Separable Representations of Task Complexity and Deliberation in Reasoning Language Models.

Quang X. Nguyen [1] **Hieu M. Vu** [1] **Dung V. Nguyen** [2] **Hai Tuan Luu** [1] **Linh Duy Tran** [3] **Tan M. Nguyen** [2]

Abstract

Reasoning-capable large language models (LLMs) produce explicit chain-of-thought (CoT) traces that scale with problem difficulty. While essential for complex reasoning, this behavior introduces unnecessary latency on simpler tasks. Traditional methods for shortening CoT, such as "think fast" prompting, often degrade accuracy on difficult problems. This paper investigates whether a targeted intervention is possible by examining how models internally represent task complexity and allocate reasoning length before generation begins. We demonstrate that task complexity is encoded with non-linear ordinal structure in pre-reasoning residual-stream activations. PCA visualizations reveal a U-shaped ordinal structure consistent with a continuous complexity gradient, which a non-linear probe can classify above chance. We then extract a "deliberation-associated" direction via contrastive prompting and apply a nullspace projection against confound directions to remove components like instruction compliance and style framing. Using a geometric alignment test, we find no evidence of geometric entanglement between the purified direction and the non-linear structure of complexity assessment. Finally, we test this separation through activation steering: negative steering along the purified direction reduces reasoning tokens by up to 22% on MATH-500 and GPQA-Diamond, with smaller effects on code generation and accuracy preserved within 1% on most settings. Our results indicate that deliberation extent and task complexity occupy separable representational subspaces, enabling targeted modulation of LLM deliberation.

[1]Independent Researcher [2]Department of Mathematics, National University of Singapore [3]AI Imaging Team, Data Solution Department, FPT Software Japan. Correspondence to: Quang X. Nguyen <ngxquang.work@gmail.com>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

1. Introduction

Recent reasoning-capable large language models (LLMs), such as DeepSeek-R1 (Guo et al., 2025) and Qwen3 (Yang et al., 2025), are trained to produce explicit chain-of-thought traces before emitting a final answer. This test-time computation is beneficial when problems require multi-step deduction, but it also means that models routinely generate long reasoning traces on problems that admit short solutions (Snell et al., 2024). The additional tokens increase inference latency and cost without necessarily improving accuracy. As reasoning models are deployed at scale, the question of whether this extended computation can be selectively reduced becomes practically relevant.

Several approaches have been proposed to reduce reasoning length. At the prompt level, budget-forcing instructions (Xu et al., 2025) discourage the model from generating long traces, while explicit "think fast" or no-thinking directives suppress the reasoning block entirely. At the training level, reinforcement learning with length penalties (Aggarwal & Welleck, 2025) and length-regularized reward shaping (Meng et al., 2024) encourage shorter outputs without accuracy loss for the training distribution. These approaches share a common limitation: reduction is applied uniformly, independent of whether the specific problem actually requires extended deliberation. We show in Section 2.2 that this matters: on harder problems (MATH-500 difficulty Level 4-5), forcing the model to skip deliberation reduces accuracy by roughly 20 percentage points relative to standard generation, consistent with prior observations on reasoning length and accuracy trade-offs (Snell et al., 2024; Li et al., 2025). The challenge is therefore not simply shortening reasoning, but doing so selectively, reducing effort on problems where it is unnecessary while preserving it on problems where it is required. Achieving this requires that the model have an internal signal about problem difficulty that can be decoupled from the mechanism governing reasoning effort.

We investigate both prerequisites. First, we find that pre-reasoning residual-stream activations encode task difficulty as a non-linear ordinal manifold - visible as a U-shaped PCA projection consistent with the Guttman effect (Guttman, 1950) and decodable by a non-linear probe. Second, we extract a candidate deliberation direction via

contrastive prompting, apply a nullspace projection to remove measurable confounds, and use a local tangent space test (Zhang & Zha, 2004) to verify that the purified direction is geometrically independent of the difficulty manifold. These findings suggest complexity perception and deliberation control are separable, in principle permitting steering one without perturbing the other.

We make the following contributions:

1. We find that pre-reasoning activations represent task complexity as a non-linear, ordinal structure, decodable via a non-linear probe (Section 3).

2. We extract a deliberation-associated direction via difference-in-means contrastive prompting and introduce a nullspace projection to remove measurable confound components (Section 4.1-4.2).

3. We apply Local Tangent Space Alignment to test if the purified direction is geometrically separable from the difficulty manifold (Section 4.3).

4. We evaluate the purified direction as a steering target and find that negative angular steering reduces thinking tokens by up to 22% on MATH-500 and GPQA-Diamond, with smaller effects on code generation and accuracy preserved within 1% on most settings, suggesting that the extracted direction modulates model's deliberation (Section 5).

2. Empirical Analysis: Behavioral Entanglement and Dynamic Allocation

2.1. Models and Data Collection

Our empirical analysis is conducted using the DeepSeek-R1-Distill-Llama-8B model (32 layers) on the MATH-500 dataset (Lightman et al., 2023), a subset of the MATH benchmark (Hendrycks et al., 2021) consisting of 500 problems. Generation hyperparameters are set to a temperature of 0.5 and a top-p of 0.9 to encourage natural yet consistent reasoning trajectories. Problems of MATH-500 are already categorized into five difficulty levels, representing a range of complexity in competition-level mathematics, from basic applications to abstract theorem-based reasoning. This categorization allows for an analysis of how problem difficulty relates to test-time compute.

2.2. Behavioral Entanglement: Compute Scales with Complexity

To systematically investigate the influence of problem complexity on the model's inference dynamics, we dissect the taxonomy of generated thought traces across varying difficulty levels. We classify the reasoning sentences into eight functional categories following Bogdan et al. (2025) and Venhoff et al. (2025b).

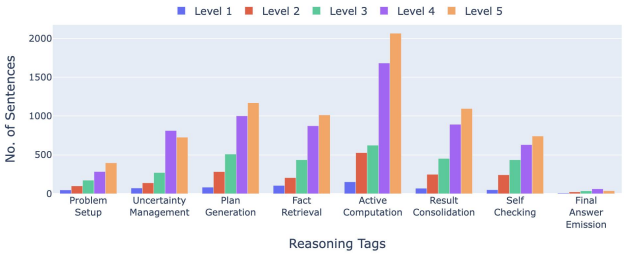


Figure 1. **Scaling of sentence counts across reasoning function tags for MATH-500 difficulty levels.** The distribution demonstrates a consistent positive correlation between problem difficulty and sentence volume across every reasoning function, indicating harder problems necessitate a higher density of reasoning steps.

