ASTER: Agentic Scaling with Tool-integrated Extended Reasoning

Anonymous ACL submission

Abstract

Reinforcement Learning (RL) has emerged as a dominant paradigm for eliciting long-horizon reasoning in Large Language Models (LLMs). However, scaling Tool-Integrated Reasoning (TIR) via RL remains challenging due to *interaction collapse*: a pathological state where models fail to sustain multi-turn tool usage, instead degenerating into heavy internal reasoning with only trivial, post-hoc code verification. We systematically study three questions: (i) how cold-start Supervised Fine-Tuning (SFT) induces an agentic, tool-using behavioral prior, (ii) how teacher-trajectory structure and interaction density shape exploration and downstream RL outcomes, and (iii) how the RL interaction budget affects learning dynamics and generalization under varying inference-time budgets. We then introduce **ASTER** (**A**gentic **S**caling with **T**ool-integrated **E**xtended **R**easoning), a framework that circumvents this collapse through a targeted cold-start strategy prioritizing structured, interaction-dense trajectories. We find that a small expert cold-start set of just 4K structurally rich, interaction-dense trajectories yields the strongest downstream performance, establishing a robust prior that enables superior exploration during extended RL training. Extensive evaluations demonstrate that ASTER-4B achieves state-of-the-art results on competitive mathematical benchmarks, reaching 90.0% on AIME2025, surpassing leading frontier open-source models, including DeepSeek-V3.2-Exp.

1 Introduction

Large Language Models (LLMs) have achieved significant reasoning improvements through Reinforcement Learning (RL), which facilitates the refinement of long Chains of Thought (CoT) (Wei et al., 2022; Yao et al., 2023). Recent systems such as OpenAI o1 (Jaech et al., 2024) and DeepSeek R1 (Guo et al., 2025) demonstrate the effectiveness of this approach. By employing analytical strategies such as problem decomposition, error correction and iterative refinement, these models exhibit strong capabilities in complex reasoning tasks (Yeo et al., 2025).

While long CoT has achieved success, text-only reasoning remains inherently fragile, as the absence of external verification allows minor errors to cascade beyond the reach of self-correction mechanisms. Tool-Integrated Reasoning (TIR) overcomes this by augmenting language models with executable tools, leveraging external feedback to enable exact computation and intermediate verification, thereby ensuring reliability in long-horizon reasoning tasks (Lin and Xu, 2025).

Existing approaches to TIR primarily differ in how tool-use behavior is initialized. Zero Tool-Integrated Reasoning (ZeroTIR) seeks to elicit tool usage directly from pretrained base models via RL, without relying on supervised cold-start data. Despite its conceptual simplicity, ZeroTIR suffers from two fundamental limitations. From the perspective of optimization, multi-turn TIR training is frequently unstable. Tool execution introduces non-stationary feedback and distributional drift toward low-probability tokens, which compounds across turns and often results in gradient explosion and poor convergence behavior (Wang et al., 2025; Mai et al., 2025; Xue et al., 2025). From the perspective of capability, RL predominantly amplifies pre-existing competencies rather than instilling fundamentally new ones (Zhang et al., 2025). As a consequence, ZeroTIR remains bounded by the representational and procedural limits of the underlying base model, and consistently struggles with long-horizon planning, coordinated tool invocation, and program-level error correction.

An alternative paradigm initializes TIR via Supervised Fine-Tuning (SFT) on carefully chosen, tool-augmented trajectories before RL (Feng et al., 2025; Shang et al., 2025; Yu et al., 2025b). These trajectories are commonly obtained either by rewrit-

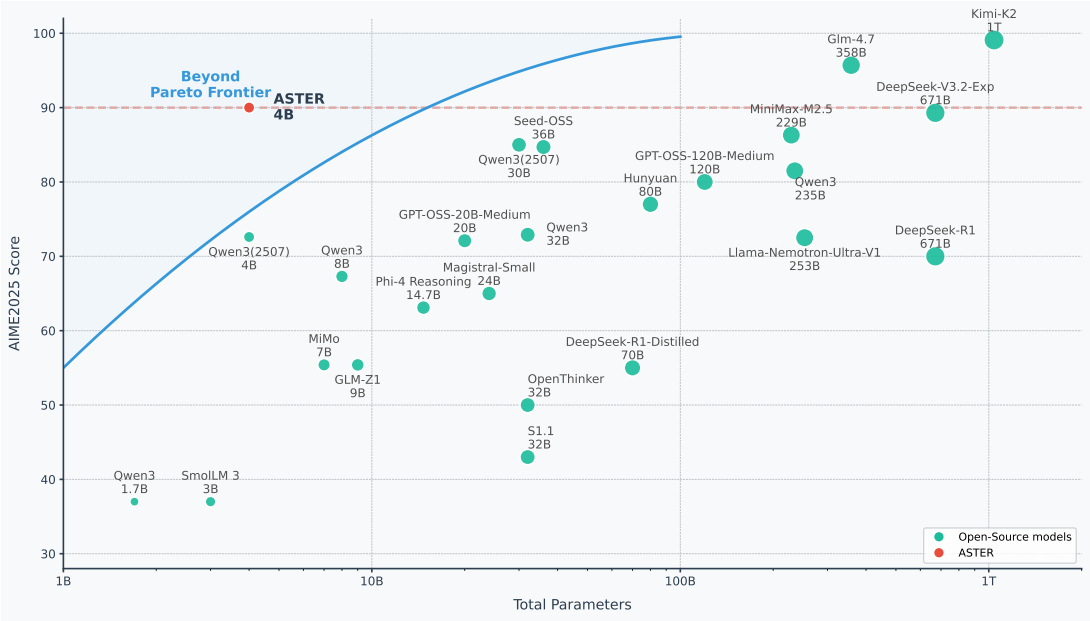


Figure 1: ASTER-4B achieves a score of 90.0% on AIME2025, outperforming DeepSeek-V3.2-Exp (89.3% with 671B parameters). Scores for compared models are taken from their official technical reports.

ing text-only CoT traces into code-interleaved format (Feng et al., 2025) or by synthesizing tool-using demonstrations with code-specialized teacher models (Yu et al., 2025b). Despite their empirical success, existing cold-start designs introduce strong inductive biases through trajectory construction. ReTool (Feng et al., 2025) largely preserves the original text-only reasoning structure, limiting the emergence of genuinely tool-driven planning. In contrast, trajectories produced by code-specialized teachers (Yu et al., 2025b) often feature short, reactive reasoning. This contradicts the deliberate, System-2 behavior expected in slow-thinking models. These inductive biases often lead to *interaction collapse* during subsequent RL training, a pathological failure mode in which models fail to sustain multi-turn tool usage and instead degenerate into short solve-to-final or finalization-heavy policies, where they make only a few tool calls before directly producing the final answer.

In this work, we introduce **ASTER** (**A**gentic **S**caling with **T**ool-integrated **E**xtended **R**easoning), a two-stage framework for studying how cold-start SFT design and RL interaction budgets shape long-horizon, multi-turn tool-using behavior. Specifically, we ask:

- **RQ1 (Cold-start SFT and behavioral priors).** How do different cold-start SFT designs, particularly their trajectory structures, shape the behavioral prior that governs downstream tool-use strategies and RL performance?
- **RQ2 (Interaction density).** How does the interaction density of cold-start trajectories influence exploration behavior and subsequent RL outcomes?
- **RQ3 (RL interaction budget).** How does the RL interaction budget impact learning dynamics and generalization, especially when inference-time budgets vary?

Our experiments show that an effective cold-start prior should not merely maximize post-SFT accuracy, but instead induce a favorable trajectory structure: exploratory, multi-stage tool use rather than a collapsed solve-to-final path. We operationalize this principle with interaction density, a practical curation signal for retaining trajectories that expose structured tool-use behavior. Specifically, we synthesize tool-augmented trajectories with GPT-OSS-20B and curate a 4K expert subset whose examples contain at least 9 tool-interaction turns for cold-start SFT. This interaction-dense initialization gives ASTER a high-entropy behavioral prior, preserving exploration capacity during RL.

Rather than only counting how often tools are called, we examine whether each trajectory follows a coherent pattern of exploration, solution advancement, and verification before producing the final answer. This analysis shows that ASTER preserves structured explore–solve–verify behavior before and after RL, whereas baseline policies concentrate into narrow finalization-heavy chains

and prematurely converge to short-horizon, suboptimal policies. Building on this structured prior, ASTER further employs a multi-stage RL procedure to stabilize and refine extended reasoning loops, allowing the model to exploit RL capability scaling and surpass baselines trained on sparse, short-horizon trajectories. Empirically, as summarized in Table 1, ASTER-4B achieves state-of-the-art results, reaching 85.0% accuracy on AIME2025 and 73.3% on HMMT2025 under a 30K inference budget. Despite its compact size, it consistently outperforms the 235B-parameter Qwen3-235B-A22B-Thinking across a diverse suite of competitive mathematics benchmarks. Moreover, ASTER exhibits strong test-time scaling: with an increased inference budget of 90K, performance improves to 90.0% on AIME2025, confirming that structured cold-start training unlocks scalable agentic intelligence. This efficiency trend is shown in Figure 1, where ASTER reaches frontier-level AIME2025 performance with substantially fewer parameters.

2 Preliminaries

Supervised Fine-Tuning SFT adapts pre-trained LLMs to specialized tasks via next-token prediction on high-quality instruction-response pairs. In the context of agentic reasoning, SFT often functions as a cold-start phase to initialize multi-turn tool-use capabilities. Common strategies for constructing such training data involve rewriting text-only CoT solutions into code-interleaved formats (Feng et al., 2025) or leveraging code-specialized teacher models to synthesize tool-using demonstrations (Yu et al., 2025b). This initialization establishes the necessary formats and baseline interaction capabilities for subsequent RL.

