THINK OR ACT? STATE-CONDITIONED REPRESENTATION STEERING FOR STRATEGIC TOOL USE IN LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents increasingly rely on external tools to access information beyond internal reasoning, yet tool use is not uniformly beneficial: a tool call can correct failed internal reasoning, but it can also introduce noise, increase cost, or corrupt an otherwise correct answer. We study this problem as a *Think-or-Act* decision: whether an agent should rely on internal reasoning or invoke an external tool for a given input. To evaluate this decision, we introduce **ToA-Eval**, a model-specific protocol that contrasts executions without and with tool use to identify when tool use helps or harms a particular agent. To control the decision, we propose **ToA-Steer**, a training-free, cluster-routed steering controller that extracts a shared THINK/ACT direction from only 50 contrastive completion pairs and dynamically selects input-specific steering strengths in latent space, without any parameter updates. Across four reasoning benchmarks and four open-weight LLMs, ToA-Steer achieves the highest accuracy in all 16 model–benchmark settings, outperforming prompt-based and static-steering baselines and even SFT and GRPO trained on the same labels; on Qwen3-8B, it raises mean accuracy from 0.585 to 0.723 with 3.5 points above GRPO. It improves accuracy by 16 points on ACT-needed instances and 9 points on THINK-preferred instances at the same time. Without any parameter updates, ToA-Steer beats RL-trained GRPO on selective tool calling in every setting. Further analysis shows that THINK/ACT signals concentrate at a consistent relative depth of 0.71–$0.72\,L$ across all four models, and that optimal steering is cluster-dependent, supporting instance-conditioned rather than globally fixed tool-use control.

1 INTRODUCTION

Large language models are increasingly deployed as interactive agents that can call external tools such as calculators, code interpreters, search engines, and retrieval systems (Yao et al., 2024; Yang et al., 2024; Jin et al., 2025). While these tools extend an agent's capabilities beyond internal reasoning, they are not uniformly beneficial: indiscriminate tool calling can waste compute and corrupt otherwise correct answers (Qian et al., 2025b; Xu et al., 2025). A tool call can fix a failed reasoning trajectory when computation or external lookup is needed, but it can also add latency, introduce noisy observations, or corrupt an answer the model would have produced correctly on its own. The bottleneck for capable agents is therefore not only tool competence, but tool-use selectivity: deciding when to rely on internal reasoning and when to invoke an external tool.

Prior studies have substantially advanced tool-augmented agents, but they provide limited support for studying strategic tool-use decision making. Most tool-use benchmarks focus on tool competence, evaluating whether a model can select the right tool, format the call correctly, or execute it successfully (Qin et al., 2023; Liu et al., 2024; Patil et al., 2024), rather than whether calling a tool improves the final answer. Recent tool-awareness and call-or-answer evaluations such as MetaTool and When2Call (Huang et al., 2023; Ross et al., 2025) move closer to tool-use selectivity, but they do not directly measure the bidirectional outcome effect of tool use: whether a tool call turns a wrong answer into a correct one, or corrupts an answer the model would otherwise get right. On the control side, prompting methods often shift the model's overall tendency to call tools rather than calibrating when tools help (Yao et al., 2022; Paranjape et al., 2023), while training-based methods require

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.

additional traces, feedback, compute, and parameter updates (Schick et al., 2023; Qian et al., 2025a; Feng et al., 2025). As a result, strategic tool use remains under-specified from both sides: we need an evaluation protocol that tells when tools help or harm a given agent, and a lightweight control mechanism that can adjust the THINK/ACT decision without retraining.

To evaluate strategic tool-use decisions, we introduce **ToA-Eval**, a model-specific Think-or-Act evaluation protocol. For each model and input, ToA-Eval compares two controlled executions of the same base model: one without tool access and one where the model must use a tool before answering. This construction identifies two outcome-changing cases: ACT, where the no-tool answer is wrong but the answer with tool use is correct, and THINK, where the no-tool answer is correct but the answer with tool use is wrong. Because the same question may benefit from tool use for one model but not for another, ToA-Eval derives these labels separately for each evaluated model. It therefore directly evaluates strategic tool-use control: a successful method should increase tool use on ACT instances, reduce tool use on THINK instances, and improve final answer accuracy.

We then address the control problem from a representation-intervention perspective (Turner et al., 2023; Zou et al., 2023). Rather than shaping the THINK/ACT decision through prompts, weight updates, or an external controller, we intervene directly in the model's latent space at inference time. We extract a lightweight THINK/ACT steering direction from contrastive completion pairs and use it to shift the model toward tool invocation or direct internal reasoning. However, a single static steering strength is insufficient: some inputs require movement toward ACT, while others require movement toward THINK. We therefore propose **ToA-Steer**, a dynamic cluster-routed steering controller that uses a shared THINK/ACT direction but selects an input-specific intervention through latent-space routing. This enables the same representation-level mechanism to encourage tool use when tools help and suppress tool use when tools harm, without updating model parameters.

In summary, our contributions are:

- We introduce **ToA-Eval**, a model-specific evaluation protocol for *strategic tool-use decision making* that explicitly measures when tool use helps or harms a given agent.
- We propose **ToA-Steer**, a dynamic, cluster-routed inference-time control method that selects input-specific steering strengths along a shared THINK/ACT direction.
- We show that ToA-Steer achieves the highest accuracy in all 16 model–benchmark settings, outperforming prompt-based, static-steering, and even training-based baselines, and beats RL-trained GRPO on selective tool calling without any parameter updates.
- We analyze the latent structure of THINK/ACT control, showing that decision-relevant signals concentrate at a consistent relative depth of 0.71–$0.72\,L$ across all four models and that optimal steering is cluster-dependent, providing evidence for instance-conditioned rather than globally fixed tool-use control.

2 ToA-Eval: A Think-or-Act Tool-Decision Evaluation Protocol

Existing tool-use evaluations mainly assess whether a model can select, format, or execute tool calls once tool use is assumed (Qin et al., 2023; Liu et al., 2024; Patil et al., 2025). In contrast, our focus is the prior decision of whether an agent should rely on internal reasoning or invoke an external tool for a given instance. This decision is not fixed at the question level: a problem that requires a tool for a smaller model may be solvable internally by a stronger one. We therefore introduce **ToA-Eval** (Think-or-Act Evaluation), a model-specific tool-use decision evaluation protocol that derives Think-or-Act decision labels by contrasting executions of the same base model without and with tool use. Rather than annotating an intrinsic tool need, ToA-Eval captures whether tool use changes the outcome for the evaluated model, yielding two decision-relevant cases: ACT, where tools fix a no-tool error, and THINK, where tools corrupt an otherwise correct answer.

Two-pass construction. For each source problem x_i, we run two controlled inference passes. In the *no-tool* pass, the model answers using the canonical benchmark prompt without external tools, and we record whether the final answer is correct as $c_i^{\mathrm{nt}} \in \{0, 1\}$. In the *with-tool-use* pass, the same model receives access to a fixed set of general-purpose tools and is required to invoke a tool before answering; we record correctness as $c_i^{\mathrm{ft}} \in \{0, 1\}$. Requiring a tool call in the second pass isolates the

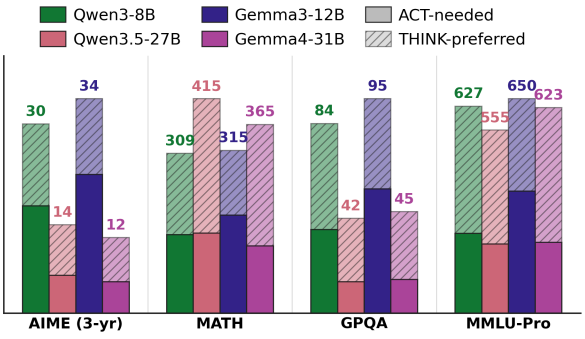


Figure 1: Breakdown of **ToA-Eval** instances across four selected source benchmarks.

Model	ACT	THINK	Total
Qwen3-8B	448	602	1050
Qwen3.5-27B	385	641	1026
Gemma3-12B	637	457	1094
Gemma4-31B	365	680	1045

Table 1: Number statistics of **ToA-Eval** across four selected base models. **ACT** denotes ACT-needed instances where tool use fixes a no-tool error. **THINK** denotes THINK-preferred instances where tool use corrupts an otherwise correct no-tool answer.

effect of using a tool: if the no-tool pass fails but the with-tool-use pass succeeds, the instance favors ACT; if the no-tool pass succeeds but the with-tool-use pass fails, the instance favors THINK.

Think-or-Act labeling. After two-pass data construction and collection, ToA-Eval keeps the two outcome-changing samples where:

- **ACT** ($c_i^{\text{nt}} = 0, c_i^{\text{ft}} = 1$): internal reasoning fails, but tool use fixes the answer. The desired decision is to invoke a tool.
- **THINK** ($c_i^{\text{nt}} = 1, c_i^{\text{ft}} = 0$): internal reasoning succeeds, but tool use corrupts the answer. The desired decision is to answer directly without relying on external tools.

The remaining cases are excluded from the main Think-or-Act evaluation: if both passes are correct, tool use is redundant for final accuracy; if both passes fail, neither decision is verifiably preferable. Thus, ACT and THINK define a decision-relevant evaluation set in which the two classes require opposite tool-use behavior, allowing ToA-Eval to evaluate strategic tool-use control: *a successful method should calibrate the model's THINK/ACT boundary by promoting tool use when it improves correctness and suppressing it when it degrades correctness.*

Source tasks and tools. We instantiate ToA-Eval from AIME2024-2026 (AIME, 2025), MATH (Hendrycks et al., 2021), GPQA (Rein et al., 2023), and MMLU-Pro (Wang et al., 2024), covering mathematical computation, symbolic reasoning, scientific knowledge, and broad expert-level multiple-choice reasoning. These source benchmarks are chosen to span both tool-beneficial settings, such as computation-heavy mathematics, and tool-risk settings, where external search or execution may distract from a correct internal answer. The with-tool-use pass exposes the same general-purpose tools across all source tasks, including Python execution and Wikipedia search, so labels are determined by whether tool use changes the evaluated model's correctness rather than by dataset-specific tool assumptions. Figure 1 summarizes the breakdown distribution for each source dataset.

Model-specific construction. Because tool necessity depends on the evaluated model, ToA-Eval is constructed separately for each base model by re-running the no-tool and with-tool-use passes and deriving ACT/THINK labels from that model's own outcomes. We evaluate four open-weight base models, Qwen3-8B (Yang et al., 2025), Qwen3.5-27B (Team, 2026), Gemma3-12B (Kamath et al., 2025), and Gemma4-31B (Google DeepMind, 2026), spanning two model families and different capability levels to test whether tool-use necessity varies across agent capabilities. We retain decision-relevant ACT and THINK instances to cover both directions of strategic tool-use control: invoking tools when they help and suppressing tools when they harm. Figure 1 and Table 1 summarize the resulting distributions, and all experiments use each model's own ToA-Eval labels. The labels are stable under repeated construction (97.3–98.9% agreement over five seeds), decoding randomness, and prompt or tool-output rewording, and the ACT/THINK contrast is driven by the content a tool returns rather than by the act of calling it (Appendix B).