As depicted in Figure 1, we observe a consistent monotonic increase in the frequency of all reasoning functions as problem complexity scales from Level 1 to Level 5. This upward trajectory is particularly pronounced in *Active Computation*, *Plan Generation*, and *Uncertainty Management*, indicating a deliberate reallocation of the token budget toward higher-order cognitive tasks. This expansion spans the entire taxonomy rather than being isolated to specific heuristic categories. Such uniform scaling suggests that heightened input complexity is associated with broad-based increases in reasoning length. This provides an evidence that the model's representation of complexity being correlated with its downstream computational budget, associated with longer reasoning trajectories prior to answer emission.

2.3. The Necessity of Dynamic Allocation

To ascertain whether this extended reasoning is computationally necessary or merely an artifact of length bias, we evaluate model accuracy under three distinct intervention strategies: **(1) Default:** We let the model perform its reasoning without any intervention, **(2) Think Fast:** A prompt-level intervention appending the instruction *"Think fast and briefly,"* explicitly constraining the reasoning trajectory, and **(3) Skip Thinking:** Following Dang et al. (2025b), the model is forced using its bias to answer by overriding its reasoning with sentences like *"I will answer him without thinking more"*.

Figure 2 illustrates a performance crossover across difficulty levels. For low-complexity tasks (Levels 1-2), the *Skip Thinking* strategy remains competitive, achieving accuracies of 93.0% and 87.8%, respectively. This suggests that for simpler problems, internal model biases may be sufficient to produce accurate results without extended reasoning trajectories. In contrast, this heuristic appears less effective as complexity increases. At Level 5, *Skip Thinking* accuracy reaches 41.0%, representing a 20.9% deficit relative to the *Default* baseline (61.9%). This divergence suggests that while latent biases provide efficient shortcuts for simpler

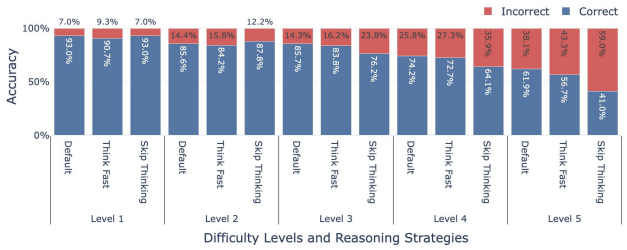


Figure 2. **Performance comparison of reasoning strategies across MATH-500 difficulty levels.** Results highlight the degradation of "Skip Thinking" as task complexity increases from Level 1 to Level 5.

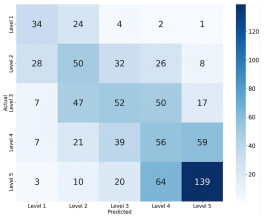


Figure 3. **Confusion matrix of the MLP Probe.**

tasks, higher-complexity problems may require the model to suppress these priors in favor of more deliberate computation. Thus, task complexity may correlate with the allocation of test-time compute.

3. The Geometry of Task Complexity

3.1. Activations Extraction

To investigate the model's intrinsic perception of problem difficulty prior to explicit reasoning, we focus exclusively on the input encoding phase. Let $P = \{p_i\}_{i=1}^{N}$ denote a corpus of N reasoning prompts sampled from the MATH-500 or LiveCodeBench (v1) datasets. For a given prompt p_i, we perform a forward pass and extract the residual stream hidden state at the final prompt token, denoted as $h_i^{(l)} \in \mathbb{R}^D$, where $l \in \{1, \ldots, L\}$ indexes the transformer layer and d is the model's hidden dimension.

By isolating the activations at the first assistant token, we capture the contextual representation of the entire question. We systematically construct a layer-specific dataset of activation-label pairs:

$$D^{(l)} = \{(h_i^{(l)}, c_i)\}_{i=1}^{N}$$

where $c_i \in C$ represents the ground-truth complexity label (e.g. Easy, Medium, Hard) associated with prompt p_i.

3.2. The U-Shaped Complexity Manifold via Principal Component Analysis

While behavioral metrics indicate that reasoning models allocate more compute to harder problems, it remains unclear how complexity is geometrically organized in the latent space. We apply Principal Component Analysis (PCA) to $D^{(l)}$ to visualize the activation structure without imposing a predefined geometric assumption.

Figure 4 projects the activations $h_i^{(l)}$ onto PC_0 and PC_3 for two models on LiveCodeBench (v1). The projection exhibits two properties relevant to subsequent analysis. First, activations corresponding to questions of similar complexity occupy contiguous regions of the projected space, with cluster centroids following a monotonic *Easy* $\rightarrow$ *Medium* $\rightarrow$ *Hard* progression. This indicates that difficulty is encoded as a continuous ordinal gradient rather than as disjoint categorical buckets. Second, the centroids trace a U-shaped curve consistent with the Guttman effect (Guttman, 1950), a signature commonly observed when an ordinal latent variable is projected to two dimensions. While the U-shape itself is a projection artifact and does not establish non-linearity in the underlying representation, it suggests difficulty is unlikely to be captured by a single dominant direction. We test this directly in Section 3.3 with a non-linear probe.

3.3. Non-Linear Probing

The U-shaped pattern in PCA suggested that difficulty assessment is unlikely to be captured by a single linear subspace. Consequently, a standard linear probe (e.g., linear regression or a single-layer classifier) is insufficient to establish an accurate decision boundary across the curvature of this space. To quantitatively validate that the model encodes complexity, we train a non-linear probe, specifically a Multi-Layer Perceptron (MLP), to map the high-dimensional residual stream activations to the categorical difficulty labels. Formally, we define the non-linear probe as a neural network $f_\theta : \mathbb{R}^D \rightarrow \Delta^{C-1}$, where D is the model's hidden dimension and C represents the discrete difficulty levels.

We train the non-linear probe on the MATH training set (Hendrycks et al., 2021). We report the confusion matrix of the probe's predictions for the DeepSeek-R1-Distill-Llama-8B model in Figure 3. The diagonal structure of this matrix indicates that task complexity is decodable in the latent space, prior to the first token generation.

4. Extracting and Purifying the Deliberation Direction

Section 3 showed that pre-reasoning activations encode task complexity. Beyond perceiving complexity, the model also allocates test-time compute, search depth, verification, verbosity, which we refer to as *deliberation*. As shown in Section 2, complexity and deliberation are strongly correlated, raising the question of whether they share a representational direction (Elhage et al., 2022) or are controlled separately. We extract a deliberation-associated direction via contrastive

(a) DeepSeek-R1-Distill-Llama-8B *(b)* Qwen3-14B

Figure 4. **PCA visualizations of difficulty-bucketed activations.** DeepSeek-R1-Distill-Llama-8B and Qwen3-14B on LiveCodeBench (v1) dataset.

prompting (Section 4.1), remove measurable confounds via nullspace projection (Section 4.2), and assess its geometric relationship to the complexity manifold (Section 4.3).

