

Conditioning Activation Interventions on Evolving Reasoning States in Large Language Models

Tae-Won Kim, Hee-Dong Kim, Jumin Han, Seong-Whan Lee*

Department of Artificial Intelligence, Korea University, Seoul, Korea
{taewon_kim, hd_kim, juminhan, sw_lee}@korea.ac.kr

Abstract

Activation steering controls language model behavior at inference time by injecting behavior-relevant directions into internal representations. In the standard steering interface, a steering direction is estimated once before generation and then applied throughout decoding. However, during multi-step reasoning, the model’s internal state shifts continuously as reasoning progresses, while standard steering continues to apply the same fixed direction across these changing states. This creates a mismatch between the model’s evolving reasoning state and the static intervention applied to it. To address this mismatch, we introduce **State-Conditioned Steering (SCS)**, which recomputes the additive update itself from the hidden states encountered during generation while keeping the same additive intervention interface. Rather than adapting when or how strongly to intervene, SCS re-estimates the update from intermediate reasoning states as generation proceeds. In a paired comparison setting, we compare fixed and state-conditioned updates and find that pre-defined directions often exhibit unstable effects, while SCS yields more consistent intervention behavior across decoding steps. Across reasoning benchmarks with explicit step structure and three instruction-tuned backbones, SCS achieves larger average gains than prompting-only decoding and baseline activation steering methods.

1 Introduction

Activation steering provides an interface for controlling language model behavior at inference time (Subramani et al., 2022; Turner et al., 2023; Zou et al., 2023; Li et al., 2023; Rinsky et al., 2024). Rather than modifying model weights, it controls generation by adding task- or behavior-specific directions to internal representations during decoding (Li et al., 2023; Rinsky et al., 2024; Chen

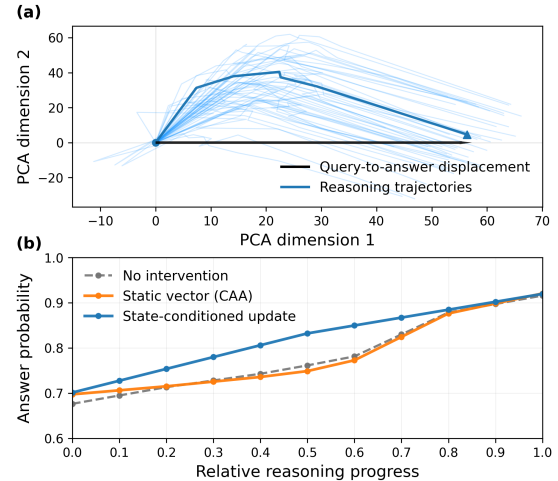


Figure 1: **Evolving Reasoning States and Local Intervention Effects.** Results shown for Gemma2-9B-IT on GSM8K. (a) visualizes evolving latent states using a two-dimensional PCA projection. The bold blue line is a representative trajectory. (b) shows the change in probability assigned to the correct answer tokens over relative reasoning progress scaled between 0 and 1 within each trace.

et al., 2025; Li et al., 2025b). Early methods relied on a single, static steering direction applied uniformly across decoding positions. To enhance this framework, recent work has moved toward adaptive or task-specific interventions. Some studies leverage steering to improve multi-step reasoning capabilities (Chen et al., 2025; Li et al., 2025b), while others design intervention policies that adapt dynamically to input semantics or task contexts (Wang et al., 2025; Stoehr et al., 2024; Scalena et al., 2024).

However, the model’s internal representations can change as intermediate steps are generated (Lightman et al., 2024; Zhou et al., 2026; Belrose et al., 2023). From this perspective, existing steering methods often rely on a representative direction estimated before generation and apply it across the entire generation trace, without explicitly consid-

*Corresponding author.

ering how reasoning states change within a single trace. As the model’s internal representations shift continuously throughout the reasoning progress, a predefined direction may become misaligned with the states encountered later in the trace. This mismatch can induce undesirable behavior. Figure 1(a) illustrates this intuition. Reasoning is represented as a trajectory in the representation space. In our visualization, this trajectory is not summarized by a single straight line from the query to the final answer state. Instead, it passes through locally varying states, suggesting that repeatedly applying the direction without considering evolving states fails to address the dynamic reasoning trace. This intuition leads us to hypothesize that conditioning the steering vector on the state can reduce the mismatch between a fixed intervention direction and the evolving states during reasoning.

To address this mismatch, we propose **State-Conditioned Steering** (SCS), which preserves the additive intervention interface while adjusting updates as reasoning progresses. We train a predictor by mapping representations extracted from visited reasoning states to their observed displacement to the subsequent state. During decoding, SCS predicts a direction from the state and refreshes the steering direction at each subsequent reasoning step. This keeps the intervention in the same additive form used by prior steering methods, while shifting the source of the vector from a predefined quantity to one directly derived from the model’s current progress.

By tracking steering behavior over reasoning progress, Figure 1(b) demonstrates that predefined directions can vary in their influence across intermediate states, whereas SCS maintains a more stable steering effect. Diagnostics at visited states further support this trend, where state-conditioned updates yield more favorable local intervention effects on answer-related proxies compared to static predefined directions. Empirically, SCS consistently improves over prompting-only decoding across reasoning benchmarks and backbones, including Llama3-8B (Grattafiori et al., 2024), Qwen2.5-7B (Yang et al., 2025), and Gemma2-9B (Gemma Team et al., 2024), and achieves higher average gains than existing steering baselines.

We summarize our contributions as follows:

1. We study the mismatch between static interventions and evolving reasoning states as a limitation of predefined additive directions in

multi-step reasoning, showing that their effects can vary across the intermediate states reached within generated traces.

2. We propose **State-Conditioned Steering**, which preserves the standard additive intervention interface while recomputing the additive update itself from reasoning states at coarse reasoning steps.
3. We evaluate SCS across multi-step reasoning benchmarks with explicit step structure and instruction-tuned backbones, showing consistent end-task gains and more favorable answer-related effects at visited states than predefined-direction baselines.

2 Related Work

Inference-time methods for LLM reasoning. A broad line of work improves LLM reasoning at inference time without updating model parameters. Prompting and decoding methods elicit, aggregate, or structure intermediate reasoning steps through the model’s input-output interface (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022; Wang et al., 2023; Zhou et al., 2023; Yao et al., 2023a,b; Besta et al., 2024). A complementary line of work studies activation-level control, where behavior-relevant directions are identified in representation space and added to internal states during generation (Subramani et al., 2022; Turner et al., 2023; Rimsky et al., 2024; Li et al., 2023; Zou et al., 2023). Our work belongs to this line of activation steering research, with a specific focus on adapting interventions to intermediate states visited during multi-step reasoning.

Interpreting intermediate model states. Intermediate computations in LLM reasoning contain structured signals that can be inspected at both representation and reasoning-step levels. Hidden-state analyses show that internal representations expose evolving prediction states and latent knowledge during decoding (Belrose et al., 2023; Ghandeharion et al., 2024; Burns et al., 2023; Azaria and Mitchell, 2023). Process supervision, reasoning-step analyses, and CoT faithfulness studies further indicate that intermediate reasoning steps contain information about answer support and reasoning dependence beyond final-answer correctness alone (Lightman et al., 2024; Chen et al., 2024; Lanham et al., 2023; Tanneru et al., 2024; Paul et al.,

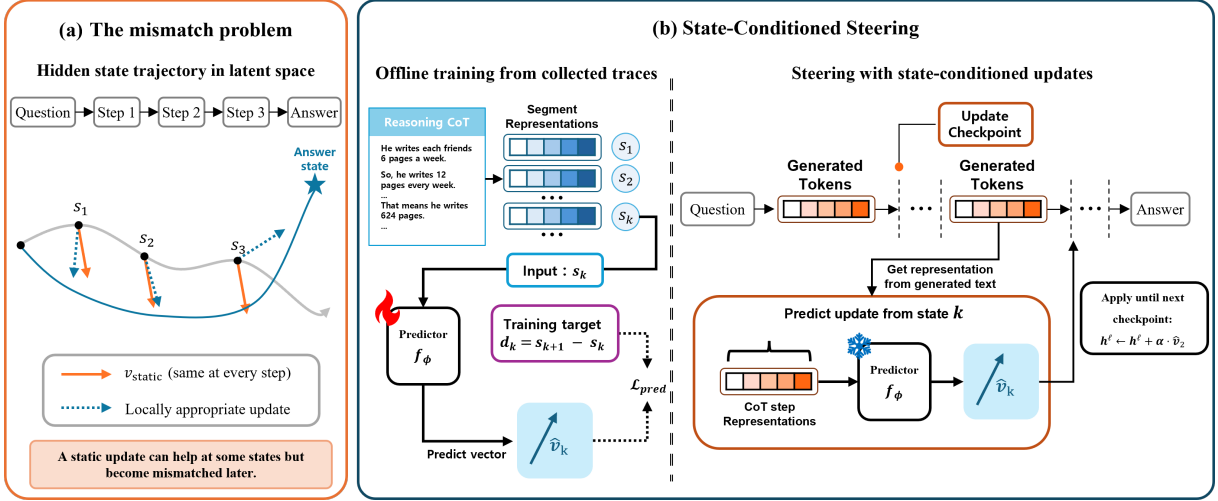


Figure 2: **Overview of State-Conditioned Steering for evolving reasoning traces.** (a) illustrates the mismatch problem where a predefined update can become locally mismatched as reasoning states evolve. (b) summarizes the State-Conditioned Steering (SCS) pipeline, with offline training from generated traces on the left side and steering interface during decoding on the right side.