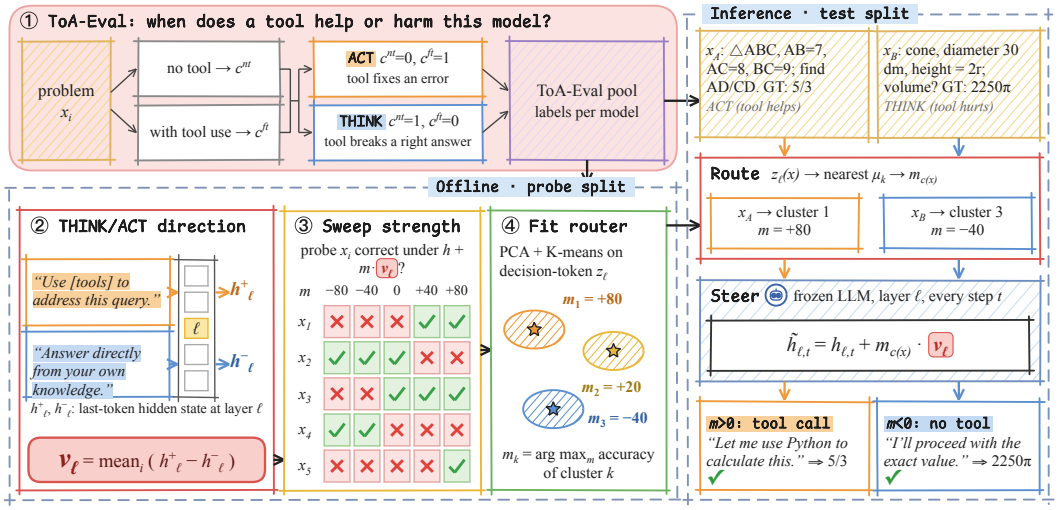


Figure 2: Overview of ToA-Eval and ToA-Steer. ① ToA-Eval labels each problem for a given model by comparing its answers without and with tool use: ACT if tool use fixes an error, THINK if it corrupts a correct answer. On the probe split, ToA-Steer ② extracts a shared THINK/ACT direction v_ℓ from contrastive completions, ③ sweeps the steering strength m, and ④ fits a PCA + K-means router that assigns each cluster the signed magnitude m_k with the best probe accuracy. At inference, each input goes to its nearest centroid and is steered by $\tilde{h}_{\ell,t} = h_{\ell,t} + m_{c(x)} v_\ell$: $m > 0$ promotes a tool call and $m < 0$ favors answering directly. Clusters and magnitudes are illustrative.

3 Methodology Design: Think-or-Act Steer

We propose **ToA-Steer**, a dynamic steering method for THINK/ACT control. At a high level, ToA-Steer first extracts a shared THINK/ACT steering direction from contrastive completion pairs that differ only in whether they endorse tool use or direct internal reasoning (Section 3.1). It then uses a cluster-routed controller to decide, for each input, whether to steer toward ACT or toward THINK and how strongly to intervene, yielding an input-specific coefficient and applying the resulting input-specific intervention to the hidden state during inference (Section 3.2).

3.1 Steering Vector Extraction

We extract a steering vector from contrastive completion pairs, following CAA (Panickssery et al., 2023). For each query x_i in the probe set, we construct two prompts that share the same `<System>` and `<User>` blocks and differ only in the final `<Assistant>` completion:

- **Positive** $(+)$**, ACT**: `<System>` a tool-aware instruction listing the available tools; `<User>` the probe query x_i; `<Assistant>` *"Use [tools] to address this query."*
- **Negative** $(-)$**, THINK**: `<System>` a tool-aware instruction listing the available tools; `<User>` the probe query x_i; `<Assistant>` *"Answer directly from your own knowledge."*

We run two forward passes, one ending with the positive completion and one ending with the negative completion, and read out the hidden state at the final completion token. Let $h_\ell^{(+)}(x_i)$ and $h_\ell^{(-)}(x_i)$ denote these last-token hidden states at layer ℓ. The steering vector is computed as the mean per-sample difference:

$$v_\ell \;=\; \frac{1}{N}\sum_{i=1}^{N}\left(h_\ell^{(+)}(x_i) - h_\ell^{(-)}(x_i)\right), \tag{1}$$

which points from the "answer-directly" representation toward the "use-the-tool" representation. Because the two forward passes share the same query and differ only in the final decision completion, this contrast reduces query-content variance and yields a decision-oriented THINK/ACT direction that is independent of the model's baseline tool-use behavior.

3.2 ToA-Steer: Cluster-Routed Dynamic Steering

Magnitude calibration. As a single global intervention strength is too coarse for THINK/ACT control, we therefore treat the intervention strength as an input-conditioned quantity. Given the THINK/ACT steering direction v_ℓ, we first calibrate a discrete set of candidate magnitudes $\mathcal{M} \subset \mathbb{R}$ on the full probe set. For each probe example x_i and each magnitude $m \in \mathcal{M}$, we replace the layer-ℓ residual at every generation step t during decoding by

$$\tilde{h}_{\ell,t} = h_{\ell,t} + m\, v_\ell, \tag{2}$$

and record the resulting final answer $\hat{a}_i^{(m)}$, which we compare with the ground-truth answer a_i^*. Positive magnitudes move the representation toward the "use tool" direction, while negative magnitudes move it toward the "answer directly" direction. This yields a calibration grid of final-answer correctness $\{\mathbf{1}[\hat{a}_i^{(m)} = a_i^*]\}_{i,\,m}$ over probe examples and candidate magnitudes, used below to assign an intervention strength to each latent region.

Latent-space routing and magnitude selection. Next, we fit the routing function. For each probe-training example, we extract a routing representation from the chosen layer $z_\ell(x) = h_{\ell,T_x}(x)$, where T_x is the index of the decision token, namely the last input token before the model emits its first token, at which the THINK/ACT decision is committed. We fit a PCA map ϕ on these representations and then run K-means on the PCA-projected probe-training representations. This produces K centroids $\{\mu_k\}_{k=1}^K$. Each held-out example x is assigned to its nearest centroid $c(x)$, and for the held-out subset $\mathcal{P}_k$ of cluster k we select the magnitude $m_k \in \mathcal{M}$ that maximizes held-out accuracy:

$$c(x) = \arg\min_{k \in \{1,\ldots,K\}} \|\phi(z_\ell(x)) - \mu_k\|_2\,, \qquad m_k = \arg\max_{m \in \mathcal{M}} \frac{1}{|\mathcal{P}_k|} \sum_{x_i \in \mathcal{P}_k} \mathbf{1}\left[\hat{a}_i^{(m)} = a_i^*\right], \tag{3}$$

where $\hat{a}_i^{(m)}$ is the final answer generated under magnitude m and a_i^* is the ground-truth answer, so each magnitude is selected by held-out final-answer accuracy rather than by agreement with ACT/THINK labels. Since m_k can be positive or negative, different clusters can be routed toward ACT or THINK using the same steering vector.

Inference-time Steering. At inference time, the routing process is fixed. For a new input x, we extract $z_\ell(x)$, project it with the fitted PCA map ϕ, assign it to the nearest centroid to obtain $c(x)$, retrieve the corresponding magnitude $m_{c(x)}$, and apply, for every generation step t,

$$\tilde{h}_{\ell,t} = h_{\ell,t} + m_{c(x)} v_\ell. \tag{4}$$

Thus, the steering direction is shared across all inputs with both the steering strength and sign selected according to the input's latent region. Since each cluster magnitude can be positive or negative, the controller can encourage tool use for ACT-needed regions while suppressing tool use for THINK-preferred regions using the same steering vector v_ℓ. More details on number of clustering selection and can be found in Appendix K.

4 Experiment

4.1 Experiment Setup

Baselines. We compare against five baselines from three families. **Base** is the unmodified model with the same tools, Python execution and Wikipedia search, exposed in the system prompt. **Prompt** additionally instructs the model to call tools only when its own reasoning is unreliable. **SFT** fine-tunes the model on tool-use traces, and **GRPO** (Shao et al., 2024) further trains it with rewards for correct THINK/ACT outcomes. **Static Steering** applies our direction v_ℓ at a fixed magnitude $m = \pm 40$, exposing the single-direction trade-off. Details are in Appendix F.

Datasets, models and metrics. We evaluate on ToA-Eval with each model's own labels (Section 2). We report **ACT**, the tool-call rate on ACT-needed instances, where higher is better; **THINK**, the tool-call rate on THINK-preferred instances, where lower is better; and final-answer **Acc**. The labeled pool is split into disjoint subsets, and *no test instance is used for steering-vector extraction or clustering* (Appendices E and G).

Table 2: Main results across models and benchmarks. **ACT** (↑) and **THINK** (↓) are tool-call rates on ACT-needed and THINK-preferred instances; **Acc** (↑) is final-answer accuracy. Row colors mark prompting, training-based, static steering, and ToA-Steer. Best in bold, second best underlined.

