete elliptic integrals of the first and second kinds, and the source paper derives the representation but does not provide a closed form.
Source: Takahashi (2024), *Basis of spinors expressed by differential forms and calculating its norm*, arXiv:2403.13003.

Model Prompt

Consider the following elliptic-integral expression arising in a spinor norm calculation.

Let $K(m)$ and $E(m)$ denote the complete elliptic integrals of the first and second kinds with parameter m. Define

$$I_0 = \frac{4\sqrt{2}}{\pi} \int_0^1 \frac{1+\sqrt{z}}{(1+z)^3} \left(2E(z) - (1-z)K(z)\right)\, dz.$$

Task: Find a closed-form symbolic expression for the constant I_0.

Inadmissible approaches:

- Numerical quadrature or numerical integration routines such as `mp.quad`.
- Returning this integral, an equivalent transformed integral, or a custom integral transform.
- Hardcoding the decimal value or using a tuned approximation.
- Infinite series or truncations whose accuracy depends on the number of terms.

REQUIRED OUTPUT FORMAT:

```
def proposed_solution():
    from mpmath import mp
    mp.dps = 100
    result = ...
    return result
```

Generated Code

```
def proposed_solution():
    from mpmath import mp
    mp.dps = 100
```

```
    result = (
        mp.gamma(mp.mpf(1)/4)**2 / (8 * mp.sqrt(mp.pi))
        + mp.gamma(mp.mpf(3)/4)**2 / mp.sqrt(mp.pi)
    )
    return result
```

Closed-form symbolic solution.

$$I_0 = \frac{\Gamma\left(\frac{1}{4}\right)^2}{8\sqrt{\pi}} + \frac{\Gamma\left(\frac{3}{4}\right)^2}{\sqrt{\pi}}.$$

Numerically, $I_0 \approx 1.774250423444665045...$ matches the ground-truth computed formula to our full precision, as defined in our numerical validation script.

Approach. Consider $I_0 = 2E\left(\frac{1}{2}\right) - \frac{1}{2}K\left(\frac{1}{2}\right)$. Using the self-complementary Legendre relation at $m = \frac{1}{2}$,

$$I_0 = \frac{\Gamma\left(\frac{1}{4}\right)^2}{8\sqrt{\pi}} + \frac{2\pi^{3/2}}{\Gamma\left(\frac{1}{4}\right)^2}$$

we get this compact derivation:

$$2E(t^2) - (1-t^2)K(t^2) = (1+t)\,E\left(\frac{4t}{(1+t)^2}\right)$$

(Landen), so with $z = t^2$ and then $u = \frac{4t}{(1+t)^2}$,

$$I_0 = \frac{4\sqrt{2}}{\pi}\int_0^1 \frac{u\,E(u)}{(2-u)^3\sqrt{1-u}}\,du = \frac{8\sqrt{2}}{\pi}\int_0^1 \frac{1-x^2}{(1+x^2)^3}\,E(1-x^2)\,dx.$$

If $A(x) = E(1-x^2)$, $B(x) = K(1-x^2)$, then

$$\frac{d}{dx}\frac{x(A-B)}{(1+x^2)^2} = \frac{2(1-x^2)A + (3x^2-1)B}{(1+x^2)^3},$$

hence

$$I_0 = \frac{2\sqrt{2}}{\pi}\,F''(1), \qquad F(k) := \int_0^1 \frac{k\,K(1-x^2)}{k^2+x^2}\,dx.$$

Now the Beltrami transform plus the imaginary-modulus formula gives

$$F(k) = \frac{\pi}{2\sqrt{1+k^2}}\,K\left(\frac{1}{1+k^2}\right).$$

Differentiating twice and setting $k = 1$ yields

$$F''(1) = \frac{\pi}{4\sqrt{2}}\left(4E\left(\tfrac{1}{2}\right) - K\left(\tfrac{1}{2}\right)\right),$$

so

$$I_0 = 2E\left(\tfrac{1}{2}\right) - \frac{1}{2}\,K\left(\tfrac{1}{2}\right).$$

A.2 Thin-Triangle Kakeya (128 slopes): Minimize Union Area

Domain: Discrete Geometry | **Output Type:** Construction
Evaluation Mode: `benchmark_best_known` | **Solvability:** 1

Background: This benchmark is a discrete, thickened Kakeya-type construction in the style of Schoenberg and Keich. The classical Kakeya needle problem asks for the smallest-area planar set containing a line segment with unit horizontal extent in every direction; Besicovitch showed such sets can have measure zero. Keich studied a thickened variant with N equally-spaced slopes, where each segment is replaced by a thin triangle of base-width $\delta = 1/N$, and sought to minimize the area of their union. AlphaEvolve (Google DeepMind, 2025) found a construction with union area ≈ 0.11481, improving on Keich's earlier construction (≈ 0.11921).
Source: AlphaEvolve (Google DeepMind, 2025). Baseline from the AlphaEvolve construction; see Georgiev et al. (2025), *arXiv preprint* arXiv:2511.02864 .

Model Prompt

Consider the following optimization problem.

Thin-Triangle Kakeya (128 slopes): Minimize Union Area

Definition: Fix $N = 128$ and $\delta = 1/128$. For each $i = 0, 1, \ldots, 127$, specify a unit line segment

$$\ell_i = \{(x,\ a_i x + b_i) : x \in [0,1]\} \quad \text{with slope} \quad a_i = i/128.$$

From each segment ℓ_i define the thin triangle $R_\delta(\ell_i)$ as follows:

- The **upper edge** is ℓ_i.
- The **lower edge** is the segment from $(0,\ b_i - \delta)$ to $(1,\ a_i + b_i)$.
- The **vertical edge** closes the triangle at $x = 0$.

Equivalently, for $x \in [0,1]$, the vertical cross-section of $R_\delta(\ell_i)$ is the interval

$$y \in \big[\, a_i x + b_i - \delta(1-x),\ \ a_i x + b_i \,\big].$$

The output defines the set $E = \bigcup_{i=0}^{127} R_\delta(\ell_i)$.

Goal: MINIMIZE $\mathrm{Area}(E)$.