2024). At the same time, studies on unfaithful rationales and monitorability suggest that surface-level text traces do not always faithfully reflect internal decision-critical factors, motivating more direct inspection of latent states (Turpin et al., 2023; Bentham et al., 2024; Wang et al., 2026). These findings motivate our approach to evaluating and guiding steering behavior based on the local states visited as reasoning unfolds.

Adaptive activation steering. Recent work has explored ways to make activation steering more adaptive. Some frameworks derive reasoning-relevant intervention signals from chain-of-thought traces (Chen et al., 2025; Nguyen and Le, 2026; Højer et al., 2025; Li et al., 2025b; Cho et al., 2026; Li et al., 2025a), while others condition steering behavior on input semantics, inference-time calibration, or localized sparse components (Wang et al., 2025; Stoehr et al., 2024; Scalena et al., 2024; Kang and Kim, 2026; Feng et al., 2026; Doan et al., 2026; Arad et al., 2025; Soo et al., 2025; Gao et al., 2025; Marks et al., 2025). These methods differ in what they adapt. Some adjust intervention timing or strength, while others gate interventions using input semantics, external verifiers, or geometric drift signals. For instance, verifier-guided latent steering rescales interventions using an external verifier (Nguyen and Le, 2026), whereas manifold-based attention steering intervenes when a proximity threshold indicates that activations have drifted from a correctness manifold (Li et al., 2026). In contrast,

SCS focuses on a narrower question of whether the additive update itself should be recomputed. SCS therefore retains the standard additive interface and changes the source of the update, rather than detecting drift, selecting among alternative reasoning paths, or revising the current trajectory.

3 State-Conditioned Steering

In this section, we first define the additive steering interface in Section 3.1 and present **State-Conditioned Steering (SCS)**. Our core motivation is to align the steering vector with the model’s ongoing reasoning progress, thereby mitigating the mismatch between a predefined update and the evolving reasoning state. Figure 2(a) illustrates this motivation by depicting how a predefined update can provide only a coarse summary of the evolving hidden states. The key distinction is that SCS estimates the additive vector from the current reasoning state rather than using a predefined vector. Finally, we describe how SCS learns state-conditioned updates from collected representations in Section 3.2 and applies them during decoding in Section 3.3.

3.1 Additive Activation Steering

We adopt the standard additive steering interface, where an intervention vector is added to the model’s internal representation during decoding (Rimsky et al., 2024; Li et al., 2023). For hidden state $h_t^{(\ell)} \in \mathbb{R}^d$ at token position t and selected

layer ℓ , the intervention is written as

$$h_t^{(\ell)} \leftarrow h_t^{(\ell)} + \alpha \cdot v, \quad (1)$$

where $v \in \mathbb{R}^d$ is the intervention direction and α controls the intervention strength. In this setting, the direction v is fixed before generation and reused across intervention positions. This provides a simple and model-agnostic control interface, but it fails to address state shifts during reasoning, thereby inducing a mismatch between the predefined vector and the evolving states. To mitigate this mismatch, we propose an approach where the steering vector is adaptively estimated from the visited state.

3.2 Offline Training from Reasoning Traces

This section briefly outlines the offline training pipeline as illustrated in Figure 2(b). We begin by collecting representations and training a predictor that estimates steering directions.

3.2.1 Constructing Dataset from Reasoning Text

To train our predictor, we collect hidden state representations. Given a training example, we divide the textual solution into sentence- or step-level units. For each unit, we obtain a representation $u_m \in \mathbb{R}^d$ by mean pooling hidden states at a selected layer ℓ over the tokens corresponding to that unit. We group adjacent units into coarser reasoning segments to make the training simple.

We compute a local coherence score $q_m = \cos(u_m, c_m)$, where c_m is the average of the neighboring unit representations u_{m-1} , u_m , and u_{m+1} , measuring how well a unit aligns with its local context. For segmentation, the relevant signal is not the absolute coherence of a single unit but whether coherence drops sharply between adjacent units. We therefore use the change $\Delta q_m = q_{m+1} - q_m$ as a boundary signal, identifying a local coherence drop and inserting a segment boundary when Δq_m falls below a threshold τ . Thus τ controls the granularity of the coarse reasoning-state segments used for representation collection, and is not a test-time steering threshold. This procedure produces sets of unit indices $P_1, \dots, P_K$. The segment representation is then computed by averaging the unit representations:

$$s_k = \frac{1}{|P_k|} \sum_{m \in P_k} u_m, \quad s_k \in \mathbb{R}^d. \quad (2)$$

The target displacement is defined as the latent displacement toward the next segment representation:

$$d_k = s_{k+1} - s_k. \quad (3)$$

Each segment and target displacement pair yields a training pair (s_k, d_k) , mapping a reasoning state to its subsequent latent displacement.

3.2.2 Predictor Training for Spatial Displacements

Given a source state s_k , we train a predictor f_ϕ to output an additive update $\hat{v}_k = f_\phi(s_k) \in \mathbb{R}^d$. It is trained to match the observed displacement $d_k = s_{k+1} - s_k$ using

$$\mathcal{L}_{\text{pred}} = \sum_k \left[\text{MSE}(\hat{v}_k, d_k) + \lambda_{\cos} (1 - \cos(\hat{v}_k, d_k)) \right]. \quad (4)$$

The first term matches the displacement magnitude, while the second term encourages directional alignment.

3.3 Steering with State-Conditioned Updates

Figure 2(b) briefly illustrates the steering procedure of SCS during inference. During decoding, we define the update checkpoints r_k using text markers, such as newlines or step-delimiters. Whenever a checkpoint r_k is encountered, we extract the hidden states corresponding to the text generated since the previous checkpoint, and mean-pool them to obtain the segment representation s_k . The predictor then takes s_k to estimate the displacement vector, which is applied to the residual stream for the subsequent tokens until the next checkpoint is reached. We substitute the static direction v in Eq. (1) with a state-conditioned vector $\hat{v}_k$:

$$h_t^{(\ell)} \leftarrow h_t^{(\ell)} + \alpha \hat{v}_k, \quad r_k < t \leq r_{k+1}. \quad (5)$$

Because each update $\hat{v}_k$ is derived from the generated prefix, SCS does not require access to future tokens. Additional implementation and configuration details are provided in Appendix A.

4 Experimental Setup

Models and Datasets We evaluate State-Conditioned Steering (SCS) on three instruction-tuned backbones: Llama3-8B (Grattafiori et al., 2024), Qwen2.5-7B (Yang et al., 2025), and Gemma2-9B (Gemma Team et al., 2024). This

Model	Method	GSM8K	MATH500	StrategyQA	Avg.
Llama3-8B	prompting-only	78.85	29.60	86.20	64.88
	CAA	79.53 ^{+0.68}	28.20 ^{-1.40}	87.44 ^{+1.24}	65.06 ^{+0.18}
	SEAL	79.00 ^{+0.15}	26.40 ^{-3.20}	—	—
	SADI	78.09 ^{-0.76}	30.60 ^{+1.00}	88.53 ^{+2.33}	65.74 ^{+0.86}
	SAE-free	79.15 ^{+0.30}	30.40 ^{+0.80}	86.67 ^{+0.47}	65.41 ^{+0.53}
	Q2Ans	79.38 ^{+0.53}	26.40 ^{-3.20}	87.51 ^{+1.31}	64.43 ^{-0.45}
	Ours	81.20 ^{+2.35}	30.00 ^{+0.40}	87.05 ^{+0.85}	66.08 ^{+1.20}
Qwen2.5-7B	prompting-only	89.99	74.80	90.46	85.08
	CAA	90.74 ^{+0.75}	75.20 ^{+0.40}	90.93 ^{+0.47}	85.62 ^{+0.54}
	SEAL	91.13 ^{+1.13}	75.80 ^{+1.00}	—	—
	SADI	90.22 ^{+0.23}	74.40 ^{-0.40}	92.95 ^{+2.49}	85.86 ^{+0.78}
	SAE-free	91.13 ^{+1.14}	75.40 ^{+0.60}	91.01 ^{+0.55}	85.85 ^{+0.77}
	Q2Ans	90.67 ^{+0.68}	73.80 ^{-1.00}	89.93 ^{-0.53}	84.80 ^{-0.28}
	Ours	91.66 ^{+1.67}	76.60 ^{+1.80}	91.63 ^{+1.17}	86.63 ^{+1.55}
Gemma2-9B	prompting-only	87.34	45.20	93.57	75.37
	CAA	87.34 ^{+0.00}	46.00 ^{+0.80}	93.57 ^{+0.00}	75.64 ^{+0.27}
	SEAL	87.64 ^{+0.30}	44.60 ^{-0.60}	—	—
	SADI	87.04 ^{-0.30}	43.80 ^{-1.40}	94.26 ^{+0.69}	75.03 ^{-0.34}
	SAE-free	87.72 ^{+0.38}	45.40 ^{+0.20}	93.57 ^{+0.00}	75.56 ^{+0.19}
	Q2Ans	87.34 ^{+0.00}	45.00 ^{-0.20}	93.41 ^{-0.16}	75.25 ^{-0.12}
	Ours	87.72 ^{+0.38}	47.20 ^{+2.00}	93.80 ^{+0.23}	76.24 ^{+0.87}

Table 1: **Exact-match (EM) accuracy on GSM8K, MATH500, and StrategyQA.** Bold indicates the best result within each model block and evaluation column. Subscripts indicate absolute gains or losses in percentage points relative to *Prompting-only*. “—” denotes settings not applicable under our evaluation pipeline.

allows us to test the method across multiple model families under the same evaluation protocol. We evaluate on three reasoning benchmarks. GSM8K (Cobbe et al., 2021) consists of grade-school math word problems. MATH500 (Lightman et al., 2024) contains competition-level math problems that often require longer reasoning chains. StrategyQA (Geva et al., 2021) is a multi-hop question answering benchmark with shorter, compositionally structured reasoning chains. Together, these datasets cover arithmetic reasoning, competition-level mathematical reasoning, and compositional question answering.