4.1. Contrastive Extraction of the Raw Direction

We extract a candidate deliberation-associated direction using the difference-in-means method (Zou et al., 2023; Turner et al., 2023; Rimsky et al., 2024; Vu & Nguyen, 2025),a standard approach for isolating behaviorally relevant directions in activation space.

Contrastive prompt construction. For each base question q_i drawn from a diverse mixture of mathematical and logical domains (MATH-500 (Lightman et al., 2023) and LiveCodeBench v1 (Jain et al., 2024)), we construct a pair of contrastive prompts by appending instructional suffixes that modulate reasoning effort:

- **Positive Prompt** (p_i^+): $q_i \oplus$*"Try different approaches and verify every step carefully."*
- **Negative Prompt** (p_i^-): $q_i \oplus$ *"Avoid overthinking, you should answer immediately when you derive the first answer."*

This suffix pair was selected for its semantic clarity: the positive suffix directly instructs expanded search and verification, while the negative suffix instructs premature convergence. We verified that p_i^+ increases generated token counts relative to the unmodified prompt, while p_i^- induces shorter reasoning traces.

Difference-in-means computation. We perform a forward pass for both prompt variants and extract the residual stream activations at the Assistant Start token across all transformer layers $l \in \{1, \ldots, L\}$. The candidate deliberation-associated direction at layer l is computed as the mean activation difference across N paired samples:

$$v_{\text{delib}}^{(l)} = \frac{1}{N} \sum_{i=1}^{N} \left(h^{(l)}(p_i^+) - h^{(l)}(p_i^-) \right)$$

Each $v_{\text{delib}}^{(l)} \in \mathbb{R}^d$ represents the average shift in activation space induced by the deliberation-modulating suffix at layer l. Layer selection is performed after purification and is described at the end of Section 4.2.

4.2. Concept Purification

The contrastive suffixes used to extract v_{delib} differ along axes beyond reasoning effort (instruction compliance, confidence, surface style), so v_{delib} likely mixes the deliberation signal with confounds. We introduce a nullspace projection procedure that removes measurable confound components, yielding a purified direction v_{pure}.

Confound construction. We apply the same difference-in-means procedure with $n = 7$ suffix pairs designed to modulate non-deliberation features, spanning surface-level stylistic variation (tone, formatting, vocabulary register; $n_s = 5$) and cognitive-level variation (instruction following, difficulty priming; $n_c = 2$). Full suffix pairs are listed in Appendix A.1. Each confound suffix was selected to have minimal measured effect on reasoning length, so that confound directions capture variation in style or framing rather than in compute allocation.

Projection We stack the n confound vectors as columns of $M_{\text{noise}} \in \mathbb{R}^{d \times n}$, compute its thin SVD $M_{\text{noise}} = U \Sigma V^\top$, and retain all components with $\sigma_i / \sigma_1 > 10^{-6}$ to form the confound column space $U_k \in \mathbb{R}^{d \times k}$. The purified vector is the orthogonal complement projection:

$$v_{\text{pure}} = \left(I - U_k U_k^\top \right) v_{\text{delib}}$$

Since $n \ll d$, the confound subspace is low-dimensional and the projection is computationally trivial.

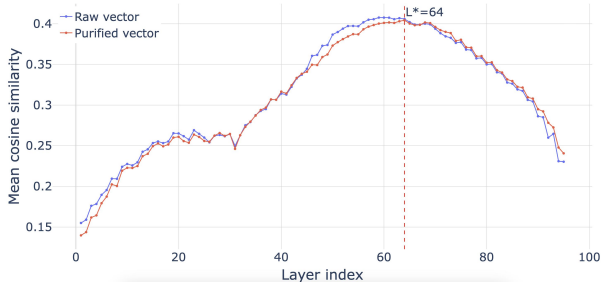


Figure 5. **Mean Pairwise Cosine Similarity of every layer against the others.** The dash line illustrates layer with the highest value.

Layer Selection. Purification is applied independently at each layer. We select layer l^* with the highest mean pairwise cosine similarity among purified vectors (Vu & Nguyen, 2025).

$$l^* = \underset{l}{\operatorname{argmax}} \frac{1}{L-1} \sum_{j \neq l} \cos\left(v_{\text{pure}}^{(l)}, v_{\text{pure}}^{(j)}\right)$$

The selected purified vector $v_{\text{pure}}^{(l^*)}$ is normalized to unit length and used as the deliberation-associated direction in all subsequent analyses.

Results Figure 5 shows the layer-wise stability analysis. The mean pairwise cosine similarity curve retains its overall shape after purification, confirming that the nullspace projection preserves cross-layer consensus structure, suggesting confound removal does not destabilize the deliberation signal.

4.3. Geometric Validation: Disentanglement from the Difficulty Manifold

To assess whether steering along the deliberation-associated direction is likely to perturb the model's difficulty encoding, we measure its geometric alignment with the local tangent space of the complexity manifold M_{diff} established in Section 3. A direction that is entangled with M_{diff} would, when used for steering, shift the model's internal assessment of task complexity, which is an undesirable side effect. We apply Local Tangent Space Alignment (LTSA) (Zhang & Zha, 2004) with an in-subspace null hypothesis suited to activation-derived vectors and test both the raw direction v_{delib} and the purified direction v_{pure}.

4.3.1. Local Tangent Space Alignment

Because M_{diff} is globally non-linear (Section 3.2), standard linear metrics such as cosine similarity with a single manifold axis are inappropriate for this geometry. Instead, we evaluate alignment locally at each point on the manifold.

For each activation $x_j \in M_{\text{diff}}$, we estimate the local tangent subspace T_{x_j} as follows:

1. **Neighborhood selection.** Identify the k-nearest neighbors $\mathcal{N}_j$ of x_j within the manifold activations using Euclidean distance.
2. **Mean centering.** Form the local patch matrix by centering the neighborhood: $\tilde{X}_j = X_{\mathcal{N}_j \cup \{j\}} - \mu_j$, where μ_j is the patch centroid.
3. **Local SVD.** Compute the thin singular value decomposition $\tilde{X}_j = U \Sigma V^\top$, yielding right singular vectors $v_1^{(j)}, \ldots, v_{k+1}^{(j)} \in \mathbb{R}^D$ ordered by descending singular value.
4. **Intrinsic dimension estimation.** Determine the local intrinsic dimension d_j as the minimum number of singular components capturing a fraction τ (variance threshold) of the total squared singular value energy:

$$d_j = \min\left\{ m : \frac{\sum_{i=1}^{m} \sigma_i^2}{\sum_{i=1}^{k+1} \sigma_i^2} \geq \tau \right\}$$