Reinforcement Learning SFT provides a cold start for agentic reasoning, but its performance is constrained by the quality and diversity of demonstrations. RL further improves the model by optimizing reward signals obtained from exploration, such as final-answer correctness or the validity of intermediate execution steps. A widely used method is Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which avoids training an explicit value critic by sampling a group of G responses $\{o_1, \ldots, o_G\}$ for each query q drawn from the prompt distribution $\mathcal{D}$ and estimating advantages from their relative rewards. GRPO's optimization objective maximizes the following surrogate loss:

$$\mathcal{J}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\Bigg[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\, \text{clip}\big(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_{i,t}\Big)\Bigg],$$

where $o_i = (o_{i,1}, \ldots, o_{i,|o_i|})$ denotes the i-th sampled response and $o_{i,<t}$ denotes its prefix before token t. The expectation uses $\pi_{\theta_{\text{old}}}(\cdot|q)$ as a sequence-level sampling distribution, whereas the policy ratio is computed at the token level under the autoregressive factorization: $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$. Here, $\hat{A}_{i,t}$ is an estimated advantage, typically derived from the group-relative reward of the full response and assigned to its response tokens. The clipping operator bounds the ratio to $[1-\epsilon, 1+\epsilon]$ to limit policy updates.

3 Towards Best Agentic Scaling Recipe

This section answers the three research questions introduced in Section 1 and uses the resulting insights to derive a concrete ASTER training recipe. We proceed from cold-start design, to interaction density, to RL interaction budget. Beyond aggregate tool-call counts, we use phase-level trajectory analysis to test whether a policy sustains structured explore–solve–verify behavior or collapses into short finalization-heavy trajectories.

3.1 Experimental Setup Summary

We provide full data construction, optimization hyperparameters, decoding settings, and baseline details in Appendix A.1. Unless otherwise specified, experiments use Qwen3-1.7B-Thinking and Qwen3-4B-Thinking-2507, a 45K GPT-OSS-20B-high synthesized tool-trajectory pool from Skywork-OR1 and AoPS/OpenMathReasoning prompts, and DAPO-Math-17K for RL. RL uses GRPO with group size $G = 8$ and a 50-call training cap. In RQ1, we use the full 45K curated pool to study cold-start teacher-pattern effects, while RQ2–RQ3 and the final recipe use controlled 4K high-interaction subsets to isolate density and budget effects. We evaluate on AIME2024, AIME2025, HMMT2025 (February), and BeyondAIME (Seed et al., 2025) with a default 32K evaluation context, comparing against published text-only and agentic baselines.

Model	AIME2024 avg@16	AIME2025 avg@16	HMMT2025 avg@16	BeyondAIME avg@16
Text-Only Reasoning Models				
Qwen3-1.7B-Thinking	47.5	38.3	25.8	20.8
Qwen3-4B-Thinking-2507	76.7	72.7	48.1	43.6
OpenAI o3-mini (medium)	79.6	77.0	53.0	–
POLARIS-4B-Preview	81.2	79.4	58.7	–
OpenReasoning-Nemotron-7B	84.7	78.2	63.5	–
Qwen3-235B-A22B-Thinking	85.7	81.5	62.5	–
Agentic Reasoning Models				
ReTool-32B	72.5	54.3	–	–
rStar2-Agent-14B	80.6	69.8	52.7	–
DemyAgent-4B	72.6	70.0	52.9†	35.3†
ASTER-1.7B-SFT	19.4	19.0	11.3	6.4
ASTER-1.7B	64.6	59.6	47.5	26.3
ASTER-4B-SFT	62.5	54.6	43.3	27.4
ASTER-4B	82.3	85.0	73.3	53.9
w/ 90K Inference Budget	**85.8**	**90.0**	**77.1**	**61.7**

Table 1: Performance comparison on competitive mathematical benchmarks. Scores are avg@16 accuracy under the DeepSeek-R1 evaluation framework (temperature=0.6, top_p = 0.95). We prioritize official performance data where available; results marked with † denote our independent evaluation using officially recommended configurations restricted to a 30K inference budget. The best scores are highlighted in **bold**, and the second-best are underlined.

3.2 Role of Cold-Start SFT and Behavioral Priors (RQ1)

To answer RQ1, we use Qwen3-4B-Thinking-2507 as the unified base model and compare different ways of entering RL under matched response budgets and tool-call caps. The variants include zero-start controls, reward/filtering controls, and cold-start SFT priors:

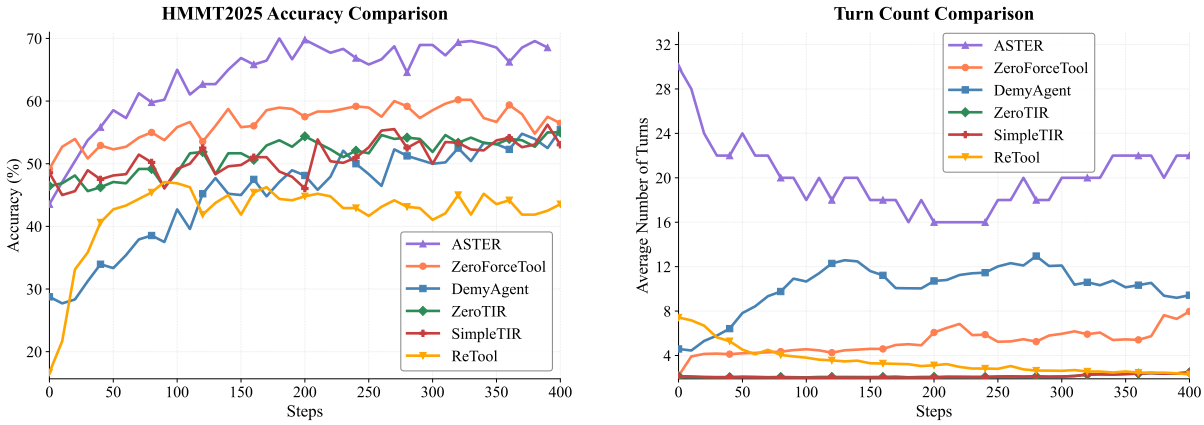
- **ZeroTIR**: Direct RL training without cold-start SFT. The reward assigns 1 to correct final answers and 0 otherwise.
- **ZeroForceTool**: Direct RL training without cold-start SFT, but with a tool-constrained reward. The reward assigns 1 only if the final answer is correct *and* explicit tool usage is observed; otherwise, it assigns 0.
- **SimpleTIR**: Direct RL training without cold-start SFT, but with trajectory filtering during RL (Xue et al., 2025). We filter out trajectories containing "void turns", i.e., turns that yield neither a code block nor a final answer.
- **ReTool**: SFT on the ReTool dataset (Feng et al., 2025)[1], which follows a "stitch-style" synthesis paradigm that rewrites computational steps from trajectories into code snippets.
- **DemyAgent**: SFT on the DemyAgent dataset (Yu et al., 2025b)[2], consisting of end-to-end synthetic trajectories generated by Qwen3-Coder-30B-A3B.
- **ASTER**: SFT on our full 45K curated dataset.

This setup makes the key comparison a question of trajectory structure rather than model scale or RL budget. We further check that the main ASTER–ReTool contrast is not explained by easier SFT problems: the two datasets have comparable difficulty under the same base model and decoding protocol. What differs is the interaction pattern exposed during SFT. ReTool and DemyAgent concentrate most demonstrations in sparse tool-use regimes, whereas ASTER contains substantially more long-horizon trajectories. Detailed dataset diagnostics are provided in Appendix A.2.

Figure 2 shows that these structural differences survive RL and translate into different operating regimes. Zero-start and sparse-prior variants either

[1] ReTool-SFT dataset

[2] Open-AgentRL-SFT-3K dataset



(a) HMMT2025 accuracy comparison.

(b) Tool calls comparison.

Figure 2: RQ1 training dynamics under different cold-start strategies. Cold-start choices lead to different downstream accuracy and tool-use regimes, even under the same base model, RL recipe, response budget, and tool-call cap.

learn limited tool use or drift toward shorter interaction patterns, while ASTER maintains substantial tool use and achieves higher downstream accuracy. Thus, cold-start SFT is not merely a source of better initial answers; it determines which tool-use behaviors RL can refine.

To understand why the same RL procedure produces such different behavior, we examine how tool use is organized within trajectories. A GPT-OSS-120B judge assigns turn-level code-purpose labels, which we map to *Explore & Expand*, *Solve & Advance*, *Verify & Repair*, and terminal *Final Answer* phases. We then build transition-probability graphs over these phases to distinguish sustained explore–solve–verify behavior from short finalization-heavy chains. Full phase definitions and aggregation details are provided in Appendix A.2.3.

Figure 3 exposes the transition prior induced before RL. We summarize each heatmap with *Destination-Final*, the share of phase transitions ending in *Final Answer*; lower values indicate continued movement among exploration, solution advancement, and verification. Before RL, ASTER has a much lower Destination-Final rate than ReTool and DemyAgent (15.7% vs. 71.6% and 76.6%), showing that its cold-start prior is distributed across reusable tool-use phases rather than concentrated into solve-to-final chains.

After RL, ASTER becomes more concise without collapsing: its Destination-Final rate rises only to 24.7%, and its destination entropy remains high. By contrast, ReTool collapses toward direct finalization (93.9% Destination-Final), ZeroForceTool learns mostly a short solve-to-final policy, and DemyAgent remains more concentrated than ASTER. Together with the phase-share view in Figure 9 in Appendix A.2.3, these results show that sparse demonstrations can bias RL toward narrow policies, whereas ASTER induces a multi-stage prior that RL can refine. We additionally test whether this cold-start effect extends beyond mathematical reasoning with a search-domain diagnostic in Appendix A.5.

Takeaway 1: Cold-start SFT Shapes the RL Prior. The benefit of cold-start SFT depends strongly on the structure of demonstrations: sparse trajectories push RL toward finalization-heavy policies, whereas ASTER induces an explore–solve–verify prior that RL refines without collapsing.