Baselines We compare SCS against prompting-only decoding and representative activation-steering baselines. *Prompting-only* generates answers using deterministic greedy decoding without any hidden-state intervention. *CAA* (Rimsky et al., 2024) represents predefined contrastive steering, adding a single static contrastive direction uniformly during decoding. *SEAL* (Chen et al., 2025) is a reasoning-calibration method that applies a pre-computed steering signal during generated reason-

ing steps. *SAE-free CoT steering* (Li et al., 2025b) constructs reasoning-oriented steering directions from chain-of-thought traces, but keeps the direction static across token positions once generation begins. *SADI* (Wang et al., 2025) represents input-adaptive steering, where the intervention signal depends on input-specific activation patterns rather than being recomputed from intermediate reasoning states reached within the same generated trace.

We also include *Q2Ans* as an internal diagnostic reference rather than a primary baseline. *Q2Ans* estimates a single coarse query-to-answer displacement direction and reuses it throughout decoding, allowing us to test whether SCS gains can be explained by a coarse start-to-end representation shift rather than by conditioning updates on the intermediate states visited during generation.

Evaluation Protocol To comprehensively assess both end-task outcomes and localized intervention dynamics, we evaluate our approach through two complementary perspectives. First, to verify whether our activation steering leads to enhanced final reasoning performance, we report exact-match

Setting	Mean Local Effect (SCS)	Mean Local Effect (CAA)	Pairwise Win Ratio	Mean Effect Difference
Qwen2.5-7B / GSM8K	0.0188	-0.1019	3.63 [2.94, 4.49]	0.1207
Qwen2.5-7B / StrategyQA	0.0695	0.0146	1.73 [1.45, 2.09]	0.0549
Llama3-8B / GSM8K	0.0044	-0.0076	3.72 [3.03, 4.68]	0.0120
Gemma2-9B / GSM8K	0.0192	-0.0475	2.82 [2.33, 3.50]	0.0667

Table 2: **Sample-level comparison of local intervention effects.** For each sample, the local intervention effect (Δm) is averaged across visited checkpoints. The pairwise win ratios (Win_{SCS}/Win_{CAA}) are reported with 95% bootstrap confidence intervals in brackets.

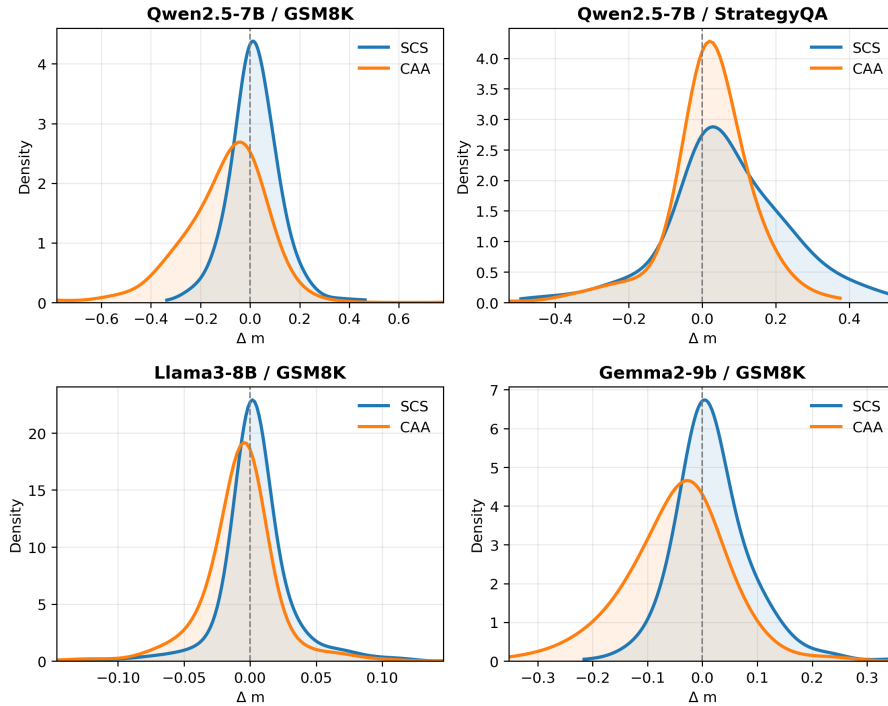


Figure 3: **Distributions of local intervention effects.** The answer-related proxy change (Δm) measures the immediate impact at each visited checkpoint. Compared to the static baseline, State-Conditioned Steering (SCS) shifts the distribution toward positive local gains.

(EM) accuracy under a shared prompt, decoding configuration, and answer-extraction pipeline across all methods. Second, we investigate how interventions immediately modify answer-related output behavior during generation. We define a local proxy m at each visited checkpoint. For GSM8K, m is the probability mass assigned to the correct numeric answer tokens. For StrategyQA, m is the probability margin between the correct and incorrect binary labels, which we use as an output-level proxy for local answer support (Kadavath et al., 2022). These answer-related local proxies m are used only as diagnostics of intervention behavior, not as substitutes for final exact-match evaluation. This diagnostic is motivated by prior work that probes intermediate reasoning traces for answer support and reasoning dependence (Lanham et al., 2023; Paul et al., 2024). However, our approach

differs in that these probes are evaluated online at visited checkpoints, rather than on separately constructed partial prefixes. Third, we report efficiency using generated-token throughput (tokens/sec) as the primary cost metric, since total wall-clock time also reflects differences in generated length and run variation. Because prompting-only, CAA, and SCS share the same generation loop, this comparison is controlled. Averaged across backbones, SCS reduces throughput by roughly 8% relative to prompting-only, while the predictor is trained offline in about one minute per setting and leaves the backbone unchanged. Details on our intervention configurations and cost measurements are provided in Appendix A and Appendix I.

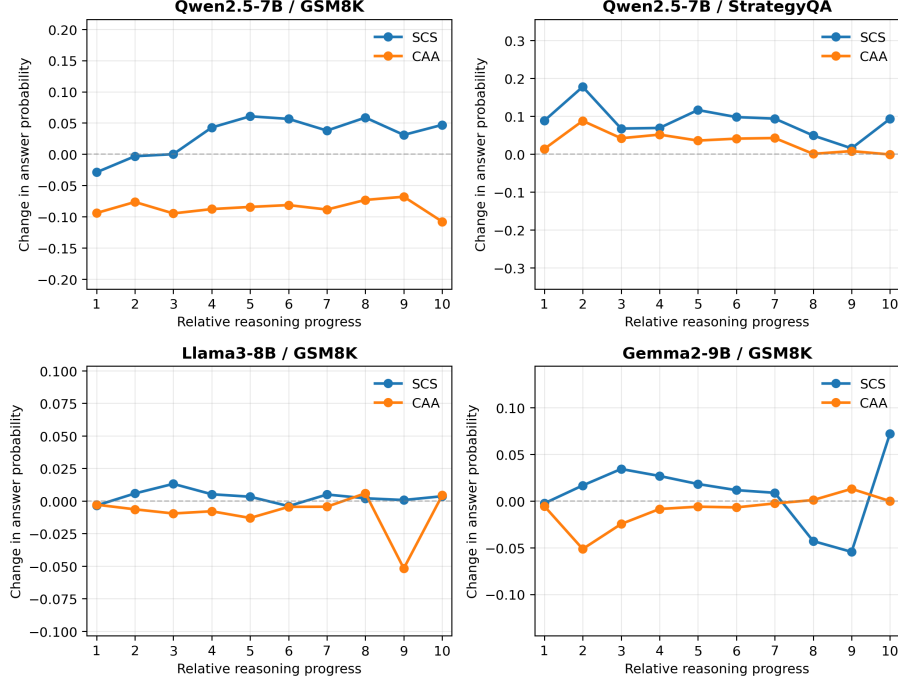


Figure 4: **Local intervention effects across reasoning progress.** Mean proxy change (Δm) is plotted against the relative position in the reasoning trace. Compared to the static baseline, State-Conditioned Steering (SCS) exhibits a more favorable trajectory of positive local gains.

5 State-Conditioned Updates in Reasoning Traces

As mentioned above, we evaluate SCS from two perspectives. We first report end-task exact-match accuracy under a shared decoding and answer-extraction protocol. We then analyze how additive updates behave at the intermediate states where they are applied. For these local analyses, we use the answer-related proxy m defined in Section 4. At each visited state, we measure the local intervention effect as

$$\Delta m = m_{\text{on}} - m_{\text{off}}, \quad (6)$$

where m_{off} and m_{on} denote the proxy m measured with and without the intervention at the same state. A positive Δm indicates that the intervention increases the answer-related proxy at the point where it is applied. We interpret Δm as an intermediate probe of answer support, not as a standalone measure of final correctness. We use CAA as the predefined-direction comparator for this diagnostic. To make this comparison controlled, SCS and CAA are evaluated on the same examples, at the same intervention layer, and with the same configured intervention strength for the corresponding run. Our local answer-related proxy analyses are computed on a fixed 500-example subset.