The entanglement score for a test vector $\hat{v}$ at point x_j is the squared projection onto the local tangent subspace:

$$S_j(\hat{v}) = \sum_{i=1}^{d_j} \left(\hat{v} \cdot v_i^{(j)} \right)^2$$

This quantity lies in $[0,1]$ for unit vectors: a value near 1 indicates the test vector lies within the local tangent space (entangled), while a value near 0 indicates orthogonality (disentangled). The aggregate entanglement score is the mean across all manifold points:

$$\bar{S}(\hat{v}) = \frac{1}{N} \sum_{j=1}^{N} S_j(\hat{v})$$

In-subspace Null Hypothesis The standard analytic baseline ($\bar{d}/D$, where $\bar{d}$ is the mean local intrinsic dimension) assumes test vectors are drawn from the full ambient space $\mathbb{R}^D$. This is invalid for activation-derived vectors, which live within the model's effective subspace ε_L of rank $r \ll D$, this baseline systematically underestimates the random-alignment expectation, producing false positives. We instead sample $M = 500$ null vectors uniformly from the unit sphere within ε_L and compute their LTSA scores. The p-value is the fraction of null scores exceeding the observed score.

4.3.2. Result

We test across two models (Qwen3-14B (Yang et al., 2025) and DeepSeek-R1-Distill-Qwen-14B (Guo et al., 2025)), two datasets (MATH-500 (Lightman et al., 2023) and LiveCodeBench v1 (Jain et al., 2024)), and both vector types,

Model	Dataset	Observed $\bar{S}$	Ratio vs Null	p-value	Entangled?
Qwen3-14B	LiveCodeBench	0.0429	1.07×	0.272	✗
	MATH-500	0.0026	0.09×	1.000	✗
R1-Distill-Qwen-14B	LiveCodeBench	0.0095	0.28×	1.000	✗
	MATH-500	0.0087	0.32×	1.000	✗

Table 1. **LTSA disentanglement results (Purified) at 80% variance threshold.** Ratio vs Null denotes the observed score divided by the null distribution mean. p-values from one-sided permutation test ($M = 500$). All results shown are for purified vectors.

totaling 8 conditions at 80% variance threshold with $k = 20$. Table 1 presents the results.

Disentanglement results. All settings fail to reject the null of no entanglement ($p > 0.05$), with observed scores falling well below the null mean (ratios 0.09×-1.07×), indicating that complexity assessment and deliberation control occupy separable subspaces.

Implication for steering. These results suggest that modulating reasoning effort via v_{pure} is unlikely to substantially shift the model's position on the difficulty manifold. In other words, the intervention modulates how much the model deliberates without measurably altering its complexity representation.

5. Causal Validation via Activation Steering

5.1. Activation Steering

Section 4.3 found no evidence of entanglement between v_{pure} and the difficulty manifold. Because of the separability, intervening along the deliberation is expected to modulate reasoning effort with limited effect on the complexity representation. This separation suggests we can modulate verification behavior and output length while preserving task comprehension.

Steering Method: To achieve control over the reasoning trajectory, we employ the Angular Steering algorithm (Vu & Nguyen, 2025). Rather than applying a static additive scalar, we rotate the activations by a target angle $\theta \in [0, 360]$ strictly toward or away from the Deliberation-associated vector. This continuous angular modulation allows us to systematically amplify (positive steering) or weaken (negative steering) the model's deliberation extent.

Models and Datasets We evaluated two widely used reasoning model families: DeepSeek-R1 Distill (Guo et al., 2025) and Qwen3 (Yang et al., 2025), at scales from 8B to 32B parameters, with generation fixed at temperature of 0.0, top-p of 0.9, and a 16,000-token budget. We evaluate on three benchmarks covering complementary reasoning demands: MATH-500 (Lightman et al., 2023)(mathematical problem solving), LiveCodeBench v4-5 (Jain et al., 2024) (code generation), and GPQA-Diamond (Rein et al., 2023)(expert-level scientific knowledge).

Evaluation Metrics: We rely on two primary metrics: **(1) Thinking length** (token count within the *<think>...</think>* block) as a proxy for allocated compute, and **(2) Task Performance** ($Pass@1$, exact-match accuracy on the final answer) to verify that steering does not degrade task comprehension. Effective steering reduces Thinking Length with limited $Pass@1$ cost. For MATH-500 and GPQA-Diamond, we evaluate $Pass@1$ via an LLM-as-a-judge protocol (Appendix C), while LiveCodeBench is evaluated via execution-based unit tests.

5.2. Performance of Deliberation Steering in Thinking Length Control

Negative steering reduces thinking tokens with minimal accuracy cost. Table 2 presents the steering results against the no-steer baseline on four models and three datasets. Across all four available models, negative steering along v_{pure} reduces the number of tokens generated within the *<think>...</think>* block, with little change in downstream accuracy. On MATH-500, token reductions range from 14% (R1-Qwen14B) to 22% (Qwen3-32B) while the corresponding accuracy changes are at most 0.2 percentage points in absolute terms, and for Qwen3-32B the accuracy increases slightly (+0.4%) alongside the 22% token reduction, suggesting that the baseline model may allocate more computation than necessary on this benchmark. On GPQA-Diamond, token reductions are 12-13% for Qwen3-14B and R1-Qwen14B, with accuracy shifts of 0.5-1.6 percentage points. Given that the GPQA-Diamond test set contains only 198 questions, differences of this magnitude correspond to 1-3 questions and should be interpreted cautiously. On LiveCodeBench, the effect on token count is more modest (2-4% for most models), and accuracy changes are mixed. Notably, Qwen3-32B shows a 7.1-point accuracy decline under negative steering on LiveCodeBench, an adverse outcome in this evaluation. We report this directly: negative steering transfers less cleanly to code generation for this model, and we do not have a principled explanation for why this benchmark is more sensitive. Positive steering increases token counts by 4-20% across conditions. Specifically, on GPQA-Diamond, small accuracy gains are observed for Qwen3-14B (+1.5%) and Qwen3-32B (+2.5%), though these differences are within the range of sampling noise at this dataset scale. On MATH-500 and LiveCodeBench,