3.3 Effect of Interaction Density of SFT (RQ2)

To isolate the effect of interaction density in cold-start SFT, we construct three controlled subsets from the 45K dataset described in Section 3.1:

- **High Density**: 4K trajectories with an average of 10.86 tool invocations per trajectory.
- **Low Density**: 4K trajectories with an average of 1.68 tool invocations per trajectory. Crucially, this subset is selected to have a nearly identical trajectory length distribution to High Density, ensuring that the two subsets differ primarily in interaction density rather than sequence length.
- **RandomDrop25**: Derived from High Density by randomly removing 25% of tool invocations from each trajectory.

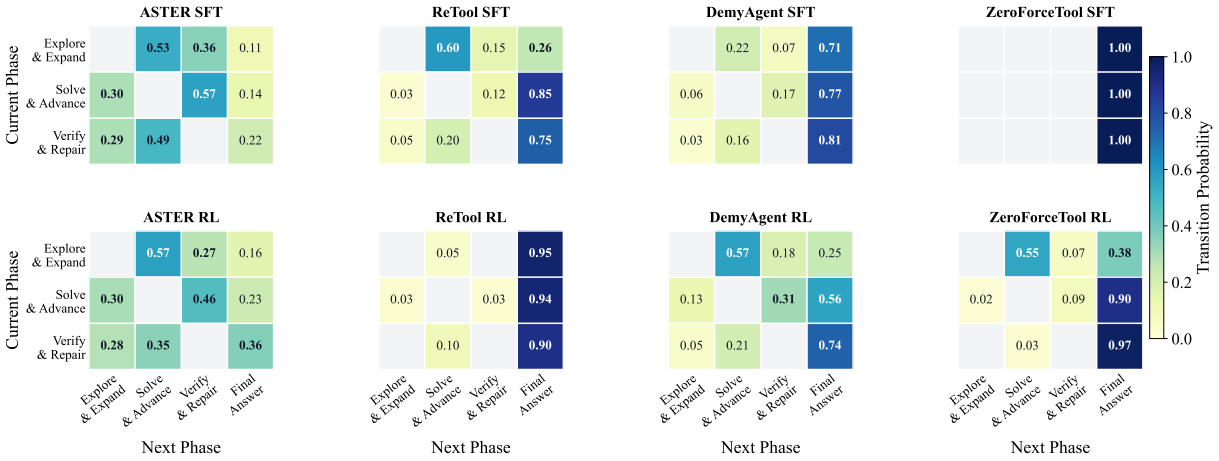


Figure 3: **Phase-transition heatmaps across SFT and RL.** Rows denote the current phase state and columns denote the next phase state; each cell reports the transition probability. *Final Answer* is retained as a destination but omitted as a source row because it is terminal. ASTER preserves distributed transitions across Explore, Solve, and Verify after RL, whereas several baselines concentrate transitions toward Final Answer.

We cold-start Qwen3-4B-Thinking-2507 with each subset and run the same RL pipeline. For this study, we use an 18K context length for both RL training and evaluation. Figure 4a plots AIME2025 accuracy during RL. We observe that High Density achieves the best final accuracy, while RandomDrop25 and Low Density converge to lower performance—despite the latter two yielding *higher* initial accuracy immediately after SFT.

We further corroborate this conclusion with two additional ablations reported in Appendix A.3: a broader stratification of the 45K dataset by tool-call count, and a teacher-model replication using Kimi-K2.5-Thinking trajectories. Both results show the same pattern: higher interaction density induces stronger exploration during RL and leads to better final performance.

> **Takeaway (RQ2): Interaction Density Matters.** Cold-start SFT with interaction-dense trajectories establishes a stronger tool-using behavioral prior, leading to persistently higher training entropy during RL and, in turn, superior final task performance.

3.4 Effect of Interaction Budget of RL Training (RQ3)

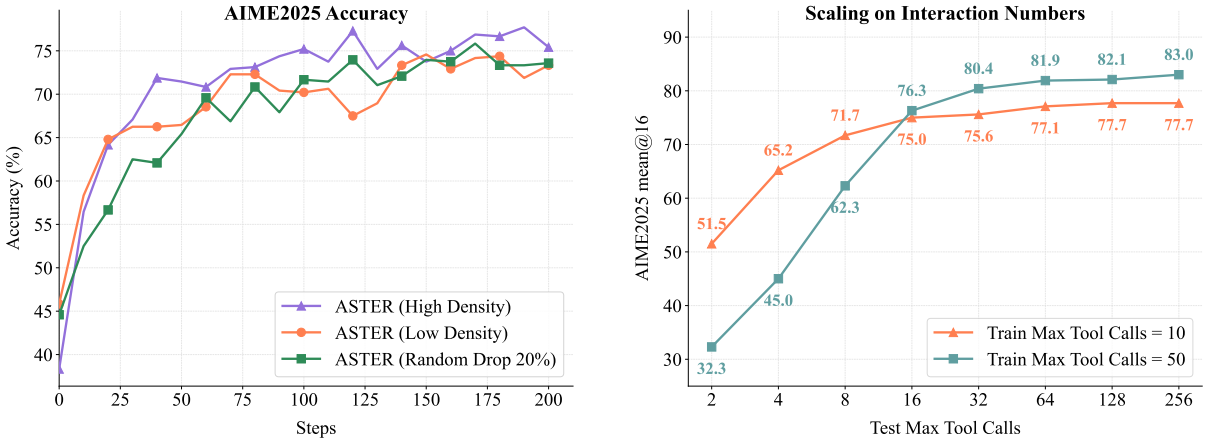
RQ3 examines how the RL interaction budget shapes reasoning behavior and test-time scaling, comparing policies learned under constrained versus generous training-time tool budgets.

We cold-start Qwen3-4B-Thinking-2507 using the 4K high-interaction trajectories introduced in Section 3.3. During RL training, we vary only the *maximum number of tool invocations per trajectory*, setting the budget to either 10 or 50 calls, while keeping all other hyperparameters identical. We systematically vary the *inference-time tool budget* from 2 to 256 calls per trajectory, allowing us to characterize how training-time interaction constraints affect performance scaling across different deployment regimes.

Figure 4b reveals a budget-alignment effect: the scaling curves intersect at intermediate inference budgets, after which the relative advantage reverses. With ample inference interaction, the generous-budget model reaches a higher peak accuracy at 256 tool calls (83.0% vs. 77.7%); under tight inference constraints, the limited-budget model is substantially stronger at 2 tool calls (51.5% vs. 32.3%).

We interpret this budget-alignment effect as consistent with two related factors. First, constrained-budget RL specializes the policy toward short-horizon tool use: even when the test-time budget is increased, the model's average number of tool calls remains nearly flat. Second, simply relaxing the inference budget can push the model into longer interaction chains that were rarely rewarded during training, making coherent long-chain tool use less reliable. Appendix A.4 provides additional tool-call statistics supporting this interpretation.

Overall, these results underscore the importance of aligning training-time and deployment-time interaction budgets in outcome-only RL for tool-augmented reasoning. Mismatch between training and inference constraints can significantly degrade performance, while budget alignment enables mod-



(a) AIME2025 accuracy during RL for cold starts with different interaction densities.

(b) AIME2025 accuracy under varying inference-time tool budgets.

Figure 4: Interaction structure and budget alignment shape downstream RL. Denser cold-start trajectories yield stronger final accuracy despite lower immediate post-SFT accuracy, while the training-time tool budget determines the test-time operating regime.

els to operate closer to their optimal regime.

> **Takeaway (RQ3): Training-Time Interaction Budget Shapes the Test-Time Operating Regime.** Training-time tool budgets do more than set a resource limit; they specialize the learned policy toward a particular interaction regime. Large-budget training enables better long-horizon test-time scaling, while small-budget training yields stronger performance under tight inference constraints but does not reliably exploit additional tools when the budget is relaxed.

4 ASTER Recipe

Building on the insights from RQ1–RQ3, we present a concrete instantiation of ASTER.

Guided by RQ1's teacher-pattern comparison and RQ2's density ablation, we train the final recipe on the 4K curated subset with at least 9 tool calls. Following RQ3, we cap the maximum number of tool calls per training trajectory at 50.

To further improve training stability and efficiency, we adopt a two-stage RL curriculum with an increasing maximum context length: 18K tokens in Stage 1 and 32K tokens in Stage 2. Before transitioning to Stage 2, we filter out prompts that consistently produce correct outcomes across all sampled trajectories in the final epoch, thereby focusing subsequent training on more challenging instances.

To evaluate the effectiveness of ASTER, we present a comparative analysis in Table 1. The results demonstrate that ASTER achieves state-of-the-art performance across all evaluated benchmarks, significantly outperforming both text-only reasoning models and existing agentic reasoning systems. Notably, ASTER-4B with a 90K inference budget achieves 85.8%, 90.0%, 77.1%, and 61.7% on AIME2024, AIME2025, HMMT2025, and BeyondAIME, respectively. When compared to existing agentic reasoning models, ASTER exhibits even more pronounced advantages: ASTER-4B outperforms the best agentic baseline (rStar2-Agent-14B) by 5.2%, 20.2%, and 24.4% on AIME2024, AIME2025, and HMMT2025, respectively. Remarkably, despite being trained from a 4B model, ASTER-4B achieves performance that exceeds much larger models, including Qwen3-235B-A22B-Thinking and OpenAI o3-mini.

We present comprehensive experimental results across two model scales (1.7B, 4B) in Figure 5 to characterize the training dynamics and behavioral patterns of our ASTER framework. Our analysis focuses on three key empirical observations that emerge during the training process.

Cold-Start Performance Recovery Following cold-start supervised fine-tuning, we observe a marked degradation in model performance on mathematical reasoning benchmarks, with accuracy *substantially falling below* the baseline performance of the pre-SFT base models. However, as reinforcement learning training progresses, model accuracy gradually recovers to match and subsequently exceed the pre-SFT baseline levels.