Method	MATH			AIME			GPQA			MMLU-Pro		
	ACT ↑	THINK ↓	Acc ↑	ACT ↑	THINK ↓	Acc ↑	ACT ↑	THINK ↓	Acc ↑	ACT ↑	THINK ↓	Acc ↑
Qwen3-8B												
Base	0.60	0.76	0.70	0.29	0.70	0.38	0.81	0.75	0.58	0.60	0.53	0.68
Prompt	0.65	0.78	0.71	0.35	0.74	0.40	0.83	0.78	0.59	0.62	0.55	0.69
SFT	0.72	0.65	0.74	0.42	0.55	0.45	0.85	0.65	0.62	0.65	0.45	0.71
GRPO	0.80	0.55	0.77	0.55	0.42	0.55	0.87	0.55	0.70	0.68	0.42	0.73
Static Steering ($m=+40$)	0.81	0.84	0.76	0.47	0.87	0.58	0.87	0.87	0.48	0.68	0.60	0.72
Static Steering ($m=-40$)	0.52	0.59	0.62	0.21	0.57	0.36	0.72	0.61	0.68	0.47	0.47	0.65
ToA-Steer (Ours)	**0.84**	**0.54**	**0.79**	**0.58**	**0.34**	**0.62**	**0.88**	**0.52**	**0.74**	**0.70**	**0.40**	**0.74**
Qwen3.5-27B												
Base	0.55	0.62	0.78	0.32	0.55	0.45	0.78	0.62	0.62	0.55	0.45	0.74
Prompt	0.59	0.65	0.78	0.36	0.58	0.46	0.80	0.65	0.62	0.58	0.48	0.74
SFT	0.64	0.55	0.80	0.42	0.45	0.50	0.81	0.55	0.65	0.60	0.40	0.76
GRPO	0.69	0.52	0.82	0.50	0.37	0.55	0.83	0.52	0.69	0.63	0.38	0.77
Static Steering ($m=+40$)	0.68	0.67	0.82	0.43	0.65	0.57	0.82	0.69	0.56	0.60	0.49	0.76
Static Steering ($m=-40$)	0.50	0.52	0.73	0.27	0.47	0.44	0.73	0.54	0.68	0.47	0.41	0.72
ToA-Steer (Ours)	**0.75**	**0.47**	**0.83**	**0.57**	**0.30**	**0.59**	**0.84**	**0.46**	**0.72**	**0.67**	**0.35**	**0.78**
Gemma3-12B												
Base	0.55	0.70	0.74	0.28	0.65	0.42	0.78	0.70	0.55	0.55	0.50	0.71
Prompt	0.58	0.71	0.74	0.32	0.66	0.43	0.79	0.71	0.55	0.57	0.51	0.71
SFT	0.65	0.60	0.78	0.41	0.49	0.52	0.81	0.60	0.62	0.60	0.44	0.73
GRPO	0.69	0.57	0.80	0.46	0.43	0.56	0.82	0.56	0.65	0.61	0.42	0.74
Static Steering ($m=+40$)	0.73	0.77	0.79	0.43	0.79	0.59	0.83	0.80	0.46	0.62	0.56	0.74
Static Steering ($m=-40$)	0.48	0.56	0.67	0.21	0.54	0.40	0.70	0.58	0.64	0.44	0.45	0.68
ToA-Steer (Ours)	**0.75**	**0.51**	**0.82**	**0.53**	**0.34**	**0.62**	**0.84**	**0.50**	**0.69**	**0.64**	**0.39**	**0.76**
Gemma4-31B												
Base	0.55	0.60	0.82	0.35	0.50	0.48	0.76	0.58	0.65	0.55	0.42	0.78
Prompt	0.59	0.62	0.82	0.39	0.52	0.49	0.78	0.60	0.65	0.57	0.44	0.78
SFT	0.61	0.54	0.85	0.43	0.41	0.54	0.78	0.52	0.69	0.58	0.38	0.79
GRPO	0.63	0.52	0.86	0.46	0.37	0.56	0.79	0.50	0.71	0.59	0.37	0.80
Static Steering ($m=+40$)	0.66	0.64	0.85	0.44	0.59	0.58	0.79	0.64	0.60	0.59	0.46	0.80
Static Steering ($m=-40$)	0.51	0.51	0.78	0.31	0.43	0.47	0.71	0.51	0.70	0.48	0.39	0.76
ToA-Steer (Ours)	**0.67**	**0.49**	**0.87**	**0.50**	**0.32**	**0.60**	**0.80**	**0.46**	**0.73**	**0.60**	**0.35**	**0.81**

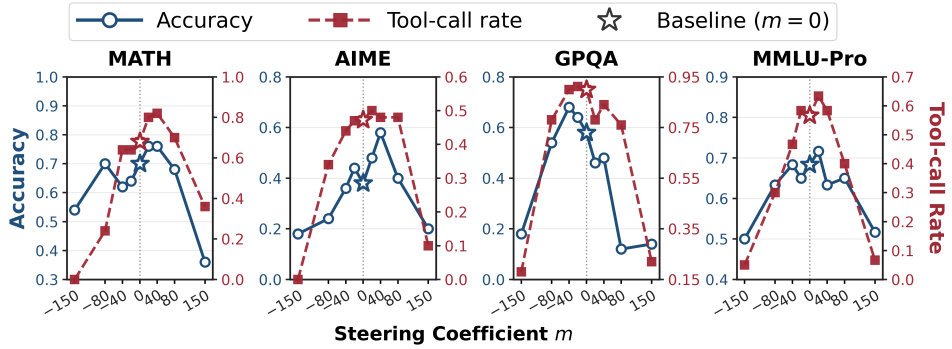


Figure 3: Final-answer accuracy and tool-call rate as the steering coefficient m is swept on Qwen3-8B across the four ToA-Eval benchmarks. Within a moderate range of $|m|$ the tool-call rate increases smoothly with m and accuracy peaks away from the no-steer baseline (m=0, star); at large $|m|$ the activation push moves the response off-manifold and both metrics collapse.

4.2 Results

ToA-Steer wins on all three metrics across every model and benchmark. On every cell of Table 2, ToA-Steer achieves the highest *Acc* and the best *ACT* and *THINK* tool-call rates. Static Steering moves both rates in the same direction, so any fixed magnitude trades one class for the other. SFT and GRPO are the strongest baselines, yet ToA-Steer beats them without any parameter updates,

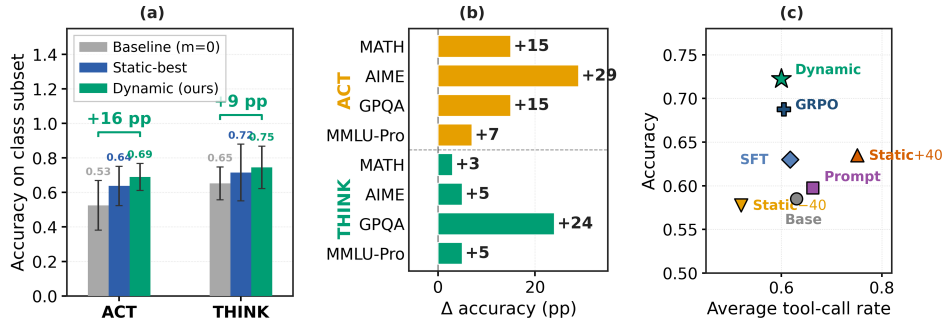


Figure 4: Class- and cost-resolved view of ToA-Steer. **(a)** Aggregate accuracy on ACT vs. THINK; ToA-Steer gains +16/+9 pp over baseline. **(b)** Per-benchmark Δ accuracy by class; gains are bidirectional on every benchmark. **(c)** Avg. accuracy vs. tool-call rate with ToA-Steer dominates.

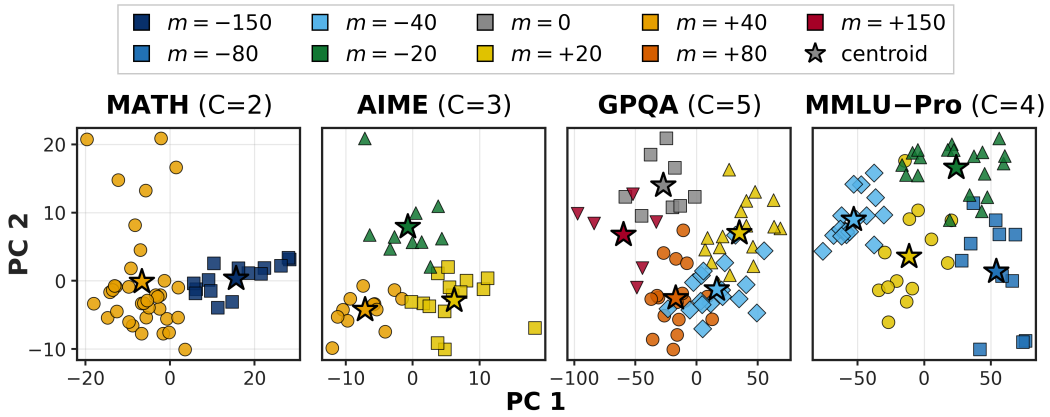


Figure 5: PCA-2D projection of decision-token activations on each benchmark, with each instance coloured by the steering magnitude m^* that maximises its accuracy. The per-benchmark K is shown in the panel header. Instances sharing the same optimal m^* form coherent clusters.

while Prompt barely moves the model. Calibrated only on GPQA, ToA-Steer also transfers to BFCL V4 live web search and scores best in all four of its settings (Appendix O).

Representation steering effectively controls tool-call behavior that translates into real accuracy gains over baseline. Fig. 3 sweeps $m \in [-150, +150]$ on Qwen3-8B. Within a moderate $|m|$ range, the tool-call rate tracks m smoothly, confirming v_ℓ as a clean control axis for tool use. Accuracy peaks above the no-steer baseline, but the peak location differs across benchmarks, so no fixed m is optimal everywhere. At $|m| \approx 150$, both signals collapse as the residual stream is pushed off-manifold.

ToA-Steer's gains are bidirectional across classes and cost-efficient against every baseline. Aggregated across the four benchmarks, ToA-Steer gains 16 points on ACT-needed and 9 points on THINK-preferred instances over the no-steer baseline (Fig. 4a), and both classes improve on every benchmark from the *same* direction routed to opposite magnitudes (Fig. 4b). ToA-Steer also reaches the highest accuracy at a moderate tool-call rate (Fig. 4c), so its gains come from better-calibrated rather than more frequent tool use (Appendix M).

5 Analysis

Do the gains hold on the full evaluation pool? We rerun every method on the full evaluation pool and group instances by the outcomes of their no-tool and with-tool-use passes (Table 3). These groupings are reliable: re-building the labels with five independent seeds reproduces 97.3 to 98.9%

Table 3: Decision quality on the full evaluation pool, pooled over the four models (mean ± SD over five independent runs). Each instance is grouped by the outcomes of its no-tool and with-tool-use passes. *Headroom recovered* is the accuracy gain over Base on decision-relevant instances divided by that of an oracle that always takes the better action. ToA-Steer has the largest gain on both decision-relevant groups and leaves decision-irrelevant instances unaffected.

Method	ACT-needed↑	THINK-preferred↑	Both-correct	Headroom recovered↑
Base	54.8 ± 2.4	71.0 ± 2.2	97.2 ± 0.6	0%
SFT	62.6 ± 2.8	66.8 ± 2.6	96.5 ± 0.7	3%
GRPO	66.0 ± 3.0	64.2 ± 2.9	96.8 ± 0.7	3%
Static-best	62.5 ± 2.6	72.9 ± 2.4	97.2 ± 0.5	14%
ToA-Steer	$\mathbf{70.4 \pm 2.7}$	$\mathbf{79.2 \pm 2.5}$	97.3 ± 0.6	**36%**

Table 5: Decision-point control on the full evaluation pool (Overall accuracy, %, mean ± SD over five independent runs). All rows share the direction v_ℓ, layer, magnitude grid, and development split. *First action only* applies the routed intervention until the model commits to its first action and then removes it. Steering the decision point alone already beats the tuned static baseline on every model, while hard routing with a linear classifier falls below it.