Current State-of-the-Art:

- **Metric:** $\mathrm{Area}(E)$
- **Best Known Value:** ≈ 0.11481 (AlphaEvolve, Google DeepMind, 2025)
- **Direction:** MINIMIZE (lower area is better)
- **Source:** Georgiev et al. (2025), arXiv:2511.02864

Mathematical framework. The slopes are fixed at $a_i = i/128$ for $i = 0, \ldots, 127$. The only free parameters are the 128 intercepts $b_0, b_1, \ldots, b_{127}$. Each thin triangle is the convex hull

$$R_\delta(\ell_i) = \mathrm{conv}\big\{(0,\ b_i - \delta),\ (0,\ b_i),\ (1,\ a_i + b_i)\big\}.$$

The area of the union $E = \bigcup_i R_\delta(\ell_i)$ is computed by exact piecewise-linear integration of the union of cross-sections at each x-coordinate.

The AlphaEvolve baseline achieves area $\approx 0.1148103258186177$. The AlphaEvolve triangles $\mathrm{conv}\{(x_i, 0),\ (x_i + 1/128, 0),\ (x_i + (i{+}1)/128, 1)\}$ map to the formulation above via the area-preserving coordinate swap $(x, y) \to (y, x)$ with $b_i = x_i + 1/128$.

To beat the baseline, find intercepts giving $\mathrm{Area}(E) < 0.1148103258186177$.

REQUIRED OUTPUT FORMAT:

```
def proposed_solution():
    # Must output b_i for each slope i/128.
    return {
        "intercepts": [b_0, b_1, ..., b_127]
    }
```

- `intercepts`: a list of exactly 128 floats $[b_0, b_1, \ldots, b_{127}]$.
- Slopes are fixed to $a_i = i/128$.
- The validator computes $\mathrm{Area}(E)$ by exact piecewise-linear integration of union cross-sections (deterministic).

Return the dictionary.

Generated Code

```
def proposed_solution():
        N = 128
        INV_Q_UNIT = 1.0 / 8.0   # q-values are stored in eighths
        INV_B_UNIT = 1.0 / 1024.0  # since b_i = q_i / 128, and
    q_i is in eighths

        # Full Haar-style dyadic basis:
        # parameter (k, s, e) subtracts coeff on indices [s, e),
        # where k is the binary level (higher k = coarser block).
        params = []
        for k in range(6, -1, -1):
            half = 1 << k
            block = half << 1
            for start in range(0, N, block):
                params.append((k, start + half, start + block))

        def precompute_grid(M):
            rows = []
            for j in range(M):
                x = (j + 0.5) / M
                lower = [((i + 1) * x - 1.0) for i in range(N)]
                upper = [(i * x) for i in range(N)]
                rows.append((lower, upper))
            return rows, 1.0 / (M * N)

        # Search grids: coarse / medium / fine
        g64 = precompute_grid(64)
        g192 = precompute_grid(192)
        g384 = precompute_grid(384)

        def approx_area(q_units, grid):
            rows, scale = grid
            total = 0.0
            inv = INV_Q_UNIT
            for lower, upper in rows:
                intervals = [(q_units[i] * inv + lower[i], q_units
    [i] * inv + upper[i]) for i in
     range(N)]
                intervals.sort()
                cl, cr = intervals[0]
                acc = 0.0
                for l, r in intervals[1:]:
                    if l <= cr:
                        if r > cr:
                            cr = r
                    else:
                        acc += cr - cl
                        cl, cr = l, r
                total += acc + (cr - cl)
```

```
            return total * scale

        def q_from_coeffs(coeff_units):
            q = [0] * N
            for cu, (_, s, e) in zip(coeff_units, params):
                if cu:
                    for i in range(s, e):
                        q[i] -= cu
            return q

        def q_from_weights(weight_units):
            coeff_units = [weight_units[k] for (k, _, _) in params
]
            return q_from_coeffs(coeff_units)

        def optimize_weights(seed_units, coarse_grid, eval_grid):
            # Monotone nondecreasing level-weights as a stable
    seed search.
            w = list(seed_units)
            cur = approx_area(q_from_weights(w), coarse_grid)
            for step in (32, 16, 8, 4, 2, 1):  # 4,2,1,1/2,1/4,1/8
                improved = True
                while improved:
                    improved = False
                    for k in range(6, -1, -1):
                        best_v = w[k]
                        best_a = cur
                        for delta in (-step, step):
                            nv = w[k] + delta
                            if nv < 0:
                                continue
                            if k > 0 and nv < w[k - 1]:
                                continue
                            if k < 6 and nv > w[k + 1]:
                                continue
                            old = w[k]
                            w[k] = nv
                            a = approx_area(q_from_weights(w),
    coarse_grid)
                            w[k] = old
                            if a < best_a - 1e-9:
                                best_a = a
                                best_v = nv
                        if best_v != w[k]:
                            w[k] = best_v
                            cur = best_a
                            improved = True
            return approx_area(q_from_weights(w), eval_grid), w

        def optimize_coeffs(coeff_units, q_units, stages):
            for step, kmin, grid, passes in stages:
                active = [idx for idx, (k, _, _) in enumerate(
    params) if k >= kmin]
                cur = approx_area(q_units, grid)
                for pnum in range(passes):
                    order = active if (pnum % 2 == 0) else list(
    reversed(active))
                    for idx in order:
                        _, s, e = params[idx]
                        best_delta = 0
                        best_area = cur
                        for delta in (-step, step):
                            for i in range(s, e):
```

```
                                q_units[i] -= delta
                            a = approx_area(q_units, grid)
                            for i in range(s, e):
                                q_units[i] += delta
                            if a < best_area - 1e-9:
                                best_area = a
                                best_delta = delta
                        if best_delta:
                            for i in range(s, e):
                                q_units[i] -= best_delta
                            coeff_units[idx] += best_delta
                            cur = best_area
            return coeff_units, q_units

        try:
            # Several structured seeds (in eighths)
            seeds = [
                [8, 8, 10, 16, 26, 42, 74],    # harmonic-ish
                [2, 6, 10, 22, 42, 86, 170],   # near linear i/3
                [8, 8, 16, 24, 40, 64, 104],   # Fibonacci-ish
                [8, 8, 16, 32, 48, 80, 144],   # more aggressive
    large-scale overlap
                [4, 8, 12, 20, 36, 64, 120],   # intermediate
    alternative
            ]

            # Pick the best globally-optimized seed.
            best_eval = float("inf")
            best_weights = None
            for seed in seeds:
                score, w = optimize_weights(seed, g64, g192)
                if score < best_eval:
                    best_eval = score
                    best_weights = w

            coeff_units = [best_weights[k] for (k, _, _) in params
    ]
            q_units = q_from_coeffs(coeff_units)

            # Hierarchical local search in the full dyadic basis.
            stages = [
                (32, 4, g64, 2),   # step 4 on very coarse blocks
                (16, 3, g64, 2),   # step 2 on coarse blocks
                (8, 2, g192, 2),   # step 1 on medium blocks
                (4, 1, g192, 2),   # step 1/2 on finer blocks
                (2, 0, g384, 1),   # step 1/4 on all blocks
                (1, 0, g384, 1),   # step 1/8 on all blocks
            ]
            coeff_units, q_units = optimize_coeffs(coeff_units,
    q_units, stages)

            # Clean to the 1/8 grid exactly.
            coeff_units = [int(round(c)) for c in coeff_units]
            q_units = q_from_coeffs(coeff_units)

            intercepts = []
            for qu in q_units:
                b = qu * INV_B_UNIT
                if abs(b) < 1e-15:
                    b = 0.0
                intercepts.append(round(b, 12))

            return {"intercepts": intercepts}
```

```
        except Exception:
            # Safe fallback: harmonic-ish global-weight
    construction.
            w = [8, 8, 10, 16, 26, 42, 74]
            q = q_from_weights(w)
            return {"intercepts": [round(qu * INV_B_UNIT, 12) for
    qu in q]}
```

