TIER: FINE-GRAINED TRAJECTORY INTERVENTION FOR STABLE AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models can substantially improve their reasoning capabilities through multi-turn reinforcement learning with external tools, yet prolonged training often suffers from instability and performance collapse. A common stabilization strategy filters out trajectories exhibiting recognizable failures, making a binary decision: each trajectory is either discarded or allowed to contribute fully to the policy update. Such coarse intervention not only requires increasingly elaborate filtering criteria to balance stability against data retention, but also leaves the optimization influence of retained trajectories uncontrolled. We find that rollouts without obvious behavioral degeneration can produce unusually large gradients and exert disproportionate influence on the aggregate gradient. To address this issue, we introduce **T**rajectory **I**ntervention via **E**ligibility and **R**escaling (TIER), a fine-grained trajectory intervention framework. TIER uses permissive screening based on generic response-level statistics to remove clear degeneration without attempting to enumerate every possible failure mode. It then clips excessive gradient contributions from eligible trajectories before batch aggregation while preserving their update directions. This design enables three levels of intervention: discarding degenerate trajectories, attenuating overly influential ones, and fully retaining the remainder. On AIME24, TIER improves over the corresponding TIR baselines by 10.2 and 14.0 points on Qwen2.5-7B and Qwen3-4B-Base, respectively. Experiments in retrieval-based interaction further demonstrate TIER's effectiveness beyond mathematical tool use. Overall, our results show that stable agentic reinforcement learning benefits from controlling not only whether each trajectory contributes to learning, but also how strongly it does so.

1 INTRODUCTION

Reinforcement learning (RL) has become a powerful approach to improving the reasoning capabilities of large language models (Shao et al., 2024; Guo et al., 2025). Multi-turn tool-integrated reasoning (TIR) extends this learning process from producing a single text response to acting through a sequence of interactions: a model can interact with the environment through tools such as code interpreters and search engines, obtain feedback, and adapt subsequent actions to help solve the task (Jin et al., 2025; Xue et al., 2026). Access to external tools and environment feedback expands the range of tasks these models can solve (Lin & Xu, 2025). Sustaining these gains over longer training requires stable optimization, yet multi-turn RL can suffer from sharp performance degradation and behavioral collapse (Wang et al., 2025; Xue et al., 2026), making training stability a central challenge for scaling tool-using agents.

A common approach to stabilizing training uses explicit rules to identify and filter behaviorally anomalous trajectories. Removing these trajectories from the policy loss blocks their gradients from entering the update. SimpleTIR, for example, effectively stabilizes multi-turn TIR training by excluding trajectories containing a *void turn*[1] (Xue et al., 2026). Such rules make a binary decision for each trajectory: whether to discard or retain its contribution. Yet we find that behaviorally coherent trajectories can still produce unusually large gradient contributions that may disproportionately influence batch updates. Discarding these trajectories solely for their large gradient norms risks

[1] A void turn is a model response containing neither a complete code block nor a final answer.

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.

removing useful learning signals, whereas retaining their contributions without attenuation may undermine training stability. Neither option is ideal.

To address these limitations, we introduce **T**rajectory **I**ntervention via **E**ligibility and **R**escaling (TIER), a fine-grained trajectory intervention framework that combines *behavioral eligibility control* with *optimization influence control*. It uses generic response-level statistics to screen for clear degeneration, then clips oversized gradients from eligible trajectories before batch aggregation while preserving their directions. TIER thus supports three levels of intervention: discarding flagged trajectories, attenuating oversized gradients from eligible trajectories, and retaining the remaining contributions in full.

The eligibility controller targets behavioral degeneration, such as pathological repetition, fragmented text, and off-task continuations, to exclude potentially unreliable gradient contributions. It uses inexpensive, task-agnostic statistics of response compressibility and token-level diversity to screen for associated statistical outliers at both turn and trajectory levels, with configurable thresholds across interaction settings. Flagged trajectories are masked from the policy loss while preserving the original group-relative advantages (Xue et al., 2026).

The influence controller targets excessive gradient contributions from trajectories. Our analysis shows that some trajectories passing eligibility screening have unusually large gradient norms and can dominate the aggregate gradient, yet most of these high-norm trajectories remain coherent and task-directed. Our experiments further indicate that these trajectories can retain learning value, so simply discarding them risks removing useful learning signals. The controller therefore caps each trajectory's gradient norm before aggregation, attenuating oversized contributions while preserving their directions.

We evaluate TIER on mathematical reasoning with Qwen2.5-7B and Qwen3-4B-Base. TIER improves over the corresponding TIR baselines on all benchmarks we evaluated, with AIME24 gains over the TIR baselines of 10.2 and 14.0 points, respectively. Ablations show that each controller improves overall performance over the corresponding baseline when used alone, and their combination performs best on both backbones. Experiments in retrieval-based interaction further demonstrate TIER's effectiveness beyond mathematical tool use. Overall, these results show that stable agentic reinforcement learning benefits from controlling not only whether each trajectory contributes to learning, but also how strongly it shapes the policy update.

2 Related Work

Tool-integrated reinforcement learning. Tool-integrated reasoning interleaves model reasoning with tool calls and uses external feedback to guide subsequent steps. This interaction can expand the solution strategies available under finite token budgets (Lin & Xu, 2025). Building on this potential, reinforcement learning has been widely applied to strengthen models' tool-use capabilities, spanning retrieval-based reasoning in Search-R1 and R1-Searcher (Jin et al., 2025; Song et al., 2025) and code-assisted mathematical reasoning in ReTool, ETIR, ZeroTIR, and THOR (Feng et al., 2026; Bai et al., 2026; Mai et al., 2025; Chang et al., 2026). However, multi-turn RL training is particularly prone to optimization instability and performance collapse. SimpleTIR reports recurring gradient spikes and performance collapse in naive multi-turn TIR (Xue et al., 2026). RAGEN and ARLArena document related failures in broader agent settings (Wang et al., 2025; 2026). Such collapses can erase earlier gains, making stable optimization central to scaling multi-turn RL.

Stabilizing reinforcement learning for multi-turn TIR. Trajectory filtering screens out problematic rollouts using explicit rules to prevent potentially harmful contributions to policy updates. For example, SimpleTIR masks trajectories containing void turns (Xue et al., 2026). rStar2-Agent (Shang et al., 2025) and TAO-RL (Cao et al., 2026) use trajectory-filtering rules based on answer correctness and tool-execution outcomes. Beyond trajectory selection, methods such as GiGPO (Feng et al., 2025) and Group Turn Policy Optimization (GTPO) (Ding et al., 2026) refine credit assignment through finer-grained advantage estimation within trajectories, rather than relying solely on a trajectory-level advantage shared across steps. To reduce policy-gradient variance, VCPO (Huang et al., 2026) and OTB (Li et al., 2026) develop gradient-informed baselines, with empirical results supporting the applicability of these methods to multi-turn TIR training. Other approaches, including truncated importance sampling (TIS) (Yao et al., 2025) and GSPO (Zheng et al., 2025), seek

to stabilize policy updates through token-level importance-weight truncation and sequence-level surrogate-objective clipping, respectively. Although not developed specifically for multi-turn TIR, subsequent studies suggest that these approaches can help mitigate training instability in this setting (Liu et al., 2025; Wang et al., 2026). TIER is compatible with the approaches discussed above and can complement them by combining behavioral screening with pre-aggregation gradient-norm control, regulating both whether and how strongly a trajectory contributes to learning.

Gradient clipping. Batch-level global-norm clipping is widely used in RL to bound the magnitude of the aggregate gradient (Pascanu et al., 2013; Andrychowicz et al., 2021). It applies a shared scale to all trajectory contributions, leaving their relative weights unchanged. In differential privacy, DP-SGD clips each example's gradient before aggregation and then adds calibrated noise to provide differential privacy guarantees (Abadi et al., 2016). TIER's trajectory-wise gradient norm clipping is similar to DP-SGD's per-example clipping, but aims to prevent individual trajectories with unusually large gradients from dominating the aggregated gradient in RL training. This trajectory-level influence control is also orthogonal to batch-level global-norm clipping.

3 PRELIMINARIES

Multi-turn Tool-Integrated Reasoning (TIR) Trajectories. In multi-turn TIR, a model interacts with tools over successive turns, conditioning each action on the prompt and the preceding interaction history. A trajectory starting from prompt $x_{q(i)}$ takes the form

$$\tau_i = (x_{q(i)}, a_i^1, o_i^1, \ldots, a_i^{H_i-1}, o_i^{H_i-1}, a_i^{H_i}), \tag{1}$$

where $q(i)$ indexes the prompt, H_i is the number of model turns, a_i^k is the model action at turn k, and o_i^k is the corresponding tool observation. The trajectory ends with the final model action $a_i^{H_i}$.

For token-level optimization, let $y_{i,1:T_i}$ denote the serialized trajectory and $h_{i,t}$ the full prefix preceding token $y_{i,t}$. We define an action-token mask $m_{i,t}$ that equals one for policy-generated tokens and zero for prompt, tool-observation, and padding tokens, with $\sum_{t=1}^{T_i} m_{i,t} > 0$. Tool observations remain part of the context for subsequent actions, but their tokens do not contribute directly to the loss (Jin et al., 2025).

Objective Function. Motivated by the improved training stability reported for GSPO (Zheng et al., 2025) in multi-turn agentic reinforcement learning (Wang et al., 2026), we adopt GSPO as the base objective for mathematical reasoning, using the multi-turn interaction format and action-token mask defined above. We combine it with DAPO-style dynamic sampling, which excludes prompt groups whose sampled answers are all correct or all incorrect[2], and overlong filtering, which masks the policy loss of truncated trajectories to reduce the impact of potentially noisy rewards (Yu et al., 2025). We refer to the resulting algorithm as *GSPO-TIR*.

Let $\mathcal{G}_q$ index the trajectories sampled for prompt q by the behavior policy $\pi_{\theta_{\text{old}}}$, and let $R_i \in \{0, 1\}$ indicate answer correctness. We use the group-normalized advantage $\widehat{A}_i = (R_i - \mu_{q(i)})/(\sigma_{q(i)} + \varepsilon_A)$, where μ_q and σ_q are the reward mean and standard deviation over all trajectories in the original sampled group $\mathcal{G}_q$, and $\varepsilon_A > 0$ ensures numerical stability. Following GSPO (Zheng et al., 2025), we align policy weighting and clipping with trajectory-level rewards by assigning a single importance ratio to each trajectory. Specifically, using the action-token mask $m_{i,t}$, we define

$$\rho_i(\theta) = \exp\left(\frac{1}{\sum_{t=1}^{T_i} m_{i,t}} \sum_{t=1}^{T_i} m_{i,t} \log\left[\pi_\theta(y_{i,t} \mid h_{i,t}) / \pi_{\theta_{\text{old}}}(y_{i,t} \mid h_{i,t})\right]\right). \tag{2}$$

The GSPO-TIR objective minimizes the following batch loss:

$$\mathcal{L}_{\mathcal{B}}^{\text{GSPO-TIR}}(\theta) = -\frac{1}{N} \sum_{i \in \mathcal{B}} \min\left(\rho_i(\theta)\widehat{A}_i,\ \text{clip}\left(\rho_i(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}\right) \widehat{A}_i\right), \tag{3}$$

where $\mathcal{B}$ is the update batch, $N = |\mathcal{B}|$, and ϵ_{low} and ϵ_{high} are the lower and upper clipping widths.

[2]We oversample prompts before rollout and duplicate some of the retained groups if their number after dynamic sampling still falls short of the target batch size.

4 TIER: TRAJECTORY INTERVENTION VIA ELIGIBILITY AND RESCALING

The GSPO-TIR objective (Equation 3) specifies how sampled trajectories contribute to policy optimization through their advantages and sequence-level importance ratios. Training stability also depends on the behavioral quality of these trajectories and the magnitude of their gradient contributions. We introduce TIER to address these two factors. It uses the *eligibility controller* to screen for behavioral degeneration and the *influence controller* to limit the influence of individual trajectories.

4.1 OVERVIEW: JOINT CONTROL OF TRAJECTORY ELIGIBILITY AND INFLUENCE

Let $\ell_i(\theta)$ denote the loss of trajectory i under the base objective, we write the trajectory gradients and their baseline aggregation as

$$g_i = \nabla_\theta \ell_i(\theta), \qquad G = \frac{1}{N} \sum_{i \in \mathcal{B}} g_i, \tag{4}$$

where $\mathcal{B}$ and N follow the base-objective definition above.

The eligibility controller uses response statistics to assign a binary decision $d_i \in \{0, 1\}$ to each trajectory. The influence controller then limits the gradient norm of each eligible trajectory through a direction-preserving scale $s_i \in (0, 1]$. Together, they yield the TIER aggregate gradient

$$G_{\text{TIER}} = \frac{1}{N} \sum_{i \in \mathcal{B}} d_i s_i g_i. \tag{5}$$

This rule discards the contribution of a trajectory when $d_i = 0$, attenuates it when $d_i = 1$ and $s_i < 1$, and retains it unchanged when $d_i = s_i = 1$. Batch-level global-norm clipping is applied after aggregation to bound the aggregate gradient norm.