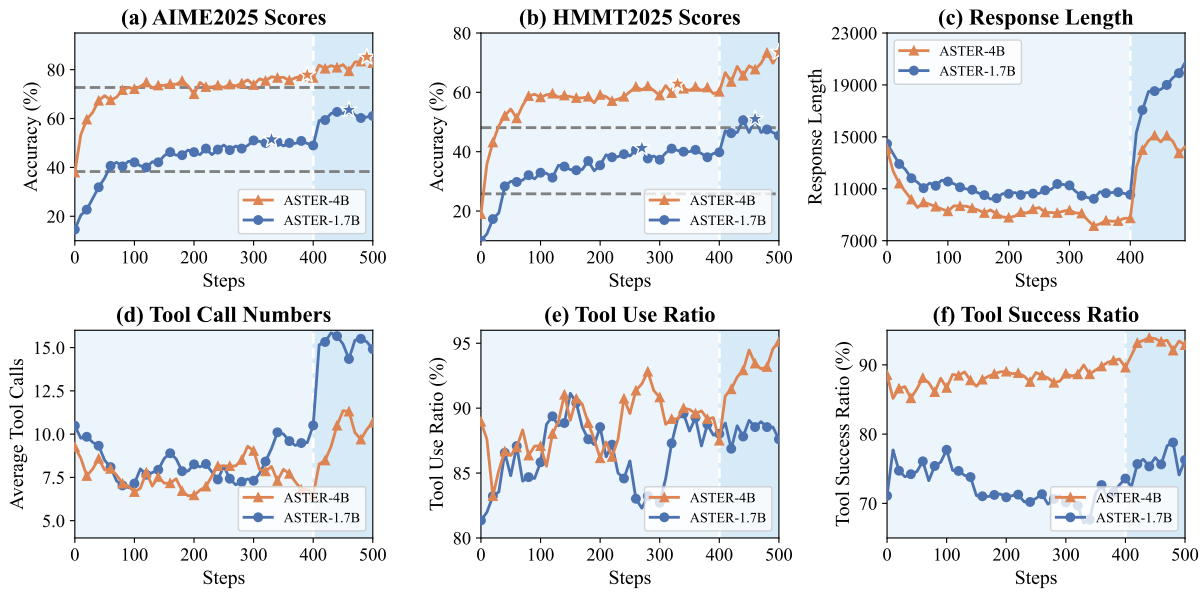


Figure 5: Training dynamics of ASTER during two-stage RL. As training progresses, ASTER gradually recovers to and then surpasses its pre-SFT performance on AIME2025 and HMMT2025. The light and dark blue shaded backgrounds denote the two RL stages: Stage 1 uses an 18K context window, while Stage 2 switches to a 32K context window and focuses on harder remaining prompts. The white dashed vertical line marks the transition between the two stages. In panels (a) and (b), gray dashed horizontal lines indicate the corresponding pre-SFT performance baselines, and colored stars mark the best checkpoints within each stage.

RL Training Dynamics Figure 5 shows a two-phase refinement pattern. In Stage 1, the average number of tool invocations decreases and then stabilizes, while accuracy recovers. In light of RQ1, we interpret this reduction as efficiency refinement rather than interaction collapse: RL prunes redundant tool calls while preserving the structured explore–solve–verify prior induced by the high-interaction cold start. After switching to Stage 2, with a longer context window and harder remaining prompts, response length, tool-use frequency, and benchmark accuracy increase together. This suggests that Stage 1 mainly improves tool-use efficiency by removing redundant calls, whereas Stage 2 allows the model to spend additional context and tool calls on harder problems that benefit from longer reasoning trajectories.

Model Scale Efficiency Differences The same recipe also exposes a scale-dependent efficiency pattern. The 1.7B model relies on more tool invocations and longer responses, suggesting that smaller models compensate for weaker internal reasoning with more external computation. By contrast, the 4B model reaches stronger performance with shorter trajectories and fewer tool calls, indicating that additional model capacity improves not only accuracy but also the selectivity of tool use. Thus, larger models requiring less external computation to achieve stronger results.

5 Conclusion

We present ASTER, a cold-start and RL framework that mitigates *interaction collapse* in tool-integrated reasoning. We find that cold-start SFT matters less through immediate accuracy than through the behavioral prior it induces for downstream RL. Interaction-dense demonstrations help select trajectories that sustain exploration, solution advancement, and verification rather than collapsing into short solve-to-final chains.

RQ1–RQ3 show how this prior interacts with RL. Structured cold starts preserve multi-stage tool use, interaction density strengthens the learned policy, and training-time tool budgets shape test-time regimes. These findings yield ASTER's recipe: a 4K high-interaction cold start followed by two-stage RL for longer-horizon reasoning.

ASTER-4B reaches 85.0% on AIME 2025 and 73.3% on HMMT 2025 under a 30K inference budget, and scales to 90.0% and 77.1% under 90K. It outperforms substantially larger text-only and agentic baselines, including Qwen3-235B-A22B-Thinking and rStar2-Agent-14B. Together, these results suggest that agentic scaling depends not only on model size or RL compute, but also on pre-RL trajectory structure.

6 Limitations

ASTER is evaluated primarily on competitive mathematical reasoning tasks with integer-answer supervision and Python-based tool use, so its effectiveness in broader domains, richer tool ecosystems, or settings with noisier rewards remains to be established. Our strongest results are obtained on Qwen3-based 1.7B and 4B backbones with a curated high-interaction cold-start dataset and substantial RL training, which means the exact gains may vary for other model families, data-construction pipelines, or tighter compute budgets. Finally, although we analyze agentic behavior through tool-use statistics, phase annotations, and a rubric-based judge, these analyses still depend on model-based annotation. More comprehensive human validation and reliability/safety evaluations for real-world tool execution are needed.

7 Ethical Considerations

This paper studies how to scale tool-integrated reasoning with RL and proposes ASTER to mitigate interaction collapse, enabling more reliable multi-step tool use and long-horizon problem solving. If adopted responsibly, these capabilities may improve the efficiency and accessibility of complex reasoning workflows such as mathematical problem solving and research assistance, potentially reducing reliance on very large models. At the same time, stronger tool-using agents can amplify misuse risks, including automating harmful tasks or interacting unsafely with external code-execution environments. We therefore emphasize that any release or deployment should rely on sandboxed tools, least-privilege access, usage monitoring, and human oversight, and we encourage future work on stronger safety evaluation protocols for tool-integrated RL agents.

References

Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. 2025. OpenCodeReasoning-II: A Simple Test Time Scaling Approach via Self-Critique. *arXiv preprint arXiv:2507.09075*.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and 1 others. 2025. Polaris: A Post-Training Recipe for Scaling Reinforcement Learning on Advanced Reasoning Models. *URL https://hkunlp.github.io/blog/2025/Polaris*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. ReTool: Reinforcement Learning for Strategic Tool Use in LLMs. *arXiv preprint arXiv:2504.11536*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, and 1 others. 2025. Skywork Open Reasoner 1 Technical Report. *arXiv preprint arXiv:2505.22312*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720*.

Heng Lin and Zhongwen Xu. 2025. Understanding Tool-Integrated Reasoning. *arXiv preprint arXiv:2508.19201*.

Xinji Mai, Haotian Xu, Zhong-Zhi Li, Xing W, Weinong Wang, Jian Hu, Yingying Zhang, and Wenqiang Zhang. 2025. Agent RL Scaling Law: Agent RL with Spontaneous Code Execution for Mathematical Problem Solving. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. 2025. AIMO-2 Winning Solution: Building State-of-the-Art Mathematical Reasoning Models with OpenMathReasoning Dataset. *arXiv preprint arXiv:2504.16891*.

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, and 1 others. 2025. Seed1.5-Thinking: Advancing Superb Reasoning Models with Reinforcement Learning. *arXiv preprint arXiv:2504.13914*.

Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, Ying Xin, Ziming Miao, Scarlett Li, Fan Yang, and Mao Yang. 2025. rStar2-Agent: Agentic Reasoning Technical Report. *ArXiv preprint arXiv:2508.20722*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan

Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning. *arXiv preprint arXiv:2504.20073*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. 2025. SimpleTIR: End-to-End Reinforcement Learning for Multi-Turn Tool-Integrated Reasoning. In *International Conference on Learning Representations (ICLR)*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying Long Chain-of-Thought Reasoning in LLMs. *arXiv preprint arXiv:2502.03373*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025a. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. 2025b. Demystifying Reinforcement Learning in Agentic Reasoning. *arXiv preprint arXiv:2510.11701*.

Charlie Zhang, Graham Neubig, and Xiang Yue. 2025. On the Interplay of Pre-Training, Mid-Training, and RL on Reasoning Language Models. *arXiv preprint arXiv:2512.07783*.

A Experimental Details

A.1 Full Experimental Setup

Training Data We construct our SFT dataset from two primary sources: (i) Skywork-OR1-RL-Data from Skywork-OR1[3] (He et al., 2025), and (ii) 93K problems from the Art of Problem Solving (AoPS) forums collected by OpenMathReasoning (Moshkov et al., 2025). We retain only English problems whose ground-truth final answers are positive integers. We then generate tool-augmented solutions using GPT-OSS-20B-high and retain only instances with both a correct final answer and a 100% tool execution success rate, yielding an initial dataset of 45K trajectories. For RL training, we employ DAPO-Math-17K dataset[4], which contains 17K prompts paired with integer ground-truth answers.

Base Models We use Qwen3-1.7B-Thinking and Qwen3-4B-Thinking-2507 as our base models.

Cold-Start SFT We employ `LLaMA-Factory`[5] for multi-turn instruction tuning. Training is conducted for 6 epochs with a global batch size of 128 and a maximum context length of 32K tokens. We adopt a constant learning rate: 3×10^{-5} for Qwen3-1.7B/4B.