5.1 End-Task Performance as Behavioral Validation

To evaluate whether intermediate intervention effects are reflected in final reasoning outcomes, we report exact-match accuracy. As reported in Table 1, SCS improves over prompting-only decoding in all evaluated settings and increases the average accuracy for each backbone. While baselines relying on predefined or static directions show inconsistent gains across datasets, SCS provides positive improvements over prompting-only across all evaluated backbone–dataset combinations, supporting the benefit of conditioning the intervention on the current reasoning state rather than reusing a fixed direction.

5.2 Local Effects at Visited Checkpoints

To track the shift in the answer-related proxy caused by interventions, we compute the local effect Δm at visited checkpoints. As illustrated by the distributions of Δm in Figure 3, SCS shifts the overall distribution toward more favorable, positive local changes compared to the predefined direction baseline. This shift indicates that updates estimated from intermediate reasoning states are better matched to the states at which they are applied.

To ensure that this distributional advantage is not

Model	Prompting	Noise	Opposite	SCS
Llama3-8B	78.85	79.61	79.53	81.20
Qwen2.5-7B	89.99	89.69	89.99	91.66
Gemma2-9B	87.34	87.19	86.81	87.72

Table 3: **Directional controls on GSM8K.** Exact-match accuracy for State-Conditioned Steering and directional controls.

Model	Prompting	Shuffled-update	SCS
Llama3-8B	78.85	78.92	81.20
Qwen2.5-7B	89.99	90.37	91.66
Gemma2-9B	87.34	87.11	87.72

Table 4: **State-pairing control on GSM8K.** Exact-match accuracy for the strict shuffled control and State-Conditioned Steering.

merely driven by a small number of large shifts, we complement this view with a paired comparison across examples. For this analysis, we average these Δm values over all valid checkpoints within each generated response, allowing us to evaluate how frequently SCS outperforms the baseline on the exact same instance. As reported in Table 2, SCS yields a favorable intervention effect across most examples. This diagnostic suggests that the local advantage of state conditioning is distributed across diverse inputs rather than concentrated on a few outliers.

5.3 Effects Along Reasoning Progress

To investigate whether intervention effectiveness is sustained as generation progresses, we group the visited checkpoints by their relative position and analyze the mean Δm conditioned on scaled relative reasoning progress. Figure 4 shows that local steering effects vary over the course of reasoning progress. The predefined-direction baseline exhibits less stable local gains across several intermediate intervals, showing that a single reused direction fails to provide uniform effect as the state shifts, whereas SCS maintains a comparatively stable and favorable effect throughout the sequence. This pattern aligns with the geometric analysis in Appendix B, where state-conditioned update centroids become increasingly separated across distant progress intervals. Because the effect of an additive update varies across reasoning progress, estimating updates from visited states provides a better matched intervention than reusing a static direction.

5.4 Vector Control

To isolate the source of these performance gains, we conduct matched-scale controls while preserving the intervention schedule and vector scale. In this section, we report the GSM8K controls and provide additional directional-control results across all evaluation datasets in Appendix F. These controls test whether SCS gains can be explained by simpler alternatives, such as matched-norm random perturbations, reversed update directions, or unpaired predicted updates.

5.4.1 Directional Controls

To test whether the orientation of the predicted update matters beyond perturbing the hidden states, we use matched directional controls that keep the intervention schedule and vector norm but remove or reverse the learned direction (noise and opposite-direction). As shown in Table 3, SCS outperforms both across all three backbones, so the gains are not explained by a norm-matched nonzero vector and the update carries task-relevant directional structure. Results on other datasets are in Appendix F.

5.4.2 State-Pairing Control

We next test whether the learned update distribution alone suffices, or whether each update must be paired with the state it was predicted from. Our shuffled-update control replaces each predicted update with a substitute drawn from updates at the same progress position on other reasoning paths, rescaled to the original norm. This keeps the progress position and scale but breaks the state-update pairing, and is stricter than noise since the substitutes come from the learned distribution. As shown in Table 4, SCS outperforms this baseline across all three backbones, where shuffling often neutralizes the gains. State-update matching is therefore a key factor behind the gains, beyond the learned distribution alone.

5.5 Reliability across Difficulty and Length

We break down accuracy by problem difficulty and generation length for Qwen2.5-7B on MATH500 (Appendix D). The largest observed gains occur in the mid-difficulty and medium- and long-generation bins, while the gains are smaller in the short-generation bin and negative at the easiest and hardest difficulty levels. These subgroup results are descriptive and do not by themselves establish why the effectiveness of SCS varies with difficulty

or generation length. A qualitative L5 example further shows that SCS can follow a fluent but invalid reasoning trace (Table 16). This example illustrates a possible failure mode rather than explaining the aggregate performance on L5 problems.

6 Conclusion

We studied activation steering for multi-step reasoning from the perspective of evolving intermediate states, focusing on benchmarks whose reasoning traces have explicit step structure. State-Conditioned Steering preserves the standard additive intervention interface while recomputing the additive update itself from visited reasoning states during generation, rather than reusing a single pre-defined direction. Across GSM8K, MATH500, and StrategyQA, SCS improves over prompting-only decoding in all evaluated backbone–dataset settings. Local diagnostics and control experiments further suggest that the gains depend not only on injecting vectors of comparable scale, but also on the learned orientation of the update and its pairing with the visited state. These results point to a promising direction for dynamic steering methods that account for the model’s evolving internal states while retaining the simplicity of additive interventions.

7 Limitations

State-Conditioned Steering is evaluated on reasoning benchmarks whose traces exhibit relatively explicit textual structure. In our main experiments, update checkpoints are defined using simple surface cues such as line breaks or step delimiters. Although the fixed-interval ablation shows that these markers are not intrinsic to state-conditioned updating, their availability provides convenient refresh points. How well SCS transfers to implicit, irregularly segmented, non-monotonic, or non-reasoning generation remains open. Such settings may require learned boundary detection or continuous update scheduling rather than manually specified textual cues.

SCS conditions its update on the currently visited trajectory but does not determine whether that trajectory is correct. In particular, it neither selects among competing hypotheses nor revises an earlier reasoning decision. This limitation becomes important in forking or garden-path settings, where several continuations may remain plausible and the appropriate path cannot be inferred from the cur-

rent state alone (Bigelow et al., 2025). A failure case on an L5 problem illustrates that SCS can follow the model into a fluent but incorrect branch. Addressing such cases would require an additional path-selection, verification, or uncertainty-aware mechanism. These capabilities are complementary to the state-conditioned update studied here.

Our experiments also use deterministic greedy decoding to isolate intervention effects from sampling variance. SCS is not intrinsically restricted to greedy decoding because its update is applied before token selection, but we have not systematically evaluated its interaction with nucleus sampling, beam search, or self-consistency. Under stochastic decoding, different sampled continuations may induce different visited states and update sequences, while search-based decoding may require maintaining separate state-conditioned updates for multiple hypotheses. Evaluating these interactions is an important direction for future work.

Our empirical claims are restricted to the evaluated 7B–9B backbones and per-setting predictors. A single larger-scale feasibility check using Llama3-70B on MATH500 changes accuracy only marginally from 50.8 to 51.2. We therefore do not treat this result as evidence of favorable scaling, and whether the update mismatch persists or attenuates in substantially larger models remains unresolved. The predictor is also trained separately for each backbone–dataset setting and behaves as a task-calibrated transition module. Cross-dataset transfer between GSM8K and MATH500 produces positive but non-uniform gains below in-domain SCS, so the current predictor should not be assumed to generalize across models or datasets without recalibration.

Finally, SCS summarizes intermediate states through mean pooling and learns displacements between adjacent segment representations. These choices provide a simple controlled interface but may discard token-level structure or represent multiple plausible transitions with a single averaged update. Moreover, our segment construction uses cosine similarity to measure local coherence between adjacent reasoning units and identify segment boundaries. Because cosine similarity in contextual representation spaces can be dominated by a small number of rogue dimensions (Timkey and van Schijndel, 2021), the resulting boundaries may be sensitive to the representation geometry and the selected threshold. Comparing standardized cosine similarity with dot-product, Eu-

clidean, learned, and subspace-based boundary criteria could clarify how sensitive the segmentation is to the choice of similarity measure. More expressive state summaries, alternative segmentation criteria, and mixture-based or distributional transition targets are therefore natural extensions.

Ethical Considerations

Like other steering methods, state-conditioned activation steering alters internal representations at inference time and could be used to change model behavior in ways that are hard for users to observe. The diagnostics here are intended to characterize intervention behavior on controlled reasoning benchmarks, not to certify the safety or reliability of steering in deployment. We therefore view SCS as an analysis and intervention framework that requires task-specific validation before practical use, and we do not evaluate it in high-stakes or safety-critical settings.

Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00457882, AI Research Hub Project) and (No. RS-2026-25522672, Development of Unified Reasoning Technology Mimicking Human Cognition for Hierarchical Understanding and Unbounded Problem Solving). This work was also partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) under the artificial intelligence graduate school program (Korea University) (No. RS-2019-II190079), the artificial intelligence star fellowship support program to nurture the best talents (IITP-2026-RS-2025-02304828) grant funded by the Korea government (MSIT), and the Institute of Information & Communications Technology Planning & Evaluation (IITP)-ITRC (Information Technology Research Center) grant funded by the Korea government (MSIT) (IITP-2026-RS-2024-00436857).

References

Dana Arad, Aaron Mueller, and Yonatan Belinkov. 2025. [SAEs are good for steering – if you select the right features](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10241–10259, Suzhou, China. Association for Computational Linguistics.