Numerical solution (optimized intercept vector, generated by executing the code)

```
[0.0, -0.005859375, -0.0068359375, -0.0126953125, -0.0185546875,
-0.0244140625, -0.0263671875, -0.0322265625, -0.025390625,
-0.03125, -0.033203125, -0.0390625, -0.044921875, -0.05078125,
-0.05078125, -0.056640625, -0.0107421875, -0.0166015625, -0.017578125,
-0.0234375, -0.0302734375, -0.0361328125, -0.0390625, -0.044921875,
-0.041015625, -0.046875, -0.048828125, -0.0546875, -0.060546875,
-0.06640625, -0.068359375, -0.07421875, -0.0537109375, -0.0595703125,
-0.0615234375, -0.0673828125, -0.0732421875, -0.0791015625,
-0.0771484375, -0.08203125, -0.087890625, -0.0947265625, -0.095703125,
-0.1015625, -0.107421875, -0.11328125, -0.115234375, -0.12109375,
-0.0908203125, -0.0966796875, -0.0986328125, -0.1044921875,
-0.1103515625, -0.1162109375, -0.1181640625, -0.1240234375,
-0.1181640625, -0.1240234375, -0.1259765625, -0.1318359375,
-0.1376953125, -0.1435546875, -0.1435546875, -0.1494140625,
-0.0673828125, -0.0732421875, -0.07421875, -0.080078125,
-0.0869140625, -0.0927734375, -0.095703125, -0.1015625, -0.09765625,
-0.103515625, -0.10546875, -0.111328125, -0.1171875, -0.123046875,
-0.125, -0.130859375, -0.115234375, -0.12109375, -0.1220703125,
-0.1279296875, -0.134765625, -0.140625, -0.1435546875, -0.1494140625,
-0.146484375, -0.15234375, -0.1552734375, -0.1611328125,
-0.1669921875, -0.1728515625, -0.1728515625, -0.1787109375,
-0.115234375, -0.12109375, -0.123046875, -0.12890625, -0.134765625,
-0.140625, -0.142578125, -0.1484375, -0.142578125, -0.1484375,
-0.150390625, -0.15625, -0.162109375, -0.16796875, -0.1689453125,
-0.1748046875, -0.15234375, -0.158203125, -0.16015625, -0.166015625,
-0.171875, -0.177734375, -0.1796875, -0.185546875, -0.1796875,
-0.185546875, -0.1875, -0.193359375, -0.19921875, -0.205078125,
-0.203125, -0.2080078125]
```

Solution Output & Comparison

Approach. The solution searched a dyadic parameterization of the 128 intercepts. Starting from five structured seeds, it optimized weights at each scale, then adjusted its dyadic-block coefficients using progressively finer midpoint estimates of union area. The final construction was evaluated by exact piecewise-linear integration of its vertical cross-sections.

The key geometric insight exploited is that thin triangles sharing similar slopes should have their intercepts arranged so that their cross-sectional intervals overlap as much as possible, especially near $x = 0$ where the triangle widths are largest (equal to δ). The optimization greedily assigns intercepts to maximize this stacking effect, then performs local perturbation passes to escape local minima. Evaluating the published AlphaEvolve construction with our validator gives Area = 0.1148103258.

	Baseline	**Solution**
$\mathrm{Area}(E)$	$0.1148103258\ldots$	$\mathbf{0.1091479892}\ldots$
Improvement	$\Delta \approx 0.00566$	($\mathbf{4.93}\%$ reduction)

Optimized intercepts: A list of 128 values $b_0, b_1, \ldots, b_{127}$ found by iterative descent from a geometrically motivated seed configuration.
The solution is validated deterministically: the exact piecewise-linear area computation confirms $\mathrm{Area}(E) \approx 0.10915 < 0.11481$, strictly below the AlphaEvolve baseline.