Reinforcement Learning Our framework inherits VeRL[6] as a submodule, ensuring compatibility with upstream updates. We perform RL training using GRPO. Following (Yu et al., 2025a), we optimize only the token-level policy gradient objective with Clip-Higher ($\epsilon_h = 0.28$) and do not include a KL divergence penalty. For data collection, we use a rollout prompt batch size of 128 and sample $G = 8$ trajectories per prompt to form the group baseline. We employ gradient accumulation with a mini-batch size of 64, yielding exactly two parameter updates per rollout step. Unless otherwise specified, step refers to a *rollout step*. We use a unified conversation template throughout our experiments. To support long-horizon tool use and iterative self-correction, we set the maximum allowed tool invocations to 50 per trajectory. Key RL hyperparameters are summarized in Table 2.

[3] Skywork-OR1 dataset
[4] DAPO-Math-17K dataset
[5] GitHub: hiyouga/LlamaFactory
[6] GitHub: volcengine/verl

Table 2: Key hyperparameters used with `verl` for RL training.

Group	Name	Value
Data	train_batch_size	128
	max_prompt_length	2048
	max_response_length	16K/30K
Actor	lr	1e-6
	ppo_mini_batch_size	64
	use_kl_loss	False
	clip_ratio_low	0.20
	clip_ratio_high	0.28
	entropy_coeff	0
Rollout	n (group size per prompt)	8
	max_assistant_turns	50

In RQ1, for the ablation on different teacher-pattern cold-start strategies, we keep the context length fixed at 32K for both training and evaluation, and hold all other hyperparameters constant.

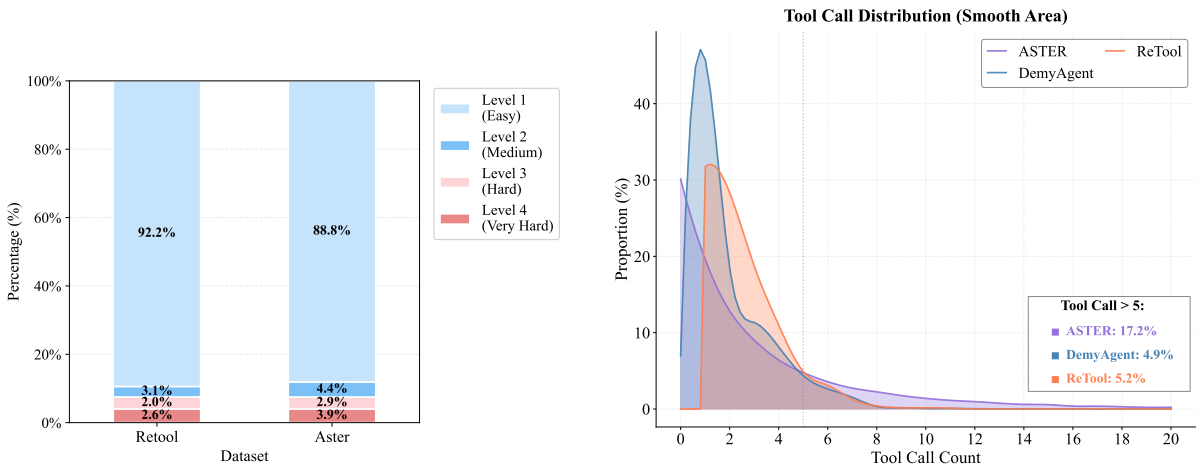
In RQ2, for the ablation on interaction density, we set the context length to 18K for both RL training and evaluation.

In RQ3, models are trained with an 18K context length, while the maximum number of tool calls is capped at 10 and 50, respectively. During evaluation, we extend the context length to 32K to assess generalization.

For the final recipe experiments, we adopt a two-stage training schedule, setting `max_response_length` to 16,384 in Stage 1 and 30,720 in Stage 2. We disable KL regularization to avoid overly constraining long-horizon exploration, especially in the early phase of RL.

Evaluation Protocol We evaluate mathematical reasoning proficiency on four competitive benchmarks: AIME2024, AIME2025, HMMT2025 (February), and BeyondAIME (Seed et al., 2025). Unless otherwise specified, we use temperature 0.6 and top_p $= 0.95$ for all evaluations. All other evaluation parameters follow the recommended settings from the respective baseline papers. By default, we use a maximum context length of 32K tokens for evaluation.

Baselines We compare our models against several state-of-the-art baselines, including OpenReasoning-Nemotron-7B (Ahmad et al., 2025), Qwen3-235B-A22B-Thinking (Yang et al., 2025), POLARIS-4B-Preview (An et al., 2025), ReTool-32B (Feng et al., 2025), as well as rStar2-Agent-14B (Shang et al., 2025) and DemyAgent-4B (Yu et al., 2025b).



(a) Problem difficulty distribution for the ReTool and ASTER datasets. Difficulty is defined by per-problem accuracy: Easy (≥ 0.75), Medium ($[0.50, 0.75)$), Hard ($[0.25, 0.50)$), and Very Hard (< 0.25).

(b) Tool-call count distribution across different cold-start datasets. ReTool and DemyAgent are biased toward sparse interactions, while ASTER contains more long-horizon trajectories.

Figure 6: Cold-start dataset diagnostics under matched modeling assumptions: comparable problem difficulty (a) and contrasting interaction-length structure (b).

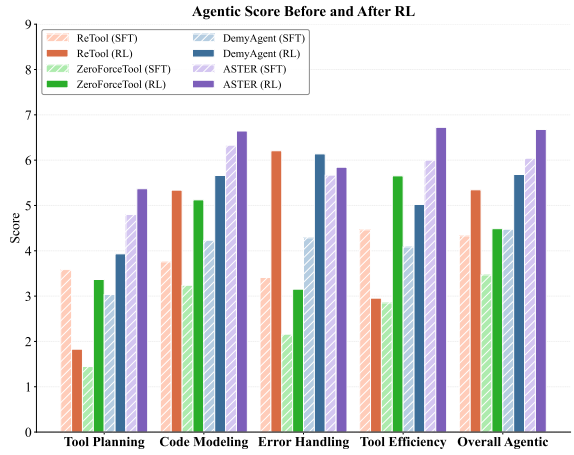


Figure 7: GPT-5 agentic judge scores over planning, code modeling, error handling, tool efficiency, and overall agentic capability.

A.2 Additional RQ1 Analyses: Cold-Start Priors

We organize the RQ1 appendix around the mechanism tested in the main text: comparable problem difficulty, different interaction structure, and the resulting changes in RL behavior and phase-level trajectory organization.

A.2.1 Dataset Controls and Trajectory Diagnostics

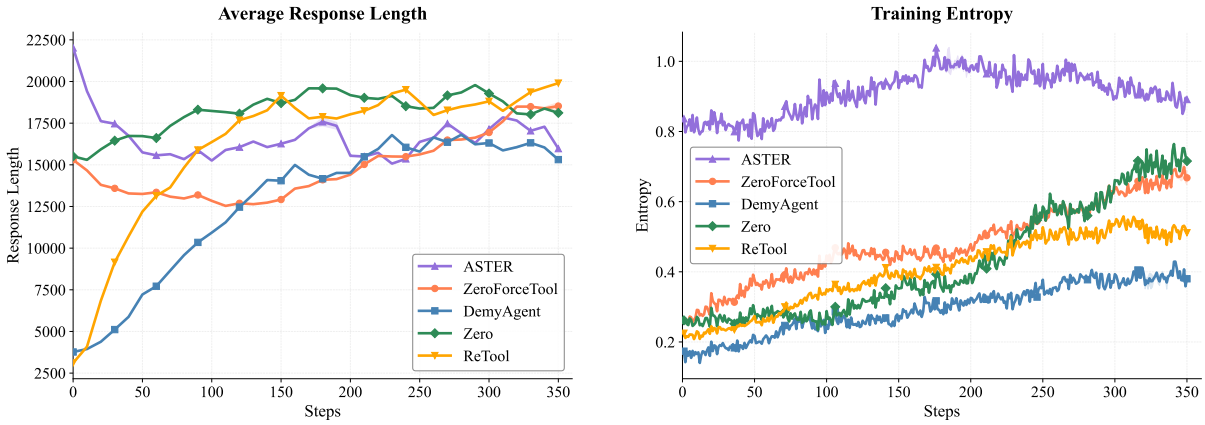
We first test whether the RQ1 comparison is confounded by dataset difficulty. For each problem in the ASTER SFT and ReTool SFT datasets, we sample eight responses from Qwen3-4B-Thinking-2507, define difficulty by per-problem accuracy, and stratify problems into four bins with 0.25-wide accuracy intervals. Figure 6a shows similar easy-problem fractions for ASTER and ReTool (92.2% vs. 88.8%), indicating comparable difficulty under the same model and decoding configuration.

With difficulty controlled, trajectory structure becomes the key distinction. Figure 6b shows that ReTool and DemyAgent are concentrated in sparse-interaction regimes, whereas ASTER contains substantially more long-horizon trajectories.

A.2.2 Training Dynamics, Behavioral Diagnostics, and Compute Controls

Having isolated interaction structure at the data level, we next ask whether the resulting RL policies differ in agentic quality, not only in raw tool-call counts. We first evaluate sampled trajectories with a GPT-5 agentic judge using the guidance in Table 3. Figure 7 shows that ASTER scores higher than the baselines on planning, code modeling, tool efficiency, and overall agentic quality, indicating that its extra tool use is better organized rather than merely more frequent. This diagnostic is intended to separate interaction quantity from interaction quality: a policy can call tools often while still using them in a reactive or poorly coordinated way. ASTER's advantage is therefore not only that it sustains tool interaction, but that the interaction remains tied to planning, computation, and verification.

We then examine the corresponding training dy-



(a) Response length on the validation set.