Amos Azaria and Tom Mitchell. 2023. [The internal state of an LLM knows when it’s lying](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Oliver Bentham, Nathan Stringham, and Ana Marasovic. 2024. [Chain-of-thought unfaithfulness as disguised accuracy](#). *Transactions on Machine Learning Research*. Reproducibility Certification.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoeffler. 2024. [Graph of thoughts: solving elaborate problems with large language models](#). In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press.

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. 2025. [Forking paths in neural text generation](#). In *International Conference on Learning Representations*, volume 2025, pages 17134–17167.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. [Discovering latent knowledge in language models without supervision](#). In *The Eleventh International Conference on Learning Representations*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jinxuan Zhou, and Wanxiang Che. 2024. [Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 54872–54904.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. [SEAL: Steerable reasoning calibration of large language models for free](#). In *Second Conference on Language Modeling*.

Seonglae Cho, Zekun Wu, and Adriano Koshiyama. 2026. [Corrsteer: Generation-time LLM steering via correlated sparse autoencoder features](#). In *Forty-third International Conference on Machine Learning*.

- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Toan Doan, Uyen Le, and Thin Nguyen. 2026. [Causal activation steering via sparse mediation](#). In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 1079–1097, Rabat, Morocco. Association for Computational Linguistics.
- Zijian Feng, Tianjiao Li, Zixiao Zhu, Hanzhang Zhou, Junlang Qian, Li Zhang, Chua Deryl, Lee Mak, Gee Ng, and Kezhi Mao. 2026. [Fine-grained activation steering: Steering less, achieving more](#). In *International Conference on Learning Representations*, volume 2026, pages 39421–39443.
- Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. [Scaling and evaluating sparse autoencoders](#). In *International Conference on Learning Representations*, volume 2025, pages 26721–26754.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies](#). *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. [Patchscopes: A unifying framework for inspecting hidden representations of language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15466–15490.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Bertram Højer, Oliver Jarvis, and Stefan Heinrich. 2025. [Improving reasoning performance in large language models via representation engineering](#). In *International Conference on Learning Representations*, volume 2025, pages 44746–44763.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.
- Minjae Kang and Jaehyung Kim. 2026. [Enhancing instruction following of llms via activation steering with dynamic rejection](#). In *International Conference on Learning Representations*, volume 2026, pages 118535–118571.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA. Curran Associates Inc.
- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.
- Ian Li, Philip Chen, Max Huang, Andrew Park, Loris D’Antoni, and Rose Yu. 2025a. [Steering LLMs’ reasoning with activation state machines](#). In *First Workshop on Foundations of Reasoning in Language Models*.
- Ian Li, Kapilesh Guruprasad, Raunak Sengupta, Niranjan Satish, Loris D’Antoni, and Rose Yu. 2026. [Manifold-guided attention steering](#). In *ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning*.
- Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. [Inference-time intervention: Eliciting truthful answers from a language model](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530. Curran Associates, Inc.
- Zihao Li, Xu Wang, Yuzhe Yang, Ziyu Yao, Haoyi Xiong, and Mengnan Du. 2025b. [Feature extraction and steering for enhanced chain-of-thought reasoning in language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10893–10913, Suzhou, China. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.
- Samuel Marks, Can Rager, Eric Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. [Sparse feature circuits: Discovering and editing interpretable causal graphs in language models](#). In *International Conference on Learning Representations*, volume 2025, pages 23888–23923.
- Tuc Nguyen and Thai Le. 2026. Atlas: Adaptive test-time latent steering with external verifiers for enhancing llms reasoning. *arXiv preprint arXiv:2601.03093*.

- Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. [Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15012–15032, Miami, Florida, USA. Association for Computational Linguistics.
- Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. [Steering llama 2 via contrastive activation addition](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.
- Daniel Scalena, Gabriele Sarti, and Malvina Nissim. 2024. [Multi-property steering of large language models with dynamic activation composition](#). In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 577–603, Miami, Florida, US. Association for Computational Linguistics.
- Samuel Soo, Wesley Teng, Chandrasekaran Balaganesh, Tan Guoxian, and Ming YAN. 2025. [Interpretable steering of large language models with feature guided activation additions](#). In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.
- Niklas Stoehr, Kevin Du, Vésteinn Snæbjarnarson, Robert West, Ryan Cotterell, and Aaron Schein. 2024. [Activation scaling for steering and interpreting language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8189–8200, Miami, Florida, USA. Association for Computational Linguistics.
- Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. [Extracting latent steering vectors from pretrained language models](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.
- Sree Harsha Tanneru, Dan Ley, Chirag Agarwal, and Himabindu Lakkaraju. 2024. [On the difficulty of faithful chain-of-thought reasoning in large language models](#). In *Trustworthy Multi-modal Foundation Models and AI Agents (TiFA)*.
- William Timkey and Marten van Schijndel. 2021. [All bark and no bite: Rogue dimensions in transformer language models obscure representational quality](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. [Steering language models with activation engineering](#). *arXiv preprint arXiv:2308.10248*.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. [Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 74952–74965. Curran Associates, Inc.
- Han Wang, Yifan Sun, Brian Ko, Mann Talati, Jiawen Gong, Zimeng Li, Naicheng Yu, Xucheng Yu, Wei Shen, Vedant Jolly, et al. 2026. [Monitorbench: A comprehensive benchmark for chain-of-thought monitorability in large language models](#). *arXiv preprint arXiv:2603.28590*.
- Weixuan Wang, Jingyuan Yang, and Wei Peng. 2025. [Semantics-adaptive activation intervention for llms via dynamic steering vectors](#). In *International Conference on Learning Representations*, volume 2025, pages 79334–79351.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2025. [Qwen2.5 technical report](#). *arXiv preprint arXiv:2412.15115*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. [Tree of thoughts: Deliberate problem solving with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. [React: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *The Eleventh International Conference on Learning Representations*.
- Yufa Zhou, Yixiao Wang, Xunjian Yin, Shuyan Zhou, and Anru Zhang. 2026. [The geometry of reasoning: Flowing logics in representation space](#). In *International Conference on Learning Representations*, volume 2026, pages 72711–72739.

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

A Implementation Details

Hardware and decoding. All experiments were conducted on an NVIDIA RTX A6000 GPU with 48GB memory. For open-ended generation, we use greedy decoding with a maximum generation length of 2048 tokens. Unless otherwise stated, all methods use the same prompt format, decoding configuration, and answer extraction procedure.

Dataset construction and segmentation. For GSM8K and MATH-style reasoning tasks, we use annotated training solutions for predictor training. For StrategyQA, we use generated reasoning traces from 1,000 training examples for predictor training and evaluate the model on the remaining 1,290 held-out examples. Given a reference reasoning text, we construct segment representations using the procedure described in Section 3.2. For segment construction, we set the segmentation boundary threshold τ to -0.03 . The final unit is always used as the endpoint of the last segment. In our experiments, we also include the displacement from the question representation to the first reasoning segment as the initial training target.

Predictor. For State-Conditioned Steering, we train a predictor for each backbone–dataset setting. The predictor takes a segment representation $s_k \in \mathbb{R}^d$ as input and outputs an additive update $\hat{v}_k \in \mathbb{R}^d$. The update prediction module is implemented with MLP layers using SiLU activations and hidden dimension $H = 512$. Dropout with rate 0.1 is applied within the encoder and decoder, and the final layer of the decoder is initialized to zero so that the initial predicted update is the zero vector. During training, we apply annealed perturbations to the intermediate latent representation as a regularizer, while inference uses the deterministic prediction path. Shared predictor-training settings are summarized in Table 5. The values of λ_{cos} used for predictor training are reported in Table 6.

Intervention configurations. During online decoding, update checkpoints determine when the completed reasoning segment is summarized and a new additive update is estimated. We instantiate checkpoints using textual cues in the generated reasoning trace. For GSM8K and StrategyQA, line-break patterns provide the main cue. For MATH500, we also use step headers, with conservative filtering for displayed math and LaTeX blocks. At the beginning of decoding,

Item	Value
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	0.01
Batch size	128
Epochs	100
Learning-rate schedule	OneCycleLR
Hidden dimension	512
Activation	SiLU
Dropout	0.1
Validation split	10% of training pairs
Checkpoint selection	Lowest validation loss
EMA decay	0.995

Table 5: **Predictor training configuration.** Shared training settings used for all predictors unless otherwise stated.

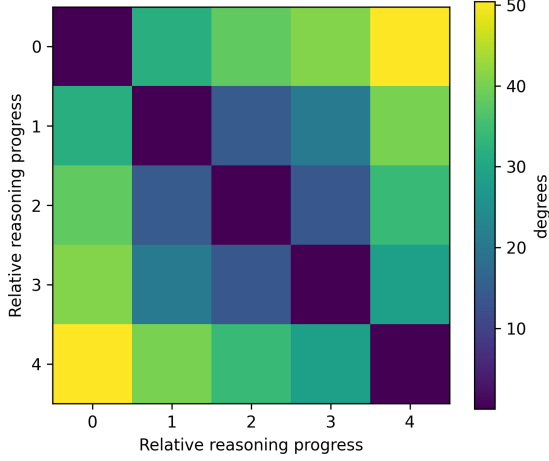
Backbone	GSM8K	MATH500	StrategyQA
Llama3-8B	2.0	2.0	2.0
Qwen2.5-7B	5.0	3.0	2.0
Gemma2-9B	2.0	2.0	2.0

Table 6: Values of λ_{cos} used in Eq. 4.