Model	Method	MATH-500		LiveCodeBench (v4-5)		GPQA-Diamond	
		Pass@1 (↑)	#Tokens	Pass@1 (↑)	#Tokens	Pass@1 (↑)	#Tokens
R1-Llama8B	**Steer (-)**	**81.8** (-0.6)	**3440** (-18%)	**29.3** (+1.3)	**11504** (-3%)	**26.1** (+0.8)	**8710** (-9%)
	Baseline	82.4	4211	28.0	11848	25.3	9537
	Steer (+)	82.8 (+0.4)	4300 (+2%)	29.5 (+1.5)	12095 (+2%)	26.8 (+1.5)	9973 (+5%)
R1-Qwen14B	**Steer (-)**	**90.6** (+0.0)	**2886** (-14%)	**39.2** (+2.6)	**10544** (-2%)	**46.5** (+1.6)	**5753** (-13%)
	Baseline	90.6	3339	36.6	10769	44.9	6642
	Steer (+)	88.0 (-2.6)	3890 (+17%)	39.9 (+3.3)	11732 (+9%)	44.3 (-0.6)	7532 (+13%)
Qwen3-14B	**Steer (-)**	**90.8** (-0.2)	**3524** (-16%)	**53.3** (+1.1)	**10210** (-4%)	**58.6** (-0.5)	**5375** (-12%)
	Baseline	91	4205	52.2	10686	59.1	6080
	Steer (+)	91.8 (+0.8)	4375 (+4%)	54.9 (+2.7)	10885 (+2%)	60.6 (+1.5)	7314 (+20%)
Qwen3-32B	**Steer (-)**	**93.2** (+0.4)	**2878** (-22%)	**46.6** (-7.1)	**10616** (-3%)	**63.6** (-1.0)	**5397** (-2%)
	Baseline	92.8	3685	53.7	10892	64.6	5518
	Steer (+)	92.8 (+0.0)	3969 (+8%)	54.1 (+0.4)	11147 (+2%)	66.2 (+2.5)	6100 (+11%)

Table 2. **Steering results across models and benchmarks.** Rows with negative steering (**Steer -**) are highlighted and bolded. Parenthetical values denote changes in Pass@1 and Token count relative to unsteered baseline.

positive steering does not reliably improve accuracy.

Continuous Causal Control of Thinking Length. The steering effect is continuous and monotonic across angles, indicating that v_{pure} produces continuous, monotonic effects on reasoning length rather than a binary switch. Figure 6 shows thinking length as a function of steering angle for Qwen3-14B across all three evaluation datasets. The effect magnitude varies across domains: token reduction is strongest on MATH-500 (16.2%) and more moderate on GPQA-Diamond (11.6%) and LiveCodeBench (4.2%). We observe similar cross-dataset patterns for the other models (see Appendix B), with Qwen3-32B showing notably stronger reduction on MATH-500 (22%) while LiveCodeBench remains consistently resistant to steering across all models. This domain variation likely reflects differences in how each benchmark encodes reasoning effort in the prompt representation. We report it as an empirical observation and leave its mechanistic explanation to future work.

Steering is uniform across difficulty levels. Figure 7 breaks down the steering effect by problem difficulty on MATH-500 (Qwen3-32B). Negative steering reduces thinking length across all difficulty levels, specifically easy, intermediate, and hard, with the absolute token reduction proportional to baseline length. Problems at Level 4-5, which have longer baseline traces, exhibit larger absolute reductions under negative steering than Level 1-2 problems, while the proportional reduction remains roughly constant across levels. This uniform proportional behavior indicates that the intervention scales with the model's existing deliberation budget rather than targeting any particular difficulty stratum. Combined with the difficulty probe from Section 3, which classifies problem complexity before reasoning begins, this property suggests a natural extension: applying steering magnitudes conditioned on predicted complexity, so that easy problems receive stronger suppression while hard problems remain unmodified. We leave this complexity-gated design to future work.

6. Related Work

6.1. Representation Analysis and Concept Isolation

A growing body of work suggests that transformer language models encode high-level concepts as approximately linear directions in the residual stream (Elhage et al., 2022; Park et al., 2023) Representation Engineering (Zou et al., 2023) operationalizes this by extracting concept vectors from contrastive prompt pairs. Our extraction procedure follows this approach. Studies on reasoning models provide further evidence that internal states encode task-relevant signals before and during generation: pre-trained priors can bias the chain of thought toward heuristic solutions (Arcuschin et al., 2025), and conflicts between a preliminary guess and the reasoning trace can trigger excessive reflection (Dang et al., 2025a). Our work examines what the model encodes about task difficulty before reasoning begins.

On the methodological side, our purification procedure is related to nullspace projection methods for concept erasure such as: INLP (Ravfogel et al., 2020), RLACE (Ravfogel et al., 2022), and LEACE (Belrose et al., 2023), which remove information about a supervised target label from representations. Our concept purification differs in that it removes components corresponding to empirically measured behavioral confounds, defined by contrastive prompt pairs verified to not affect chain-of-thought length, rather than requiring labeled data for each confound type.

6.2. Activation Steering and Reasoning Control

Activation steering modulates model behavior by modifying residual-stream representations at inference time, without

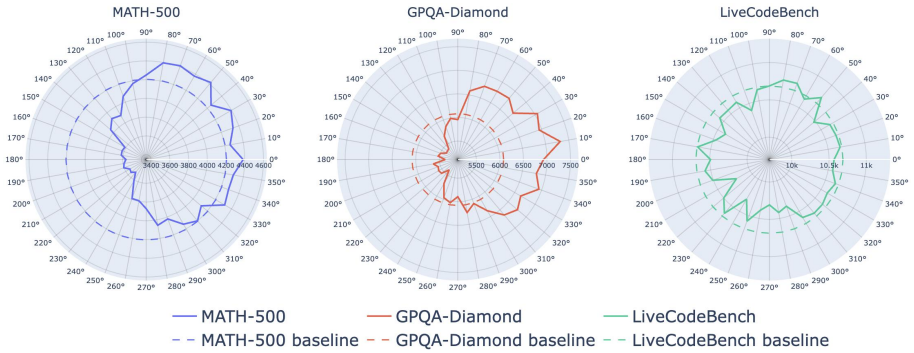


Figure 6. **Thinking length as a function of steering angle for Qwen3-14B across datasets.**

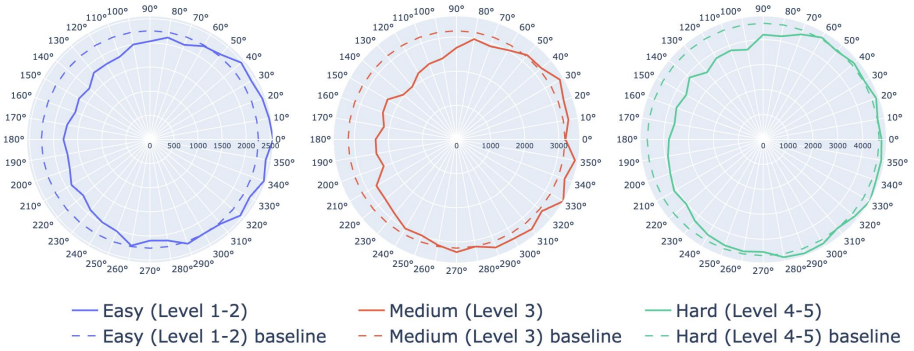


Figure 7. **Thinking length by difficulty level under steering.**

retraining (Turner et al., 2023; 2024). Nguyen et al. (Vu & Nguyen, 2025) introduced angular steering, a method that rotates activations toward or away from a target direction. We adopt this steering approach in our Section 5.