Method	Qwen3-8B	Qwen3.5-27B	Gemma3-12B	Gemma4-31B	Average
Static-best	65.9 ± 1.3	79.5 ± 1.1	72.1 ± 1.3	84.3 ± 1.0	75.5 ± 1.0
Linear router	62.7 ± 1.8	76.1 ± 1.7	68.3 ± 1.9	80.8 ± 1.6	72.0 ± 1.4
ToA-Steer (first action only)	66.8 ± 1.5	81.2 ± 1.4	73.2 ± 1.5	85.9 ± 1.3	76.8 ± 1.1
ToA-Steer (full)	$\mathbf{67.3 \pm 1.4}$	$\mathbf{82.2 \pm 1.2}$	$\mathbf{73.8 \pm 1.4}$	$\mathbf{86.9 \pm 1.1}$	$\mathbf{77.6 \pm 1.0}$

of them on every model (Table 4). SFT and GRPO gain on ACT-needed instances but lose on THINK-preferred ones, so each recovers only 3% of the oracle headroom. ToA-Steer improves both groups, by 15.6 and 8.2 points, and recovers 36%, more than twice as much as the best static magnitude, while leaving both-correct instances unchanged. Its full-pool gains therefore come from the instances where the THINK/ACT decision matters.

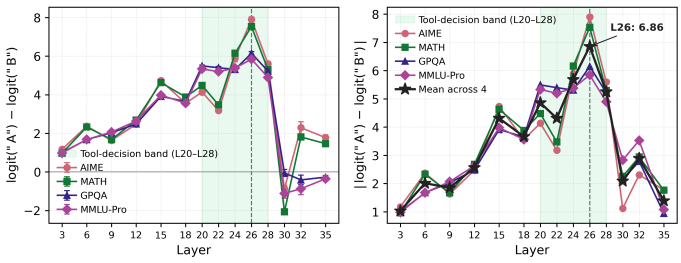
Does ToA-Steer control the decision or perturb the whole trajectory? Since ToA-Steer steers every decoding step, its gains could come from altering later reasoning rather than the decision itself. We therefore stop steering once the model commits to its first action, a tool call or an answer. This variant alone beats Static-best on all four models and recovers 62 to 65% of the full method's gain over it (Table 5). A linear router that hard-classifies each input as ACT or THINK instead falls 3.5 pp below Static-best, since every misclassified input is pushed the wrong way. The gain therefore comes mainly from steering the decision point, not from perturbing the whole generation.

Table 4: Label agreement (%) with the original ToA-Eval labels over five re-built label sets.

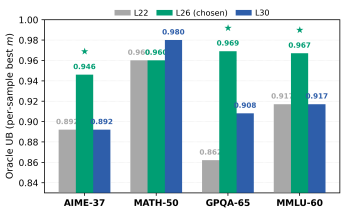
Model	Agree.↑	ACT rec.↑	THINK rec.↑
Qwen3-8B	97.8 ± 0.6	96.9 ± 0.9	98.5 ± 0.5
Qwen3.5-27B	98.6 ± 0.4	97.9 ± 0.6	99.1 ± 0.3
Gemma3-12B	97.3 ± 0.8	96.1 ± 1.1	98.2 ± 0.6
Gemma4-31B	98.9 ± 0.3	98.3 ± 0.5	99.3 ± 0.2

Where is the THINK/ACT boundary encoded? A logit lens at the prompt-end token shows that $|\,\mathrm{logit}(\text{tool}) - \mathrm{logit}(\text{skip})\,|$ stays near zero in early layers, rises sharply across $\ell \in [20, 28]$, and peaks at ℓ=26 (Fig. 6a). The per-sample oracle bound agrees: among $\ell \in \{22, 26, 30\}$, ℓ=26 is highest on three of four benchmarks (Fig. 6b). We therefore steer at ℓ=26 on Qwen3-8B, and the same analysis places the peak at 0.71–0.72 L on all four models (Appendix I).

How many decision regimes does each benchmark contain, and is cluster-routing feasible? Fig. 5 projects decision-token activations onto two PCs and colours each instance by its best steering magnitude. Instances sharing the same best magnitude form coherent regions, so an unsupervised partition can recover useful routing groups. The number of regimes differs by benchmark, from K=2 on MATH to 5 on GPQA, suggesting richer structure on knowledge-heavy tasks. Routing is necessary and robust: a random direction, random clusters, or removing a routing component costs 4.9 to 13.2 points, while a fixed K or imbalanced calibration costs at most 1.8 (Appendix L).



(a) Logit-lens decoding of the tool/skip preference at the prompt-end residual stream. The signal rises through $\ell \in [20, 28]$ and peaks at ℓ=26.



(b) Per-sample oracle upper bound at $\ell \in \{22, 26, 30\}$. ℓ=26 wins on 3/4 benchmarks.

Figure 6: Layer-wise justification for steering at ℓ=26. (a) The logit lens shows the THINK/ACT signal forming inside $\ell \in [20, 28]$ and peaking at ℓ=26. (b) The per-sample oracle ranks ℓ=26 highest on 3/4 benchmarks. The two analyses converge on ℓ=26.

How many contrastive pairs are needed to find the steering direction? We re-fit v_ℓ from $N \in \{5, 10, 25, 50, 100, 200\}$ contrastive pairs and re-run cluster-routed steering with each resulting direction. Accuracy rises sharply from N=5 to N=50 and then plateaus on every benchmark (Fig. 8), so we use N=50 pairs throughout the paper. Fitting the direction therefore takes only 100 forward passes and no gradient computation, far less supervision than the tool-use traces and rollouts needed to train SFT and GRPO. This makes ToA-Steer cheap to calibrate for each new model and benchmark.

6 Related Work

Evaluating tool-use decisions. Most tool-use benchmarks measure tool competence, that is, whether a model selects, formats, and executes calls correctly (Qin et al., 2023; Liu et al., 2024; Patil et al., 2024). MetaTool and When2Call (Huang et al., 2023; Ross et al., 2025) ask whether a model should call a tool at all, but do not measure whether a call changes the correctness of the final answer. ToA-Eval measures this outcome effect in both directions and separately for each model.

Adaptive tool use in LLM agents. Tool use is elicited through prompting (Yao et al., 2022; Paranjape et al., 2023), supervised fine-tuning (Schick et al., 2023; Qin et al., 2023; Liu et al., 2024; Patil et al., 2024), or reinforcement learning (Li et al., 2025b; Feng et al., 2025; Qian et al., 2025a). As the bottleneck shifts from tool competence to tool *necessity*, recent work learns when to call tools through reflection tokens (Asai et al., 2023) or metacognitive cost-benefit policies (Qian et al., 2025b; Li et al., 2025a), typically with additional training or an auxiliary controller. We instead control the decision at inference time inside the model's own latent space.

Latent representation steering. Activation steering adds a contrastive direction to hidden states to control attributes such as truthfulness, refusal, sentiment, or persona without parameter updates (Zou et al., 2023; Turner et al., 2023; Panickssery et al., 2023; Chen et al., 2025). These methods assume a monotonic control axis, where moving further along the direction strengthens the target behaviour. Tool-use decisions break this assumption, because ACT-needed and THINK-preferred inputs require opposite signs along the same axis. ToA-Steer therefore extends steering from monotonic behavioural control to instance-dependent agent decisions (Appendix D).

7 Conclusion

We studied strategic tool use in LLM agents as a Think-or-Act decision. ToA-Eval measures, for each model, when tool use helps or harms, and shows that the answer depends on the model. ToA-Steer controls this decision with a single THINK/ACT direction extracted from about fifty contrastive pairs and routes each input to a signed steering strength. Without any parameter updates, it achieves the highest accuracy in all 16 model–benchmark settings and beats RL-trained GRPO on selective tool calling. Extending instance-conditioned steering from single-turn tasks to multi-step agent trajectories is a natural next step.

Ethics Statement

This work uses publicly available benchmarks (AIME, MATH, GPQA, MMLU-Pro, and BFCL V4) and open-weight language models, and involves no human subjects or personally identifiable data. ToA-Steer changes when a model calls external tools rather than adding new capabilities, and its steering strengths are selected to improve answer correctness. As with other activation-steering methods, the same mechanism could in principle be used to push a model toward or away from tools for other objectives, so steered models should be evaluated for their intended use before deployment.

Reproducibility Statement

The ToA-Eval construction and the disjoint vector-train, router-train, development, and test splits are described in Section 2 and Appendix G, and the benchmarks in Appendix E. The prompts, training hyperparameters, and settings of all baselines are given in Appendix F. Steering-vector extraction, including the contrastive completions and the number of pairs, is described in Section 3.1 and Appendix H; layer selection for each model in Appendix I; and the choice of K and the magnitude grid in Appendix K. Label stability, routing ablations, and sensitivity analyses are reported over five independent runs in Appendices B and L, and the transfer setup in Appendix O.

AI Use Statement

We used AI tools to polish the writing of this paper, improving clarity, grammar, and readability.

References

AIME. AIME Problems and Solutions. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`, 2025. Accessed: 2026-05-07.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv:2507.21509*, 2025.

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. *arXiv preprint arXiv:2504.11536*, 2025.

Google DeepMind. Gemma 4. `https://deepmind.google/models/gemma/gemma-4/`, 2026. Accessed: 2026-05-07.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram'e, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gael Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai

Zhai, Anton Tsitsulin, Róbert Istvan Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, Andr'as Gyorgy, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Boxi Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, Cj Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Pluci'nska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, J. Michael Wieting, Jonathan Lai, Jordi Orbay, Joe Fernandez, Joshua Newlan, Junsong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Ping mei Xu, Piotr Stańczyk, Pouya Dehghani Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Ardeshir Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vladimir Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab S. Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and L'eonard Hussenot. Gemma 3 technical report. *ArXiv*, abs/2503.19786, 2025. URL `https://api.semanticscholar.org/CorpusID:277313563`.

Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. *arXiv preprint arXiv:2409.05907*, 2024.