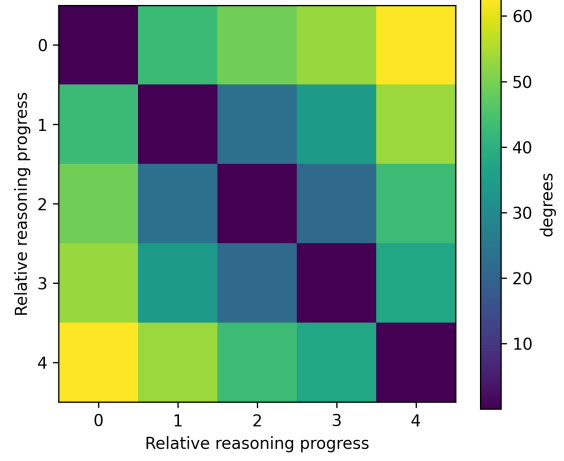
SCS initializes the intervention from the available prompt prefix and then updates it at subsequent checkpoints. For exact-match evaluations, we use the final layer as the target intervention layer across all EM reports. Rather than optimizing the depth for each backbone individually, we made this choice based on identifying the overarching average validation trends across all models. Specifically, we conducted a preliminary sweep over three relative-depth candidates, $\{0.5, 0.75, 1.0\}$, on a 500-example validation subset of GSM8K under varying steering strengths $\alpha \in \{0.1, 0.5, 1.0\}$. After identifying the final layer (1.0) as the most robust region on average, we fixed this target layer and selected α for each backbone–dataset setting on its corresponding held-out validation split using the search grids reported in Table 7. Our code is available at <https://github.com/Tae-Won-Kim96/State-Conditioned-Steering>.

B Update Geometry Across Reasoning Progress

We further examine whether predicted updates remain geometrically similar as generation proceeds. Following the analysis in Section 5, we group visited checkpoints by relative reasoning progress and compare the geometry of predicted updates across



(a) Qwen2.5-7B



(b) Llama3-8B

Figure 5: **Principal angle geometry of predicted updates across reasoning progress.** Predicted vectors are grouped by relative reasoning progress intervals. For each interval, we estimate an update subspace from the predicted updates and compute principal angles between subspaces from different intervals.

Backbone	Search grid	GSM8K	MATH500	StrategyQA
Llama3-8B	{0.05, 0.1, 0.15, 0.2, 0.3}	0.15	0.15	0.2
Qwen2.5-7B	{0.5, 0.75, 1.0, 1.25, 1.5}	1.25	0.75	1.25
Gemma2-9B	{0.5, 0.75, 1.0, 1.5, 2.0}	1.0	1.5	1.5

Table 7: **Steering strengths for exact-match evaluation.** Search grids and selected values of α used at the target layer for SCS in final EM reports.

Backbone	Gap 1	Gap 2	Gap 3	Gap 4
Qwen2.5-7B	0.061	0.146	0.266	0.475
Llama3-8B	0.202	0.412	0.672	0.936

Table 8: **Predicted-update geometry across reasoning progress.** We divide the reasoning trace into five sequential intervals (0 to 4) based on relative progress. We compute the cosine distance between the predicted-update centroids of any two intervals, and average these distances by the distance gap between the intervals.

progress intervals. This diagnostic tests whether the learned updates are well described by a single progress-invariant distribution or instead vary over the generated reasoning trace. For each relative reasoning progress interval, we collect predicted updates and estimate a low-dimensional update subspace using PCA. We then measure principal angles between update subspaces from different intervals. Smaller angles indicate more similar update subspaces, while larger angles indicate greater differences between the directions spanned by predicted updates across reasoning progress. We treat these analyses as update vector geometry diagnostics rather than evidence for discrete reasoning stages. Figure 5 shows the principal angle structure for

Backbone	Spearman ρ	Bootstrap 95% CI
Qwen2.5-7B	0.146	[0.062, 0.229]
Llama3-8B	0.235	[0.151, 0.313]
Gemma2-9B	0.161	[0.071, 0.242]

Table 9: **Consistency between local effects and final output proxies.** For each sample, we compute the mean local intervention effect over visited checkpoints and correlate it with the final answer-related proxy. We report Spearman rank correlations with bootstrap 95% confidence intervals.

Qwen2.5-7B and Llama3-8B. Table 8 confirms this geometric shift at the centroid level, where the cosine distance between respective update centroids grows substantially as the interval gap increases. This monotonic increase indicates that intervals farther apart in the generation trace employ predicted updates with progressively distinct average directions. Together, these results suggest that predicted updates vary over the generated reasoning trace, supporting the view that the learned update is not described as a single progress-invariant direction.

C Local Effects and Final Output Proxies

We additionally examine whether intermediate answer-support diagnostics show any association with final output behavior. Prior work suggests that intermediate reasoning traces can carry information about answer support and final-answer dependence, while cautioning that such signals should not be treated as definitive evidence of faithful reasoning (Lanham et al., 2023; Paul et al., 2024). Motivated

Model	Dataset	Prompting-only	Noise Direction	Opposite Direction	SCS
Llama3-8B	GSM8K	78.85	79.61	79.53	81.20
	MATH500	29.60	27.80	30.00	30.00
	StrategyQA	86.20	86.51	86.59	87.05
Qwen2.5-7B	GSM8K	89.99	89.69	89.99	91.66
	MATH500	74.80	75.80	76.00	76.60
	StrategyQA	90.46	90.93	91.01	91.63
Gemma2-9B	GSM8K	87.34	87.19	86.81	87.72
	MATH500	45.20	44.80	45.20	47.20
	StrategyQA	93.57	93.72	93.64	93.80

Table 10: **Directional-control analysis across models and datasets.** Exact-match accuracy comparing Prompting-only, Noise direction, Opposite direction controls, and State-Conditioned Steering (SCS).

Model	Prompting-only	Unit-level update	Segment-level update
Llama3-8B	78.85	80.06	81.20
Qwen2.5-7B	89.99	90.67	91.66
Gemma2-9B	87.34	87.49	87.72

Table 11: **Ablation of coherence-based segment construction on GSM8K.** The unit-level variant removes coherence-based grouping and trains updates between adjacent reasoning units.

by this view, we use this analysis only as a post-hoc consistency check.

For each sample i , we compute the mean local intervention effect over visited checkpoints:

$$\Delta \bar{m}_i = \frac{1}{K_i - 1} \sum_{k=1}^{K_i-1} (m_{i,k}^{\text{on}} - m_{i,k}^{\text{off}}), \quad (7)$$

where K_i denotes the total number of update checkpoints visited during the generation of sample i , and $k \in \{1, 2, \dots, K_i\}$ indexes the sequential checkpoints. The terms $m_{i,k}^{\text{on}}$ and $m_{i,k}^{\text{off}}$ represent the answer-related proxies measured with and without the intervention at visited checkpoint k . We intentionally exclude the final checkpoint K_i from this average to prevent measuring the local effect and the final output proxy at the identical decoding position. We then measure the final answer-related proxy m_i^{final} at the end of the trace for the same sample. We report Spearman rank correlations between $\Delta \bar{m}_i$ and m_i^{final} , with bootstrap 95% confidence intervals over samples. The resulting associations are positive across all three backbones, but small in magnitude. We therefore interpret this result as a weak consistency signal between local answer-related gains and final output behavior.

	Split	Prompting-only	SCS
Length	Short	91.02	91.62 ^{+0.60}
	Medium	82.04	84.43 ^{+2.40}
	Long	51.20	53.61 ^{+2.41}
Difficulty	Easy (L1–2)	90.98	90.23 ^{-0.75}
	Mid (L3–4)	77.68	82.40 ^{+4.72}
	Hard (L5)	53.73	52.99 ^{-0.75}

Table 12: **Reliability by generation length and problem difficulty (Qwen2.5-7B, MATH500).** Exact-match accuracy with gains over Prompting. Length bins split all 500 examples into equal-size groups, and difficulty follows the MATH levels.

D Reliability across Difficulty and Length

Table 12 breaks down accuracy by problem difficulty and generation length on MATH500 with Qwen2.5-7B. To construct the length bins, we sort all 500 examples by generation length and divide them into three equal-sized groups. SCS improves over prompting-only in all three bins, with larger gains in the medium and long bins. Across difficulty levels, the largest gain occurs on mid-level (L3–4) problems, while performance decreases slightly on easy (L1–2) and hard (L5) problems. These results show that the overall improvement is concentrated in medium- and long-generation examples and mid-level problems. SCS does not consistently improve performance at the easiest or hardest difficulty levels.

E Layer Sensitivity

Figure 6 reports layer-sensitivity results for State-Conditioned Steering. The purpose of this analysis is to examine whether the observed gains are restricted to a single favorable deployment layer. The

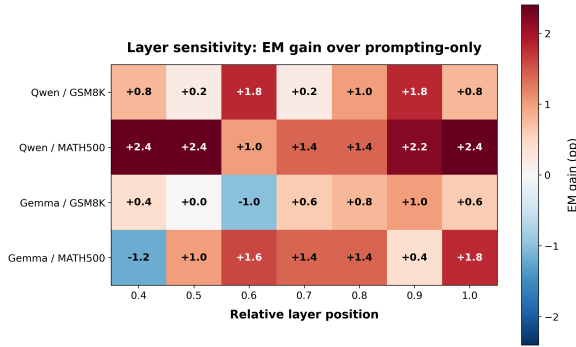


Figure 6: **Layer sensitivity of State-Conditioned Steering.** We report EM gains over prompting-only decoding across relative layer depths. Rows correspond to backbone–dataset settings, columns correspond to relative layer depths, and each cell shows the absolute EM gain in percentage points.

best layer varies across backbones and datasets, and some mid-layer settings underperform prompting-only decoding. However, the late layer deployment used in the main experiments remains competitive across the evaluated settings.