(b) Training entropy over RL training steps.

Figure 8: Training dynamics under different teacher-pattern cold-start strategies. (a) Response length on the validation set. (b) Training entropy over RL training steps.

Table 3: Guidance used by the GPT-5 agentic judge. Each dimension is scored from weak to strong behavior; the overall score is judged holistically and is not the arithmetic mean of the other dimensions.

Dimension	Scoring focus
Tool planning	Whether tool use is deliberate, sequenced, and tied to problem-solving goals rather than reactive or accidental.
Computational modeling	Whether code reflects the mathematical structure of the problem, using suitable abstractions or algorithms instead of calculator-style snippets.
Error diagnosis	Whether the model identifies root causes, tests hypotheses, and validates repairs rather than repeating errors or only fixing syntax.
Tool-use efficiency	Whether tools are invoked at appropriate moments for non-trivial operations while avoiding unnecessary calls.
Overall quality	Whether the full trajectory coherently coordinates planning, modeling, debugging, verification, and final answering.

Table 4: Rollout cost comparison across RQ1 methods. Average response length and wall-clock time per rollout step are computed over the last 50 RL steps under the same 30K response budget and 50-round tool-call cap.

Method	Avg. Length (tokens)	Avg. Time/Step (s)
ASTER	16,048	988
DemyAgent	16,338	1,012
ZeroForceTool	17,549	969
ZeroTIR	18,089	1,013
ReTool	20,301	1,017

Table 5: Pre-RL Pass@K exploration analysis on HMMT2025. We sample 64 trajectories per problem from each initialization before RL training.

Cold Start	Pass@1	Pass@16	Pass@64
ZeroTIR	39.43%	60.59%	66.67%
ASTER	36.20%	64.91%	70.00%

namics. Figure 8 reports response length and training entropy for the same cold-start priors. ASTER sustains the highest training entropy throughout RL, consistent with a prior that preserves exploration instead of collapsing early into finalization-heavy behavior.

These behavioral differences are not a byproduct of larger rollout budgets. All methods use the same 30K response budget and 50-round tool-call cap; the 90K inference setting in Table 1 is used only to probe ASTER's scaling ceiling, not for the controlled RQ1 comparison. As shown in Table 4, ASTER does not spend more rollout compute than the baselines: its average response length and wall-clock time per rollout step are comparable to, and in some cases lower than, the alternatives.

Pre-RL Exploration. We also test whether the high-entropy prior reflects useful exploration rather than noisier generation. Before RL, we sample 64 trajectories per HMMT2025 problem from each initialization; for the zero-start baseline, this is simply the base model. Table 5 shows that ASTER has slightly lower Pass@1 than the zero-start model but higher Pass@16 and Pass@64. Thus, ASTER initializes a policy with broader support over multi-turn solution paths, giving RL more useful behavior to refine.

Table 6: Phase definitions used in the transition analysis. The judge emits a richer code-purpose label set, but the transition matrix retains these three tool-purpose phases plus terminal final-answer turns.

Phase	Definition
Explore & Expand	Code/tool use to broaden the search space: enumerate cases, simulate, inspect patterns, or probe structure before fixing a solution path.
Solve & Advance	Code/tool use to advance a chosen derivation: compute values, solve equations, simplify expressions, or execute subproblems.
Verify & Repair	Code/tool use to validate answers, check constraints, debug errors, retry failed outputs, or recover from inconsistent paths.
Final Answer	Commitment to the final answer. Retained as a transition destination but omitted as a source row because it is terminal.

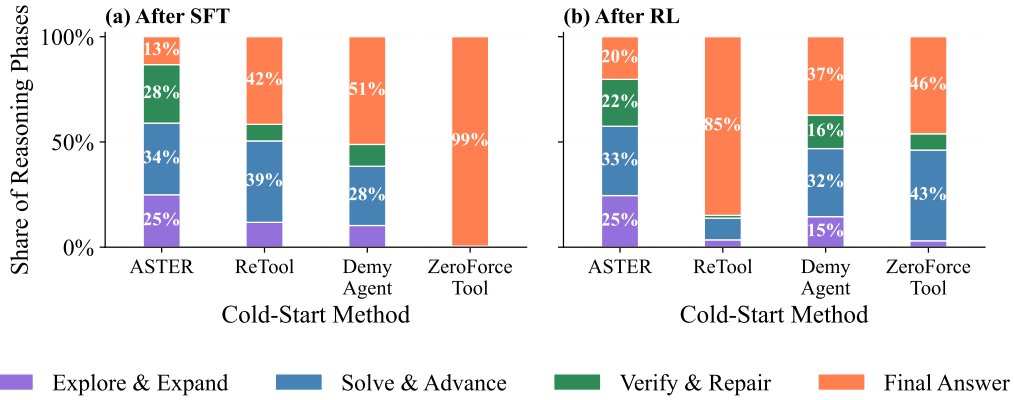


Figure 9: Phase composition of reasoning chains after SFT and RL. Each bar sums to 100% and shows the share of phase segments assigned to *Explore & Expand*, *Solve & Advance*, *Verify & Repair*, and *Final Answer*.

Table 7: Summary statistics for the phase-transition analysis. Destination-Final is the share of phase transitions whose destination is *Final Answer*. Entropy is the normalized entropy of the transition-destination distribution. In the dominant motifs, E, S, V, and F denote *Explore & Expand*, *Solve & Advance*, *Verify & Repair*, and *Final Answer*, respectively.

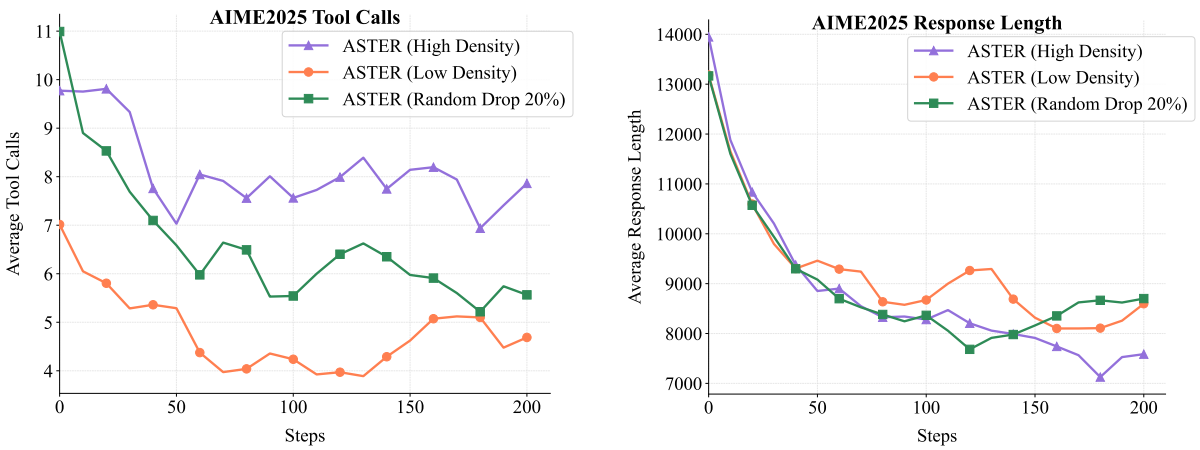
Run	Tool Turns	Phases	Destination-Final	Entropy	Dominant Motif
ASTER-SFT	14.51	5.72	15.7	0.972	$S \to V \to F$: 7.4%
ReTool-SFT	2.53	2.33	71.6	0.617	$S \to F$: 54.1%
DemyAgent-SFT	1.22	1.87	76.6	0.564	$S \to F$: 29.0%
ZeroForceTool-SFT	0.11	1.00	100.0	0.000	F: 80.8%
ASTER-RL	10.25	4.39	24.7	0.995	$S \to F$: 12.1%
ReTool-RL	0.14	1.15	93.9	0.213	F: 58.8%
DemyAgent-RL	3.43	2.36	53.8	0.841	$S \to F$: 26.6%
ZeroForceTool-RL	1.56	2.11	87.7	0.353	$S \to F$: 71.7%

A.2.3 Phase-Transition Analysis Details

Finally, phase-transition analysis connects the aggregate training behavior to the internal organization of each trajectory. Figure 9 complements the transition heatmaps in Figure 3 by reporting the aggregate share of each phase in the reasoning chain.

We use GPT-OSS-120B annotations only for analysis; they are not used as rewards, filters, or training signals. All SFT and RL runs in Figure 3 are annotated with the same fixed turn-level protocol. For each parsed turn, the judge assigns one `code_intent` label: the three tool-purpose labels in Table 6, `final_answer`, `non_tool_reasoning`, or `format_other`.

The judge receives the problem, trajectory metadata, and one parsed turn at a time, and returns strict XML containing the assigned `code_intent`. We omit `non_tool_reasoning` and `format_other` from the transition matrix, so the reported transitions focus on tool-use purpose and finalization rather than generic natural-language reasoning turns.



(a) Tool-call count over training steps.

(b) Response length over training steps.

Figure 10: Training dynamics for the controlled RQ2 interaction-density ablation.

Table 8: Length and interaction statistics of the OSS-generated High Density and Low Density 4K subsets used in RQ2. Character-length statistics are computed over full trajectories.