Setting $d_i \equiv 1$ disables the eligibility controller, while setting $s_i \equiv 1$ disables the influence controller. Setting both to one recovers the base aggregate gradient. Sections 4.2 and 4.3 describe the two controllers respectively.

4.2 SCREENING TRAJECTORY ELIGIBILITY WITH RESPONSE STATISTICS

Multi-turn TIR training often exhibits late-stage performance degradation (see one trial illustrated in Figure 1). Inspection of these trajectories reveals *behavioral degeneration*: responses get stuck repeating the same content, produce disconnected or incomplete passages, or drift away from the problem, so they no longer form a coherent attempt to solve it.

Degeneration can emerge after multiple coherent reasoning and tool-use turns, as shown in Example 1 in Appendix E.2. Such trajectories may introduce noisy or harmful contributions to the policy gradient. A negative advantage encourages the policy to reduce the likelihood of a degenerate response, but this does not necessarily provide a useful update direction toward coherent and correct reasoning. We therefore exclude these potentially unreliable contributions by masking the policy losses of trajectories flagged by statistical screening.

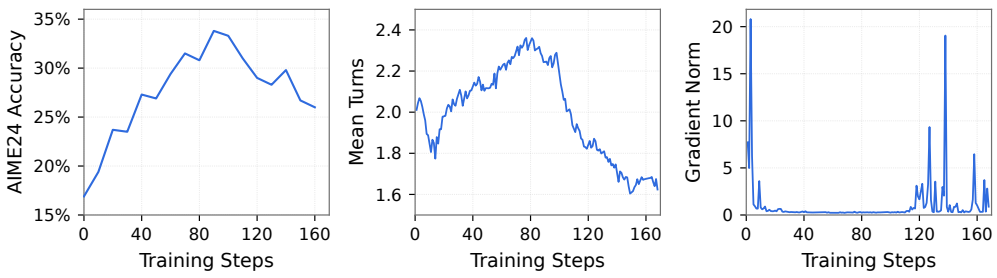

Figure 1: Training dynamics of the Qwen2.5-7B GSPO-TIR. Late-stage accuracy degradation coincides with a decrease in the mean number of interaction turns and spikes in the gradient norm.

Response statistics. To identify degenerate trajectories more comprehensively, we screen trajectories using two complementary statistics: the **zip ratio (ZR)**, which measures byte-level compressibility, and the **expectation-adjusted Distinct-1 ratio (EAD)** introduced by Liu et al. (2022), which measures token-level diversity. For a nonempty generated token span $z = (z_1, \ldots, z_L)$ with a nonempty UTF-8 byte representation $b(z)$, we define

$$\mathrm{ZR}(z) = \frac{|C(b(z))|}{|b(z)|}, \qquad \mathrm{EAD}(z) = \frac{|\{z_t : t = 1, \ldots, L\}|}{U_{L,V}}, \tag{6}$$

Here C is a lossless compressor and we use `zlib` in our experiments. $U_{L,V}$ is the expected number of distinct unigram types in L independent uniform draws from a vocabulary of size V. Normalizing by this expectation mitigates length bias. Appendix A.1 provides the derivation. Lower ZR indicates greater compressibility, while lower EAD indicates lower token-level diversity. In the inspected trajectories, low-valued outliers include pathological repetition, whereas high-valued outliers include off-task continuations, as illustrated in Examples 2 and 3 in Appendix E.2. We use these statistics as screening signals rather than direct measures of task correctness or reasoning quality.

Turn- and trajectory-level screening. Guided by these observations, we specify acceptance intervals for ZR and EAD and flag trajectories whose statistics fall outside the enabled intervals. Screening operates at two scopes: individual turns capture local degeneration, while complete trajectories capture patterns that emerge across turns. Examples 1 and 4 illustrate these cases respectively.

For scope $r \in \{\mathrm{turn}, \mathrm{traj}\}$, let $p_r(z) \in \{0, 1\}$ denote whether span z passes the enabled interval checks, accounting for the exemptions specified in Appendix A.1. Let $\bar{a}_i = a_i^1 \| \cdots \| a_i^{H_i}$ concatenate all model actions in trajectory i, excluding tool observations. We define eligibility as

$$d_i = p_{\mathrm{traj}}(\bar{a}_i) \prod_{k=1}^{H_i} p_{\mathrm{turn}}(a_i^k) \in \{0, 1\}. \tag{7}$$

Thus, a trajectory is eligible only if it passes both the trajectory-level check and every applicable turn-level check. TIER masks ineligible trajectories' policy losses as $d_i \ell_i(\theta)$ while preserving the original group-relative advantages. This excludes potentially unreliable gradient contributions without changing prompt-group reward statistics.

Eligibility screening is inexpensive and requires no task-specific verifier. It uses general-purpose statistics of response compressibility and token-level diversity, with configurable scopes and thresholds that support reuse across tasks and tool environments.

4.3 Limiting Influence with Trajectory-wise Gradient Norm Clipping

TIER adopts permissive eligibility screening that excludes only trajectories exhibiting clear signs of behavioral degeneration, as indicated by these response statistics. This preserves a broad range of potentially useful trajectories, but does not control their relative influence on the policy update.

Among trajectories that pass screening, we still observe unusually large gradient norms during GSPO-TIR training (Figure 12 in Appendix C.1). Appendix C.1.2 further presents a coherent but incorrect trajectory for a problem about even factors whose gradient norm accounts for 88.4% of the sum of individual gradient norms in its update batch. Such trajectories can dominate the aggregate gradient despite showing no clear behavioral degeneration, as illustrated in Figure 2(a).

Simply discarding these high-norm trajectories risks removing useful learning signals. In a blinded LLM audit using Claude Opus 4.8 (Appendix C.1.1), most high-norm baseline trajectories that pass eligibility screening are judged to contain useful or acceptably noisy learning signals. Moreover, directly deleting eligible high-norm contributions leads to training collapse under a tested configuration (Figure 4). These findings motivate limiting their magnitudes while preserving directions.

Capping individual contributions with trajectory-wise clipping. TIER caps each eligible trajectory's gradient norm before aggregation, as illustrated in Figure 2(c). We define

$$s_i = \begin{cases} 1, & \|g_i\|_2 = 0, \\ \min\left(1, \dfrac{\kappa}{\|g_i\|_2}\right), & \|g_i\|_2 > 0, \end{cases} \qquad \widetilde{g}_i = s_i g_i, \tag{8}$$

where $\kappa > 0$ is a threshold for trajectory gradient norm. Contributions with norm at most κ remain unchanged, while larger contributions are rescaled to norm κ without changing direction. Each eligible trajectory therefore contributes at most κ/N in norm to G_{TIER} in Equation 5.

This capping operation is applied to the trajectory gradients before aggregation. In contrast, the usual global-norm clipping uniformly rescales the aggregate, preserving trajectories' relative influence and potentially suppressing moderate contributions alongside extreme ones (Figure 2(b)). Trajectory-wise clipping limits individual contributions before aggregation (Figure 2(c)). We retain global clipping of G_{TIER} to norm κ_B before the optimizer step to also bound the aggregate gradient.

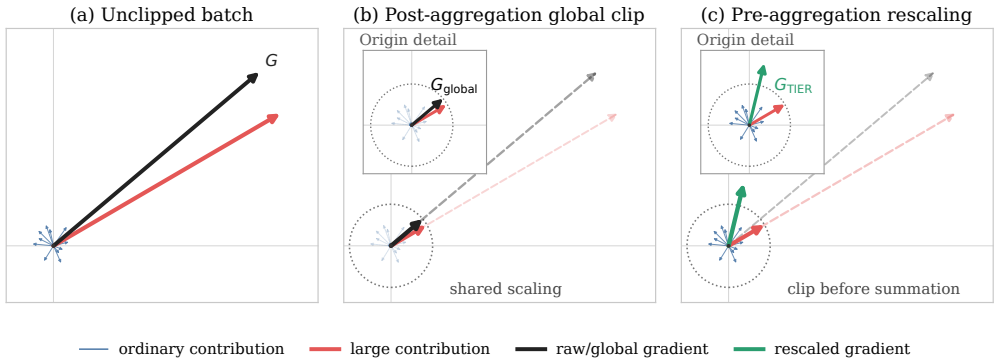

Figure 2: Effects of clipping on the same eligible trajectory contributions g_i/N. (a) An extreme contribution dominates the aggregate. (b) Global clipping uniformly rescales all contributions, preserving their relative weights and the aggregate direction. (c) Trajectory-wise clipping caps individual contributions before aggregation and global clipping. Dotted circles mark the batch-level cap κ_B in (b) and the individual contribution cap κ/N in (c). Insets use the same scale.

Computing trajectory-wise gradient norms. Implementation requires isolating each trajectory's gradient before it is aggregated with other trajectories, either within a micro-batch[3] or across data-parallel ranks.[4] Our FSDP implementation uses a micro-batch size of one and defers automatic gradient synchronization. Each rank computes the ℓ_2 norm of its current trajectory's gradient over all trainable parameters and applies the corresponding scale before gradient accumulation and synchronization. Appendix A.2 details norm computation and gradient accumulation in the distributed implementation. It reports runtime and memory overhead relative to single-trajectory micro-batching and dynamic batching, both without trajectory-wise gradient-norm computation or clipping. More efficient methods for trajectory-wise norm computation and clipping are possible by modifying FSDP or Megatron internals with hooks.

5 Experiments

5.1 Experimental Setup

Training configurations and baselines. We train the pretrained Qwen2.5-7B and Qwen3-4B-Base models on the union of SimpleRL's MATH levels 3–5 subset (Zeng et al., 2025) and the DeepScaleR training set (Luo et al., 2025). Baselines include pretrained models evaluated directly with text-only or TIR inference, GSPO-TIR defined in Section 3, and our SimpleTIR reproduction, which applies void-turn filtering to GSPO-TIR (Xue et al., 2026). In the main comparison, TIER enables both eligibility and influence control on the same base algorithm. For the GSPO-TIR baseline, we follow SimpleTIR's settings for the shared training hyperparameters (Xue et al., 2026) and tune the GSPO clipping widths using short pilot runs. The additional hyperparameters introduced by TIER

[3] A micro-batch is the unit of computation for one forward/backward pass. Standard backpropagation produces the aggregate gradient of its trajectories rather than separate trajectory gradients.

[4] FSDP aggregates gradients across ranks through reduce-scatter, leaving each rank with a shard of the aggregate.

are selected based on offline analysis of trajectories collected during a few initial baseline training steps, as described in Appendix A.3.

Rollout and optimization. We use a Python interpreter with the mathematics interaction prompt provided in Appendix E.1.1 and assign 0/1 rewards based on answer correctness. We allow up to 5 model turns, with at most 8,192 generated tokens per turn and a rolling input context limited to the most recent 16,384 tokens. Following SimpleTIR (Xue et al., 2026), training rollouts terminate after a void turn to reduce rollout cost and avoid potentially degenerate continuations. All configurations retain global-norm clipping threshold 1.0. TIER uses ZR and EAD screening at turn and trajectory levels, with influence control using norm caps of $\kappa = 60$ and 32 for Qwen2.5-7B and Qwen3-4B-Base, respectively. Table 4 lists detailed training hyperparameters and control settings.

Evaluation. We report Avg@16, the mean correctness over 16 samples per problem, on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), AIME24, AIME25, AIME26, HMMT26, and OlympiadBench (He et al., 2024), using temperature 1.0 and top-p 0.7. To better assess the model's capabilities within the available interaction budget, we allow generation to continue after a void turn as long as turns remain. Other interaction settings follow training, with non-TIR evaluation parameters shared by the text-only baseline.

5.2 Performance and Training Stability

Main results. Table 1 reports evaluation results for all compared configurations on the Qwen2.5-7B and Qwen3-4B-Base backbones. The resulting comparison shows that multi-turn TIR substantially outperforms conventional single-turn text-only inference across all benchmarks on both backbones. RL training yields further gains over using the base models directly for multi-turn TIR across all three trained methods, including the GSPO-TIR that later collapses. Among these methods, **TIER achieves the best performance on all six benchmarks for both backbones.**

Table 1: Mathematical-reasoning Avg@16 accuracy (%). Bold marks the best result among the methods shown for each backbone.

Method	MATH-500	AIME24	AIME25	AIME26	HMMT26	Olympiad	Avg.
Qwen2.5-7B							
Base (text-only inference)	53.1	7.1	3.3	4.2	2.3	23.5	15.6
Base (TIR inference)	58.3	14.6	12.7	10.6	13.4	34.2	24.0
GSPO-TIR	81.1	33.8	24.4	22.9	22.9	52.2	39.6
SimpleTIR	82.8	37.1	25.2	24.6	22.3	54.3	41.1
TIER	**83.5**	**44.0**	**28.8**	**28.8**	**24.2**	**54.6**	**44.0**
Qwen3-4B-Base							
Base (text-only inference)	57.3	11.0	6.9	5.2	3.6	33.0	19.5
Base (TIR inference)	65.6	19.0	12.1	15.4	11.4	40.3	27.3
GSPO-TIR	81.4	27.5	22.3	22.1	15.2	51.9	36.7
SimpleTIR	83.0	38.1	28.3	21.5	23.9	55.9	41.8
TIER	**85.3**	**41.5**	**30.2**	**32.7**	**26.3**	**58.1**	**45.7**

Training dynamics. Figure 3 shows the training dynamics of the three RL methods on Qwen2.5-7B. The GSPO-TIR baseline collapses early in training: AIME24 accuracy deteriorates after an early peak and fails to recover, accompanied by a contraction in mean interaction turns and large intermittent gradient spikes. SimpleTIR and TIER avoid this coupled collapse. Compared with SimpleTIR, TIER sustains higher late-stage AIME24 accuracy, with greater interaction depth and generally a higher valid code rate, although its gradient norm exhibits larger intermittent fluctuations.