F Additional Directional-Control Results

Table 10 reports additional directional-control results across all evaluation datasets. The controls use the same intervention schedule and matched vector norms as State-Conditioned Steering, but replace the predicted update with either a random vector or the opposite direction. These controls occasionally improve individual settings, consistent with prior observations that non-targeted activation perturbations can affect generation behavior (Rimsky et al., 2024; Wang et al., 2025). However, in our experiments, the directional controls do not consistently yield the performance gains achieved by State-Conditioned Steering. Overall, State-Conditioned Steering remains the strongest or tied for the strongest across all settings, suggesting that the learned update provides a more reliable intervention signal than generic perturbations.

We apply the same matched-norm noise and opposite-direction controls to the SAE-free CoT baseline on GSM8K (Table 13). The original SAE-free update achieves higher accuracy than both controls on all three backbones, although the margins are small. This result provides a directional sanity check showing that neither matched-norm noise nor the opposite direction reaches the performance of the original update.

Model	SAE-free	Noise Direction	Opposite Direction
Llama3-8B	79.15	78.85	79.00
Qwen2.5-7B	91.13	90.97	90.52
Gemma2-9B	87.72	87.49	87.57

Table 13: **Directional controls for SAE-free CoT on GSM8K.** Exact-match accuracy for the original SAE-free update and its matched-norm noise and opposite-direction controls.

Model	Prompting	SCS (checkpoint)	SCS (fixed token)
Llama3-8B	78.85	81.20	80.97
Qwen2.5-7B	89.99	91.66	90.67
Gemma2-9B	87.34	87.72	87.64

Table 14: **Checkpoint cue ablation on GSM8K.** Exact-match accuracy when updates are refreshed at textual checkpoints or every 16 generated tokens.

G Checkpoint Ablation

SCS refreshes its update at checkpoints defined by textual cues such as newlines and step delimiters. These markers provide practical refresh points but are not intrinsic to the method, whose core operation is to re-estimate the update from a visited state. We replace these textual checkpoints with checkpoints placed every 16 generated tokens on GSM8K (Table 14). The fixed-interval variant remains above prompting-only on all three backbones. Explicit textual markers therefore provide useful timing cues but are not required for state-conditioned updating.

H Ablation on Segment Construction

To understand the effect of our segment construction, we compare our segment-level approach against a unit-level variant. In this setup, we do not group adjacent units into segments. Instead, the predictor is trained on the displacements between adjacent mean-pooled unit representations. As shown in Table 11, the unit-level variant outperforms prompting-only across all three backbones, while the segment-level approach further improves over the unit-level variant. This result indicates that steering does not depend entirely on the specific segmentation rule, although grouping states into coherent segments can provide a clearer training signal.

I Inference and Training Cost

Table 15 reports inference cost across all backbone–dataset settings. We use generated-token through-

Model	Method	GSM8K		MATH500		StrategyQA	
		Wall	tok/s	Wall	tok/s	Wall	tok/s
Llama3-8B	Prompting	1:09:12	35.18	1:17:06	35.37	0:44:17	35.40
	CAA	1:07:07	35.37	1:10:19	35.56	0:44:00	35.54
	SCS	1:14:15	32.27	1:07:33	32.78	0:48:00	32.57
Qwen2.5-7B	Prompting	1:45:21	39.32	1:58:06	38.27	1:08:11	38.65
	CAA	1:45:30	39.18	2:02:20	39.18	1:10:17	38.67
	SCS	1:53:38	35.97	2:09:20	36.21	1:15:04	35.30
Gemma2-9B	Prompting	2:07:46	25.32	1:35:12	25.65	1:31:16	13.86
	CAA	2:10:34	24.69	1:34:45	25.60	1:28:31	14.32
	SCS	2:20:53	22.92	1:46:52	22.86	1:41:28	12.45

Table 15: **Inference cost across datasets.** For each backbone and method we report wall-clock time and generated-token throughput (tok/s) on GSM8K, MATH500, and StrategyQA.

put as the primary metric because wall-clock time also varies with output length. Prompting-only, CAA, and SCS use the same intervention-aware generation loop. Across the nine evaluated settings, SCS reduces throughput by approximately 8% relative to prompting-only. Its predictor is trained offline once for each backbone–dataset setting, requiring 28–76 seconds in our experiments, while the backbone parameters remain unchanged.

J Qualitative Case Studies

We provide qualitative case studies on GSM8K and MATH500 to illustrate how SCS affects generated reasoning traces. Tables 17–22 report output comparisons across backbone–dataset settings.

Problem	Prompting-only output	SCS output
In triangle ABC , $AB = 17$, $AC = 8$, and $BC = 15$. Let D be the foot of the altitude from C to AB . Find the area of triangle ACD . Gold: 3840/289	The altitude h is the same as the length of CD . Since D is the foot of the altitude from C to AB , triangle ACD is a right triangle with CD as the height and AD as the base. $\dots AD^2 + (120/17)^2 = 8^2$ $AD^2 = 64 - 14400/289 = 4096/289$, so $AD = 64/17$. Area of triangle $ACD = \frac{1}{2} * AD * CD = \frac{1}{2} * 64/17 * 120/17 = 3840/289$. The final answer is 3840/289. (correct)	Since $AB^2 = AC^2 + BC^2$, triangle ABC is a right triangle with the right angle at C . The area of triangle ABC is $\frac{1}{2} * AC * BC = \frac{1}{2} * 8 * 15 = 60$. Since D is the foot of the altitude from C to AB , the area of triangle ACD is half the area of triangle ABC , because D divides AB into two segments and the altitude from C is shared. Thus the area of triangle $ACD = \frac{1}{2} * 60 = 30$. The final answer is 30. (incorrect)

Table 16: SCS follows a fluent but geometrically invalid area-bisection shortcut, while prompting-only reaches the correct answer. This case illustrates that SCS can follow the visited reasoning trajectory without detecting or revising an incorrect reasoning branch.

Question	Prompting-only output	SCS output
Luke is spending time at the beach building sandcastles. He eventually notices that each level of a sandcastle will have half the square footage as the level below it. If he makes a 4 leveled sandcastle where the top level has a square footage of 16, what is the average square footage of a level?	Step 1: Determine the square footage of each level. - The top level has a square footage of 16. - The second level has half the square footage of the top level, so it's $16/2 = 8$. - The third level has half the square footage of the second level, so it's $8/2 = 4$. - The fourth level has half the square footage of the third level, so it's $4/2 = 2$. Step 2: Calculate the total square footage of the sandcastle. - Total square footage = $16 + 8 + 4 + 2 = 30$. Step 3: Calculate the average square footage of a level. - Average square footage = Total square footage / Number of levels = $30/4 = 7.5$. The final answer is 7.5.	Step 1: Determine the square footage of each level. - The top level has a square footage of 16. - The second level has twice the square footage of the top level, so $16 * 2 = 32$. - The third level has twice the square footage of the second level, so $32 * 2 = 64$. - The fourth level has twice the square footage of the third level, so $64 * 2 = 128$. Step 2: Calculate the total square footage. - Total square footage = $16 + 32 + 64 + 128 = 240$. Step 3: Calculate the average square footage. - Average square footage = Total square footage / Number of levels. - Average square footage = $240/4 = 60$. The final answer is 60.

Table 17: Qualitative output comparison on GSM8K / Qwen2.5-7B.

Question	Prompting-only output	SCS output
Mrs. Tatiana owns a grocery store that sells different fruits and vegetables, which includes carrots. The price of carrots in the grocery store increases by 5% of the original price every year. What would be the price of carrots after three years if it was \$120 initially? (Round to the nearest integer)	A: The initial price of carrots is \$120. The price increases by 5% each year. To find the new price after 1 year, multiply the initial price by 1.05: $120 * 1.05 = 126$ The new price is \$126. To find the new price after 2 years, multiply the new price by 1.05 again: $126 * 1.05 = 131.7$ The new price is \$131.7 (round to the nearest integer: \$132). To find the new price after 3 years, multiply the new price by 1.05 again: $132 * 1.05 = 138.6$ The new price is \$138.6 (round to the nearest integer: \$139). The final answer is 139.	A: The initial price of carrots is \$120. The price increases by 5% each year. To find the new price after 1 year, multiply the initial price by 1.05 (100% + 5%): $\$120 * 1.05 = \$126$ The new price is \$126. To find the new price after 2 years, multiply the new price by 1.05 again: $\$126 * 1.05 = \$131.7$ The new price is \$131.7. To find the new price after 3 years, multiply the new price by 1.05 again: $\$131.7 * 1.05 = \$137.585$ Rounded to the nearest integer, the final answer is: The final answer is 138.

Table 18: Qualitative output comparison on GSM8K / Llama3-8B.