Wenjun Li, Dexun Li, Kuicai Dong, Cong Zhang, Hao Zhang, Weiwen Liu, Yasheng Wang, Ruiming Tang, and Yong Liu. Adaptive tool use in large language models with meta-cognition trigger. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13346–13370, 2025a.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025b.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*, 2024.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

Jayneel Parekh, Pegah Khayatan, Mustafa Shukor, Arnaud Dapogny, Alasdair Newson, and Matthieu Cord. Learning to steer: Input-dependent steering for multimodal llms. *arXiv preprint arXiv:2508.12815*, 2025.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025a.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Smart: Self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4604–4621, 2025b.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3391–3409, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

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.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Hongshen Xu, Zihan Wang, Zichen Zhu, Lei Pan, Xingyu Chen, Shuai Fan, Lu Chen, and Kai Yu. Alignment for efficient tool calling of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17787–17803, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A LIMITATIONS

Our study focuses on open-weight LLM agents and a controlled tool-use setting with two general-purpose tools. This design allows us to isolate the THINK/ACT decision and evaluate whether representation steering can improve strategic tool use, but future work can extend the framework to richer tool ecosystems, longer multi-step agent trajectories, and interactive environments where tool decisions interact with planning, memory, and feedback. A further direction is to study how repeated THINK/ACT interventions accumulate over extended agent workflows, where early tool-use decisions may influence later context, confidence, and downstream actions.

B LABEL RELIABILITY DETAILS

Each ToA-Eval label comes from one no-tool and one with-tool-use execution. We test whether the labels are reproducible and what drives the contrast between the two passes.

Stability under re-construction. We re-run both passes with five independent seeds and compare the re-built labels with the original ones. *Agreement* is the fraction of instances whose label is reproduced; *ACT/THINK recall* is the fraction of original ACT-needed/THINK-preferred instances that keep their label. Agreement is 97.3–98.9% across models (Table 4), 99.9% under greedy decoding and above 95% up to temperature 0.4 (Table 6), and 97.5–98.6% under semantically equivalent prompt paraphrases and tool-output renderings (Table 7). ACT labels are slightly more sensitive than THINK labels because tool-using trajectories add variation through tool selection, arguments, observations, and post-tool reasoning.

Table 6: Label agreement (%) under decoding randomness (five seeds per setting, averaged over models and benchmarks).

Decoding	Agree.	ACT rec.	THINK rec.
Greedy (T=0)	99.9 ± 0.1	99.8 ± 0.2	100.0 ± 0.0
T=0.2, p=0.95	98.4 ± 0.5	97.6 ± 0.8	99.0 ± 0.4
T=0.4, p=0.95	96.6 ± 1.1	95.2 ± 1.6	97.7 ± 0.8
T=0.7, p=0.95	92.7 ± 1.9	90.8 ± 2.6	94.1 ± 1.5

Table 7: Label agreement (%) under semantically equivalent prompt paraphrases and tool-output renderings (five paired reconstructions).

Prompt	Tool output	Agree.	ACT rec.	THINK rec.
Canonical	Canonical	99.2 ± 0.3	98.7 ± 0.5	99.6 ± 0.2
Paraphrased	Canonical	98.6 ± 0.5	97.8 ± 0.8	99.1 ± 0.4
Canonical	Reformatted	98.2 ± 0.6	97.3 ± 0.9	98.9 ± 0.5
Paraphrased	Reformatted	97.5 ± 0.8	96.4 ± 1.2	98.4 ± 0.6

Tool content versus the act of calling a tool. A with-tool-use execution differs from a no-tool one not only in the information the tool returns but also in the tool schema, the generated call, the interrupted trajectory, and the continuation after the call. To separate these factors, we rerun the with-tool-use pass in two controls. In the *empty-result* control, the tool-calling procedure, tool-message structure, and post-tool continuation are kept, but the tool returns no content. In the *hidden-result* control, the model-generated tool call is executed but its output is hidden from the model. Table 8 reports accuracy on the two label subsets. Calling a tool without receiving its content changes accuracy by at most 3.2 pp relative to the no-tool pass, whereas exposing the actual result changes it by $+86.8$ pp on ACT-needed and -87.7 pp on THINK-preferred instances relative to the hidden result. The label contrast is therefore carried by what the tool returns, not by the tool-calling scaffold. Because both subsets are defined by the outcomes of the with-tool-use pass, this audit explains what drives the contrast under our protocol; it does not measure tool utility independently of the protocol.

C CASE STUDY

Fig. 7 contrasts PROMPT and our TOA-STEER on two short MATH problems for Qwen3-8B. Each panel quotes the model's actual tool-decision sentence verbatim from its generation. In Case 1 PROMPT commits to a Python call and ends with the wrong decimal, while steering pushes TOA-STEER toward an in-context derivation that recovers the exact answer; in Case 2 the directions are reversed, PROMPT stays inside symbolic trigonometry and runs out of budget, while steering triggers a Python call that closes the ratio. The two cases need opposite tool decisions, so a single fixed-magnitude steering cannot recover both.

Table 8: Tool content versus the act of calling a tool: accuracy (%, five seeds) on the two label subsets. The *No tool* and *Actual result* columns are close to 0 or 100 by construction, since the subsets are defined by these two outcomes. The informative comparison is between the no-tool pass and the two controls that call a tool without receiving its content (Δ_{call}), and between the hidden and the actual result (Δ_{content}).

Subset	No tool	Empty result	Hidden result	Actual result	Δ_{call}	Δ_{content}
ACT-needed	4.6 ± 1.2	6.1 ± 1.8	7.4 ± 2.0	94.2 ± 1.7	$+1.5$ / $+2.8$	$+86.8$
THINK-preferred	96.8 ± 0.9	95.1 ± 1.4	93.6 ± 1.9	5.9 ± 1.6	-1.7 / -3.2	-87.7

Δ_{call}: empty result − no tool / hidden result − no tool. Δ_{content}: actual result − hidden result.

Case 1: cone volume (over-use).
The diameter of a cone is 30 decimeters. If the height is two times the radius, what is the volume of the cone in cubic decimeters? GT: $\mathbf{2250}\pi$.

PROMPT: *"Let me write a Python snippet to compute that."*
Python returns 7068.58 ⇒ 7068.58 ×
tool call broke the symbolic form.

TOA-STEER (OURS): *"I'll proceed with the exact value."*
no tool ⇒ $\mathbf{2250}\pi$ ✓
skipping the call kept π symbolic.

Case 2: chord ratio (under-use).
$\triangle ABC$, AB=7, AC=8, BC=9; *find* AD/CD. GT: $\mathbf{5/3}$.

PROMPT: *"Let me try to calculate $\sin(\phi)$ directly."*
no tool ⇒ no answer ×
stuck in trig identities, budget exhausted.

TOA-STEER (OURS): *"Let me use Python to calculate this."*
Python returns 1.281 rad ⇒ $\mathbf{5/3}$ ✓
tool call closed the computation.

Figure 7: Two MATH cases on Qwen3-8B with verbatim model output. Both cases show opposite tool-use errors by PROMPT, whereas TOA-STEER corrects both by suppressing the harmful call and triggering the helpful one.

D EXTENDED RELATED WORK

Adaptive Tool Use in LLM Agents. External tools extend LLMs beyond parametric knowledge, with early prompting-based methods eliciting tool use through in-context demonstrations and reasoning formats (Yao et al., 2022; Paranjape et al., 2023), and later training-based approaches improving tool competence via supervised tool-use traces (Schick et al., 2023; Qin et al., 2023; Liu et al., 2024; Patil et al., 2024) or reinforcement learning from environmental feedback (Li et al., 2025b; Feng et al., 2025; Qian et al., 2025a). As agents become more capable, the key challenge shifts from tool competence to tool necessity: unnecessary calls waste resources and may corrupt correct answers, while missed calls leave models relying on unreliable internal reasoning. Recent work addresses this decision problem with learned reflection tokens (Asai et al., 2023) or metacognitive policies that trade off internal knowledge against query cost (Qian et al., 2025b; Li et al., 2025a; Xu et al., 2025), typically via fine-tuning, reinforcement learning, or an auxiliary controller. In contrast, we frame tool-use control as an inference-time intervention within the model's latent space itself, using dynamic steering to modulate the model's THINK/ACT tendency without parameter updates.

Latent Representation Steering. Latent representation steering controls LLM behaviour by intervening on hidden activations at inference time, typically deriving steering directions from contrastive examples to modulate attributes such as truthfulness, refusal, toxicity, sentiment, or style without updating model parameters (Zou et al., 2023; Turner et al., 2023; Panickssery et al., 2023; Chen et al., 2025). Recent work further explores more conditional forms of activation control, such as input-dependent gating or modulation of steering strength across generation (Lee et al., 2024; Parekh et al., 2025). However, these applications usually assume a relatively monotonic control axis, where moving further along a direction consistently strengthens or suppresses the target behavior. Tool-use control poses a different challenge: the desired direction depends on the instance, since the model should move toward ACT when tools are helpful but toward THINK when tools are unnecessary or harmful. Therefore, we explore whether representation steering can be extended from monotonic

behavioral control to instance-dependent agent decisions, using dynamic steering to calibrate the model's latent THINK/ACT tendency without globally increasing or suppressing tool use.

E BENCHMARK DETAILS

We evaluate on four reasoning benchmarks that span computation-heavy and knowledge-heavy regimes. **AIME** consists of competition mathematics problems drawn from the AIME 2023, 2024, and 2025 contests; **MATH** (Hendrycks et al., 2021) covers algebra, geometry, and number theory at high-school competition level; **GPQA** (Rein et al., 2023) is a graduate-level science question-answering benchmark across physics, chemistry, and biology; **MMLU-Pro** (Wang et al., 2024) is a more challenging variant of MMLU with longer questions and harder distractors, from which we use a balanced subset across professional and STEM categories. The per-model ToA-Eval pool sizes obtained by retaining only ACT-needed and THINK-preferred instances after ToA-Eval labelling are shown in the main text (Table 1).

F BASELINE DETAILS

Base. The unmodified target model under the default tool-available setting, with Python execution and Wikipedia search exposed via the system prompt.

Prompt. Same setting as Base, but the system prompt prepends the instruction shown in the box below. No other change is made to the input or the decoding configuration.

Prompt baseline: system instruction

You have access to two tools: a Python interpreter (`python`) and a Wikipedia search engine (`wikipedia_search`). Call a tool only when (i) the problem requires precise numerical computation you cannot reliably perform yourself, or (ii) the problem requires a fact you are not confident about. If you can solve the problem confidently from internal reasoning, do not call any tool.

SFT. We fine-tune the target model on supervised tool-use traces drawn from the ToA-Eval training partition, keeping only trajectories whose final answer matches the ground truth. Each trace is rendered in the model's chat template with the structure shown in the box below. ACT-needed traces contain at least one `<tool_call>...</tool_call>` block; THINK-preferred traces contain none. Training uses LoRA (rank 16, α=32) on the attention and MLP projections, AdamW with learning rate 2×10^{-5}, batch size 32, cosine schedule with 3% warmup, 2 epochs over the training pool, and a maximum sequence length of 4096 tokens.