A recent line of work applies such interventions specifically to reasoning models. Venhoff et al. (Venhoff et al., 2025b;a) identify linear directions mediating reasoning behaviors (backtracking, uncertainty) in DeepSeek-R1-Distill models, and show that applying steering vectors to base models can recover up to 91% of the thinking-model performance gap on MATH-500. Azizi et al. (Azizi et al., 2025) extract a steering vector that compresses chain-of-thought traces by up to 67% while preserving accuracy. CREST (Zhang et al., 2025) identifies attention heads associated with verification and backtracking and intervenes on them to reduce token generation. These concurrent works confirm that linear directions mediate reasoning behaviors. Our work differs in two respects: we verify geometric independence of the steering direction from the difficulty manifold before applying it (Section 4.3), and we purify the raw contrastive vector of measurable confound components (Section 4.2).

7. Conclusion

This paper investigated two questions about reasoning-capable language models: whether task complexity is measured before reasoning begins, and whether a separable internal direction governs the models' deliberation decision. On the first question, we found that pre-reasoning activations at the first assistant token carry a non-linear, ordinally structured representation of problem difficulty, which is detectable via a non-linear probe and visible as a U-shaped manifold under PCA. On the second, we extracted a deliberation-associated direction via contrastive prompting and showed that nullspace projection removes measurable confound components without distorting the cross-layer consensus structure. An in-subspace LTSA test found no evidence of entanglement between the purified direction and the difficulty manifold in all four model-dataset conditions tested. Taken together, these findings suggest that complexity perception and deliberation control occupy distinct representational subspaces. We tested this separation through activation steering: negative angular steering along the purified direction reduces thinking tokens by up to 22% across four models on mathematical and scientific benchmarks with smaller effects on code generation and accuracy preserved within 1% on most settings. These results are consistent with the view that pre-reasoning activations encode both problem difficulty and a separable signal governing how many tokens the model will generate, though the precise mechanisms underlying this separation remain to be understood. Whether analogous structure generalizes to other model families, larger scales, or domains beyond mathematics and code is left to future work.

Declaration of LLM Usage

Large language models were used only as a writing assistant to help edit and polish the manuscript, including improving clarity, grammar, concision, and organization. All technical ideas, theoretical results, proofs, experiments, and interpretations were developed and verified by the authors. The authors take full responsibility for the content of the paper.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aggarwal, P. and Welleck, S. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. *URL https://arxiv. org/abs/2503.04697*, 2(3), 2025.

Arcuschin, I., Janiak, J., Krzyzanowski, R., Rajamanoharan, S., Nanda, N., and Conmy, A. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.

Azizi, S., Potraghloo, E. B., and Pedram, M. Activation steering for chain-of-thought compression. *arXiv preprint arXiv:2507.04742*, 2025.

Belrose, N., Schneider-Joseph, D., Ravfogel, S., Cotterell, R., Raff, E., and Biderman, S. Leace: Perfect linear concept erasure in closed form. *Advances in Neural Information Processing Systems*, 36:66044–66063, 2023.

Bogdan, P. C., Macar, U., Nanda, N., and Conmy, A. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

Dang, R., Huang, S., and Chen, J. Internal bias in reasoning models leads to overthinking. *arXiv preprint arXiv:2505.16448*, 2025a.

Dang, R., Li, Z., Huang, S., and Chen, J. The first impression problem: Internal bias triggers overthinking in reasoning models. *arXiv preprint arXiv:2505.16448*, 2025b.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL "https://transformer-circuits.pub/2022/toy_model/index.html".

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Guttman, L. The principal components of scale analysis. *Measurement of prediction*, pp. 312–361, 1950.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Li, J., Zhao, W., Zhang, Y., and Gan, C. Steering llm thinking with budget guidance. *arXiv preprint arXiv:2506.13752*, 2025.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Meng, Y., Xia, M., and Chen, D. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37: 124198–124235, 2024.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Ravfogel, S., Elazar, Y., Gonen, H., Twiton, M., and Goldberg, Y. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7237–7256, 2020.

Ravfogel, S., Twiton, M., Goldberg, Y., and Cotterell, R. D. Linear adversarial concept erasure. In *International Conference on Machine Learning*, pp. 18400–18421. PMLR, 2022.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Rimsky, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., and Turner, A. Steering llama 2 via contrastive activation addition. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pp. 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U., and MacDiarmid, M. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Mini, U., and MacDiarmid, M. Activation addition: Steering language models without optimization. 2024.

Venhoff, C., Arcuschin, I., Torr, P., Conmy, A., and Nanda, N. Base models know how to reason, thinking models learn when. *arXiv preprint arXiv:2510.07364*, 2025a.

Venhoff, C., Arcuschin, I., Torr, P., Conmy, A., and Nanda, N. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint arXiv:2506.18167*, 2025b.

Vu, H. M. and Nguyen, T. M. Angular steering: Behavior control via rotation in activation space. *arXiv preprint arXiv:2510.26243*, 2025.

Xu, S., Xie, W., Zhao, L., and He, P. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhang, Z. and Zha, H. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004.

Zhang, Z., Wu, X., Zhou, Z., Wu, Q., Zhang, Y., Ponnusamy, P., Subbaraj, H., Wang, J., Song, S. L., and Athiwaratkun, B. Understanding and steering the cognitive behaviors of reasoning models at test-time. *arXiv preprint arXiv:2512.24574*, 2025.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Table 3. Contrastive suffix pairs for confound direction extraction. Each pair is appended to the base prompt p_i (the original problem with no suffix). The difference-in-means between positive and negative activations yields one confound direction.