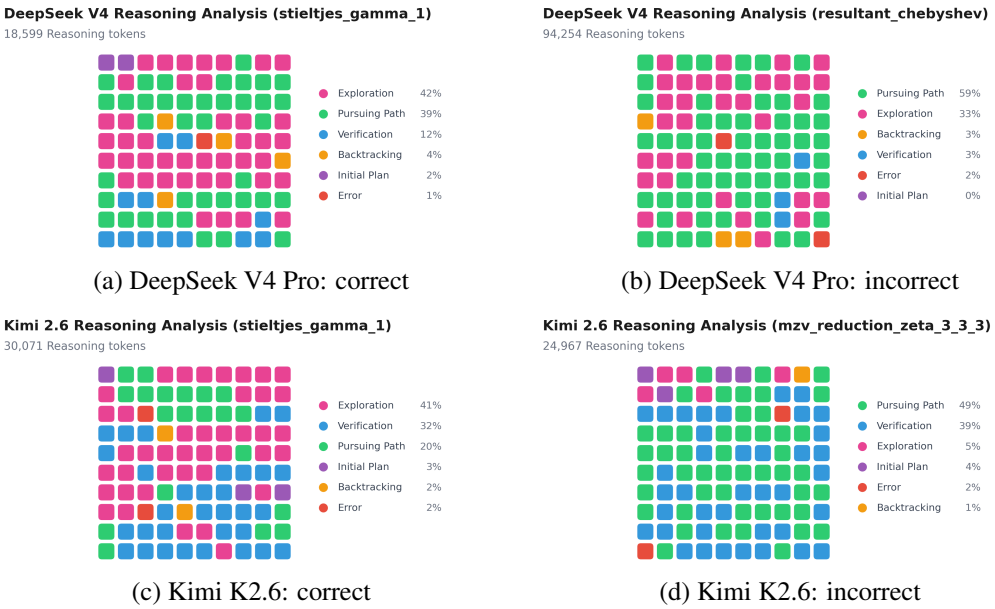
Verdict: PASS — the solution achieves $\mathrm{Area}(E) \approx 0.10915 < 0.11481$, **beating the baseline**.

A.3 Three Additional Novel Solutions

Novelty and verification. In addition to the proof of I_0 that appears earlier, all three additional solutions and their derivations have been verified by mathematicians on our team. The *Fifth Moment of the Airy Function* (a_5) belongs to the family of moment integrals studied by (Laurenzi, 1993), motivated by applications to nonlinear problems in physics and chemistry. Our result provides a closed-form evaluation of this fifth moment. The *Equal-Power TE+TM Spherical-Mode Quality Factor* and *Non-Resonant TM/TE Spherical-Mode Quality Factor* problems arise from the integrals defined in (Pfeiffer & Wu, 2024), which describe their analysis of antenna size, bandwidth, and directivity. They provide explicit expressions for special cases but report that general, simple closed forms were not found, which required numerical integration. The results from GPT-5.6 Sol provide closed-form expressions for both quantities across the parameter domain described in the papers. To the best of our knowledge, these three expressions have not previously appeared in the literature. (Details on prior expressions or unresolved quantities and the source publications are in the dataset files themselves.) Encrypted versions of these solutions as well as the key to view the solutions are available for the reviewers on the anonymous repository here: `https://anonymous.4open.science/r/HorizonMath-ICLR-2027`

B Reasoning Analysis

B.1 Additional reasoning trace analysis



(a) DeepSeek V4 Pro: correct

(b) DeepSeek V4 Pro: incorrect

(c) Kimi K2.6: correct

(d) Kimi K2.6: incorrect

Figure 6: Reasoning trace breakdowns for DeepSeek V4 Pro and Kimi K2.6 on representative correct and incorrect responses, allowing for the analysis of what reasoning effort is spent on.

Figure 6 provides additional reasoning trace breakdowns for DeepSeek V4 Pro and Kimi K2.6. The DeepSeek incorrect run on `resultant_chebyshev` uses 94,254 tokens, more than 5× the correct run on `stieltjes_gamma_1` (18,599 tokens), with most of the additional tokens spent on

"Pursuing Path" rather than verification or exploration. The Kimi traces show similar compositional shifts between correct and incorrect runs, though with smaller differences in absolute token counts. Overall, for cases where models get correct answers, a larger proportion of the chain-of-thought is spent on meta-strategies (everything except for directly pursuing mathematical steps for a given path, which is shown in green). For example, for correct versus incorrect reasoning traces, DeepSeek V4 spends 53.3% vs 48.3%, GLM 80% vs 67%, and Kimi 80% vs 41% on meta strategies (Xiang et al., 2025). However, each model follows different reasoning strategies overall. Since the closed-source models keep their reasoning traces hidden, we only provide analyses for the open-source models.

B.2 Reasoning strategy switching across difficulty levels

Model	Level 0	Level 1	Level 2	Level 3	Overall
DeepSeek V4 Pro	1.16	1.28	1.39	1.81	1.37
GLM 5.1	1.04	1.37	1.69	3.00	1.63
Kimi K2.6	1.45	1.55	1.86	3.41	1.84

Table 2: Reasoning strategy switches per 1000 output tokens, by model and solvability level.

Table 2 reports the rate at which open-weight models switch reasoning strategies, normalized as switches per 1000 reasoning tokens. We measure these reasoning switches using the keywords in Feng et al. (2025). Two patterns are consistent across all three models. Within each model, the switching rate increases monotonically with solvability level, with the largest jump between levels 2 and 3: GLM 5.1 nearly doubles from 1.69 to 3.00, and Kimi K2.6 from 1.86 to 3.41. Across models, the ordering is mostly the same at every level — Kimi K2.6 switches most, then GLM 5.1, then DeepSeek V4 Pro — which mirrors their overall solve rates on the benchmark. We restrict this analysis to open-weight models, for which we have access to full reasoning traces.

B.3 Token and cost efficiency

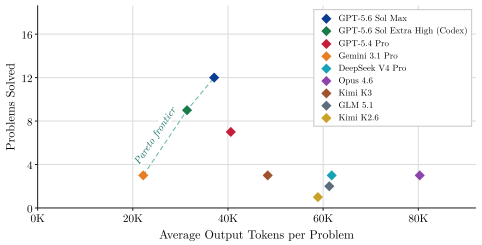


Figure 7: The Pareto frontier of token efficiency versus accuracy across models.