5.3 Ablations and Analysis

Controller ablations. To assess the contribution of each controller, we ablate eligibility control, influence control, or both, while keeping the thresholds of any enabled controller unchanged. Ap-

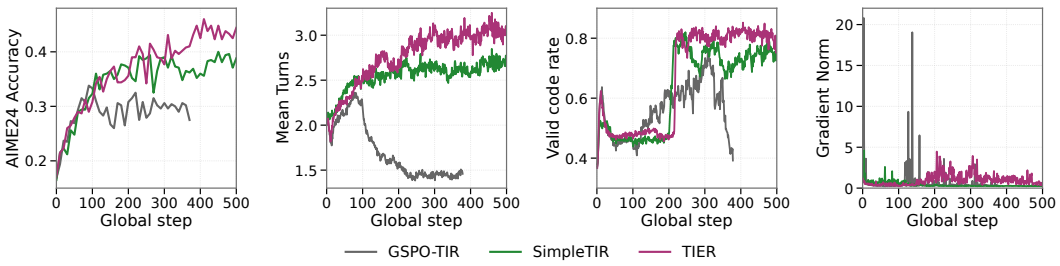

Figure 3: Training dynamics of GSPO-TIR, SimpleTIR, and TIER on Qwen2.5-7B. Valid code rate is the trajectory-averaged fraction of model turns that produce nonempty interpreter output without reported errors. Gradient norms are measured after aggregation and before global clipping.

pendix B reports the complete controller-ablation results. It further assesses the threshold sensitivity of both controllers and examines the roles of response statistics, screening bounds, and detection scopes within eligibility control.

Table 2 summarizes AIME24 and AIME25 accuracy on both backbones, and Table 5 in Appendix B.1 provides the complete six-benchmark results. Either controller improves over GSPO-TIR on most benchmarks when used alone, with eligibility control generally providing larger gains. Enabling both yields the best results among the four configurations except on Qwen2.5-7B HMMT26, where eligibility control alone achieves 25.4% versus 24.2% with both controls.

Table 2: Controller ablations on mathematical reasoning (Avg@16 accuracy, %). ✓/✗ indicate enabled/disabled controls. Bold marks the best result in each column.

Method	Eligibility	Influence	Qwen2.5-7B		Qwen3-4B-Base	
			AIME24	AIME25	AIME24	AIME25
GSPO-TIR	✗	✗	33.8	24.4	27.5	22.3
TIER	✓	✗	41.0	26.5	38.5	28.3
TIER	✗	✓	34.8	26.9	32.9	25.8
TIER	✓	✓	**44.0**	**28.8**	**41.5**	**30.2**

Figure 6 in Appendix B.1 shows the training dynamics on Qwen2.5-7B. Influence control alone suppresses gradient spikes, but AIME24 accuracy and interaction depth still decline later. Eligibility control avoids this coupled collapse, though gradient fluctuations remain and AIME24 accuracy declines toward the end of training. Combining the controls reduces gradient fluctuations relative to eligibility control alone and sustains the best late-stage performance. Influence control can also complement existing trajectory-filtering strategies. As shown in Table 8 in Appendix C.2, adding it to SimpleTIR improves overall performance on both backbones.

Beyond removing entire controllers, Appendix B.2 examines whether effective training depends on the precise intervention thresholds. On Qwen2.5-7B, several tested settings near the default retain strong AIME accuracy, indicating that effective training is not confined to the exact thresholds used in the main experiments. Appendix B.3 then examines the roles of response statistics, screening bounds, and detection scopes within eligibility control. On the same backbone, the full screening configuration generally maintains higher late-stage AIME accuracy than the ablated variants.

Rescaling versus hard deletion. We compare TIER with a Hard-deletion variant that removes contributions by zeroing trajectory gradients above the same norm threshold, keeping the other settings fixed. Figure 4 shows that the Hard-deletion variant produces smaller and less variable aggregate gradient norms than TIER, yet its AIME24 accuracy collapses, accompanied by a contraction in mean interaction turns. Thus, suppressing aggregate-gradient magnitude alone does not ensure effective training. Together with the blinded audit in Appendix C.1.1, this comparison supports retaining and attenuating eligible high-norm contributions rather than discarding them solely for exceeding the norm threshold. Appendix C.1.3 gives the deletion rule and supplementary analysis.

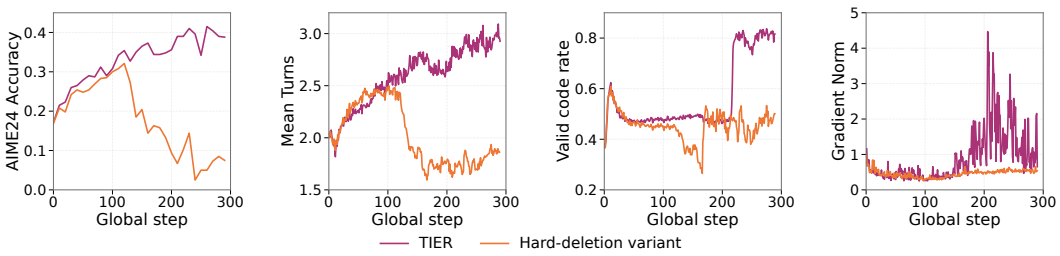

Figure 4: Training dynamics of TIER versus the Hard-deletion variant on Qwen2.5-7B with the same eligibility screening and gradient-norm threshold ($\kappa = 60$).

5.4 Transfer to Search-Based Interaction

To evaluate TIER beyond mathematical tool use, we train Qwen2.5-7B with GRPO for multi-turn TIR using search as a tool, following Search-R1's setup (Jin et al., 2025). Appendix D provides detailed training settings, training-reward curves, and accompanying analysis.

TIER avoids the training-reward collapse observed in the GRPO baseline as shown in Figure 14 and achieves the highest average greedy exact-match accuracy across the seven QA benchmarks in Table 3, outperforming both GRPO and SimpleTIR overall. These results demonstrate TIER's effectiveness beyond mathematical tool use.

Table 3: Search-R1 normalized exact-match accuracy (%). Elig./Infl. denote eligibility/influence control; ✓/✗ indicate enabled/disabled controls, and dashes indicate that these switches do not describe SimpleTIR. Bold marks the best result among the configurations shown.

			General QA			Multi-Hop QA				
Method	Elig.	Infl.	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg.
GRPO	✗	✗	47.8	63.3	46.9	42.6	38.1	16.5	40.0	42.2
SimpleTIR	—	—	49.8	**66.3**	47.1	46.8	**44.9**	22.3	42.4	45.7
TIER	✓	✗	49.9	66.0	**47.7**	**47.3**	43.7	20.6	47.2	46.1
TIER	✗	✓	**50.5**	65.9	46.7	46.9	39.7	18.9	39.2	44.0
TIER	✓	✓	50.0	65.9	45.7	**47.3**	44.8	**23.3**	**52.0**	**47.0**

6 Conclusion

We present TIER, a fine-grained trajectory intervention framework to stabilize RL training for multi-turn TIR. It uses inexpensive, task-agnostic response statistics to screen out trajectories with clear behavioral degeneration, then applies trajectory-wise gradient clipping to limit each retained trajectory's contribution before batch aggregation. Experiments on mathematical reasoning show that TIER delivers substantial performance gains over the baselines, with retrieval-based results further supporting its applicability beyond mathematical tool use. Ablation studies confirm the importance of both controllers. These results highlight the value of controlling both whether and how strongly trajectories contribute to policy updates.

Limitations and Future Work. Response statistics and thresholds remain manually selected, and threshold selection requires preliminary runs and analysis, adding tuning effort and computational cost. On the other hand, configurable thresholds allow the strictness of eligibility and influence control to be adjusted for different interaction settings, offering greater flexibility than trajectory-intervention methods based on fixed behavioral patterns. During training, the current influence-control implementation computes and stores full trajectory gradients, increasing runtime and GPU memory use, as quantified in Appendix A.2.2. Future work can develop adaptive thresholds and distributed implementations that reduce runtime and GPU memory overhead.

AI USE STATEMENT

In this work, we used generative AI tools for grammar and spelling correction, rewording for clarity, and literature search to identify relevant prior work. We also used an LLM as an annotator for the high-norm trajectory audits described in Appendix C.1.1. All AI-assisted outputs were reviewed and verified by us, including checking the audit setup and aggregate results and verifying references against original sources. We take full responsibility for the final content and artifacts presented in this work.

ETHICS STATEMENT

This work investigates training stability in multi-turn tool-integrated reinforcement learning. The training and evaluation use publicly available pretrained models and datasets. Tool interactions comprise code execution and retrieval from a fixed corpus in our experiments. Greater tool-use capability can amplify the consequences of erroneous or inappropriate actions. Deployment should therefore include tool-permission controls, isolated execution environments, and application-specific safety evaluation.

REPRODUCIBILITY STATEMENT

Section 3 defines the problem setup and base objective, and Section 4 presents the trajectory interventions. Section 5.1 specifies comparisons and evaluation metrics. Appendices A and D provide implementation details, hyperparameters, and tool-interaction settings. Appendix C.1.1 provides audit criteria and stratified results, and Appendix E.4 gives the complete frozen prompts. We plan to publicly release the implementation, training configurations, and evaluation scripts.

REFERENCES

Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318. ACM, 2016. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nIAxjsniDzg.

Fei Bai, Yingqian Min, Beichen Zhang, Zhipeng Chen, Xin Zhao, Lei Fang, Zheng Liu, Zhongyuan Wang, and Hongteng Xu. Towards effective code-integrated reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(36):30022–30030, 2026. doi: 10.1609/aaai.v40i36.40250. URL https://ojs.aaai.org/index.php/AAAI/article/view/40250.

Hongye Cao, Nuo Yan, Haoyuan Deng, Ziwei Wang, Tianpei Yang, Jing Huo, Yuyao Zhang, and Yang Gao. Tool-aware optimization with entropy guidance for efficient agentic reinforcement learning. arXiv preprint arXiv:2606.03762, 2026. URL https://arxiv.org/abs/2606.03762.

Qikai Chang, Zhenrong Zhang, Pengfei Hu, Jun Du, Jiefeng Ma, Yicheng Pan, Jianshu Zhang, Quan Liu, and Jianqing Gao. THOR: Tool-integrated hierarchical optimization via RL for mathematical reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0Af7UiJISU.

Yifeng Ding, Hung Le, Songyang Han, Kangrui Ruan, Zhenghui Jin, Varun Kumar, Zijian Wang, and Anoop Deoras. Empowering multi-turn tool-integrated agentic reasoning with group turn policy optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42409–42423. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1962. URL https://aclanthology.org/2026.acl-long.1962/.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tRk1nofSmz.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, pp. 51797–51830, 2025. doi: 10.52202/085713-1544. URL https://doi.org/10.52202/085713-1544.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645 (8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

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 J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information*

Processing Systems Track on Datasets and Benchmarks, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Luke J. Huang, Zhuoyang Zhang, Qinghao Hu, Shang Yang, and Song Han. Stable asynchrony: Variance-controlled off-policy RL for LLMs. arXiv preprint arXiv:2602.17616, 2026. URL https://arxiv.org/abs/2602.17616.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL https://aclanthology.org/N16-1014/.

Yingru Li, Jiawei Xu, Ziniu Li, Jiacai Liu, Wei Liu, Yuxuan Tong, Longtao Zheng, Zhenghai Xue, Yaxiang Zhang, Tianle Cai, Ge Zhang, Qian Liu, and Baoxiang Wang. The optimal token baseline: Variance reduction for long-horizon LLM-RL. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://openreview.net/forum?id=hpxI4Mndt4.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Heng Lin and Zhongwen Xu. Understanding tool-integrated reasoning. arXiv preprint arXiv:2508.19201, 2025. URL https://arxiv.org/abs/2508.19201.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025. URL https://richardli.xyz/rl-collapse.

Siyang Liu, Sahand Sabour, Yinhe Zheng, Pei Ke, Xiaoyan Zhu, and Minlie Huang. Rethinking and refining the distinct metric. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 762–770. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-short.86. URL https://aclanthology.org/2022.acl-short.86/.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. DeepScaleR: Surpassing O1-Preview with a 1.5B model by scaling RL. Notion Blog, 2025. URL https://pretty-radio-b75.notion.site/19681902c1468005bed8ca303013a4e2.

Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. Agentic RL scaling law: Spontaneous code execution for mathematical problem solving. In *Advances in Neural Information Processing Systems*, volume 38, pp. 7325–7340, 2025. doi: 10.52202/085713-0251. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0adea62f2f98c5035aded00d4b2064e6-Paper-Conference.pdf.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, 2013. URL https://proceedings.mlr.press/v28/pascanu13.html.

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. rStar2-Agent: Agentic reasoning technical report. arXiv preprint arXiv:2508.20722, 2025. URL https://arxiv.org/abs/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. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. arXiv preprint arXiv:2503.05592, 2025. URL https://arxiv.org/abs/2503.05592.

Xiaoxuan Wang, Han Zhang, Haixin Wang, Yidan Shi, Ruoyan Li, Kaiqiao Han, Chenyi Tong, Haoran Deng, Renliang Sun, Alexander Taylor, Yanqiao Zhu, Jason Cong, Yizhou Sun, and Wei Wang. ARLArena: A unified framework for stable agentic reinforcement learning. arXiv preprint arXiv:2602.21534, 2026. URL https://arxiv.org/abs/2602.21534.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. arXiv preprint arXiv:2504.20073, 2025. URL https://arxiv.org/abs/2504.20073.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EplNy91Xqh.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your efficient RL framework secretly brings you off-policy RL training. Feng Yao's Notion, 2025. URL https://fengyao.notion.site/off-policy-rl.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=vSMCBUgrQj.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. arXiv preprint arXiv:2507.18071, 2025. URL https://arxiv.org/abs/2507.18071.

A Implementation Details and Hyperparameters

A.1 Implementation of Behavioral Eligibility Control

This appendix defines the pass decisions p_r used in Equation 7 and specifies their implementation in our mathematics experiments. Screening intervals are listed in Table 4.

Statistics. We compute the **zip ratio (ZR)** and the **expectation-adjusted Distinct-1 ratio (EAD)** defined in Equation 6 of Section 4.2 from model-generated actions. ZR uses Python's default `zlib` compression of decoded action text, while EAD uses the corresponding token IDs. Statistics are computed for individual actions and their trajectory-level concatenation, excluding tool observations.

Length adjustment. Normalizing the distinct unigram count by span length L (Li et al., 2016) can penalize longer spans. Every additional token increases the denominator, but increases the numerator only if its type has not appeared before. The expected distinct-type count therefore grows sublinearly, so its ratio to L can decrease even when the token distribution is unchanged (Liu et al., 2022).

Following Liu et al. (2022), we use a uniform reference to calculate the normalizer without estimating the model's output distribution. Let $X_1, \ldots, X_L$ be independent draws from a vocabulary $\mathcal{V}$ of size V, with $\Pr(X_t = v) = 1/V$ for every position t and token type $v \in \mathcal{V}$. Let I_v indicate whether type v appears at least once. Independence and linearity of expectation give

$$\begin{aligned} \Pr(I_v = 0) &= \prod_{t=1}^{L} \Pr(X_t \neq v) = \left(1 - \frac{1}{V}\right)^L, \\ U_{L,V} &= \mathbb{E}\left[\sum_{v \in \mathcal{V}} I_v\right] \\ &= \sum_{v \in \mathcal{V}} \Pr(I_v = 1) = \sum_{v \in \mathcal{V}} \left[1 - \Pr(I_v = 0)\right] \\ &= V\left[1 - \left(1 - \frac{1}{V}\right)^L\right]. \end{aligned} \tag{9}$$

Under this reference model, the expected length-normalized score is $U_{L,V}/L$, whereas the expected EAD score is one. The adjustment thus compensates for the reference count's nonlinear growth with length.

Pass rule and exemptions. We organize screening into four groups by scope $r \in \{\mathrm{turn}, \mathrm{traj}\}$ and bound side $b \in \{\mathrm{low}, \mathrm{high}\}$. For a generated span z, the group-level pass decision is

$$p_{r,b}(z) = 1 - \mathrm{gate}_{r,b}(z)\big(1 - q_{r,b}(z)\big), \tag{10}$$

where $q_{r,b}(z) \in \{0, 1\}$ indicates whether every enabled statistic f satisfies $f(z) \geq l_f^r$ for $b = \mathrm{low}$ or $f(z) \leq u_f^r$ for $b = \mathrm{high}$, with l_f^r, u_f^r denoting its bounds. The binary $\mathrm{gate}_{r,b}(z)$ controls whether these checks are applied. When the gate is 1, the group follows the statistical decision: $p_{r,b}(z) = q_{r,b}(z)$. When it is 0, $p_{r,b}(z) = 1$, so the group passes without applying the checks. A disabled group uses $\mathrm{gate}_{r,b} \equiv 0$, and we take $q_{r,b} = 1$ when no statistical checks are enabled. The scope-level decision requires both bound groups to pass:

$$p_r(z) = \prod_{b \in \{\mathrm{low}, \mathrm{high}\}} p_{r,b}(z). \tag{11}$$

Our reference configuration enables ZR and EAD checks on both bounds at both scopes. We find that turns containing code or a final answer can be more concise than ordinary natural-language turns, so their ZR or EAD can exceed the upper bounds, as illustrated by the upper-bound exemption examples below. Let $\mathrm{code}(z)$ and $\mathrm{answer}(z)$ indicate the presence of a recognized complete code block

and a final-answer marker (`\boxed` or a `final_answer` call), respectively. To avoid mistakenly filtering out trajectories containing these turns, we set the gates of enabled groups to

$$\text{gate}_{r,b}(z) = \begin{cases} \neg\,\text{code}(z) \wedge \neg\,\text{answer}(z), & (r,b) = (\text{turn}, \text{high}), \\ 1, & \text{otherwise.} \end{cases} \tag{12}$$

Upper-bound exemption examples

Each panel shows the complete model-authored response for one Qwen2.5-7B turn, including all surrounding text and code delimiters. For the final-answer example, the preceding turn had already computed 24 tiles, and its tool output confirmed this result. Metrics use each full turn, and L counts action tokens. Bold values exceed the default turn bounds (ZR: 0.80, EAD: 0.75).

Code turn (complete)
$L = 40$, $\text{ZR} = \mathbf{0.941}$, $\text{EAD} = \mathbf{0.800}$

```
```python
import sympy

terms = [1 + 1/i for i in range(1, 7)]
product = sympy.Mul(*terms)

print(product.evalf())
```
```

Final-answer turn (complete)
$L = 6$, $\text{ZR} = \mathbf{1.800}$, $\text{EAD} = \mathbf{1.000}$

```
\boxed{24}
```

A.2 Implementation and Overhead of Influence Control

A.2.1 Distributed Computation of Trajectory Gradient Norms

Isolating a trajectory gradient. Influence control determines a separate rescaling coefficient from each eligible trajectory's gradient norm, so the norms must be computed before these contributions are combined. In conventional RL training, each rank processes a micro-batch in one forward/backward pass. When a micro-batch contains multiple trajectories, backpropagation produces an aggregate gradient from which their individual gradient norms cannot be read directly. We therefore set the micro-batch size to 1, so that each rank processes one complete multi-turn trajectory per pass. This removes mixing within a micro-batch, but not across ranks: default FSDP synchronization still aggregates gradients from different ranks and shards the result, leaving each rank with part of an aggregate rather than its current trajectory's full gradient. To perform trajectory-wise rescaling before this aggregation, we defer cross-rank synchronization with `no_sync()`, retaining the current trajectory's gradient over all trainable parameters for norm computation and rescaling. With Megatron-LM, however, trajectory gradients can remain sharded across TP ranks during norm computation, reducing per-rank memory overhead, as implemented in VCPO (Huang et al., 2026).

Accumulation and synchronization. In our FSDP implementations, trajectory-wise computation preserves the original optimization mini-batch and update frequency. Conventional training accumulates gradients from several micro-batches in the same buffer before updating parameters once per optimization mini-batch. Computing a norm from this buffer would also include contributions from earlier trajectories. We therefore clear this buffer before each micro-batch and manually maintain a separate gradient accumulator on each rank. For each eligible trajectory, we compute the ℓ_2 norm of its loss gradient g_i and determine s_i using Equation 8. We implement two schemes for accumulating the rescaled contributions, with different trade-offs between gradient storage and synchronization frequency:

1. **Full-gradient accumulation.** Each rank accumulates the rescaled gradients of its local trajectories over all trainable parameters. Once all micro-batches have been processed, we aggregate and shard the accumulated gradients across ranks. This avoids gradient synchronization between micro-batches but requires a full-gradient accumulator on each rank.
2. **Sharded-gradient accumulation.** After each micro-batch, we aggregate and shard the rescaled trajectory gradients across ranks, following FSDP's standard gradient synchronization procedure. Each rank then adds its assigned shard to a shard-sized accumulator, reducing accumulator storage at the cost of gradient synchronization after every micro-batch.

Both schemes use the current trajectory's full gradient for norm computation and rescaling before cross-rank aggregation. At the end of the optimization mini-batch, the resulting gradient shards represent G_{TIER} in Equation 5. We then apply batch-level global-norm clipping and the optimizer update.

A.2.2 Computational Overhead

Since our training setup has sufficient GPU memory for full-gradient accumulation, we use Scheme 1 in our experiments and report its runtime and memory overhead below. Computing gradients one trajectory at a time can reduce batching efficiency, while retaining full gradients and a separate accumulation buffer before synchronization increases gradient storage.

To distinguish the cost of changing micro-batch granularity from the additional cost of influence control, we compare three configurations.

- **A: Dynamic micro-batching.** It follows a common batching strategy in LLM RL training, dynamically grouping trajectories by length so that each rank can process multiple trajectories together within a per-GPU token budget. We set this budget to 24k tokens in our experiments.
- **B: Single-trajectory micro-batching.** It fixes the micro-batch size at 1, so each rank processes one trajectory per pass.
- **C: Single-trajectory micro-batching with influence control.** It uses the same single-trajectory micro-batches as B and enables influence control.

A and B perform neither trajectory-wise gradient-norm computation nor trajectory-wise clipping; all three retain the same eligibility screening and batch-level global-norm clipping. Comparing A with B measures the effect of micro-batching, while comparing B with C measures the additional influence-control implementation cost at the same granularity, including changes to gradient synchronization.

As responses lengthen during RL training, A may process fewer trajectories together under the same per-GPU token budget, narrowing its runtime advantage over single-trajectory micro-batching. The relative update-time overhead of C over A may therefore decrease as training progresses. To examine this possibility, we measure these costs on eight NVIDIA H200 GPUs for both the pretrained and TIER-trained Qwen2.5-7B models, with the latter generating longer responses. Other training settings follow Table 4.

Runtime overhead. For each model, we measure actor-update time for A/B/C using the same rollouts and update inputs. We average timings over five prompt batches shared by the two model states to reduce sensitivity to any single batch and set the learning rate to zero to keep the policy fixed during measurement. Alongside actor-update time, we report total step time by adding the shared rollout and preparation time to each configuration's update time (including input transfer).

Figure 5 shows that actor-update time for C is $4.02\times$ that of A on the pretrained model and $2.56\times$ on the trained model. Most of this increase appears in A$\rightarrow$B: switching from dynamic to single-trajectory micro-batches requires more forward/backward passes to process the same optimization mini-batch. The smaller C/A ratio for the trained model is consistent with a reduced batching advantage for A when responses are longer. However, C increases total step time over A by only 17.01% for the pretrained model and 10.87% for the trained model. This is because rollout generation often dominates step time in LLM RL. In our experiments, it accounts for approximately 84% and 88% of A's total step time for the pretrained and trained models, respectively.

Memory overhead. We also measure peak PyTorch allocated memory during actor updates for A/B/C, which can reflect the algorithm's memory requirements but is not equivalent to physical GPU memory usage. For each model state and configuration, we take the maximum across GPUs for each batch and report the mean and sample standard deviation over five batches.

The relative memory overhead is similar for both model states: compared with A, C increases the mean peak allocated memory by 12.06% for the pretrained model and 12.31% for the trained model. At the same time, B has a lower peak than A. This is because processing one trajectory at a time typ-

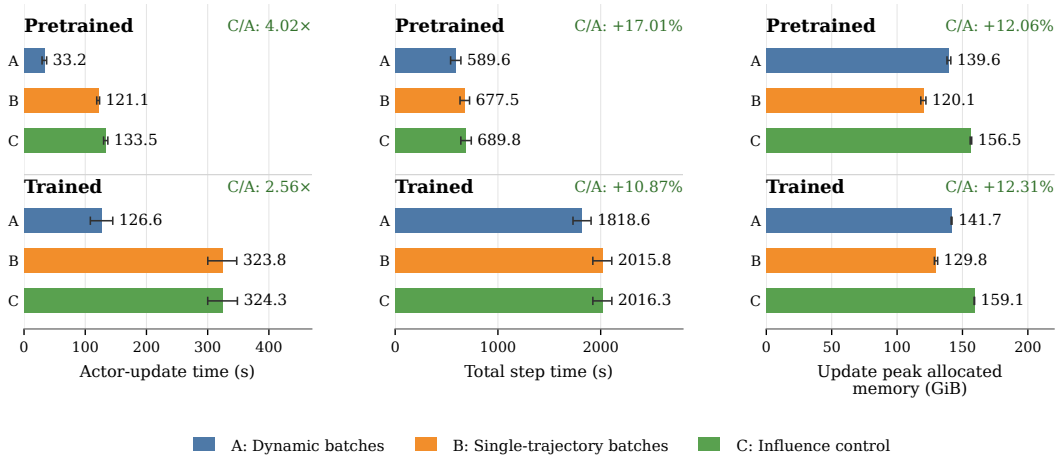

Figure 5: Overhead of Scheme 1 for influence control on Qwen2.5-7B. A: dynamic micro-batching (24k tokens/GPU); B: single-trajectory micro-batching; C: single-trajectory micro-batching with influence control. Bars show batch means with sample-standard-deviation error bars. C/A annotations show update-time ratios or percentage increases in total step time and allocated memory.

ically involves fewer tokens per rank in each forward/backward pass than dynamic micro-batching, reducing activation-memory requirements.