Feature	Positive suffix	Negative suffix
Surface-level stylistic confounds		
Tone	*"Explain the solution in a highly enthusiastic, and friendly tone."*	*"State the solution in a cold, blunt, and strictly robotic tone."*
Formatting	*"Please format your final explanation using bullet points for each step."*	*"Please format your final explanation as a single continuous paragraph without any line breaks."*
Perspective	*"Write the explanation in the first person ('I will calculate…', 'I can see that…')."*	*"Write the explanation in the third person passive voice ('It is calculated…', 'It can be seen that…')."*
Vocabulary	*"Use highly formal and advanced mathematical terminology to describe the operations."*	*"Explain the operations using simple, everyday words as if talking to a beginner."*
Capitalization	*"Write your entire response in ALL CAPITAL LETTERS."*	*"Write your entire response in all lowercase letters, including the start of sentences and proper nouns."*
Cognitive confounds		
Instruction compliance	*"Follow the instructions carefully and step by step."*	(no suffix)
Question type framing	*"This is a critical exam question that will be graded carefully."*	*"This is just a casual practice exercise."*

A. Purification Details

This appendix provides the full list of confound suffix pairs used in Linear Concept Purification (Section 4.2), the norm ratio between the purified and raw deliberation-associated directions for each model, and the cross-layer stability curves before and after purification.

A.1. Confound Suffix Pairs

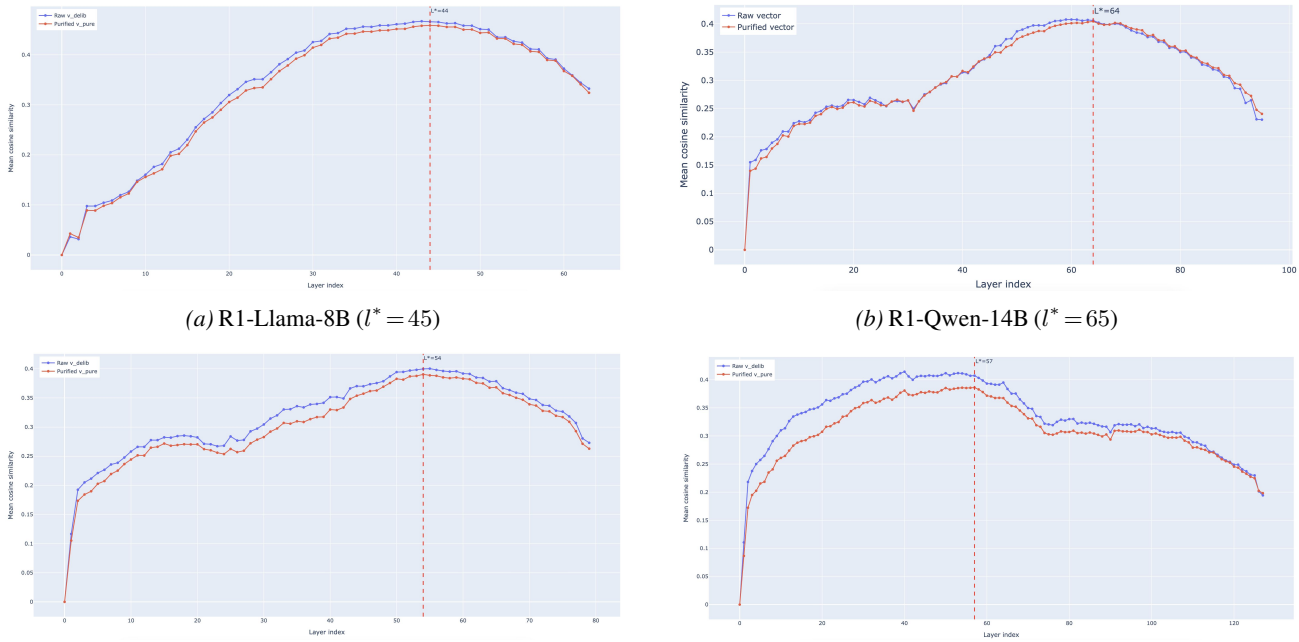
Table 3 lists all contrastive suffix pairs used to construct the confound direction matrix M_{noise}. Each pair defines a difference-in-means direction targeting a specific non-deliberation feature. The first five pairs (Tone through Capitalization) target surface-level stylistic features; the remaining two (Instruction Compliance and Question Type Framing) target deeper cognitive features. All confound suffixes were verified to have negligible effect on chain-of-thought length prior to inclusion.

A.2. Purification Norm Ratio

Table 4 reports the selected layer l^* and the norm ratio $\|v_{\text{pure}}\|/\|v_{\text{raw}}\|$ at that layer for each model. The norm ratio quantifies how much of the raw contrastive direction's magnitude is retained after projecting out all 7 confound directions. Values range from 0.868 to 0.905, indicating that approximately 10-13% of the raw direction's magnitude is attributable to confound components. The selected layer l^* corresponds to the layer with the highest mean cosine similarity between its purified direction and those at all other layers (i.e., the most stable layer after purification).

Table 4. Selected layer and purification norm ratio for each model. The norm ratio $\|v_{\text{pure}}\|/\|v_{\text{raw}}\|$ measures the fraction of the raw direction retained after nullspace projection of 7 confound directions.

Model	Selected layer l^*	$\|v_{\text{pure}}\|/\|v_{\text{raw}}\|$
R1-Llama-8B	45	0.905
R1-Qwen-14B	65	0.893
Qwen3-14B	55	0.868
Qwen3-32B	58	0.884



(a) R1-Llama-8B ($l^* = 45$)

(b) R1-Qwen-14B ($l^* = 65$)

(c) Qwen3-14B ($l^* = 55$)

(d) Qwen3-32B ($l^* = 58$)

Figure 8. Cross-layer stability for raw (v_{raw}, blue) and purified (v_{pure}, orange) directions across model families. Vertical dashed lines indicate the selected layer l^* for each architecture.

A.3. Cross-Layer Stability Curves

Figures 8a–8d show the cross-layer stability curves for both the raw (v_{raw}) and purified (v_{pure}) directions across all four models. Each curve plots the mean cosine similarity between the direction at layer l and the directions at all other layers. The curves confirm that purification preserves the overall cross-layer consensus structure: the shape of the stability curve is maintained after projection, indicating that the nullspace projection removes layer-specific confound components without distorting the shared deliberation signal.

B. Additional Steering Results

This appendix presents the full steering results for all four models across all benchmarks. Each polar chart shows the normalized thinking token count (radial axis) as a function of steering angle θ (angular axis), with separate traces for each benchmark.