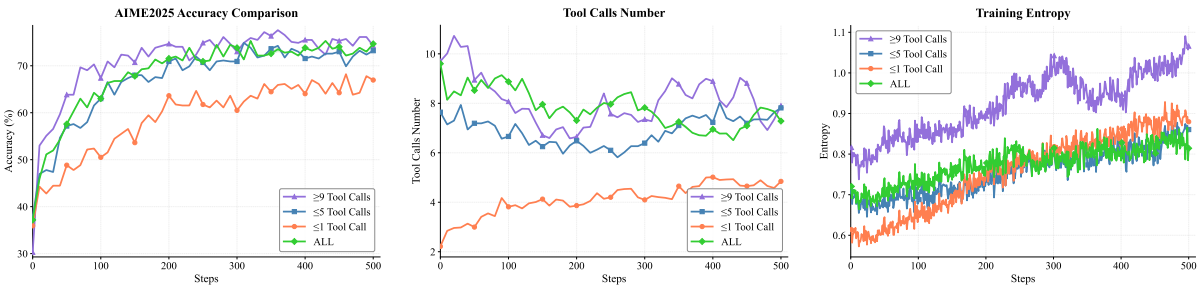
Statistic	High Density	Low Density
Tool rounds	7–20	0–4
Mean Tool rounds	10.86	1.68
Character length mean	42,564	42,700
Character length median	36,204	36,078
Character length std.	18,346	18,726

Transition Aggregation. For each annotated trajectory, we build a four-phase sequence by retaining labels in {`explore_expand`, `solve_advance`, `verify_repair`, `final_answer`}. If a final-answer label appears, the sequence is truncated after its first occurrence. We then merge consecutive identical labels into one phase segment and count transitions between adjacent phase segments. The heatmaps report row-normalized transition probabilities, while Table 7 additionally reports average tool turns, average phase segments, the share of transitions whose destination is *Final Answer*, normalized destination entropy, and the dominant phase motif for each run.

A.3 Additional RQ2 Experimental Results

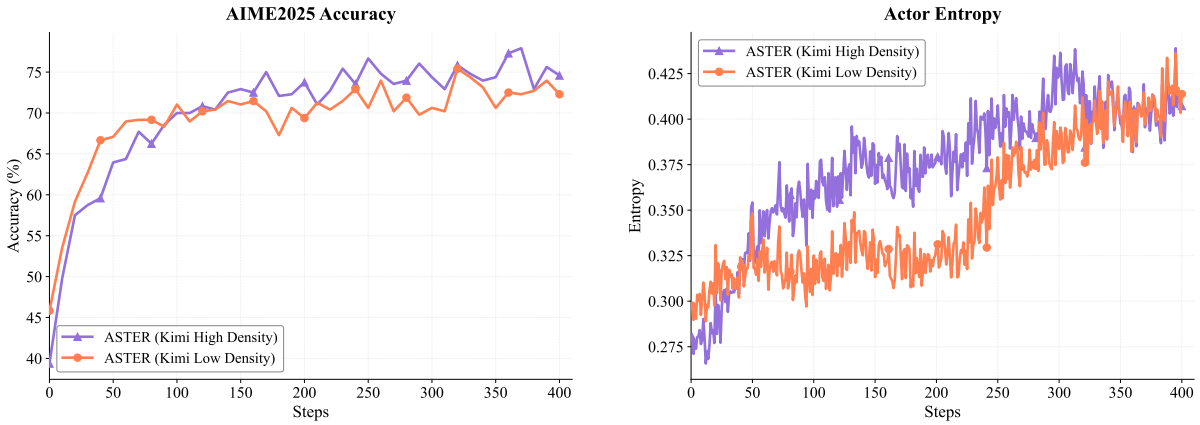
To further validate the role of interaction density in cold-start SFT, we provide three complementary analyses. First, we examine the behavioral dynamics of the controlled density ablation in Section 3.3, where High Density, Low Density, and Random-Drop25 are trained under the same RL pipeline. To rule out trajectory length as the primary confounder, we construct the High Density and Low Density 4K subsets with closely matched character-length distributions. As shown in Table 8, the two subsets have nearly identical mean, median, and standard deviation of character length, while differing substantially in tool-interaction rounds. Thus, the comparison primarily isolates interaction density rather than overall response length. Figure 10 complements the AIME2025 accuracy results in Figure 4a by reporting tool-call count and response length during RL training. The high-density initialization preserves a stronger tendency to invoke tools, while response length decreases as training progresses, suggesting that RL does not simply reward longer traces; instead, it converts the dense cold-start prior into more selective and efficient tool use.

Second, we perform a broader stratification of the 45K trajectories described in Section 3.1 by their number of tool invocations. We construct three subsets with at most 1 tool call (22K trajectories), at most 5 tool calls (37K trajectories), and at least 9 tool calls (4K trajectories), and compare them with the full dataset under the same 18K-context RL setup. Figure 11 shows a consistent ordering: the ≥ 9 subset achieves the strongest final AIME2025 accuracy, followed by the full dataset and then the lower-density subsets. This ordering is mirrored by the behavioral metrics: higher-density cold starts induce persistently higher tool-call counts and maintain higher training entropy, indicating stronger exploration and less premature collapse.



(a) AIME2025 accuracy. (b) Tool-call count. (c) Training entropy.

Figure 11: Training dynamics under cold starts from subsets stratified by tool-call count. From left to right: AIME2025 accuracy, tool-call count, and training entropy over RL training steps.



(a) AIME2025 accuracy. (b) Training entropy.

Figure 12: Training dynamics under Kimi-K2.5-Thinking cold-start trajectories. From left to right: AIME2025 accuracy and training entropy over RL training steps.

These results support the view that interaction density is not only correlated with final accuracy, but directly shapes the policy prior that RL subsequently amplifies.

Finally, we test whether the same conclusion holds when the cold-start trajectories are produced by a different teacher model, Kimi-K2.5-Thinking. Figure 12 shows that the higher-density initialization again yields stronger RL improvement and sustains higher training entropy than its lower-density counterpart. This teacher-model replication suggests that the effect is not an artifact of GPT-OSS-20B trajectory style; rather, interaction-dense demonstrations provide a robust behavioral prior that improves downstream RL exploration.

A.4 Additional RQ3 Analysis: Budget Mismatch

To better understand why mismatched training-time and test-time tool budgets hurt performance, we measure the realized number of tool calls made by the budget-10 and budget-50 models under different inference-time budgets. Table 10 reports both the mean and maximum tool-call counts over the evaluation set.

These statistics suggest that the degradation under budget mismatch is not explained by a single factor. First, there is clear evidence of policy specialization: the budget-10 model's mean tool calls plateau at around 3.5–3.7 even when the test-time budget increases to 256, whereas the budget-50 model scales to around 7.6 calls on average. This indicates that constrained-budget RL induces a short-horizon interaction policy rather than a policy that can smoothly exploit additional tools at inference time. Second, there is also a distribution-shift component: the budget-10 model occasionally produces much longer trajectories, with maximum tool-call counts reaching 28–32 under large test budgets, but these long chains lie outside the interaction regime consistently rewarded during training. As a result, simply relaxing the inference-time budget is insufficient unless the model has learned to maintain coherent long-chain tool use.

Table 9: Search-domain cold-start diagnostic. Accuracy is reported as mean@1 percentage on selected validation domains. Tool-call statistics are estimated from the logged interaction-turn metric by dividing by two, since each tool invocation contributes one model-side and one tool-side interaction event. Validation caps tool calls at 30 per trajectory.

Method	NQ	PopQA	HotpotQA	MuSiQue	Mean Tool Calls	Max Tool Calls
Zero-RL	28.82	31.80	26.49	9.60	2.67	7
Kimi-SFT+RL	28.03	32.97	28.92	12.49	14.38	30
Δ	-0.79	+1.17	+2.43	+2.89	+11.71	+23

Table 10: Realized tool-call counts under different inference-time budgets. The budget-10 model remains concentrated around short interaction chains even when more tools are available, while the budget-50 model makes more effective use of larger inference budgets.

Test Budget	B10 Mean	B10 Max	B50 Mean	B50 Max
2	1.4	2	1.7	2
4	2.3	4	3.1	4
8	3.1	8	4.7	8
16	3.5	16	6.5	16
32	3.7	26	7.1	32
64	3.5	28	7.6	64
128	3.6	32	7.3	53
256	3.7	28	7.6	55

A.5 Search-Domain Cold-Start Diagnostic

We further conduct a diagnostic study in a search-based QA setting to examine whether cold-start SFT changes the interaction policy beyond the mathematical reasoning domain. The training prompts are randomly sampled from `PeterJinGo/nq_hotpotqa_train`, yielding 21,201 training examples. We compare two search-RL initializations: **Zero-RL**, which starts RL directly without supervised cold start, and **Kimi-SFT+RL**, which first applies cold-start SFT on search trajectories synthesized by Kimi-K2.5-Thinking and then uses the same RL pipeline. For both methods, RL uses a 4K prompt budget and a 16K maximum response length. During training, we allow up to 20 tool calls per trajectory; during validation, we increase the cap to 30 tool calls to test whether the learned policy can exploit a larger search budget.

Table 9 reports validation performance on selected search-domain benchmarks, together with the estimated number of tool calls. The selected domains cover single-hop factual QA (NQ and PopQA) as well as multi-hop or compositional QA (HotpotQA and MuSiQue). The Kimi-based cold start improves several domains, especially HotpotQA and MuSiQue, while remaining comparable on NQ. More importantly for the cold-start question, it substantially changes the learned interaction regime: the average number of estimated tool calls increases from 2.67 to 14.38, and the maximum observed tool-call count increases from 7 to 30. This suggests that supervised search trajectories can induce a much longer-horizon search prior before RL, although its accuracy benefit is domain-dependent rather than uniformly positive.

B Example Output

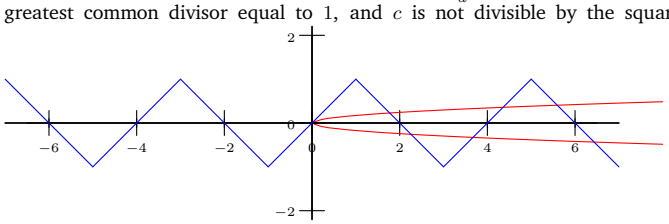
Problem statement of AIME25 for the case study.