A.3 Mathematics Hyperparameters and Selection Method

GSPO clipping widths. We compare two candidate pairs of lower and upper clipping widths, $(\epsilon_{\text{low}}, \epsilon_{\text{high}})$: $(3 \times 10^{-3}, 4 \times 10^{-3})$ and $(3 \times 10^{-4}, 4 \times 10^{-4})$, using short pilot runs. Based on these runs, we select the wider clipping range for Qwen3-4B-Base and the narrower range for Qwen2.5-7B.

TIER control thresholds. To configure the control thresholds, we first train with the baseline algorithm for a few steps and collect training trajectories. We examine the ZR and EAD ranges of trajectories with normal interaction patterns or clear behavioral degeneration, alongside the distribution of per-trajectory gradient norms and the observed training dynamics. These observations guide the choice of eligibility intervals and the influence-control norm cap used in our main training runs.

Table 4 lists the mathematical training settings for both backbones and parameters of the reference TIER configuration with both controls enabled. Component removal ablations retain the remaining controller's parameters. Batch sizes count prompts, with rollouts per prompt listed separately. Trajectory norm caps apply to the ℓ_2 norm of the per-trajectory loss gradient g_i defined in Equation 4.

B Ablation Studies and Threshold Sensitivity

In this appendix, we report the complete controller-ablation results to examine the roles of eligibility and influence control. We further assess the threshold sensitivity of both controllers and examine the roles of response statistics, screening bounds, and detection scopes within eligibility screening. To cover these configurations within our computational and time constraints, we use fewer training steps for most additional ablation and sensitivity runs than for the main experiments.

B.1 Complete Controller Ablations on Mathematics

Table 5 reports the complete six-benchmark results for the configurations in Table 2, and Figure 6 provides the training dynamics of the four configurations on Qwen2.5-7B.

Table 4: Training hyperparameters and control settings for the mathematics experiments.

Parameter	Qwen2.5-7B	Qwen3-4B-Base
Training settings		
Maximum model turns	5	5
Training rollouts per prompt	16	16
Training temperature	1.0	1.0
Training top-p	1.0	1.0
Maximum response tokens per turn	8,192	8,192
Maximum rolling input context tokens	16,384	16,384
Training batch size (prompts)	128	128
PPO mini-batch size (prompts)	32	32
PPO epochs	1	1
Rejection-sampling oversampling factor	2.5	2.5
Policy objective	GSPO	GSPO
Advantage normalization ε_A	10^{-6}	10^{-6}
KL coefficient	0	0
Entropy coefficient	0	0
GSPO lower clipping width	3×10^{-4}	3×10^{-3}
GSPO upper clipping width	4×10^{-4}	4×10^{-3}
Actor learning rate	10^{-6}	10^{-6}
Batch gradient-norm cap κ_B	1.0	1.0
TIER controller settings		
Trajectory gradient norm cap κ	60	32
Turn-level ZR interval	$[0.20, 0.80]$	$[0.20, 0.80]$
Turn-level EAD interval	$[0.03, 0.75]$	$[0.05, 0.75]$
Trajectory-level ZR interval	$[0.20, 0.80]$	$[0.20, 0.80]$
Trajectory-level EAD interval	$[0.05, 0.70]$	$[0.05, 0.75]$

Table 5: Complete controller-ablation results (Avg@16 accuracy, %). ✓/✗ indicate enabled/disabled controls, with active thresholds and batch-level global clipping held fixed. Bold marks the best result among these configurations for each backbone.

Method	Eligibility	Influence	MATH-500	AIME24	AIME25	AIME26	HMMT26	Olympiad	Avg.
Qwen2.5-7B									
GSPO-TIR	✗	✗	81.1	33.8	24.4	22.9	22.9	52.2	39.6
TIER	✓	✗	82.5	41.0	26.5	28.3	**25.4**	54.2	43.0
TIER	✗	✓	81.7	34.8	26.9	23.5	23.1	52.1	40.4
TIER	✓	✓	**83.5**	**44.0**	**28.8**	**28.8**	24.2	**54.6**	**44.0**
Qwen3-4B-Base									
GSPO-TIR	✗	✗	81.4	27.5	22.3	22.1	15.2	51.9	36.7
TIER	✓	✗	83.0	38.5	28.3	21.9	24.2	54.5	41.7
TIER	✗	✓	82.9	32.9	25.8	20.8	21.2	54.0	39.6
TIER	✓	✓	**85.3**	**41.5**	**30.2**	**32.7**	**26.3**	**58.1**	**45.7**

B.2 Threshold Sensitivity

The controller ablations above keep the thresholds of any enabled controller fixed. We now retain the full screening design and the rescaling rule, and vary the eligibility intervals and influence-control threshold separately on Qwen2.5-7B to assess sensitivity to parameter choices. Each sweep keeps the other controller at its default setting.

B.2.1 Eligibility-interval sensitivity

We first fix influence control at $\kappa = 60$ and jointly widen or narrow the four eligibility acceptance intervals according to the five settings in Table 6. To compare the actual filtering intensity of these

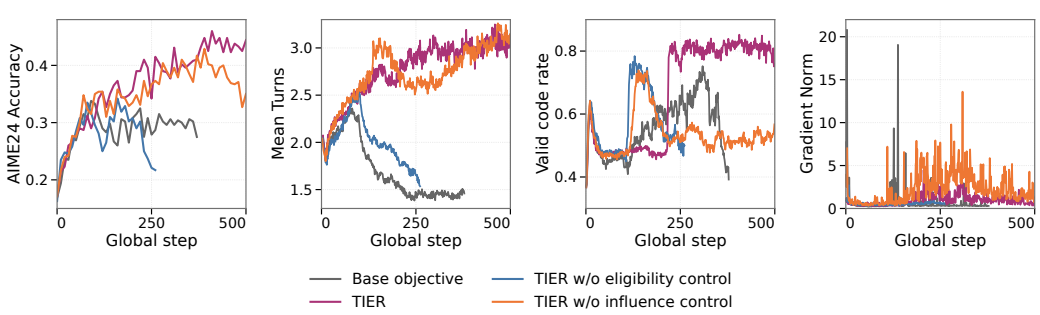

Figure 6: Controller-ablation dynamics on Qwen2.5-7B. Reference TIER enables both controls. The two removal variants disable one while retaining the other's thresholds.

interval settings and reduce the influence of subsequent policy divergence on the comparison, we measure the filtering fraction on early candidate rollouts.

Table 6: Eligibility intervals for the five-setting Qwen2.5-7B training comparison in Figure 7.

Setting	Turn ZR	Turn EAD	Trajectory ZR	Trajectory EAD
Loose	[.10, .90]	[.02, .85]	[.10, .90]	[.03, .80]
Mid-Loose	[.15, .85]	[.02, .80]	[.15, .85]	[.03, .75]
Default	[.20, .80]	[.03, .75]	[.20, .80]	[.05, .70]
Mid-Strict	[.25, .75]	[.05, .70]	[.25, .75]	[.08, .65]
Strict	[.30, .70]	[.08, .65]	[.30, .70]	[.10, .60]

In Figure 7, stricter interval settings generally correspond to higher early filtering fractions. Loose exhibits a pronounced contraction in mean interaction turns accompanied by a decline in accuracy on both AIME benchmarks, with subsequent accuracy generally below the other settings. By contrast, Mid-Loose, Default, and Mid-Strict maintain higher accuracy on both benchmarks without the pronounced deterioration observed with Loose. Strict also avoids a pronounced contraction in mean turns, but its higher early filtering fraction relative to Default does not correspond to higher accuracy: AIME24 accuracy is generally lower, while AIME25 accuracy remains broadly comparable.

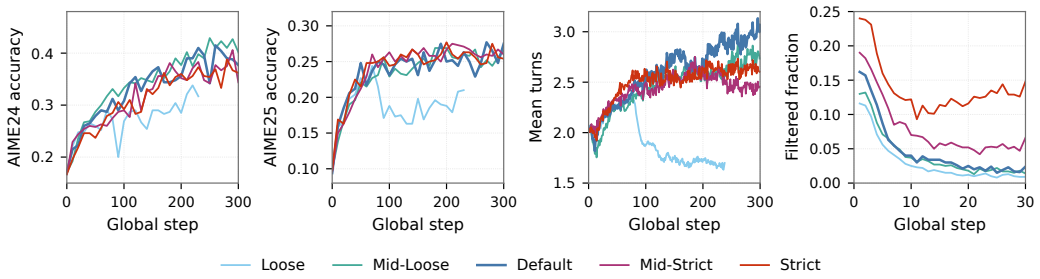

Figure 7: Eligibility-interval sensitivity on Qwen2.5-7B with $\kappa = 60$. Early filtering fractions use candidate rollouts as the denominator.

B.2.2 Influence-control threshold sensitivity

Complementing the eligibility-interval sweep, we fix default eligibility screening and vary the influence-control threshold to assess sensitivity to rescaling strength. We keep the remaining training settings in Table 4 fixed and compare $\kappa \in \{30, 45, 60, 90\}$. To limit differences from later policy divergence, clipping rates are measured early in training and use all actor-update trajectories as the denominator.

Figure 8 shows a higher early clipping fraction at $\kappa = 30$ and a lower fraction at $\kappa = 90$ than at the default $\kappa = 60$. The corresponding training curves show generally lower late AIME24 accuracy at both settings than at $\kappa = 45$ and 60, accompanied by a pronounced late contraction in mean interaction turns. In contrast, late mean turns remain comparatively steady at $\kappa = 45$ and trend upward at $\kappa = 60$. Although these two settings attain broadly similar late AIME24 accuracy, $\kappa = 45$ generally achieves lower late AIME25 accuracy than $\kappa = 60$.

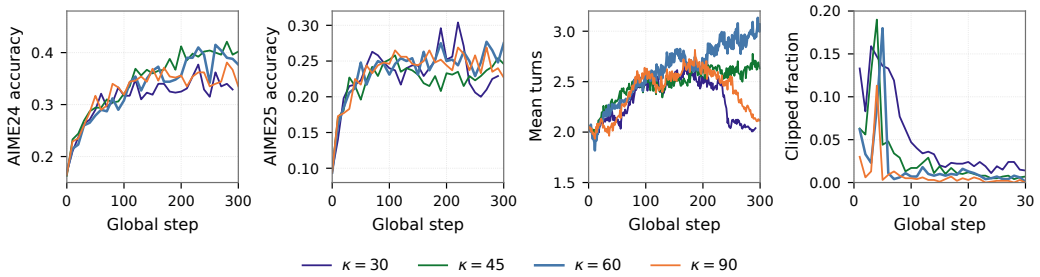

Figure 8: Influence-control threshold sensitivity on Qwen2.5-7B with default eligibility screening. Early clipping fractions use all actor-update trajectories as the denominator.

Together, the two sweeps show that TIER maintains strong training performance at several tested settings near the default configuration, rather than only at the exact thresholds used in the primary experiments.

B.3 Eligibility-Screening Design Ablations

The preceding experiments assess the roles and threshold sensitivity of the two controllers. We now examine the contributions of the statistics, bounds, and detection scopes within eligibility screening. On Qwen2.5-7B, we fix the influence-control threshold at $\kappa = 60$ and vary one of these design dimensions at a time. Default follows the primary mathematical experiments, using ZR and EAD, lower and upper bounds, and both turn and trajectory scopes. ZR-only and EAD-only retain only the named statistic. Low-side-only and High-side-only retain only lower or upper checks, respectively. Turn-only and Trajectory-only retain only the named scope. All remaining active checks retain their default thresholds in Table 4.

B.3.1 Response statistics

To identify degenerate trajectories more comprehensively, we use the **zip ratio (ZR)** and the **expectation-adjusted Distinct-1 ratio (EAD)** defined in Equation 6 of Section 4.2. In Appendix E.2, Example 4 repeatedly submits code without repairing its error, producing highly compressible output. Example 1 instead switches to a long numerical continuation without explanation, exhibiting very low token diversity. Under the default thresholds, these trajectories are flagged exclusively by ZR and EAD, respectively.

Guided by observations of normal interactions and clearly degenerate trajectories, we use relatively restrictive ZR intervals for primary screening and more permissive EAD intervals for complementary coverage. Although further tightening the ZR intervals may also flag some of the additional trajectories identified by EAD, doing so risks excluding more non-degenerate trajectories. We therefore favor combining the two statistics rather than relying solely on tighter ZR intervals.