SFT chat-template trace

```
<system> You are a problem-solving assistant with access to two tools
(python, wikipedia_search).  Use them only when external computation
or factual lookup would help.  </system>
<user> {problem} </user>
<assistant> {chain_of_thought}
<tool_call>{name}({args})</tool_call>
<tool_result>{output}</tool_result>
{continued_reasoning}
\boxed{{final_answer}} </assistant>
```

GRPO. We further train the SFT model with group-relative policy optimization (Shao et al., 2024) on tool-use rollouts. For each prompt x with ToA-Eval label $y \in \{\text{ACT}, \text{THINK}\}$, every group of G=8 trajectories is constructed to contain both tool-using and tool-skipping behaviors so that the relative-advantage estimate has gradient signal for either decision: 4 trajectories are sampled under the standard tool-available chat template (under which the policy may emit a `<tool_call>` block), and 4 are sampled under a no-tool variant of the same prompt in which the tool schema is removed from the system message, forcing a fully internal trajectory. Letting $c(\tau) \in \{0, 1\}$ denote

final-answer correctness and $t(\tau) \in \{0, 1\}$ denote whether trajectory τ contains a tool call, the per-trajectory reward is

$$r(\tau) \;=\; c(\tau) \;+\; \lambda\big(2\mathbb{1}[\,t(\tau) = \mathbb{1}[y{=}\text{ACT}]\,] - 1\big), \tag{5}$$

which gives $+1$ for a correct final answer plus $\pm\lambda$ depending on whether the trajectory's tool-use action matches the ToA-Eval label; we set $\lambda{=}0.5$. Within each group, the relative advantage is computed across all $G{=}8$ rewards (mixing tool-using and tool-skipping trajectories), so the policy is updated toward whichever behavior accumulates more reward on this particular prompt. We optimize with KL coefficient $\beta{=}0.04$ against the SFT reference policy, learning rate 1×10^{-6}, group size $G{=}8$, prompt batch size 64, and 2 epochs over the ToA-Eval training pool. Rollout decoding uses temperature 0.7 and top-p 0.95.

Static Steering. Static Steering applies the same direction v_ℓ as our method at the same layer ($\ell{=}26$) but uses a single fixed magnitude $m \in \{+40, -40\}$ for all inputs, with no clustering or per-instance routing. *Static-best* (Table 5) instead selects the single magnitude with the highest development-split accuracy from the same grid $\mathcal{M}$, using the same direction, layer, development split, and calibration budget as ToA-Steer.

Linear router. A logistic-regression classifier on the decision-token representation $z_\ell(x)$, trained on the same development split and labels available to ToA-Steer, predicts ACT or THINK; the input is then steered with the signed magnitude selected for that class from $\mathcal{M}$ by held-out final-answer accuracy.

First-action-only steering. Routing, magnitudes, and layer are identical to ToA-Steer, but the intervention is applied only until the model emits its first action (the first `<tool_call>` token or the first answer token) and is removed afterwards.

G ToA-Eval Construction and Train/Probe/Test Splits

ToA-Eval labels are obtained by running the target model without and with tool use on each benchmark and keeping only the disagreements: ACT-needed (no-tool fails, with-tool-use succeeds), THINK-preferred (no-tool succeeds, with-tool-use fails). Items where both modes succeed (tool-redundant) and items where both fail are excluded from the supervision pool.

Four disjoint splits. From the per-model labelled pool we form four disjoint, stratified-by-class splits, each used exclusively at one stage of the pipeline. Their roles are summarised in Table 9; the key property is that *no Test instance is used for steering-vector extraction, PCA / K-means fitting, K selection, magnitude-grid search, prompt selection, SFT, or GRPO training, for ToA-Steer or any baseline.* Concretely, the Test split is held out before any other split is constructed, and final-answer accuracy and ACT/THINK tool-call rates in Table 2 are computed only on Test. The Dev/probe split contains 50 instances per benchmark (25 ACT-needed and 25 THINK-preferred) drawn balanced by class; for benchmark/model combinations with fewer than 25 instances in either class after ToA-Eval filtering (e.g., AIME on smaller models), we use the maximum balanced subset available.

H Steering Vector Extraction

The steering vector v_ℓ is extracted from N contrastive completion pairs that share the same system and user blocks but differ in the final assistant completion (use tools versus answer directly; Section 3.1), following CAA-style activation differencing. Per-benchmark accuracy as a function of N is shown in Fig. 8; accuracy saturates by $N \approx 50$ on every benchmark, so we use $N{=}50$ pairs throughout the paper. The layer choice $\ell{=}26$ is justified in Section 5.

I Per-Model Layer Selection

The main-text logit-lens analysis (§5) is run on Qwen3-8B and identifies $\ell{=}26$ ($\approx 0.72\,L$) as the depth at which the THINK/ACT signal is most crisply written into the residual stream. To check that

Table 9: Per-(model, benchmark) data split for ToA-Steer and all baselines. The four splits are disjoint and stratified by class: *no Test instance is used for vector extraction, PCA / K-means fitting, K selection, magnitude-grid search, prompt selection, SFT, or GRPO training.*

Split	Used for	Uses ACT/THINK label?	In Table 2?
Vector-train	Contrastive vector v_ℓ (Eq. 1); SFT supervised traces; GRPO rollout prompts	label-free contrast / yes (training)	no
Router-train	PCA fit and K-means centroids $\{\mu_k\}$ (Eq. 3)	label-free (unsupervised)	no
Dev/probe	Per-cluster magnitudes m_k (Eq. 3); per-benchmark K; static-best m for the Static Steering baseline	yes (used for selection)	no
Test	Final-answer Acc and ACT/THINK tool-call rates	yes (only for evaluation)	yes

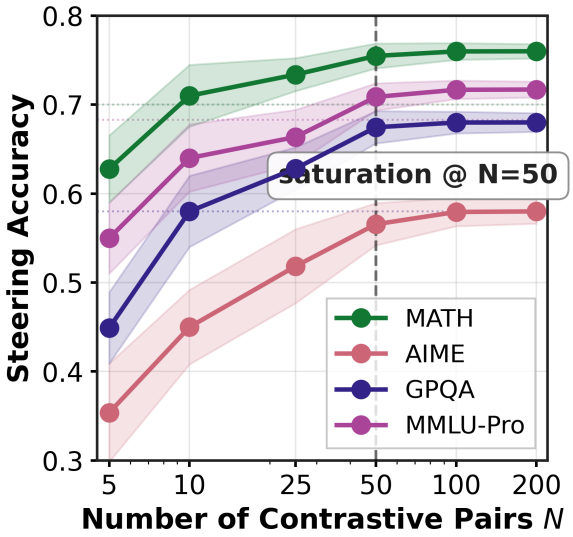


Figure 8: Sample efficiency: steering accuracy as a function of the number of contrastive pairs N used to fit the steering vector, on the four ToA-Eval benchmarks (Qwen3-8B). Accuracy saturates by $N{\approx}50$ on every benchmark, so the per-benchmark steering vectors throughout the paper are fitted with $N{=}50$ pairs.

the same selection rule transfers to the other three backbones, we re-run the same analysis (logit-lens of $|\,\mathrm{logit}(\text{tool}) - \mathrm{logit}(\text{skip})\,|$ at the prompt-end token across the four ToA-Eval benchmarks) on each of Qwen3.5-27B, Gemma3-12B, and Gemma4-31B. Figs. 9–11 show the resulting curves. The qualitative pattern matches Qwen3-8B on every model: $|\,\mathrm{logit}(\text{tool}) - \mathrm{logit}(\text{skip})\,|$ stays small in early layers, rises through a contiguous mid-to-late band, peaks at a single layer, and collapses (often flipping sign) in the last few layers as the residual is rewritten for token-level prediction. The peak lands at $\ell{=}46$ for Qwen3.5-27B (64 layers, $\approx 0.72\,L$), $\ell{=}34$ for Gemma3-12B (48 layers, $\approx 0.71\,L$), and $\ell{=}43$ for Gemma4-31B (60 layers, $\approx 0.72\,L$). All four peaks fall within ≈ 0.71–$0.72\,L$, supporting the choice of an analogous proportional-depth layer per model rather than a single absolute index.

J Steering Magnitude Sweeps by Layer Group

To complement the logit-lens evidence (Appendix I) with a causal probe of where steering actually changes behaviour, we sweep the intervention magnitude m at every candidate layer on Qwen3-8B and plot the resulting probe-set accuracy and tool-call rate, grouped into early ($\ell \in [3, 15]$), mid ($\ell \in [18, 28]$), and late ($\ell \in [30, 35]$) bands. Three regimes emerge. **Early layers** (Fig. 12, left;

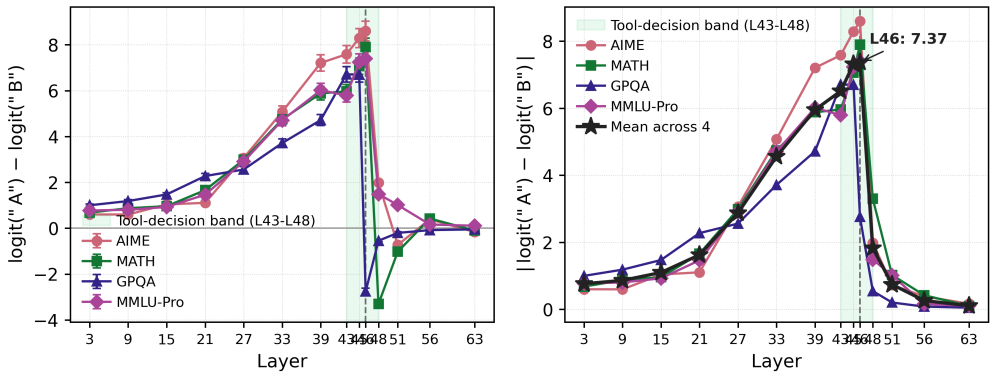


Figure 9: Logit-lens decoding of the tool/skip preference at the prompt-end residual stream on **Qwen3.5-27B** (64 layers). All four ToA-Eval benchmarks show a tightly concentrated rise inside the tool-decision band $\ell \in [43, 48]$ followed by a sharp post-peak collapse, with several benchmarks flipping sign within two layers of the peak; the cross-benchmark mean magnitude (right, black stars) places ℓ=46 as the unique peak.