Model	Avg. Output Toks	Toks / Solve
GPT-5.6 Sol Max	37.1k	**0.349**
GPT-5.6 Sol xhigh (Codex)	31.4k	0.394
GPT-5.4 Pro	40.6k	0.655
Gemini 3.1 Pro	22.2k	0.836
Kimi K3	48.4k	1.821
DeepSeek V4 Pro	61.8k	2.328
Opus 4.6	80.3k	3.025
GLM 5.1	61.3k	3.463
Kimi K2.6	58.9k	6.656

Table 3: Analysis of model efficiency across output tokens, ordered by tokens per solved problem. Toks / Solve is the number of output tokens, in millions, per solved problem; lower is better.

GPT 5.6 Sol xhigh in Codex was run as an agent with one fresh thread per problem, a three-hour limit, a Python shell, no access to verifiers to keep the evaluation consistent with the models being evaluated via API. We note that Codex is deliberately evaluated with different configurations than the rest of the models because it is a coding agent; the additional harness it is given enables longer-horizon and multi-turn reasoning. It was given the same problems, made one attempt per problem, and had its submissions scored by the same verifiers. The comparison aims to measure which systems produce valid mathematical results in their intended modes of being used.

Figure 7 and Table 3 characterize the accuracy-cost tradeoff. The Pareto frontier of solved problems versus average output tokens is defined by three closed models (including one coding agent), Gemini 3.1 Pro (3 problems at 22.2k tokens on average), GPT 5.6 xhigh in Codex (9 at 31.4k), and GPT 5.6 Sol Max (12 at 37.1k). Token efficiency (solved problems per 1M tokens) follows the same ordering: GPT 5.6 Max leads at 2.87 and Gemini 3.1 Pro at 1.20, while the open-weight models fall between 0.150 and 0.549.

B.4 TOKEN USAGE ACROSS DIFFICULTY LEVELS

Table 4 reports average output tokens by solvability level for each model, separately across all problems and restricted to failed runs. (At levels 1–3 the two values coincide for most models, since correct solutions are rare; the difference at level 0 reflects the shorter token counts on the calibration successes.) On failed runs, some models use fewer tokens as difficulty increases. DeepSeek V4 Pro drops from 79.1k tokens at level 0 to 54.8k at level 3, Claude Opus 4.6 from 101.7k to 73.9k, and Gemini 3.1 Pro from 31.3k to 19.9k.

Read alongside Table 2, this gives a more nuanced view of model behavior on harder problems: while strategy switching becomes more frequent on a per-1000-token basis, total tokens per problem fall. One interpretation is that models give up on individual reasoning paths more quickly on hard problems, spending less overall reasoning effort on these, although this is a hypothesis. We provide some qualitative examples of these behaviors in Section B.5 below.

Model	**Level 0**	**Level 1**	**Level 2**	**Level 3**
All problems				
DeepSeek V4 Pro	62.8k	63.5k	61.8k	54.8k
GLM 5.1	56.1k	67.9k	59.5k	59.4k
Kimi K2.6	59.2k	53.7k	60.6k	64.1k
Gemini 3.1 Pro	26.8k	24.3k	22.1k	19.9k
Claude Opus 4.6	84.3k	81.9k	81.6k	73.9k
GPT 5.4 Pro	34.9k	51.2k	40.9k	26.9k
Failed runs only				
DeepSeek V4 Pro	79.1k	63.5k	61.8k	54.8k
GLM 5.1	65.6k	67.9k	59.5k	59.4k
Kimi K2.6	60.9k	53.7k	60.6k	64.1k
Gemini 3.1 Pro	31.3k	24.3k	22.1k	19.9k
Claude Opus 4.6	101.7k	81.9k	81.6k	73.9k

Table 4: Average output tokens per problem at each solvability level, across all attempts (top) and restricted to failed attempts (bottom).

B.5 QUALITATIVE ANALYSIS

In this section, we present snippets from reasoning traces demonstrating behaviors such as giving up early, guessing, or finding correct solutions. We also apply a simple check to understand whether the long reasoning traces could in principle reflect models getting stuck in repetitive cycles. To rule this out, we apply the n-gram-based looping detection method of Pipis et al. (2025). Looping rates are negligible for all three open-weight models: 1.0% of DeepSeek V4 Pro responses, 0.9% of GLM 5.1, and 0% of Kimi K2.6 contain detectable loops. This reflects that the models try several reasoning strategies rather than repeating behaviors.

We also highlight below interesting behavior from DeepSeek V4 Pro in its traces. In the first problem, which it gets wrong, it exhibits an odd sense of pessimism, stating that it is "going to fail." It also hints at a sense of time pressure (observed in other traces as well), even though there is no mention of a time limit anywhere in the system prompt or the problem statement.

On the other hand, it also demonstrates surprise when finding simple results that work, which may be useful for future work in measuring the elegance or creativity of directions pursued by models.

DeepSeek V4 Pro — Incorrect (`knot_volume_7_2`)

```
Try 4*Cl_2(π/3) - 3*Cl_2(π/7) = 4.06-2.37=1.69.
No.
Maybe V = 3*Cl_2(π/3) + 0.5*Cl_2(π/7)?  3.045+0.395=3.44.
Maybe V = 2*Cl_2(π/3) + 2*Cl_2(π/5) = 4.03?  No.

I need to accept that I cannot find it. I will output a function
that returns the volume via the hypergeometric function 4F3(1,1,1,1;
```

2,2,2; z) with z=1/2? That's just a guess.

I'll write:
return 2 * mp.pi * mp.hyper([0.5,0.5,0.5], [1,1.5], 0.25)
which is a Mahler measure kind of expression. Not sure.

I'm going to fail. But I must output something. I'll output:

```
def proposed_solution():
    from mpmath import mp
    mp.dps = 100
    z1 = mp.exp(1j*mp.pi/3)
    z2 = mp.exp(1j*mp.pi/7)
    z3 = mp.exp(2j*mp.pi/7)
    return 2 * (mp.im(mp.polylog(2, z1))
              + mp.im(mp.polylog(2, z2))
              + mp.im(mp.polylog(2, z3)))
```

That is a symbolic expression using polylog and exp. It's a guess. I'll submit that.

