All 4 models land within 0.146% on founder arithmetic; only 1 of them is exact 10 times out of 10

10 seeded SaaS problems, every tool and MCP server disabled, ground truth computed in code

Dmitriy Semenkevich · ORCID 0009-0009-3013-4978

16 August 2026 · dimhold.by

ABSTRACT

The shorthand that language models cannot do arithmetic is stated far more often than it is measured on the arithmetic people actually delegate. This study put 10 seeded problems of the kind a founder asks, compounding MRR with churn applied monthly, multi year cash forecasts against a fixed burn, LTV, a growth rate that switches partway through the timeline and runway under a stepped burn, to 4 Anthropic models with every built in tool and every MCP server disabled, so each model had to do the arithmetic itself. Answers were single shot with no working shown. Expected values were computed in code and the model that answered was verified from the API response rather than assumed from the flag. Every model answered every question. The worst relative error anywhere in the run was 0.145127%, so every answer landed within 0.146% of the truth. Separation appears only at the cent. Exact matches, meaning a relative error at or under 1e-6, were 10 of 10, 9 of 10, 9 of 10 and 3 of 10. The weakest model was never wildly wrong; it was slightly wrong 7 times out of 10. On one 36 month cash forecast that slight wrongness came to $8,451 with nothing in the reply to signal it. It also spent the most reasoning of any model, 58,027 tokens against 7,771 for the most accurate one. More reasoning went with less accuracy across these 4 models. That is a direction rather than a lever: n is 4 and token spend is inseparable from model identity.

1. The question

“Language models cannot do arithmetic” is one of the most repeated claims about them and one of the least often measured on the arithmetic that gets delegated in practice. A founder does not ask a model to multiply 7 digit primes. They ask what MRR looks like after 36 months of compounding growth net of churn and when the cash runs out under a burn that steps up every quarter.

Those problems are short. They are also iterative, which means a rounding decision in month 3 is still present in month 60. This study asks how far off the answers land and where the models differ from each other.

The tools are taken away on purpose. In production nobody would run these without a calculator. That is exactly why the tool free case is worth measuring: it isolates the model.

2. Method

2.1 Isolation, verified rather than assumed

Built in tools are disabled with `--tools ""` and MCP servers are disabled with `--strict-mcp-config --mcp-config no-mcp.json`. Both are needed. `--tools ""` alone leaves a connected MCP server reachable. An MCP that can run SQL would quietly do the arithmetic.

```
claude -p --output-format json --model <model> \
  --tools "" --strict-mcp-config --mcp-config no-mcp.json
```

Isolation was checked with a probe rather than assumed: asked to read a local file containing a random string, the model answers that it has no Read tool. That probe was run by hand outside the harness. Its reply was not stored, so the check is reported here rather than reproducible from the repository.

2.2 The problems

10 problems, generated from seed `20260812`, so the same command reproduces the same set. Each states its own rule, which matters more than it sounds: a wrong answer then means an arithmetic error rather than a disagreement about how to model a business.

ID	SHAPE	HORIZON
`mrr-0`, `mrr-5`	MRR grows, then churn is applied to the grown figure	36 and 60 months
`cash-1`, `cash-6`	revenue compounding against a fixed monthly burn	60 and 36 months
`ltv-2`, `ltv-7`	LTV as ARPU times gross margin over monthly churn	one division
`two-phase-3`, `two-phase-8`	growth rate switches at month 6, churn constant	60 and 36 months
`runway-4`, `runway-9`	burn steps up every 3 months; name the month cash goes negative	integer answer

A representative one, `cash-6`:

> *A company starts with $420,000 in cash. It spends a fixed $40,000 every month. Revenue is $18,000 in month 1 and grows 11.0% every month after that (month 2 revenue = month 1 revenue x 1.11).*

It then defines cash after 36 months as starting cash plus all revenue collected less all spend and asks for that figure.

2.3 The prompt, the same for everyone

```
Answer the question below. Do not show any working, do not explain.
Reply with exactly one line in the form:
ANSWER: <number>
```

Models: `claude-opus-5`, `claude-sonnet-5`, `claude-haiku-4-5` and `claude-fable-5`, through the Claude Code CLI on 16 August 2026. Same questions, same prompt, same isolation, single shot, no retries on content.

The model that answered is verified against `modelUsage` in the CLI response rather than assumed from the `--model` flag. A call that never reached the requested model fails

loudly instead of being scored. Rate limits are retried with backoff and never counted as wrong answers.

2.4 The metric

Expected values are computed in `run.ts`. No model grades anything.

Relative error is the absolute difference between the answer and the expected value, divided by the expected value. **Exact** means the answer differs from the computed value by no more than display rounding, a relative error at or under 1e-6. The bar has to be that tight to separate anything. Loosening it by 1 order of magnitude, to 1e-5, would score the $37.90 that `claude-haiku-4-5` missed the 60 month cash forecast by as exact, on a forecast running to $4.86M.