Question	Prompting-only output	SCS output
Melanie is a door-to-door saleswoman. She sold a third of her vacuum cleaners at the green house, 2 more to the red house, and half of what was left at the orange house. If Melanie has 5 vacuum cleaners left, how many did she start with?	Here's how to solve this problem step-by-step: **1. Orange House:** Melanie had 5 vacuum cleaners left after selling half at the orange house, so she had $5 * 2 = 10$ vacuum cleaners before that. **2. Red House:** Before selling to the red house, she had $10 + 2 = 12$ vacuum cleaners. **3. Green House:** Before selling at the green house, she had $12 * 3 = 36$ vacuum cleaners (since she sold a third at the green house). **The final answer is 36.**	Here's how to solve this problem step-by-step: **1. Work backward:** Since Melanie has 5 left, and half of what was left at the orange house was sold, she had $5 * 2 = 10$ vacuum cleaners before the orange house. **2. Add back the red house sales:** Before selling to the red house, she had $10 + 2 = 12$ vacuum cleaners. **3. Calculate the initial amount:** She sold a third of her vacuum cleaners at the green house, meaning the 12 she had represents two-thirds of her original amount. Let x be the original number of vacuum cleaners. So, $(2/3) * x = 12$ **4. Solve for x:** Multiply both sides of the equation by $(3/2)$ to isolate x : $x = 12 * (3/2) = 18$ The final answer is 18.

Table 19: Qualitative output comparison on GSM8K / Gemma2-9B.

Problem	Prompting-only output	SCS output
Let $x^8 + 3x^4 - 4 = p_1(x)p_2(x) \cdots p_k(x),$ where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.	To solve the problem, we start by factoring the polynomial $x^8 + 3x^4 - 4$. We can make a substitution to simplify the polynomial. Let $y = x^4$. Then the polynomial becomes: $y^2 + 3y - 4.$ We can factor this quadratic polynomial as follows: $y^2 + 3y - 4 = (y + 4)(y - 1).$ Substituting back $y = x^4$, we get: $x^8 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1).$ Next, we factor $x^4 - 1$ further. We recognize that it is a difference of squares: $x^4 - 1 = (x^2 + 1)(x^2 - 1).$ The term $x^2 - 1$ is also a difference of squares and can be factored as: $x^2 - 1 = (x + 1)(x - 1).$ So, the complete factorization of the original polynomial is: $x^8 + 3x^4 - 4 = (x^4 + 4)(x^2 + 1)(x + 1)(x - 1).$ We need to check if any of these factors can be factored further over the integers. The polynomial $x^4 + 4$ cannot be factored further over the integers because it does not have any real roots (the discriminant of the corresponding quadratic in x^2 is negative). The polynomials $x^2 + 1$, $x + 1$, and $x - 1$ are already in their simplest forms. Thus, the factorization of $x^8 + 3x^4 - 4$ into monic polynomials with integer coefficients is: $x^8 + 3x^4 - 4 = (x^4 + 4)(x^2 + 1)(x + 1)(x - 1).$ We are asked to find the sum of the values of these polynomials evaluated at $x = 1$: $p_1(1) + p_2(1) + p_3(1) + p_4(1) = (1^4 + 4) + (1^2 + 1) + (1 + 1) + (1 - 1) = 5 + 2 + 2 + 0 = 9.$ Therefore, the answer is: 9	To solve the problem, we start by factoring the polynomial $x^8 + 3x^4 - 4$. We can make a substitution to simplify the polynomial. Let $y = x^4$. Then the polynomial becomes: $y^2 + 3y - 4.$ We can factor this quadratic polynomial as follows: $y^2 + 3y - 4 = (y + 4)(y - 1).$ Substituting back $y = x^4$, we get: $x^8 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1).$ Next, we factor each of these terms further. The term $x^4 - 1$ is a difference of squares: $x^4 - 1 = (x^2 + 1)(x^2 - 1).$ The term $x^2 - 1$ is also a difference of squares: $x^2 - 1 = (x + 1)(x - 1).$ So, we have: $x^4 - 1 = (x^2 + 1)(x + 1)(x - 1).$ The term $x^4 + 4$ can be factored using the Sophie Germain identity: $x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2).$ Thus, the complete factorization of the original polynomial is: $x^8 + 3x^4 - 4 = (x^2 + 2x + 2)(x^2 - 2x + 2)(x^2 + 1)(x + 1)(x - 1).$ The polynomials $p_1(x)$, $p_2(x)$, $\dots$, $p_k(x)$ are $x^2 + 2x + 2$, $x^2 - 2x + 2$, $x^2 + 1$, $x + 1$, and $x - 1$. We need to find the sum of the values of these polynomials evaluated at $x = 1$: $p_1(1) = 1^2 + 2 \cdot 1 + 2 = 1 + 2 + 2 = 5,$ $p_2(1) = 1^2 - 2 \cdot 1 + 2 = 1 - 2 + 2 = 1,$ $p_3(1) = 1^2 + 1 = 1 + 1 = 2,$ $p_4(1) = 1 + 1 = 2,$ $p_5(1) = 1 - 1 = 0.$ Adding these values together, we get: $p_1(1) + p_2(1) + p_3(1) + p_4(1) + p_5(1) = 5 + 1 + 2 + 2 + 0 = 10.$ Therefore, the answer is: 10

Table 20: Qualitative output comparison on MATH500 / Qwen2.5-7B.

Problem	Prompting-only output	SCS output
For some real numbers a and b, the equation $8x^3 + 4ax^2 + 2bx + a = 0$ has three distinct positive roots. If the sum of the base-2 logarithms of the roots is 5, what is the value of a?	A: Let the roots be r_1, r_2, and r_3. Then, we have that $r_1 + r_2 + r_3 = -\frac{4a}{8} = -\frac{a}{2}$ and $r_1 r_2 r_3 = -\frac{a}{8}$. Since the sum of the base-2 logarithms of the roots is 5, we have that $\log_2 r_1 + \log_2 r_2 + \log_2 r_3 = 5$. Using the property of logarithms, we have that $\log_2(r_1 r_2 r_3) = 5$, so $\log_2(-\frac{a}{8}) = 5$. This implies that $a = -2^5 = -32$. Final Answer: -32	A: Let the three roots be r_1, r_2, and r_3. Then, we have that $r_1 + r_2 + r_3 = -\frac{4a}{8} = -\frac{a}{2}$, and $r_1 r_2 r_3 = -\frac{a}{8}$. Since the sum of the base-2 logarithms of the roots is 5, we have that $\log_2 r_1 + \log_2 r_2 + \log_2 r_3 = 5$. Using the property of logarithms that $\log_2 r_1 + \log_2 r_2 + \log_2 r_3 = \log_2(r_1 r_2 r_3)$, we have that $\log_2(-\frac{a}{8}) = 5$. Taking 2^5 to both sides, we have that $-\frac{a}{8} = 32$, so $a = -256$. Final Answer: -256

Table 21: Qualitative output comparison on MATH500 / Llama3-8B.

Problem	Prompting-only output	SCS output
What is the smallest positive integer n such that all the roots of $z^4 + z^2 + 1 = 0$ are n^{th} roots of unity?	Here's how to solve this problem: **1. Recognize the Pattern** The equation $z^4 + z^2 + 1 = 0$ looks like a quadratic if we make a substitution. Let $w = z^2$. Substituting, we get: $w^2 + w + 1 = 0$ **2. Solve the Quadratic** This quadratic doesn't factor easily, so we use the quadratic formula: $w = \frac{-1 \pm \sqrt{1^2 - 4 \cdot 1 \cdot 1}}{2 \cdot 1} = \frac{-1 \pm \sqrt{-3}}{2} = \frac{-1 \pm i\sqrt{3}}{2}$ **3. Find the Roots of the Original Equation** Since $w = z^2$, the roots of the original equation are: $\begin{aligned} * z &= \sqrt{\frac{-1+i\sqrt{3}}{2}} \\ * z &= \sqrt{\frac{-1-i\sqrt{3}}{2}} \\ * z &= -\sqrt{\frac{-1+i\sqrt{3}}{2}} \\ * z &= -\sqrt{\frac{-1-i\sqrt{3}}{2}} \end{aligned}$ **4. Determine the Order of Roots of Unity** These roots are complex numbers. Notice that they are all complex conjugates of each other. This means they lie on the unit circle in the complex plane. Furthermore, they are evenly spaced around the circle. To find the order of these roots of unity, we need to find the smallest integer n such that raising each root to the nth power gives us 1. Since there are four roots evenly spaced around the unit circle, the smallest n that works is **4**. Final Answer: $n = \boxed{4}$	Here's how to solve this problem: **1. Recognize the Pattern** The equation $z^4 + z^2 + 1 = 0$ looks like a quadratic if we let $y = z^2$. Substituting, we get: $y^2 + y + 1 = 0$ **2. Solve the Quadratic** This quadratic doesn't factor easily, so we use the quadratic formula: $y = \frac{-1 \pm \sqrt{1^2 - 4 \cdot 1 \cdot 1}}{2 \cdot 1} = \frac{-1 \pm \sqrt{-3}}{2} = \frac{-1 \pm i\sqrt{3}}{2}$ **3. Find the Roots of the Original Equation** Since $y = z^2$, the roots of the original equation are: $\begin{aligned} * z &= \sqrt{\frac{-1+i\sqrt{3}}{2}} \\ * z &= \sqrt{\frac{-1-i\sqrt{3}}{2}} \\ * z &= -\sqrt{\frac{-1+i\sqrt{3}}{2}} \\ * z &= -\sqrt{\frac{-1-i\sqrt{3}}{2}} \end{aligned}$ **4. Determine the Order of Roots of Unity** These roots are complex numbers, and they are all sixth roots of unity. You can verify this by raising each root to the power of 6 and seeing if you get 1. **5. The Answer** The smallest positive integer n for which all the roots are nth roots of unity is $\boxed{6}$.

Table 22: Qualitative output comparison on MATH500 / Gemma2-9B.