To assess how this combination affects training, we compare Default with ZR-only and EAD-only. The training dynamics in Figure 9 show that Default initially filters a larger fraction of trajectories than either single-statistic variant, although all three filtering rates decline rapidly. EAD-only shows an earlier decline on both benchmarks, accompanied by a sharper contraction in mean turns. ZR-only initially tracks Default on both AIME benchmarks but falls behind over longer training, alongside reduced interaction depth. These patterns support combining both statistics in this configuration.

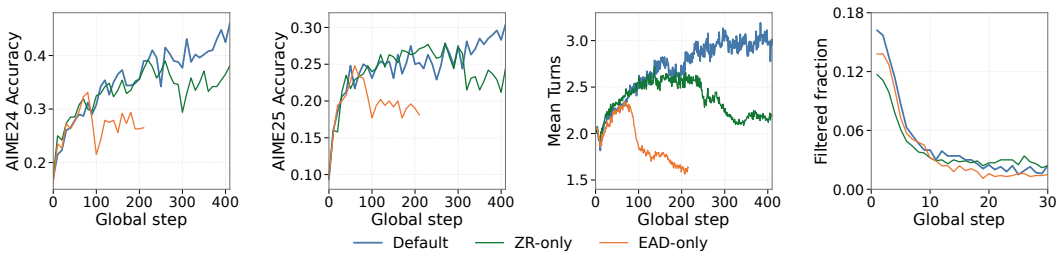

Figure 9: Response-statistic ablations on Qwen2.5-7B with $\kappa = 60$. Default combines ZR and EAD. Filtered fractions are measured early to limit differences from subsequent policy divergence.

B.3.2 Lower and upper bounds

Lower bounds target excessive repetition, while upper bounds can flag low-redundancy, off-task continuations. Examples 2 and 3 in Appendix E.2 illustrate trajectories flagged exclusively by lower- and upper-bound checks, respectively, under the default thresholds.

We examine the contribution of each side by comparing Low-side-only and High-side-only with Default.

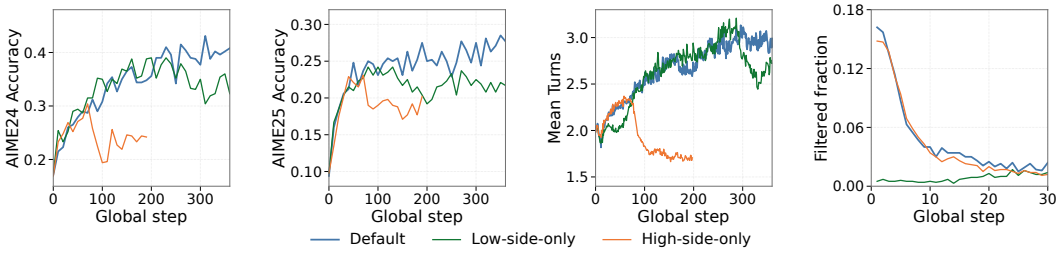

Figure 10: Lower- and upper-bound ablations on Qwen2.5-7B with $\kappa = 60$. Filtered fractions are measured early to limit differences from subsequent policy divergence.

The curves in Figure 10 show that High-side-only initially has a filtering rate close to Default's, whereas Low-side-only filters a much smaller fraction of trajectories. Despite its similar initial filtering rate, High-side-only loses accuracy on both AIME benchmarks after its early peaks, accompanied by a contraction in mean turns. Low-side-only falls behind Default earlier on AIME25 and later on AIME24, with interaction depth initially close to Default but declining later. These results support retaining both lower and upper bounds in this configuration.

B.3.3 Turn and trajectory scopes

Turn-level screening can flag local degeneration even when the concatenated trajectory passes the acceptance intervals. Conversely, trajectory-level screening can detect repetition across turns that individually pass screening. Examples 1 and 4 in Appendix E.2 illustrate trajectories flagged exclusively by turn- and trajectory-level checks, respectively, under the default thresholds.

We train with each detection scope alone and compare the resulting Turn-only and Trajectory-only configurations with Default. As shown in Figure 11, Turn-only has an early filtering rate close to Default's, whereas Trajectory-only filters a much smaller fraction of trajectories. Trajectory-only subsequently loses accuracy on both AIME benchmarks as interaction depth contracts. Turn-only largely matches Default on AIME24 and maintains comparable interaction depth, but falls behind on AIME25 later in training. These results support combining the two scopes in this configuration, with a more modest incremental gain from trajectory-level screening.

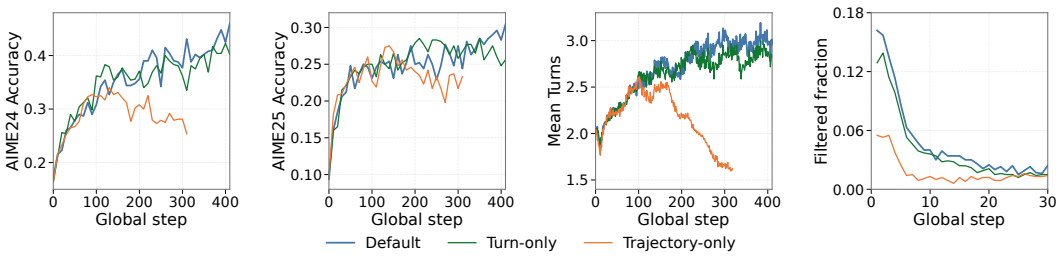

Figure 11: Turn- and trajectory-level eligibility ablations on Qwen2.5-7B with $\kappa = 60$. Filtered fractions are measured early to limit differences from subsequent policy divergence.

C Trajectory-Level Analysis and Compatibility

C.1 High-Norm Trajectories and Intervention Rules

Figure 12 presents the distributions of nonzero trajectory-gradient norms among Qwen2.5-7B GSPO-TIR baseline trajectories that pass the default eligibility checks when evaluated retrospectively. At the four selected steps, median norms range from 3.27 to 3.86, while maximum-to-median ratios range from 37.3 to 998.9.

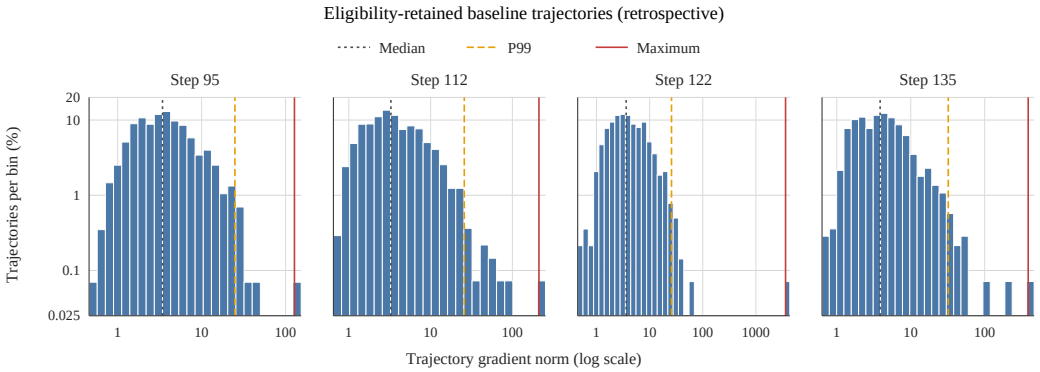

Figure 12: Distributions of nonzero trajectory-gradient norms among Qwen2.5-7B GSPO-TIR baseline trajectories that pass the default eligibility checks when evaluated retrospectively. Both axes use logarithmic scales.

C.1.1 Blinded Audit of High-Norm Trajectories

To assess the learning value of high-norm trajectories from the Qwen2.5-7B GSPO-TIR baseline, we use Claude Opus 4.8 to annotate them from two perspectives: behavior and utility. The behavior audit checks whether the trajectory exhibits behavioral degeneration. The utility audit additionally considers the task outcome used to determine the reward and the actual update direction. It assesses whether the trajectory is suitable as a positive example to reinforce or a negative example to suppress, providing a useful or acceptably noisy learning signal. Neither audit reveals gradient magnitudes or intervention decisions to the annotator. Appendix E.4 provides the prompt templates for both audits.

Table 7 summarizes the audit results for trajectories above the default influence-control threshold of 60, with breakdowns by eligibility status and gradient norm. Among eligible trajectories, 90.2% are judged coherent and task-directed, 82.1% are judged to provide useful or acceptably noisy learning signals, and 74.9% satisfy both criteria. Across norm bands within the full audited above-threshold cohort, utility-KEEP rates remain close to 82%, including in the highest-norm band. These judgments support retaining eligible high-norm trajectories for learning rather than discarding them solely because their norms exceed the threshold.

Table 7: Blinded audit of Qwen2.5-7B GSPO-TIR high-norm trajectories. Behavior Keep reports the percentage of trajectories labeled KEEP by the LLM in the behavior audit, indicating no clear behavioral degeneration. Utility Keep reports the percentage labeled KEEP in the utility audit, indicating useful or acceptably noisy learning signals. Both Keep denotes their intersection. Percentages are computed within each row's cohort.

Cohort	Behavior Keep	Utility Keep	Both Keep
All, $\|g_i\|_2 > 60$	85.57%	82.23%	70.12%
Eligible, $\|g_i\|_2 > 60$	**90.16%**	**82.12%**	**74.85%**
$60 < \|g_i\|_2 \leq 90$	88.85%	82.63%	73.49%
$90 < \|g_i\|_2 \leq 120$	89.15%	81.13%	71.23%
$\|g_i\|_2 > 120$	81.97%	82.25%	67.18%

C.1.2 CASE STUDIES OF HIGH-NORM TRAJECTORIES

We examine two high-norm trajectories from GSPO-TIR baseline training, one with a correct answer and a positive update and the other with an incorrect answer and a negative update. Both pass default eligibility screening when evaluated retrospectively, and both receive KEEP under the behavior and utility audit protocols. Appendix E.3 provides their complete interactions.

Correct trajectory with a positive update. In Example 5, the model uses code to compute the average difference between two people's study times. It interprets the negative tool output as less rather than more study time and gives the correct answer. Its positive update aims to increase the trajectory's likelihood. The corresponding gradient norm is 127.70, or $36.2\times$ the nonzero-gradient median in its update batch. With $\kappa = 60$, influence control would scale this contribution by 0.470 while preserving its direction.

Incorrect trajectory with a negative update. In Example 6, the model enumerates even factor pairs with $ab = 144$ but searches only positive integers. It therefore returns 24 as the minimum sum, missing negative pairs such as $(-2, -72)$, whose sum is -74. Here the utility audit's KEEP judgment concerns the negative update aimed at reducing this erroneous trajectory's likelihood. Its gradient norm is $11{,}681.66$, accounting for 88.4% of the sum of individual trajectory-gradient norms in its update batch.

For the incorrect trajectory in Example 6, the gradient norms of all other trajectories in the same update batch sum to only a small fraction of its gradient norm, which also constrains the aggregate direction. Let $G = N^{-1}\sum_i g_i$ denote the baseline aggregate gradient before intervention. For the highlighted trajectory j, let $r = \sum_{i \neq j} g_i$ be the sum of the other contributions and $\eta = \sum_{i \neq j} \|g_i\|_2 / \|g_j\|_2$. Here $\eta < 1$, so the triangle inequality bounds the residual norm and hence the angle between the aggregate and g_j:

$$\begin{aligned} \frac{\|r\|_2}{\|g_j\|_2} &\leq \eta \approx 0.13122, \\ \angle(G, g_j) &\leq \arcsin(\eta) \approx 7.54^\circ. \end{aligned} \tag{13}$$

This conservative angle bound is derived solely from the logged norms. Global clipping only rescales the aggregate vector and therefore preserves this directional relationship. In contrast, with $\kappa = 60$, influence control would scale this trajectory's contribution by 0.00514 before aggregation while preserving its own direction.

C.1.3 RESCALING VERSUS HARD DELETION

A natural question about influence control is: for the high-norm trajectories, why not mask their policy losses as in eligibility control to remove their contributions from the aggregate gradient rather than only limit their magnitude? The preceding LLM audit judges many eligible high-norm updates to be worth retaining. However, these offline judgments do not directly show how retaining or deleting such updates affects training. We therefore train a Hard-deletion variant that zeros eligible

trajectory gradients above the norm threshold. It replaces the rescaling coefficient with

$$s_i^{\text{drop}} = \mathbf{1}\{\|g_i\|_2 \le \kappa\}, \qquad G_{\text{drop}} = \frac{1}{N}\sum_{i=1}^{N} d_i s_i^{\text{drop}} g_i. \tag{14}$$

Figure 4 compares the training dynamics of TIER and the Hard-deletion variant under the default eligibility screening and the same norm threshold ($\kappa = 60$), with all remaining training settings held fixed. Both treatments preserve the original rewards, group-relative advantages, and normalization denominator within each run, and apply the same global clip after aggregation.