Wall clock time is deliberately not measured, because it measures the connection as much as the model. Reasoning token counts are reported instead: they come out of the API response and do not depend on the link.

3. Results

Every model answered every question and every answer landed within 0.146% of the truth. Exact matches were 10 of 10 for `claude-opus-5` against 3 of 10 for `claude-haiku-4-5`.

	`CLAUDE-OPUS-5`	`CLAUDE-SONNET-5`	`CLAUDE-HAIKU-4-5`	`CLAUDE-FABLE-5`
answered	10/10	10/10	10/10	10/10
exact	**10/10**	9/10	**3/10**	9/10
within 1%	10/10	10/10	10/10	10/10
worst relative error	0.00009%	0.0225%	**0.1451%**	0.0005%
reasoning tokens, total	**7,771**	34,520	**58,027**	8,274

Nobody refuses and nobody collapses. Every answer from every model is inside 1% of the truth, which is already at odds with the shorthand.

Separation is at the cent. The 3 problems `claude-haiku-4-5` got exactly right are 1 of the 2 LTV divisions and the 2 integer month answers. Every problem requiring a compounded sequence came back slightly wrong. So did the other division, `ltv-7`, missed by $0.52 with no compounding in it at all, so compounding is where most of the drift sits rather than all of it.

TASK	EXPECTED	`CLAUDE-HAIKU-4-5`	OFF BY
`cash-6`, 36 months	5,822,959.30	5,831,410	$8,450.70
`two-phase-3`, 60 months	18,578.75	18,553	$25.75
`cash-1`, 60 months	4,864,506.92	4,864,544.82	$37.90

The $8,451 miss on `cash-6` is a relative error of 0.1451%, the worst anywhere in the run. It reads as a perfectly reasonable number. Nothing in the reply signals that it is off. The prompt asked for no working, so there is nothing to inspect either.

More reasoning went with less accuracy. `claude-haiku-4-5` spent 58,027 reasoning tokens, roughly 7.5 times the 7,771 of `claude-opus-5`, then finished last on exactness. `claude-fable-5` reached 9 of 10 on 8,274 tokens while `claude-sonnet-5` reached the same 9 of 10 on 34,520. Across these 4 points the model that thought least was the most accurate.

Difficulty concentrates where the rule changes. The problems whose growth rate switches at month 6 and the long cash forecasts produced the largest errors. Straight compounding and single divisions were handled cleanly by all 4.

4. Why a near miss is the dangerous shape

A model that refuses is handled by a fallback. A model that returns 42 where the answer runs into the millions is caught by any sanity check. Neither happened here.

What happened is a number that is right to 3 significant figures and wrong underneath. It survives eyeballing. It survives an order of magnitude check. It survives a reviewer who knows roughly what the answer should be, because roughly what the answer should be is exactly what it is. In a 36 month cash forecast that shape of error was $8,451.

This is a different engineering problem from hallucination. There is nothing to detect in the text, while asking for working would only produce more text to check. The defence is arithmetic done outside the model, with the model used for setting the problem up rather than for evaluating it.

The reasoning result points the same way with a much weaker claim attached. If accuracy came from thinking longer, the fix would be a budget. Across these 4 points it did not: the model with the largest reasoning spend placed last. That comparison runs between models rather than inside one. n is 4 and token spend is perfectly confounded with model identity, so nothing here says what a larger thinking budget would do to any single model. Section 5 puts the same limit on it.

5. Threats to validity

10 problems put to 4 models in 1 sitting, single shot. This compares 4 models on identical tasks. It does not establish a failure rate for any of them. 10 problems cannot separate 9 of 10 from 10 of 10 with any confidence. The gap between 3 of 10 and 10 of 10 is wider than the design can produce by chance, but the ordering of the middle 2 is not a result.

The problems come from 1 generator with 1 seed. Their shapes are the ones a founder asks about, chosen rather than sampled. A different family of problems could rank the models differently.

Nothing here says anything about behaviour with tools enabled, which is how anyone would actually run these. The measurement is of the model alone, on purpose.

The CLI ran at its default decoding settings, which is what practitioners run but not a controlled temperature.

Reasoning token counts come from the API response for a single sitting. They are a spend figure rather than a measure of reasoning quality, while 10 problems per model is a thin basis for the inverse relationship reported above. It is a direction, not a law.

An earlier pass on 12 August 2026 ran only `claude-opus-5` and `claude-haiku-4-5` and reproduced the headline of 10 of 10 against 3 of 10. That pass is recorded in the repository README, but its raw data is not in the repository, so nothing here rests on it.

The repository notes that `claude-fable-5` is also the model that drove the session in which this run was made. That does not affect the score, because expected values are computed in `run.ts` and no model grades itself: it submits an answer that the script checks.

One correction to `RESULTS.md` in the repository belongs here. Its per task table labels the horizon of `mrr-5`, `cash-6` and `two-phase-8` wrongly and its prose calls `cash-6` a 5 year forecast. The README is not the file at fault. Read from the questions stored in `results.json`, `mrr-5` runs 60 months, `cash-6` runs 36 and `two-phase-8` runs 36. The expected values, the answers and the errors are unaffected; every number in this paper was recomputed from `results.json` rather than copied from that table.