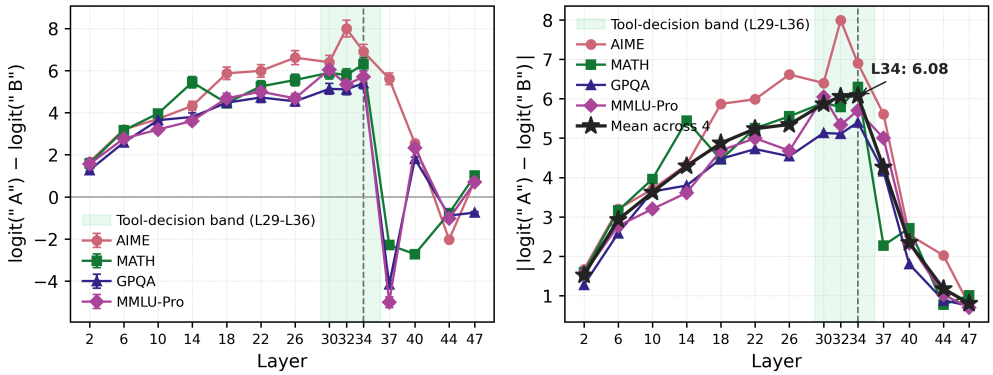


Figure 10: Logit-lens decoding of the tool/skip preference at the prompt-end residual stream on **Gemma3-12B** (48 layers). The signal rises gradually from a non-trivial early-layer baseline and forms a broad shoulder over $\ell \in [29, 36]$ before peaking at ℓ=34; the post-peak decay is gentle and largely positive, in contrast to the sharp dive seen on Qwen models.

Fig. 13, left) are inert: both accuracy and tool-call rate are essentially flat in m, with all curves collapsing onto the baseline at m=0. The residual at these depths does not yet carry the THINK/ACT decision, so additive intervention has no leverage over downstream behaviour. **Mid layers** (centre panels) show the opposite: tool-call rate is strongly and monotonically modulated by m (negative m suppresses tool calls, positive m encourages them) and accuracy traces a clear non-trivial profile with sharp drops at extreme magnitudes and a peak near a layer-specific small $|m|$. This is the layer band at which the decision feature is causally accessible to additive steering. **Late layers** (right panels) move tool-call rate but accuracy degrades and becomes noisy: at these depths the residual is already being rewritten for token-level prediction, so a magnitude push perturbs the committed plan rather than redirecting the underlying decision. The mid band, and within it the layer chosen by the logit-lens analysis (ℓ=26 on Qwen3-8B), is therefore the only depth at which steering changes behaviour controllably.

K Cluster-Routed Controller

Choice of K per benchmark. We do not use a single K across benchmarks. Instead, the cluster count is set per benchmark following the structure recovered from the PCA-2D projection of decision-token activations (Fig. 5, §5): MATH at K=2, AIME at K=3, MMLU-Pro at K=4, and GPQA

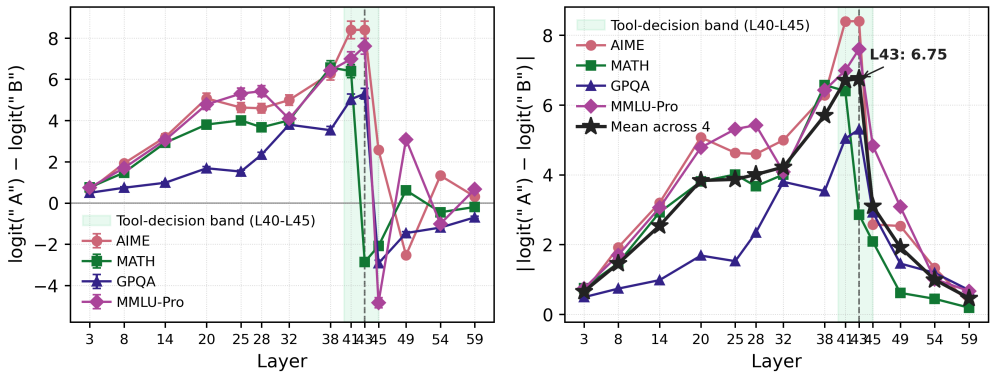


Figure 11: Logit-lens decoding of the tool/skip preference at the prompt-end residual stream on **Gemma4-31B** (60 layers). Several benchmarks show a non-monotonic pre-peak shape, with an intermediate sub-peak around $\ell \approx 25$ followed by a brief dip before the true peak inside $\ell \in [40, 45]$; the cross-benchmark mean places the peak at $\ell = 43$, and the post-peak signal is oscillatory rather than uniformly negative.

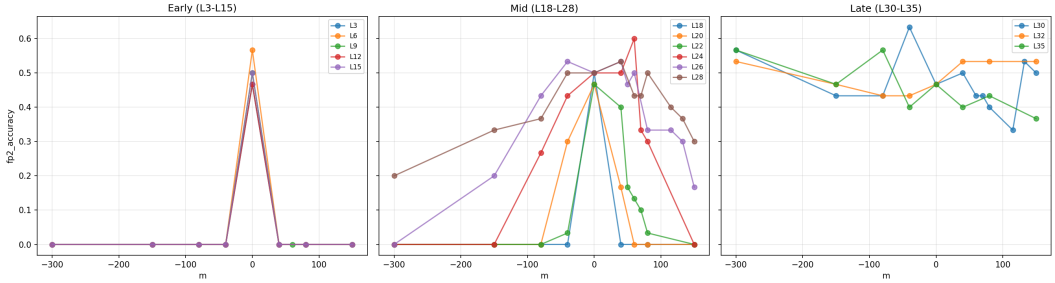


Figure 12: Probe-set accuracy as a function of steering magnitude m on Qwen3-8B, grouped by layer band: early ($\ell \in [3, 15]$, left), mid ($\ell \in [18, 28]$, centre), and late ($\ell \in [30, 35]$, right). Early-layer curves are flat in m; mid-layer curves form a non-trivial profile with sharp drops at extreme magnitudes; late-layer curves move with m but become noisy and drift below the $m=0$ baseline.

at $K=5$. These values are the smallest K at which held-out accuracy of cluster-routed steering crosses the static-best baseline (Fig. 14); larger K yields no reliable further gains on held-out data and widens the overfit gap of the per-cluster magnitude search. The main results in Table 2 use these per-benchmark values throughout.

Distribution of cluster magnitudes. Each cluster magnitude m_k is selected on the probe set from the discrete grid $\{-150, -80, -40, -20, 0, +20, +40, +80, +150\}$, which includes $m_k=0$ (no intervention) and signed values that encourage ($m_k>0$) or suppress ($m_k<0$) tool use. Across the four benchmarks on Qwen3-8B (with the per-benchmark K above, so 14 selected magnitudes in total), the selected $\{m_k\}$ span both signs and a range of strengths: clusters whose centroids correspond to ACT-needed regions select $m_k>0$, clusters in THINK-preferred regions select $m_k<0$, and a small fraction of clusters select $m_k=0$ when the local population is balanced enough that no single magnitude improves held-out accuracy. The mass concentrates on $|m_k| \in \{20, 40, 80\}$, with extreme magnitudes (± 150) chosen only on AIME, the smallest benchmark. This split, with the same direction v_ℓ used in both signs depending on cluster identity, is the empirical signature of bidirectional control along a single steering axis.

L Routing Necessity and Sensitivity

We test whether the learned direction and the learned routing are both necessary, and how sensitive ToA-Steer is to its calibration choices. All experiments use Qwen3-8B on the ToA-Eval test split.

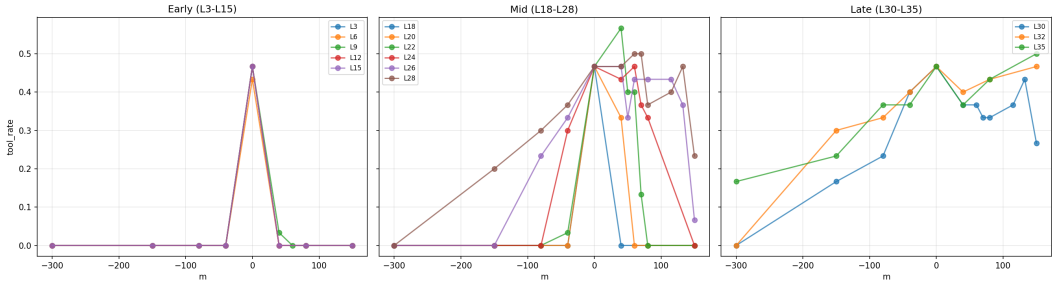
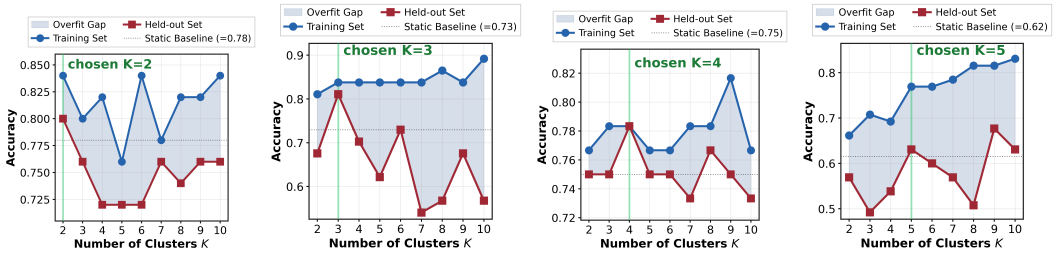


Figure 13: Tool-call rate as a function of steering magnitude m on Qwen3-8B, with the same layer grouping as Fig. 12. Early-layer tool-call rates are unaffected by m (curves collapse onto the baseline). Mid-layer tool-call rates are monotonically and strongly modulated by m, increasing toward 1 for $m{>}0$ and decreasing toward 0 for $m{<}0$. Late-layer tool-call rates also move with m but with weaker leverage and at the cost of accuracy (Fig. 12, right).



(a) MATH ($K{=}2$). (b) AIME ($K{=}3$). (c) MMLU-Pro ($K{=}4$). (d) GPQA ($K{=}5$).

Figure 14: Per-benchmark K selection for the cluster-routed controller on Qwen3-8B. Each panel sweeps $K \in [2, 10]$ and plots cluster-routed steering accuracy on the training partition (blue) and the held-out probe partition (red); the dotted line is the static-best steering baseline on that benchmark, and the green vertical line marks the chosen K. The chosen K is the smallest value at which the held-out (red) curve crosses the static-best baseline; beyond this point the shaded *overfit gap* between training and held-out accuracy widens with no reliable held-out gain. The four benchmarks select different K values ($2/3/4/5$ for MATH/AIME/MMLU-Pro/GPQA), confirming that decision-regime granularity is benchmark-specific and that a single fixed K would either under-fit (too few clusters for knowledge-style tasks) or over-fit (too many for computation-heavy MATH).

The random controls are averaged over five independent randomizations and go through the same accuracy-gated calibration pipeline as ToA-Steer.

Direction and routing are both necessary. A norm-matched random direction reaches 0.591, barely above no steering at 0.585, so the learned THINK/ACT direction accounts for 13.2 of ToA-Steer's 13.8-point gain over Base (Table 10). Shuffling the cluster assignments while keeping the direction and re-selecting the magnitudes recovers only part of the gain and trails the full method by 4.9 points. Removing PCA, replacing K-means with PC1 quantile routing, or sharing one magnitude across all clusters costs 5.0, 8.0, and 8.8 points on average, so the gain comes from routing inputs to appropriate signed strengths rather than from steering alone.