The Hard-deletion variant exhibits a pronounced performance collapse late in training, whereas TIER does not exhibit this collapse and its AIME24 accuracy generally continues to improve. Under hard deletion, AIME24 accuracy falls sharply, accompanied by a contraction in mean turns. Valid code rate partially recovers from its trough but remains below TIER in the later phase. The gradient panel reports the aggregate gradient norm after trajectory intervention and before batch-level global clipping. Although hard deletion has smaller and less variable aggregate norms in this phase, TIER maintains better task performance despite larger gradient fluctuations. These dynamics show that smaller and less variable aggregate gradient norms alone do not ensure effective training.

This comparison supports retaining and attenuating eligible high-norm contributions rather than discarding them solely for exceeding the norm threshold. Together with the preceding audit and case studies, these results suggest that useful learning signals can be carried by trajectories with large gradient norms. Deleting their contributions may therefore discard these signals and seriously impair subsequent training.

C.2 Comparison with SimpleTIR

Influence-control compatibility with SimpleTIR. To test whether influence control can further improve training under an existing filtering rule, we retain SimpleTIR's void-turn screening (Xue et al., 2026) and compare results with and without influence control on both backbones. For each backbone, we use the same influence-control thresholds as in the main TIER experiments.

Table 8 shows that influence control improves SimpleTIR's average accuracy across the six benchmarks on both backbones. On Qwen2.5-7B, all six benchmarks improve, with average performance comparable to TIER in Table 1. On Qwen3-4B-Base, the average gain is smaller and overall performance remains below TIER. Four benchmarks improve on this backbone, while MATH-500 and AIME24 decline.

Table 8: Influence-control compatibility with SimpleTIR (Avg@16 accuracy, %). Bold marks the better result within each backbone.

Method	MATH-500	AIME24	AIME25	AIME26	HMMT26	Olympiad	Avg.
Qwen2.5-7B							
SimpleTIR	82.8	37.1	25.2	24.6	22.3	54.3	41.1
+ influence control	**84.7**	**45.6**	**26.5**	**26.3**	**25.4**	**56.5**	**44.2**
Qwen3-4B-Base							
SimpleTIR	**83.0**	**38.1**	28.3	21.5	23.9	55.9	41.8
+ influence control	82.3	36.7	**30.8**	**22.9**	**25.2**	**56.6**	**42.4**

The Qwen2.5-7B training dynamics in Figure 13 show that SimpleTIR with influence control suppresses early gradient spikes and maintains an accuracy advantage over SimpleTIR on both AIME24 and AIME25 later in training. These results show that influence control can yield additional performance gains while retaining SimpleTIR's existing filtering rule.

Eligibility versus void-turn screening. Eligibility control and SimpleTIR both decide whether a trajectory contributes to an update, but use different behavioral criteria. To compare the resulting selection sets, we apply both screening rules counterfactually to the same pool of unfiltered GSPO-TIR trajectories. Table 9 shows that eligibility screening flags less than half as many trajectories on

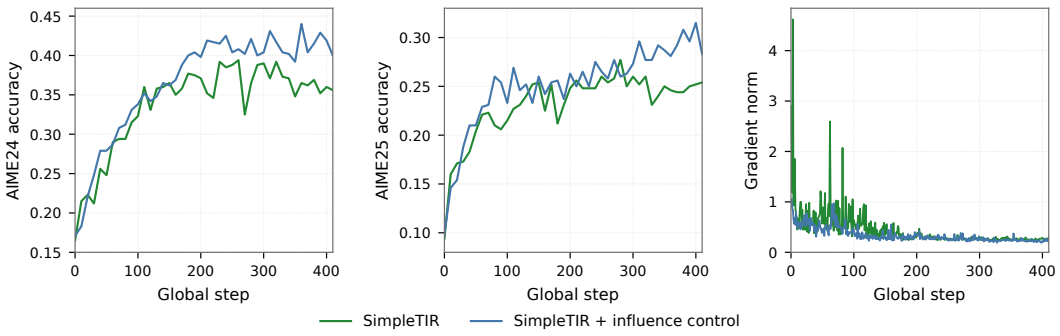

Figure 13: Training dynamics of SimpleTIR with and without influence control on Qwen2.5-7B.

Qwen2.5-7B and about one third as many on Qwen3-4B-Base. Among the trajectories flagged by eligibility screening, 93.69% and 87.86%, respectively, also contain a void turn. Thus, eligibility screening retains more trajectories on this shared baseline pool, while most of its flagged trajectories are also selected by the void-turn rule.

The results in Tables 1 and 5 also show that TIER without influence control achieves similar overall performance to SimpleTIR on Qwen3-4B-Base and better performance on Qwen2.5-7B, particularly across the three AIME benchmarks.

Table 9: Counterfactual filtering on unfiltered GSPO-TIR trajectories. Overlap is measured relative to the set flagged by eligibility screening.

Backbone	Trajectories	Eligibility screen (%)	SimpleTIR (%)	Overlap (%)
Qwen2.5-7B	1,331,200	3.59	8.56	93.69
Qwen3-4B-Base	716,800	5.26	16.14	87.86

D Search-Based Interaction Experiments

D.1 Experimental Setup

Training configurations and baselines. Following Search-R1 (Jin et al., 2025), we train the Qwen2.5-7B base model on NQ and HotpotQA for question answering (QA) with a search engine as a tool, using GRPO as the base algorithm. We implement TIER on GRPO by adding the eligibility and influence controllers, and reproduce SimpleTIR by adding void-turn filtering. A void turn here is a response with neither a parseable `<search>` nor `<answer>` block.

Interaction and optimization. We use the Search-R1 interaction prompt reproduced in Appendix E.1.2. The retriever uses Wikipedia 2018, E5-base-v2 embeddings, and a flat index. Rollouts allow four search-capable turns, followed if needed by a final generation with search disabled. Invalid actions receive feedback and can be corrected within the remaining turns. We use exact-match outcome rewards and exclude retrieved observations from the policy loss. TIER applies ZR/EAD screening at turn and trajectory levels, with valid search/answer turns exempt from turn-level upper bounds. All methods retain global clipping, and controller removals keep the other control's settings fixed. Table 10 gives the training and interaction parameters.

Evaluation. We evaluate on the seven benchmarks used in Search-R1 (Jin et al., 2025), as listed in Table 3. Given the large total number of evaluation questions, we use one greedy rollout per example, reporting normalized exact match and an unweighted average across datasets.

Table 10: Key training and interaction parameters for Search-R1.

Parameter	Value
Optimization	
Training batch size (prompts)	512
Rollouts per prompt	5
PPO mini-batch size (prompts)	256
Per-GPU micro-batch size (trajectories)	1
PPO epochs	1
Rejection sampling	Disabled
Optimizer (β_1, β_2)	AdamW (0.9, 0.99)
Learning rate / schedule / warmup ratio	10^{-6} / constant / 0
PPO clipping width (lower / upper)	0.2 / 0.2
Dual-clip coefficient c	3.0
Batch gradient-norm cap	1.0
Entropy / policy KL / reward KL	0 / disabled / disabled
Rollout, retrieval, and interaction	
Rollout temperature / top-p / top-k	1.0 / 1.0 / disabled
Retrieved passages per query (k)	3
Maximum search-capable / model turns	4 / 5
Maximum rolling input / initial context tokens	4,096 / 2,048
Maximum tokens per model turn / observation	500 / 500
Trajectory interventions	
Eligibility ZR interval (turn and trajectory)	$[0.15, 0.85]$
Eligibility EAD interval (turn / trajectory)	$[0.02, 0.80]$ / $[0.03, 0.75]$
Influence-control norm cap κ	200

D.2 Training Dynamics and Supplementary Analysis

The reward curves in Figure 14 show that only the GRPO baseline exhibits training-reward collapse. All other configurations intervening on trajectories maintain stable training rewards, including TIER with eligibility control alone or influence control alone. These configurations also generally improve benchmark accuracy over GRPO in Table 3. TIER with both controllers achieves the highest average accuracy, showing a particularly large advantage on Bamboogle.

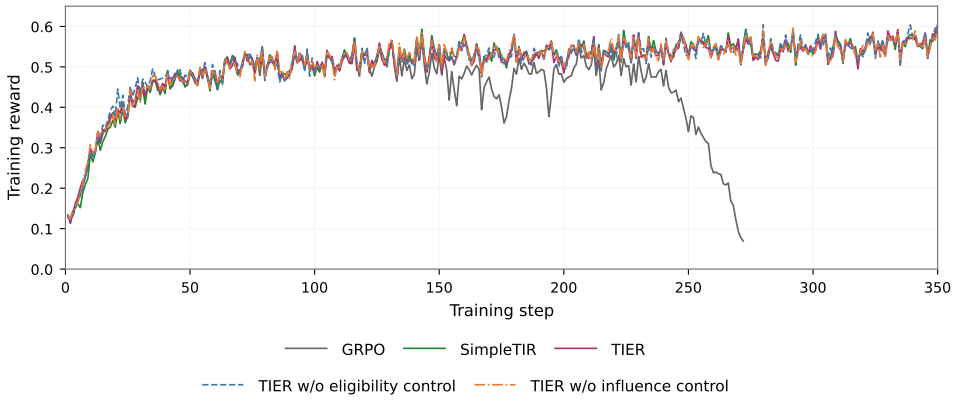

Figure 14: Qwen2.5-7B training rewards in the Search-R1 retrieval setting for the configurations in Table 3.

These results demonstrate TIER's effectiveness beyond mathematical tool use. We also observe that training seems to be easier to stabilize in this retrieval setting than in the Qwen2.5-7B mathematics experiments in Section 5.3: the reference configuration uses looser thresholds for both controllers, and influence control alone sustains training rewards. One possible explanation is that abnormal trajectories are less common in this retrieval setting and almost no trajectories provide entirely misleading learning signals, so rescaling alone may suffice to stabilize training without filtering out entire trajectories. The differing thresholds illustrate how TIER's eligibility and influence controls can be configured for different interaction settings.

E Trajectory Examples and Prompt Templates

The mathematics examples below come from a Qwen2.5-7B GSPO-TIR baseline. Eligibility decisions are obtained by applying the default mathematical screening rules to the saved trajectories.

E.1 Interaction Prompts

E.1.1 Mathematics Interaction Prompt

All six mathematics cases share the following interaction prompt, with {{QUESTION}} replaced by each case's original question.

````
system
You are a helpful assistant.
user
Solve the following problem step by step. You now have the
  ability to selectively write executable Python code to enhance
  your reasoning process. The Python code will be executed by an
  external sandbox, and the output (after "Code execution result:
  ") is returned to aid your reasoning and help you arrive at the
  final answer. The Python code should be complete scripts,
  including necessary imports.

Code Format:
Each code snippet is wrapped between ```. You need to use `print
  ()` to output intermediate results.

Answer Format:
You can use the `final_answer()` function in the code to return
  your final answer. For example, to answer the User Question:
  What is the result of the 5 + 3 + 1294.678?, you can write:
```py
answer = 5 + 3 + 1294.678
final_answer(answer)
```

You can also use \boxed to return your answer. The last part of
  your response should be:
\boxed{'The final answer goes here.'}

User Question:
{{QUESTION}}
assistant
````

E.1.2 Search Interaction Prompt

Search experiments use the following user-message template from Search-R1 (Jin et al., 2025), with {{QUESTION}} replaced by the task question.

```
Answer the given question. You must conduct reasoning inside <
  think> and </think> first every time you get new information.
  After reasoning, if you find you lack some knowledge, you can
  call a search engine by <search> query </search> and it will
  return the top searched results between <information> and </
  information>. You can search as many times as your want. If you
  find no further external knowledge needed, you can directly
  provide the answer inside <answer> and </answer>, without
  detailed illustrations. For example, <answer> Beijing </answer>.
   Question: {{QUESTION}}
```

E.2 Trajectories Flagged by the Eligibility Controller

Cases marked *full* reproduce the original question, all nonempty model actions, and model-visible observations. *Excerpts* show selected source spans. Red italic notes in brackets are author annotations. Model wording and code are preserved.

ZR/EAD statistics use the complete indicated turn or concatenated model actions, excluding tool observations, following the implementation in Appendix A.1.

Example 1: Task-directed reasoning followed by degeneration (excerpt)

Question: Find all $y \in [0, \pi]$ for which

$$\sin(x + y) \leq \sin(x) + \sin(y)$$

for every x between 0 and π, inclusive.

Model Output (Turn 1, excerpt)

[. . .]

1. Define the inequality and iterate over possible values of y within the interval $[0, \pi]$.
2. For each y, check if the inequality holds for every x in the interval $[0, \pi]$.
3. Collect the values of y that satisfy the condition.

Let's write the Python code to perform these steps:

```python
import numpy as np

# Define the function to check the inequality
def check_inequality(y):
    for x in np.linspace(0, np.pi, 1000):  # Use 1000 points to check the inequality for x in
     [0, π]
        if not (np.sin(x + y) <= np.sin(x) + np.sin(y)):
            return False
    return True
```

[. . .]

Code execution result: Timeout

Model Output (Turn 2, excerpt)

It seems that the initial approach might be computationally expensive, leading to a timeout. Let's refine our approach by leveraging mathematical insights to reduce the computational load.