6. Prior work

Arithmetic without a calculator is a well populated field. This study is a comparison inside 1 vendor family rather than a new benchmark.

- How well do Large Language Models perform in Arithmetic tasks? (arXiv:2304.02015, March 2023) is the direct ancestor of the question: arithmetic ability measured on its own, across operation types, with no tools involved.
- GPT Can Solve Mathematical Problems Without a Calculator (arXiv:2309.03241, September 2023) argues the opposite of the shorthand quoted at the top of this paper, showing accurate multi digit arithmetic from a fine tuned model without external tools.
- FinanceReasoning: Benchmarking Financial Numerical Reasoning More Credible, Comprehensive and Challenging (arXiv:2506.05828, June 2025) benchmarks financial

numerical reasoning at scale, in a program of thought setting where the model may emit code.

- BankMathBench: A Benchmark for Numerical Reasoning in Banking Scenarios (arXiv:2602.17072, February 2026) is the closest in domain, covering everyday banking computations that need exponents and geometric progressions.
- wesm/llm-arithmetic-benchmark (December 2025) is the nearest neighbour and it is code rather than a paper. It measures tool free arithmetic aggregation, a sum grouped by key over CSV rows, under the same no calculator condition used here. Its `benchmark.py` lists 6 Anthropic models beside OpenAI and locally hosted ones, so a within family comparison on identical arithmetic tasks already exists.
- Inverse Scaling in Test-Time Compute (arXiv:2507.14417, July 2025) constructs tasks on which extending a reasoning model's chain of thought lowers accuracy while naming 5 distinct failure modes for it. That is the published form of the reasoning result above. Their tasks are built to produce the effect, with distractors injected on purpose. Here the effect is incidental, on clean and well posed problems, which makes it weaker evidence on a more ordinary task.

1 thing was searched for and not found. Nothing scores a near miss separately from an exact match, meaning no paper reporting "within 0.1% relative" and "exact to the cent" as different numbers on the same problem set; every hit on rounding turned out to be about quantization.

An earlier version of this section claimed a second gap: that nothing compares several models of 1 vendor family on identical arithmetic tasks. The repository's own prior work list names `wesm/llm-arithmetic-benchmark` doing exactly that, so the claim is withdrawn rather than narrowed. What is particular here is the problem set, 10 questions a founder types rather than aggregation over CSV rows, together with an isolation that cuts MCP as well as built in tools. This is a small comparison rather than a benchmark.

Searched arXiv, Semantic Scholar and GitHub on 29 August 2026. Semantic Scholar rate limited most queries and general web search was unavailable that day, so the open web outside those sources is not claimed as checked. The prior work list kept in the repository was compiled on 27 August 2026, 11 days after the run rather than before it.

7. Availability

The harness, the seeded task generator, the structured results and the full transcript of every question and reply are in the repository. The archived release carries the DOI above.

8. References

1. Zheng Yuan and others. How well do Large Language Models perform in Arithmetic tasks? arXiv preprint, 2023. arXiv:2304.02015. https://arxiv.org/abs/2304.02015
2. Zhen Yang and others. GPT Can Solve Mathematical Problems Without a Calculator. arXiv preprint, 2023. arXiv:2309.03241. https://arxiv.org/abs/2309.03241
3. Zichen Tang and others. FinanceReasoning: Benchmarking Financial Numerical Reasoning More Credible, Comprehensive and Challenging. arXiv preprint, 2025. arXiv:2506.05828. https://arxiv.org/abs/2506.05828
4. Yunseung Lee and others. BankMathBench: A Benchmark for Numerical Reasoning in Banking Scenarios. arXiv preprint, 2026. arXiv:2602.17072. https://arxiv.org/abs/2602.17072
5. Wes McKinney. llm-arithmetic-benchmark. GitHub repository, December 2025. https://github.com/wesm/llm-arithmetic-benchmark
6. Aryo Pradipta Gema and others. Inverse Scaling in Test-Time Compute. arXiv preprint, 2025. arXiv:2507.14417. https://arxiv.org/abs/2507.14417
7. Dmitriy Semenkevich. llm-arithmetic: harness, seeded task generator and raw results. GitHub, 2026. https://github.com/dimhold/llm-arithmetic
8. Dmitriy Semenkevich. All 4 models land within 0.146% on founder arithmetic; only 1 of them is exact 10 times out of 10. Zenodo, 2026. doi:10.5281/zenodo.22128841. https://doi.org/10.5281/zenodo.22128841

CITE THIS

```
Semenkevich, D. (2026). All 4 models land within 0.146% on founder arithmetic; only 1 of them is exact 10 times out of 10. dimhold.by. https://doi.org/10.5281/zenodo.22128841
```

LLM evaluation · arithmetic reasoning · SaaS metrics · tool free evaluation · reasoning tokens · numerical precision