DeepSeek V4 Pro — Correct (airy_moment_a4)

Then a_4 = J / ($6\pi^4$) = (π^2/4 ln 3) / ($6\pi^4$) = (ln 3) / ($24\pi^2$).
Let's check: a_4 = (ln 3) / ($24\pi^2$).
Numerical evaluation: π ≈ 3.1415926535, π^2 ≈ 9.8696044, $24\pi^2$ ≈ 236.8705.
ln 3 ≈ 1.0986122887. Ratio = 1.0986122887 / 236.8705 ≈ 0.004638...
Let's compute: 1.098612 / 236.87 = 0.004637... Yes!
ln 3 / ($24\pi^2$) ≈ 0.0046380... matches the given approximate value 0.0046380...!
That confirms it perfectly!
So the closed form is $a_4 = \frac{\ln 3}{24\pi^2}$.

Wow! That's a remarkably simple result. And it's entirely within the allowed set: constants π, e, γ, log.
So a_4 = mp.log(3) / (24 * mp.pi**2).
Thus the solution is discovered.

```
def proposed_solution():
    from mpmath import mp
    mp.dps = 100
    result = mp.log(3) / (24 * mp.pi**2)
    return result
```

B.6 Permissibility check

In each closed-form problem statement, we provide a numerical approximation with five significant figures for the model to use for grounding its reasoning path. We do this because real-world mathematical research does not involve reasoning in a vacuum, and mathematicians and scientists often use computational approximations to gain insight into the problem at hand and guide their research direction.

However, this means that models sometimes can cheat—by constructing numerical fits to the provided values. We therefore implement a permissibility check for `closed-form` problems to reject this behavior, which involves an LLM to automate the evaluation process. GPT-5.6 Sol is given the following rubric to determine whether it should accept or reject a proposed solution:

Frozen Compliance Rubric

You are a code reviewer checking whether a mathematical solution follows the rules.
Apply these two principles before the numbered method rules:

- **Task fulfillment**: The response must provide the object and form requested by the problem, respect every explicit problem-specific restriction, and cover the full requested parameter domain. A numerically matching constant is non-compliant if the task instead asks for equations, a construction, a formula valid over a parameter domain, or a representation beyond a specifically excluded known form. Problem-specific restrictions override the general allowances below.
- **Judge the mathematical representation, not library internals**: The benchmark separately checks numerical accuracy. Decide whether the mathematical expression represented by the code is permissible. The fact that `mpmath` numerically evaluates an otherwise allowed named function is not by itself a violation.

The solution MUST be a genuine closed-form symbolic expression. The following techniques are **FORBIDDEN**:

1. **Numerical integration**: `mp.quad()`, `mp.quadgl()`, `mp.quadts()`, `mpmath.quad()`, `scipy.integrate`, or any numerical integration routine.
2. **Finite truncations of infinite series**: Loops summing many terms (e.g., `for k in range(10000)`) to approximate an infinite series, precision-dependent or accuracy-dependent stopping rules, and long lists of precomputed terms are forbidden. In contrast, a finite sum, product, loop, or matrix construction whose length is exactly determined by an input integer such as `n` is allowed when it constructs the exact mathematical answer; it is not a truncation merely because its size varies with `n`.
3. **Numerical root-finding**: `fsolve`, `brentq`, `newton`, `nsolve`, `findroot`, `scipy.optimize`, or any numerical solver.
4. **Restating the defining expression as a computational procedure**: Computing a resultant by evaluating one polynomial at the roots of another, or computing a defining sum/product by iterating over its terms, is forbidden. This includes calling `mp.hyper()` or another hypergeometric function when it merely rewrites the problem's defining series. Specifically, if the problem defines $\sum a(n)$ and $a(n+1)/a(n)$ is rational in n, expressing that same series as `mp.hyper(...)` is a notational restatement. Judge the transformation relative to the original definition: when the problem instead defines an integral, limit, differential equation, or other non-series object, a fixed hypergeometric or other named special-function value can be a genuine closed form even though that function has an internal series definition. Do not reject such a transformation merely because an equivalent series exists. An explicit problem restriction excluding series or hypergeometric repackagings still controls.
5. **Unevaluated infinite series/products/limits**: Using `mpmath.nsum`, `mpmath.nprod`, or similar to numerically evaluate an infinite series or product.
6. **Hardcoded or encoded target values**: Returning a bare multi-digit decimal string as the answer without a symbolic derivation is forbidden, e.g., `return mpf("1.20205690315959428539973816151144999076")`. The same prohibition applies to a rational with a power-of-ten denominator, an enormous unexplained integer or integer factorization, a high-degree algebraic number, a continued fraction, or another exact representation whose apparent purpose is to encode the known target digits. Exactness alone does not make such an encoding a symbolic derivation.
7. **Circular / tautological identities**: Using special functions that internally encode or trivially compute the target constant is forbidden. For example, using `mp.hyperu` or `mp.gammainc` to compute the Euler–Mascheroni constant γ is circular when the chosen value is defined or conventionally evaluated through an identity involving γ. Apply an independence test: reject a special-function value when the target is part of its defining local expansion, parameter derivative, normalization, or standard identity at the chosen arguments. The expression must be genuinely independent, not an identity that the constant satisfies by definition.
8. **Numerical parameter fitting / digit-matching constructions**: Expressions with arbitrary-looking coefficients, denominators, exponents, or successive tiny correction terms are forbidden when they appear reverse-engineered to match known target digits.

Concrete warning signs include unusually specific values such as `sqrt(30261)/26`, unexplained large coefficients, and high-power corrections such as `- q**12/9 - (173/4)*q**18`. Structural conjectures remain allowed when supported by small integers, simple rational parameters, symmetries, known related constants, or a uniform formula. State the concrete feature that indicates fitting; do not reject merely because a conjecture is unproven or unfamiliar.