Example Case Study: AIME Problem

Question:
A piecewise linear function is defined by

$$f(x) = \begin{cases} x & \text{if } -1 \le x < 1 \\ 2 - x & \text{if } 1 \le x < 3 \end{cases}$$

and $f(x+4) = f(x)$ for all real numbers x. The graph of $f(x)$ has the sawtooth pattern depicted below. The parabola $x = 34y^2$ intersects the graph of $f(x)$ at finitely many points. The sum of the y-coordinates of all these intersection points can be expressed in the form $\frac{a+b\sqrt{c}}{d}$, where a, b, c, and d are positive integers such that a, b, d have greatest common divisor equal to 1, and c is not divisible by the square of any prime. Find $a+b+c+d$. Graph



Labeled Answer:
$\boxed{259}$

Figure 13: AIME2025-I problem 11.

Trajectory of ASTER-4B on AIME 2025. This is Part I of the 48-turn interaction.

Model Output by ASTER-4B (Turn 1) [Initial analysis]

(Handling each periodic line segment separately and solve the resulting quadratics on each segment)
...We need to treat each segment separately for $n \geq 0$. But also consider that for n large, constraints may fail; but we need to find until constraints fail.But we must also consider n negative? We already determined $n \geq 0$. But just to confirm, let's test n negative for segment A with n=-1 etc. No real solutions. So only $n \geq 0$.Now we can brute force for n from 0 to say 10, compute solutions and check constraints. Let's do that using Python.

```
import sympy as sp
n = sp.symbols('n', integer=True, nonnegative=True)
solutions = []
# Let's compute for n from 0 to maybe 10
for n_val in range(0, 10):
    # Segment A
    y1 = (1 + sp.sqrt(1 + 544*n_val)) / 68
    y2 = (1 - sp.sqrt(1 + 544*n_val)) / 68
    for y in [y1, y2]:
        x_val = 34 * y**2
        # check interval [4n-1, 4n+1)
        okA = (sp.S(4*n_val - 1) <= x_val) and (x_val < sp.S(4*n_val + 1))
        if okA:
            solutions.append((n_val, 'A', sp.simplify(y), sp.simplify(x_val)))
    # Segment B
    D = 544*n_val + 273  # numeric for each n_val
    y3 = (-1 + sp.sqrt(D)) / 68
    y4 = (-1 - sp.sqrt(D)) / 68
    for y in [y3, y4]:
        x_val = 34 * y**2
        okB = (sp.S(4*n_val + 1) <= x_val) and (x_val < sp.S(4*n_val + 3))
        if okB:
            solutions.append((n_val, 'B', sp.simplify(y), sp.simplify(x_val)))
solutions[:10]
```

Execution Result: (0, 'A', 1/34, 1/34), (0, 'A', 0, 0), ...

Model Output by ASTER-4B (Turn 8) [Identifying the Finite Range]

The model hypothesizes that while the parabola extends infinitely, intersections only happen for a limited range of periods n. It writes code to sweep n until no solutions are found.

```
max_n=0
for n_val in range(0,100):
found=False
# Check Segment A and B for existence of valid solutions
# ... (code calculating y1, y2, y3, y4 and checking intervals) ...
if found:
    max_n=n_val
max_n
```

Execution Result: 8

4

Trajectory of ASTER-4B on AIME 2025. This is Part II of the 48-turn interaction.

Model Output by ASTER-4B (Turn 14) [Symbolic Summation]

Having established that intersections only occur for $n \in [0, 8]$, the model proceeds to calculate the exact symbolic sum of all y-coordinates.

```
sols=[]
for n_val in range(0,9):
# Segment A calculations
y1 = (1 + sp.sqrt(1+544*n_val))/68
y2 = (1 - sp.sqrt(1+544*n_val))/68
for y in [y1,y2]:
# ... check constraints ...
    sols.append(sp.simplify(y))
# Segment B calculations
# ... check constraints ...
sols
```

Execution Result: [1/34, 0, -1/68 + sqrt(273)/68, ..., -1/68 + 5*sqrt(185)/68]

Model Output by ASTER-4B (Turn 15) [Initial Result]

```
sum_y = sum(sols)
sum_y
```

Execution Result: 1/68 + 5*sqrt(185)/68

The model has theoretically found the correct sum here, but subsequently enters a long verification phase where it attempts to check each solution numerically.

Model Output by ASTER-4B (Turn 40 & 41) [The Pivot: Self-Correction]

```
# Manually checking the solution (2, 'A', -8/17)

n_val, seg, y_sym, x_sym = (2, 'A', -8/sp.Integer(17), 128/sp.Integer(17))
y_val=float(y_sym.evalf())
x_val=float(x_sym.evalf())
y_val, x_val
```

Execution Result: (-0.470588..., 7.529411...)

Analysis: Now numeric_check: $n = 2$, seg='A', x in $[4(2) - 1, 4(2) + 1) = [7, 9)$. Indeed $x = 7.529$ is within $[7, 9)$. So constraints are satisfied.
But earlier numeric check flagged None because we used `floor(x/4)`... `floor(7.529/4)=1`. That gave $n = 1$ incorrectly. The solution belongs to $n = 2$.
Conclusion: My earlier numeric check had a bug. All 35 solutions are indeed valid.

5

Trajectory of ASTER-4B on AIME 2025. This is Part III of the 48-turn interaction.

Model Output by ASTER-4B (Turn 42) [Re-Confirmation]

With the verification bug resolved, the model trusts the original set of 35 solutions.

```
sum_y = sum([y_sym for *,*,y_sym,_ in solutions])
sum_y
```

Execution Result: 1/68 + 5*sqrt(185)/68

Model Output by ASTER-4B (Turn 48 - Final)

The sum of the y-coordinates is $\frac{1+5\sqrt{185}}{68}$. Matching the form $\frac{a+b\sqrt{c}}{d}$:

- $a = 1$
- $b = 5$
- $c = 185$ (square-free, 5×37)
- $d = 68$

Checking gcd: $\gcd(1, 5, 68) = 1$. The conditions are satisfied.
Calculate final value: $a + b + c + d = 1 + 5 + 185 + 68 = 259$.
The final answer is $\boxed{259}$.

6

Trajectory of Retool-4B on AIME 2025. This is Part I of the 2-turn interaction.

Model Output by Retool-4B (Turn 1) [Initial analysis]

(Handle each periodic line segment separately, then solve the induced quadratics and filter by the segment's x-interval)
...Therefore, in total, for x [0,34], we have:9 increasing intervals and 9 decreasing intervals, with the last decreasing interval cut off at 34.Therefore, we need to set up equations for each of these 9 increasing and 9 decreasing intervals and solve the quadratic equations.However, manual calculation would be error-prone. Let's generate the equations and solve them programmatically to find the valid solutions for each interval.

```
import math
# Define the intervals and solve quadratics for each n in increasing and decreasing
    cases

# Increasing intervals: defined by n from 0 to8
def solve_increasing():
    solutions = []
    for n in range(0, 9):
        a = 34
        b = -1
        c = -4 * n
        discriminant = b**2 - 4*a*c
        sqrt_disc = math.sqrt(discriminant)
        y1 = (-b + sqrt_disc) / (2*a)
        y2 = (-b - sqrt_disc) / (2*a)
        solutions.append((n, y1, y2))
    return solutions

# Decreasing intervals: defined by n from0 to8 (last cut off at x=34)
def solve_decreasing():
    solutions = []
    for n in range(0, 9):
        a = 34
        b = 1
        c = - (4 * n + 2)
    discriminant = b**2 - 4*a*c
    sqrt_disc = math.sqrt(discriminant)
    y1 = (-b + sqrt_disc) / (2*a)
    y2 = (-b - sqrt_disc) / (2*a)
    solutions.append((n, y1, y2))
    return solutions

# Calculate the valid y coordinates
valid_y = []

# Process increasing intervals
inc_solutions = solve_increasing()
for n, y1, y2 in inc_solutions:
    x1 = 34 * y1**2
    x2 = 34 * y2**2
    interval_start = 4*n - 1
    interval_end = 4*n + 1

    # Adjust intervals for n=0
    if n == 0:
        valid_x_interval = (0, 1)
    else:
        valid_x_interval = (interval_start, interval_end)
```

7

Model Output by Retool-4B (Turn 1) [Initial analysis]

```
    # Check each solution
    for y in [y1, y2]:
        if -1 <= y <= 1:
            x = 34 * y**2
            if valid_x_interval[0] <= x < valid_x_interval[1]:
                valid_y.append(y)

# Process decreasing intervals
dec_solutions = solve_decreasing()
for n, y1, y2 in dec_solutions:
    x1 = 34 * y1**2
    x2 = 34 * y2**2
    interval_start = 4*n + 1
    interval_end = 4*n + 3

    # Adjust for n=8 (truncated at34)
    if n == 8:
        valid_x_interval = (33, 34)
    else:
        valid_x_interval = (interval_start, interval_end)

    for y in [y1, y2]:
        if -1 <= y <= 1:
            x = 34 * y**2
            if valid_x_interval[0] <= x < valid_x_interval[1]:
                valid_y.append(y)

# Sum all valid y coefficients
sum_y = sum(valid_y)

# Approximate the sum to verify
approx_sum = 4 * (sqrt(2) - 1)  # Approximation from original thinking

print("Valid y-coordinates:", valid_y)
print("Sum of y-coordinates:", sum_y)
print("Approximate sum:", approx_sum)
```

Execution Result: name 'sqrt' is not defined

Model Output by Retool-4B (Turn 2 - Final)

This error suggests that the code attempted to use sqrt but forgot to import it. Let's correct this by replacing any manual sqrt calls with math.sqrt(n). Also, let's rewrite the symbolic calculation. Wait, but in the code above, we already use math.sqrt(discriminant). Therefore, perhaps the issue is somewhere else. Wait, ...

Therefore, in the `valid_y` list, we have the accumulated y values from all intervals. ...

But according to our code output earlier, sum was 1.1469...

Answer is $\boxed{24}$.

8