Sensitivity to K and to the calibration distribution. Using one fixed K between 2 and 5 for all benchmarks stays within 0.8 to 1.8 points of the benchmark-specific K, and calibrating on the natural or a 70:30 imbalanced ACT/THINK ratio instead of a balanced one costs 0.7 to 1.4 points (Table 11).

M Per-Benchmark Accuracy–Cost Trajectories

Fig. 15 shows the per-benchmark trajectories from Baseline through Static-best to ToA-Steer on Qwen3-8B. ToA-Steer reaches the highest accuracy on every benchmark, and the trajectory shape

Table 10: Controls and component ablation of the routing pipeline (Qwen3-8B, ToA-Eval test split, accuracy). Random controls report mean ± SD over five randomizations.

Variant	MATH	AIME	GPQA	MMLU-Pro	Average
ToA-Steer (full)	**0.790**	**0.620**	**0.740**	**0.740**	**0.723**
No steering (Base)	0.700	0.380	0.580	0.680	0.585
Random direction, norm-matched	0.704 ± 0.011	0.394 ± 0.018	0.589 ± 0.014	0.676 ± 0.012	0.591 ± 0.009
Random clusters, magnitudes re-selected	0.753 ± 0.013	0.558 ± 0.021	0.672 ± 0.016	0.713 ± 0.014	0.674 ± 0.011
w/o PCA: K-means on raw hidden states	0.747	0.557	0.684	0.702	0.673
w/o K-means: PC1 quantile routing	0.719	0.521	0.651	0.681	0.643
w/o cluster-specific magnitudes: shared strength	0.716	0.498	0.641	0.686	0.635

Table 11: Sensitivity to the cluster count and to the ACT/THINK ratio of the calibration data (Qwen3-8B, ToA-Eval test split, accuracy). The default uses benchmark-specific K and a balanced calibration set.

Factor	Setting	MATH	AIME	GPQA	MMLU-Pro	Average
Default	$K^*=(2,3,5,4)$, balanced	0.790	0.620	0.740	0.740	0.723
Cluster count	Fixed $K=2$	0.790	0.598	0.708	0.723	0.705
	Fixed $K=3$	0.781	0.620	0.726	0.731	0.715
	Fixed $K=4$	0.776	0.607	0.733	0.740	0.714
	Fixed $K=5$	0.767	0.595	0.740	0.735	0.709
Calibration ratio	Natural ratio	0.784	0.611	0.735	0.734	0.716
	ACT-heavy 70:30	0.778	0.600	0.728	0.731	0.709
	THINK-heavy 30:70	0.782	0.604	0.724	0.736	0.712

(right then up-left) shows that ToA-Steer does not simply increase tool use; on every benchmark it finds an accuracy higher than Static-best at a tool-call rate close to Baseline. Fig. 16 additionally plots the aggregate accuracy–cost scatter for all baselines and ToA-Steer on Qwen3-8B, with the per-benchmark scatter overlaid in lighter shades.

N Static-Steering Frontier and Per-Sample Oracle Headroom

To quantify how much accuracy a dynamic, instance-conditioned controller can in principle recover beyond any single-magnitude static intervention, we plot, per benchmark and on Qwen3-8B, the static-steering frontier traced by sweeping m at $\ell{=}26$ (Fig. 17, blue line), the baseline at $m{=}0$ (black star), and the per-sample oracle (red cross) which selects the best m independently for each test instance. The orange band marks the accuracy region a successful dynamic controller should reach. The oracle exceeds the best static point by $+38$ pp on AIME-2026, $+20$ pp on GPQA, $+9$ pp on MATH, and $+5$ pp on MMLU-Pro: the harder, more ACT-needed benchmarks have the largest static-to-oracle gap, indicating that on these tasks the optimal magnitude varies sharply across instances and a single global m cannot capture it. This headroom is the upper bound that ToA-Steer's per-cluster routing aims to recover with K clusters per benchmark; the gap between the static-best point and the oracle measures, in accuracy terms, the value of dynamic routing at this layer.

O Transfer to a Realistic Web-Search Environment

To test whether ToA-Steer generalizes beyond the clean tools and benchmarks it is calibrated on, we apply it to the Web Search category of BFCL V4, a live web-search environment, on Qwen3.5-27B. The steering direction, layer, routing, and per-cluster magnitudes are calibrated on GPQA with the Wikipedia tool and applied without re-calibration. *Web Search Overall* is the unweighted mean of the official Snippet and No-Snippet settings, *50% Fetch Success* evaluates the No-Snippet setting when each URL fetch succeeds with probability 0.5, and *Irrelevance* measures whether the model abstains from calling tools that are irrelevant to the query. ToA-Steer achieves the best mean score in all four settings (Table 12). It improves over Base by 4.8 points on Web Search Overall and by 5.0 points under 50% fetch failure, and stays ahead of GRPO by 0.8 to 1.1 points in every setting.

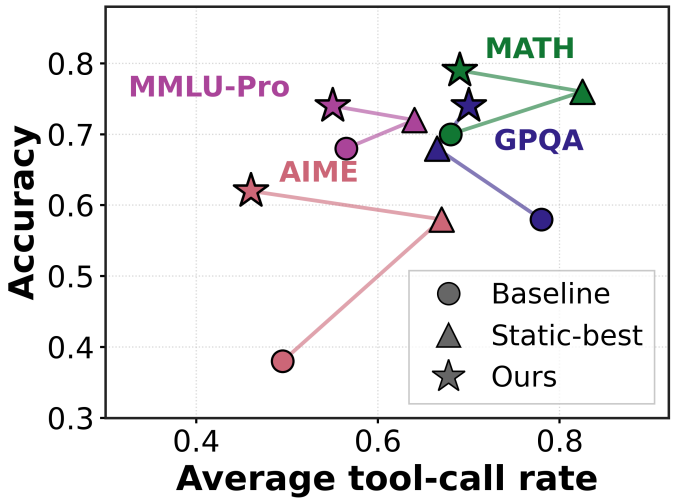


Figure 15: Per-benchmark accuracy–cost trajectories on Qwen3-8B: Baseline (•) → Static-best (▲) → ToA-Steer (★). ToA-Steer reaches the highest accuracy on every benchmark; the trajectory shape (right then up-left) shows that ToA-Steer does not simply increase tool use: on every benchmark it finds an accuracy higher than Static-best at a tool-call rate close to Baseline.

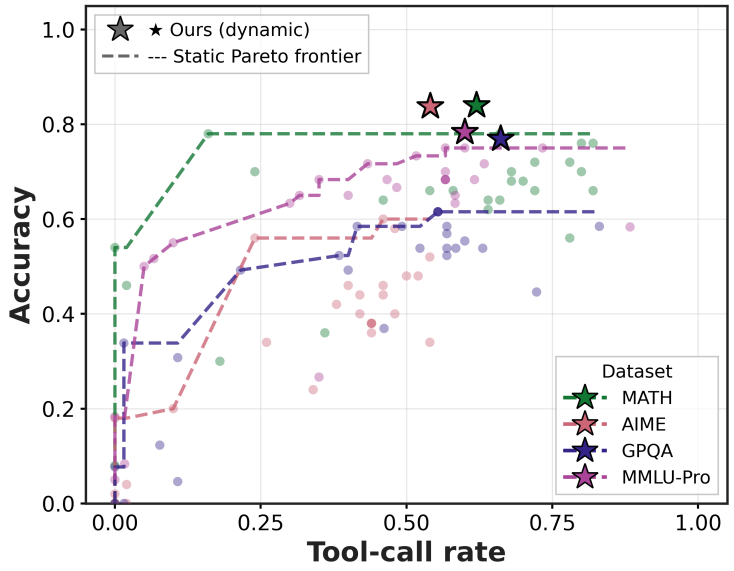


Figure 16: Aggregate accuracy versus tool-call rate across all four benchmarks (Qwen3-8B). Each marker is one method; per-benchmark scatter is shown in light shades for context.

Table 12: Transfer from GPQA to BFCL V4 Web Search without re-calibration (Qwen3.5-27B, accuracy %, mean ± SD over five seeds).

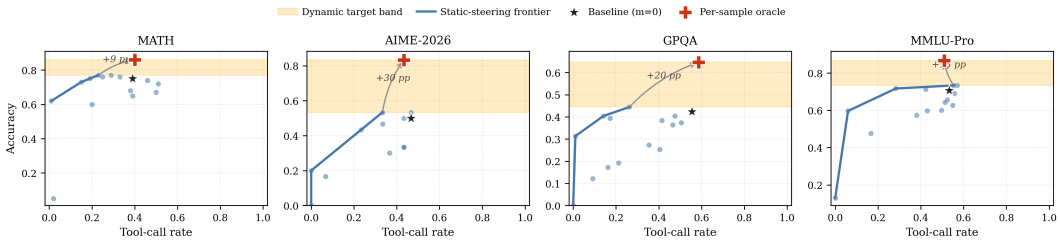


Figure 17: Per-benchmark accuracy vs. tool-call rate on Qwen3-8B at ℓ=26. Each panel shows the static-steering frontier obtained by sweeping a single magnitude m (blue line, scatter), the baseline at m=0 (★), and the per-sample oracle that picks the best m for each test instance (red cross). The orange band marks the accuracy region a dynamic controller should reach. The static-to-oracle gap (+38 pp AIME-2026, +20 pp GPQA, +9 pp MATH, +5 pp MMLU-Pro) is the headroom available to instance-conditioned routing beyond any single-magnitude static intervention.

Method	**Web Search Overall↑**	**No-Snippet↑**	**50% Fetch Success↑**	**Irrelevance↑**
Base	43.8 ± 2.1	39.2 ± 2.6	31.0 ± 3.0	87.9 ± 0.8
Prompt	44.6 ± 2.3	40.0 ± 2.8	31.7 ± 3.2	88.8 ± 0.8
SFT	46.4 ± 2.0	41.9 ± 2.5	33.6 ± 2.9	90.7 ± 0.7
GRPO	$\underline{47.5 \pm 2.1}$	$\underline{43.1 \pm 2.6}$	$\underline{34.9 \pm 3.1}$	$\underline{91.8 \pm 0.6}$
Static-best	47.4 ± 1.9	42.6 ± 2.4	34.0 ± 2.8	91.2 ± 0.6
ToA-Steer	$\mathbf{48.6 \pm 1.9}$	$\mathbf{44.1 \pm 2.3}$	$\mathbf{36.0 \pm 2.7}$	$\mathbf{92.6 \pm 0.5}$