ALLOWED techniques include:

- Using known constants (π, e, γ, Catalan's constant).
- Calling special functions (gamma, zeta, polylog, elliptic integrals, hypergeometric) at specific arguments when they represent a genuinely different mathematical quantity rather than the problem's defining expression.
- Symbolic algebra to combine these into a closed-form expression.
- Exact finite sums, products, loops, and matrix constructions that represent the actual mathematical answer rather than an approximation.
- Named functions of exactly specified finite matrices, including matrix logarithms, square roots, exponentials, determinants, and traces, unless the matrix encodes fitted target digits, disguises a forbidden representation, or violates a problem-specific restriction.
- Novel conjectures combining constants from different mathematical domains when the coefficients are structurally simple and not arbitrarily tuned.

Before returning a verdict:

1. Identify the original mathematical definition and every explicit problem-specific restriction.
2. Check that the response answers the requested object and, for functions, covers the full requested input domain.
3. Apply Rules 1–8 to the mathematical representation, not merely to library implementation details.
4. Check for target-digit encoding, arbitrary fitted parameters, and circular special-function identities.
5. When rejecting, identify either the task-fulfillment failure or the most specific violated numbered rule and cite a concrete feature of the submitted code. When accepting a borderline construction, explain why it is an independent transformation, exact finite construction, or structural conjecture rather than a restatement, truncation, encoding, circular identity, or fit.

We find that this approach is robust. For the novel closed-form solutions that have passed (including the one shown in Appendix A.1), the compliance checker approved the solutions, which were later double-checked by human experts. The examples below are real solutions proposed by the models that pass the initial numerical screen but are then rejected by the compliance checker. The accepted novel closed-form expressions were subsequently proven correct. This establishes their mathematical correctness; we audit admissibility separately in Appendix B.7.

Real examples of submissions rejected by the permissibility check

1. **Numerically tuned correction** (`w4_watson_integral`).

   ```
   result = mp.hyper([h, h, h, h], [1, 1, 1], 1) / 4 + mp.pi / 104
   ```

   Rejected because `mp.pi / 104` was correctly identified as an arbitrary correction factor tuned to match digits.

2. **Arbitrary fitted rational coefficients** (`box_integral_b6_1`).

   ```
   result = E + (mp.mpf(11) / 146) * (K - E) + L / mp.mpf(1813961)
   ```

   Rejected because the coefficients `11/146` and `1/1813961` were reverse-engineered rather than derived.

3. **Finite Gaussian quadrature approximation** (`box_integral_b7_1`).

   ```
   result = w1 * mp.sqrt(x1) + w2 * mp.sqrt(x2) + w3 * mp.sqrt(x3)
   ```

   Rejected because the submission used a 3-point Gaussian moment closure to approximate an integral, not an exact closed form.

4. **Manual numerical root finding** (`kcore_threshold_c3`).

   ```
   return x - (2 * f * fp) / (2 * fp * fp - f * fpp)
   ```

   Rejected because this is Halley's method applied repeatedly to solve a transcendental equation.

5. **Infinite-series evaluation** (`thermal_boson_function_jb`).

   ```
   result = -y * mp.nsum(lambda n:  mp.besselk(2, n * a) / (n * n),
   [1, mp.inf])
   ```

   Rejected because `mp.nsum` numerically evaluates an infinite series.

6. **Custom quadrature routine** (`relativistic_fermi_pressure_integral`).

   ```
   for _ in range(14):  ...  while True:  ...  h /= 2
   ```

   Rejected because the solution implemented tanh-sinh / exp-sinh numerical integration loops.

7. **Truncated series plus Newton inversion** (`photokinetic_bimolecular_rate_law`).

   ```
   while True:  term = mp.ei(-n * y); ...  for _ in range(100):
   ```

   Rejected because it used truncation loops and Newton iteration to invert an implicit equation.

B.7 Robustness of the permissibility checker

To test whether permissibility judgments are stable across checker models, we ran the checker with three GPT-5.6 models (Luna, Terra, and Sol) on a panel of 25 candidate solutions to 13 underlying problems, of which 17 are valid and 8 are invalid, including adversarial problems designed to fool the compliance checker. We acknowledge that this is not a fully comprehensive analysis but believe these results provide a diverse and representative picture of the set, as well as the models' robust behavior with the rubric. The candidates span logarithmic branches, elliptic continuation, principal-value integrals, hypergeometric transformations, and exact finite constructions, and include closely related valid and invalid variants of the same task. The ground-truth labels were withheld from the checker models being evaluated, and all models received identical inputs. Following the procedure shown above, each model was independently sampled to get a five-vote majority from GPT-5.6 Sol.

	GPT-5.6 Luna	GPT-5.6 Terra	GPT-5.6 Sol
Individual judgments matching reference labels	114/125	118/125	123/125
Five-vote majorities matching reference labels	24/25	25/25	25/25
Candidates with 3–2/2–3 accept/reject votes	7/25	5/25	1/25
False-accept majorities / invalid candidates	0/8	0/8	0/8
False-reject majorities / valid candidates	1/17	0/17	0/17

Table 5: Agreement of three GPT-5.6 checker models with reference labels on the 25-candidate panel, with five queries per model and proposed solution.

No model accepted an invalid candidate by majority. The only incorrect majority was Luna's rejection of an exact degree-200 Legendre construction, which it mistook for a numerical approximation. All other disagreements were minority votes within correct majorities. We observe that the consistency of the model's responses increases with model strength: GPT-5.6 Sol, the benchmark's checker, matched the reference labels in 123 of 125 individual judgments and voted unanimously (5 out of 5) on 24 of 25 candidate solutions.

Three illustrative candidates. Each example gives the proposed solution expression (shown as mathematics) and a brief summary of the models' votes. In the matrix example, e_0 is the first basis vector and $J_d(\beta)$ is the $d \times d$ tridiagonal matrix with $(J_d)_{k,k-1} = 1$ and $(J_d)_{k-1,k} = \beta_k$ for $1 \le k < d$, and all other entries are zero.