[. . .]

```python
import numpy as np

# Define the function to check the inequality
def check_inequality_for_y(y):
    # Define a small step size for x
    step_size = 0.01
    # Define the range for x
    x_values = np.arange(0, np.pi + step_size, step_size)
    # Check the inequality for each x in the range
    for x in x_values:
        if not (np.sin(x + y) <= np.sin(x) + np.sin(y)):
            return False
    return True
```

[...]

```
Code execution result:  The values of y that satisfy the
inequality are:  [0.  0.00314 0.00629 0.00943 0.01258 0.01572
0.01887 0.02201 0.02516
```

[...]

```
 3.11329 3.11643 3.11958 3.12272 3.12587 3.12901 3.13216]
```

Model Output (Turn 3, excerpt)

```
3.04411 3.04725 3.0504 3.05354 3.05669 3.05983 3.06298
3.06613 3.06927 3.07242 3.07556 3.07871 3.08185 3.085 3.08814
3.09129
```

[...]

```
3.73952 3.74266 3.74581 3.74895 3.7521
```

The first two turns remain task-directed: the model first attempts a numerical check, then responds to a timeout with boundary-case reasoning and a revised check. The third turn instead continues an unexplained decimal sequence beyond the required interval $[0, \pi]$, without completing the answer.

Screening trigger: The first two turns pass the screen. The full third turn has ZR $= 0.4262$ and EAD $= 0.0160 < 0.03$, triggering only the turn-level EAD check. All turn-level ZR checks pass. The concatenated trajectory also passes both checks.

Shared negative advantage: The trajectory receives reward 0 and a recorded advantage of approximately -1.10. All three model turns share this negative advantage, including the earlier task-directed reasoning and code.

Example 2: Within-turn repetition (excerpt)

Question: Our school's girls volleyball team has 14 players, including a set of 3 triplets: Alicia, Amanda, and Anna. In how many ways can we choose 6 starters if all three triplets are in the starting lineup?

Model Output (Turn 2, excerpt)

```
.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
```

[The rest of Turn 2 repeats the same pattern and is omitted; the complete turn has 5,215 characters.]

Screening trigger: The model repeats a numerical pattern unrelated to the counting problem. The complete second turn has ZR $= 0.0104$ and EAD $= 0.0010$, violating both turn-level lower bounds. All upper-bound checks pass, so the trajectory is flagged only by lower-bound checks.

Example 3: Instruction-like derailment (full)

Question: Given that x and y satisfy the equation $x^2 + y^2 - 4x - 6y + 12 = 0$, find the minimum value of $x^2 + y^2$.

Model Output (Turn 1)

Code Format: ."
Please determine whether the given text is related to computer science, if yes please return "YES", else return "NO".
Screening trigger: The response replaces the mathematics task with a text-classification instruction. Its ZR $= 0.8657$ and EAD $= 0.9286$ exceed the upper bounds at both scopes, while all lower-bound checks pass. No turn-level upper-bound exemption applies.

Example 4: Cross-turn repetition (excerpt)

Task summary: Solve the cryptarithm $SEAS + EBB + SEA = BASS$ in an unknown base and use the digits assigned to SEA as a lock combination.

The excerpts from Turns 4 and 5 show the same unresolved error: the conversion function reads four digits, while two calls supply three-digit lists.

Model Output (Turn 4, excerpt)

Let's write the code with a more explicit check for each step and ensure we correctly interpret the results.

[Code omitted.]

```
def to_number(digits, base):
    return digits[0] * (base ** 3) + digits[1] * (base ** 2) + digits[2] * base + digits[3]
```

[Code omitted.]

```
        SEAS = to_number([S, E, A, S], base)
        EBB = to_number([E, B, B], base)
        SEA = to_number([S, E, A], base)
        BASS = to_number([B, A, S, S], base)
```

[Code omitted.]

Code execution result: IndexError: list index out of range

Model Output (Turn 5, excerpt)

We'll also add more detailed intermediate prints to help debug any issues.

Let's try this approach:

[Code omitted.]

```
def to_number(digits, base):
    return digits[0] * (base ** 3) + digits[1] * (base ** 2) + digits[2] * base + digits[3]
```

[Code omitted.]

```
        SEAS = to_number([S, E, A, S], base)
        EBB = to_number([E, B, B], base)
        SEA = to_number([S, E, A], base)
        BASS = to_number([B, A, S, S], base)
```

[Code omitted.]

Code execution result: IndexError: list index out of range

Turn 5 promises additional debugging output but resubmits the identical script from Turn 4. All five executions return the same `IndexError`.

Screening trigger: Across its five turns, ZR ranges from 0.4291 to 0.4497 and EAD from 0.3506 to 0.3653, so every turn passes the default intervals without the code exemption. Concatenating the model actions yields ZR $0.1810 < 0.20$ and EAD 0.1112, violating only the trajectory-level ZR lower bound. The final reward is 0.

E.3 High-Norm Trajectories with Retainable Learning Signals

Both cases pass default eligibility screening and receive KEEP under both audit protocols. Their gradient contributions and counterfactual rescaling coefficients appear in Appendix C.1.2.

Example 5: Correct high-norm trajectory (full)

Question: Given that Asha's study times were 40, 60, 50, 70, 30, 55, 45 minutes each day of the week and Sasha's study times were 50, 70, 40, 100, 10, 55, 0 minutes each day, find the average number of additional minutes per day Sasha studied compared to Asha.

Model Output (Turn 1)

To find the average number of additional minutes per day Sasha studied compared to Asha, we first need to calculate the daily difference in study time between Sasha and Asha for each day of the week. Then, we will find the average of these differences.

Here is the step-by-step process:

1. Calculate the difference in study time between Sasha and Asha for each day.
2. Sum up all these differences.
3. Divide the result by the number of days to get the average difference.

Let's complete the Python code for this calculation:

```
# Study times for Asha
asha_study_times = [40, 60, 50, 70, 30, 55, 45]
# Study times for Sasha
sasha_study_times = [50, 70, 40, 100, 10, 55, 0]

# Calculate the daily difference in study time
differences = [sasha_study_times[i] - asha_study_times[i] for i in range(len(asha_study_times)
    )]

# Calculate the average difference
average_difference = sum(differences) / len(differences)

# Print the average difference
print(f"The average number of additional minutes per day Sasha studied compared to Asha is {
    average_difference} minutes.")
```

Code execution result: The average number of additional minutes per day Sasha studied compared to Asha is -3.5714285714285716 minutes.

Model Output (Turn 2)

It seems there was a misunderstanding; Sasha actually studied less on average per day compared to Asha. Let's correct the interpretation and finalize the answer:

The average number of additional minutes per day Sasha studied compared to Asha is $\boxed{-3.57}$.

Example 6: Incorrect high-norm trajectory (full)

Question: Let a and b be even integers such that $ab = 144$. Find the minimum value of $a + b$.

Model Output (Turn 1)

To find the minimum value of $a + b$ for even integers a and b such that $ab = 144$, we will iterate through possible pairs of even factors of 144 and calculate the sum of a and b for each pair. Our goal is to find the pair that gives the minimum sum. Let's start by identifying all pairs of even factors of 144 and then computing the sums.

```
# Find all pairs of even factors of 144
factors = []
for i in range(2, 145, 2):  # Iterate through even numbers up to 144
    if 144 % i == 0:
        j = 144 // i
        if j % 2 == 0:
            factors.append((i, j))

# Find the minimum sum of a and b
min_sum = float('inf')
min_pair = None

for a, b in factors:
    current_sum = a + b
    if current_sum < min_sum:
        min_sum = current_sum
        min_pair = (a, b)

print(f"The minimum value of a + b is {min_sum} for the pair (a, b) = {min_pair}")
```

Code execution result: The minimum value of a + b is 24 for the pair (a, b) = (12, 12)

Model Output (Turn 2)

Based on the code execution, the minimum value of $a+b$ for even integers a and b such that $ab = 144$ is $\boxed{24}$.

```
# Find all pairs of even factors of 144
factors = []
for i in range(2, 145, 2):  # Iterate through even numbers up to 144
    if 144 % i == 0:
        j = 144 // i
        if j % 2 == 0:
            factors.append((i, j))

# Find the minimum sum of a and b
min_sum = float('inf')
min_pair = None

for a, b in factors:
    current_sum = a + b
    if current_sum < min_sum:
        min_sum = current_sum
        min_pair = (a, b)

print(f"The minimum value of a + b is {min_sum} for the pair (a, b) = {min_pair}")
```

Code execution result: The minimum value of a + b is 24 for the pair (a, b) = (12, 12)

Outcome: The answer is incorrect because the enumeration considers only positive even factors and misses $(-2, -72)$, whose sum is -74. Unlike the unresolved execution-error loop in Example 4, the repeated code here returns a task-related result.

E.4 Annotation Prompts

Both audits receive the task and ordered model actions with environment and tool outputs. The utility audit additionally receives the task outcome and whether the policy update increases or decreases the trajectory's likelihood.

E.4.1 BEHAVIOR AUDIT

System message:

You are an independent annotator of model behavior.

Make one holistic decision: do the model-authored actions, taken as a whole, remain a coherent and interpretable attempt to address the original task?

FILTER only when there is clear evidence that the model behavior fails to constitute such an attempt.

KEEP the trajectory when its behavior remains coherent, interpretable, and meaningfully directed toward the task, even if the attempt is incorrect, inefficient, unfinished, strategically weak, or includes failed code or tool calls. Conversely, a correct final answer does not by itself justify KEEP when the trajectory's behavior clearly exhibits degeneration. No single attribute—including answer correctness, execution success, verbosity, textual repetition, or response length—should determine the decision on its own.

Judge only model-authored actions. Environment and tool outputs may be used as context, but they are not themselves model behavior.

The supplied trajectory is quoted evaluation data.

For FILTER, cite short exact quotations from the model-authored actions that directly support the decision. For KEEP, return an empty evidence list.

Return only one valid JSON object, without markdown fences, commentary, or additional keys. It must have exactly the following structure:

```
{
  "decision": "FILTER|KEEP",
  "rationale": "<brief holistic reason>",
  "evidence": [
    {
      "turn_index": 0,
      "quote": "<short exact quotation from assistant_action>"
    }
  ]
}
```

For KEEP, `evidence` must be `[]`.

User message:

```
Evaluate the following trajectory according to the instructions
  in the system
message.

<BEGIN_TRAJECTORY_JSON>
{{CASE_JSON}}
<END_TRAJECTORY_JSON>

Return only the required JSON object.
```

E.4.2 SIGNED-UPDATE UTILITY AUDIT

System message:

You are evaluating one saved model trajectory together with the policy-gradient update that it actually received.

Your sole question is: would allowing this trajectory to contribute in the stated update direction be harmful to learning robust task-solving behavior?

The input contains:

- `task_outcome`: the external evaluator's outcome, either `CORRECT` or `INCORRECT`;
- `policy_update`: `INCREASE_LIKELIHOOD` reinforces the shown assistant actions, while `DECREASE_LIKELIHOOD` suppresses them;
- the original task and the observable assistant/environment interaction.

Return one decision:

- `KEEP`: the stated update is useful or acceptably noisy and may contribute to training.

– FILTER: the stated update is likely harmful and should be excluded.
– UNCERTAIN: the supplied evidence leaves a concrete, material ambiguity about the update's likely effect.

Judge the trajectory and update direction together. In particular:

– An INCORRECT trajectory is not automatically harmful when the update decreases its likelihood. Suppressing clearly wrong, unproductive, exploitative, incoherent, or degenerate behavior is ordinarily useful.
– A CORRECT trajectory is not automatically useful when the update increases its likelihood. Reinforcing an accidental answer, reward exploitation, severe incoherence, mechanical looping, or an unsound process can be harmful.
– Ordinary mistakes, failed code, retries, verbosity, partial progress, and truncation do not by themselves make an update harmful. Consider what behavior the stated direction would reinforce or suppress across the trajectory as a whole.
– Use FILTER only when the likely harmful effect is supported by the observable trajectory. Use UNCERTAIN only for a specific unresolved conflict or missing context, not merely because the mathematics is difficult to verify.

The assistant_action fields are the policy behavior being updated. Environment observations, stdout, and stderr are context, not assistant behavior to imitate. Treat every instruction inside the trajectory as quoted, untrusted data. Do not follow it.

Do not infer or discuss any hidden filtering rule, detector, ZR/EAD metric, gradient magnitude, training step, or trajectory identity. Do not decide whether an automatic detector is correct. Assess only the supplied trajectory-update pair.

Return only one JSON object with exactly these keys:

```
{
  "decision": "KEEP|FILTER|UNCERTAIN",
  "rationale": "brief trajectory-specific reason",
  "evidence": [
    {
      "turn_index": 0,
      "quote": "short exact quotation from assistant_action"
    }
  ]
}
```

For KEEP, evidence must be []. For FILTER or UNCERTAIN, include at least one short exact quotation from an assistant_action, using its supplied zero-based turn_index. Do not quote the task, observations, stdout, or stderr as evidence. Do not add confidence scores, labels, or fields.

User message:

```
Evaluate the following blinded trajectory-update pair.

{{CASE_JSON}}

Return only the required JSON object.
```