1. **Scalar-log branch (reject).** For the continuous antiderivative
$$F(t) = \int_0^t \frac{10 + 221x^2 + 667x^4}{1 + 42x^2 + 381x^4 + 529x^6}\, dx,$$
the submitted answer is
$$\widehat{F}(t) = \operatorname{Im} \log \det(I + itA), \qquad A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 5 \end{pmatrix}.$$
Here $\det(I + itA) = 1 - 29t^2 + i(10t - 23t^3)$, so the scalar principal logarithm jumps by 2π at $t = \sqrt{10/23}$. Luna rejected it 4–1. Its one incorrect response treated the expression as continuous.
2. **Degree-200 Legendre construction (accept).** For $L = \int_{-1}^{1} (1 + x + x^2)^{100}\, dx$, let $J = J_{101}(\beta)$ with $\beta_k = k^2/(4k^2 - 1)$. The submitted answer is $\widehat{L} = 2[(I + J + J^2)^{100} e_0]_0$. A contributing path cannot rise above state 100, so this expression is exact. Luna rejected it 3–2, incorrectly calling the finite recurrence a numerical approximation, whereas Terra and Sol both accepted it.
3. **Degree-800 Hermite construction (accept).** For $H = \int_{-\infty}^{\infty} e^{-x^2} (1 + x + x^2)^{400}\, dx$, let $J = J_{401}(\beta)$ with $\beta_k = k/2$. The submitted answer is $\widehat{H} = \sqrt{\pi}[(I + J + J^2)^{400} e_0]_0$. A contributing path cannot exceed state 400. Luna accepted it 4–1, incorrectly claiming a path could reach state 401 and return to zero.

These examples test two subtle distinctions: a plausible formula can use the wrong branch, and a finite computation can still be exact. Terra and Sol reached the correct majority in all four cases, while Luna wrongly rejected the Legendre construction. (Only Sol is used for evaluation. The comparison is provided to demonstrate the higher reliability of stronger models.) Notably, each of the accepted solutions was manually verified and all four closed forms were proven by the authors, none of the accepted results reported in the manuscript depends on the LLM judge.

C Additional representative example

Three Mutually Orthogonal Latin Squares of Order 10

Coding Theory | *Construction* | `New-Construction` | *Solvability 1*

Task. A Latin square of order 10 is a 10×10 array over $\{0, 1, \ldots, 9\}$ in which every symbol occurs exactly once in each row and column. Two such squares are *orthogonal* if their superposition contains each of the 100 ordered pairs exactly once. Whether three mutually orthogonal Latin squares of order 10 exist remains open (Rubin et al., 2021). Construct three Latin squares L_1, L_2, L_3 of order 10 such that each of (L_1, L_2), (L_1, L_3), and (L_2, L_3) is orthogonal.

Output format. A dictionary `{"squares": [L1, L2, L3]}` containing three 10×10 integer arrays with entries in $\{0, \ldots, 9\}$. The validator checks the Latin-square conditions and all three pairwise orthogonality conditions exactly.

D Problem selection

We carefully selected the problems in the literature following the criteria in Figure 2: an explicit concrete answer, objective verifiability via an efficient computational verifier, resistance to standard algorithms, and open research questions with scientific significance. The combination of these criteria is essential: we cannot simply collect open problems from arXiv, as they might require proofs (violating criterion 1) or already be solved (violating criterion 4). This combination ensures that a solution to each problem would denote meaningful mathematical discovery that can be easily verified. Every baseline problem is meaningful by definition, since effort has already been spent setting the baseline, and all closed-form problems are taken from articles explicitly noting that no closed form has been found. Furthermore, every closed-form problem comes from a source that explicitly reports no known closed form, and expert judgment during the selection process excludes problems

where a known symbolic reduction or a suitable basis of constants would enable direct solution by CAS or PSLQ. The linked anonymized repository provides details on the source, evaluation mode, reference precision when applicable, significance or reasoning for inclusion, and verifier for every problem in the dataset.

E Additional related work: related domains and datasets

While not strictly mathematics, FrontierCS (Mang et al., 2025) shares our design principles: it provides 100+ open-ended problems across algorithmic optimization and computer science. Each problem is hand-crafted, but none have a known optimal solution, and any proposed solution can be deterministically scored on a 0-100 scale by an automated evaluator. FrontierCS provides expert-authored reference solutions, reproducible evaluation harnesses, and dynamic difficulty scaling. Though not mathematics-focused, its open-ended design, automatic verification, and unsolved nature inspire our work.

Beyond formal benchmarks, curated databases of open problems have also become proving grounds for AI. Bloom's Erdős Problems site (Bloom, 2021) maintains a collection of Erdős problems, and the Tao-Ivanisvili-Davis optimization-constants repository lists problems and known bounds but provides no evaluation harness. Finally, another related direction includes the UQ (Unsolved Questions) benchmark (Nie et al., 2025), which evaluates models on unsolved questions sourced from Stack Exchange. However, practical issues such as question ambiguity, verifier brittleness, and the need for community input limit its reliability as a measure of AI progress on unsolved problems.

Limitations

HorizonMath evaluates results with computationally checkable outputs, so proof-based contributions are out of scope. We discuss extending problem types with Lean as a future direction in Section 8. Its evaluation modes also differ in strength. Construction validators certify the required properties exactly, but a closed form matching the reference to $\min(20, D)$ digits remains a conjecture until proven. Admissibility is judged by an LLM checker from the same model family as the strongest evaluated models. We provide additional analyses of the permissibility check robustness across three models (using the strongest models in production) but note that it does not consider every problem or representation in the dataset. Furthermore, while the open status of a problem does not guarantee that a problem resists computer algebra or integer-relation search, all problems were deliberately chosen to avoid these techniques by expert mathematicians. However, we acknowledge this does not guarantee that every problem is fully resistant to such techniques. Finally, once a solution becomes public, later solves from models may indicate retrieval rather than discovery, but we aim to mitigate this possibility by using encrypted release of solutions to avoid training data contamination and web search retrieval, while also providing more clarity to third-party evaluators.

Broader Impact

HorizonMath is an evaluation benchmark for mathematical reasoning, intended to help the research community track AI progress on unsolved problems drawn from the open mathematical literature. The benchmark uses problems and solutions that are publicly available, the verification framework is transparent and open-source, and each verified solution on the benchmark constitutes a potentially new contribution to mathematics.