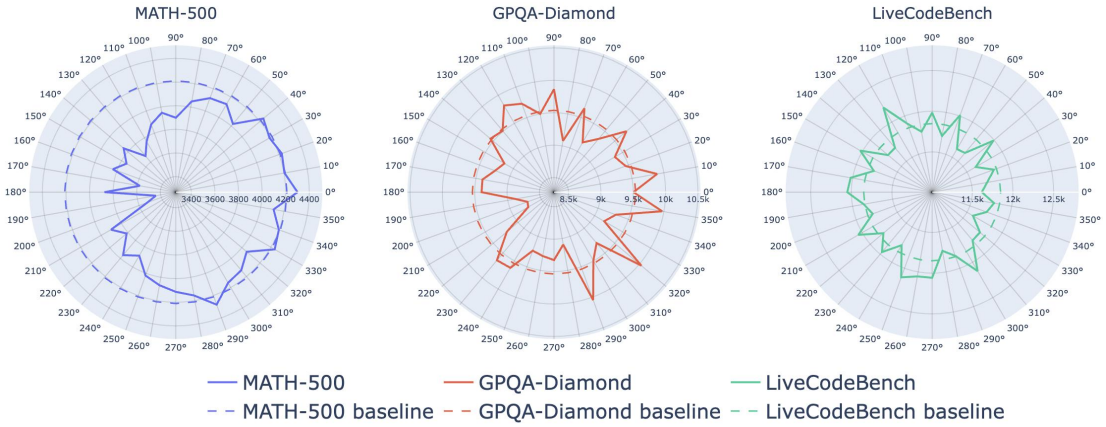


Figure 9. Steering results for DeepSeek-R1-Distill-Llama-8B across MATH-500, LiveCodeBench, and GPQA-Diamond.

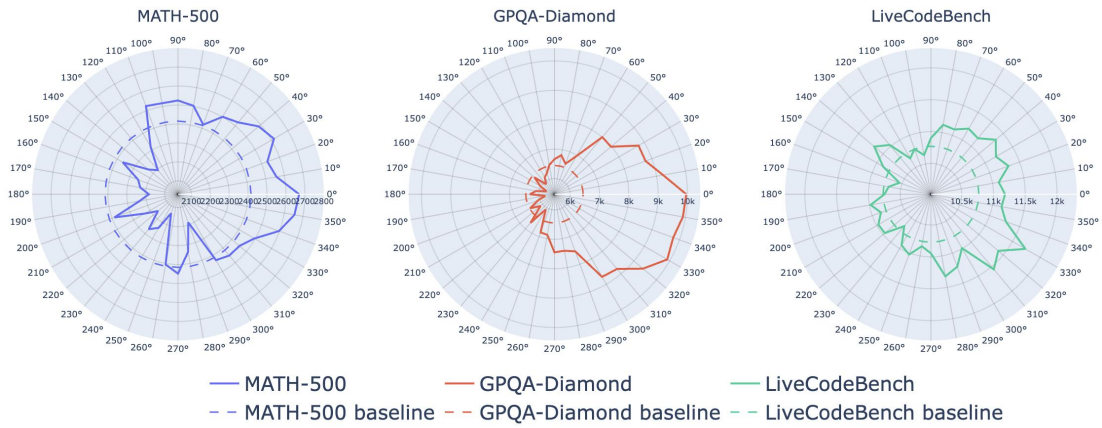


Figure 10. Steering results for DeepSeek-R1-Distill-Qwen-14B across MATH-500, LiveCodeBench, and GPQA-Diamond.

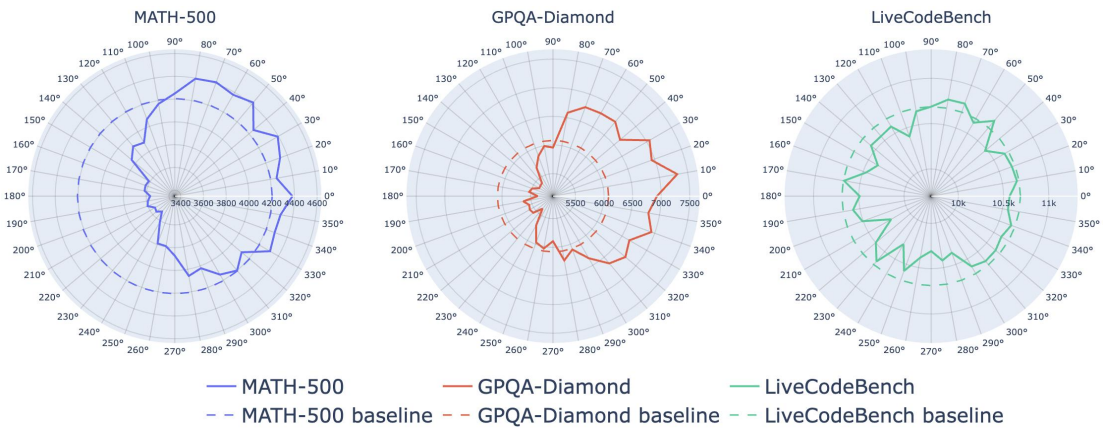


Figure 11. Steering results for Qwen3-14B across MATH-500, LiveCodeBench,and GPQA-Diamond.

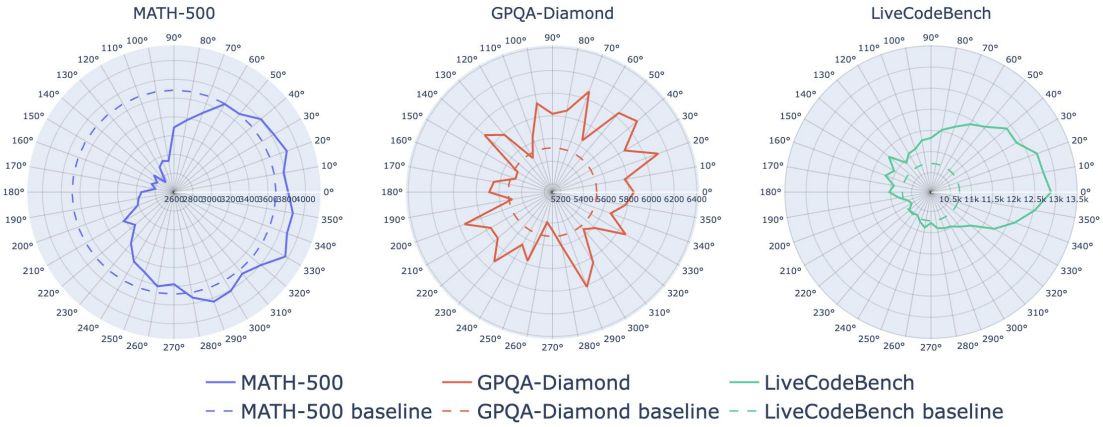


Figure 12. Steering results for Qwen3-32B across MATH-500, LiveCodeBench, and GPQA-Diamond.

C. LLM-as-a-Judge Evaluation

For MATH-500 and GPQA-Diamond, model outputs are open-ended text rather than structured tokens, so exact-string matching against the gold answer is unreliable. We therefore use an LLM-as-a-judge protocol to evaluate Pass@1. For

each benchmark, we prompt a judge model with the gold answer and a batch of candidate responses, and ask it to label each candidate as `Correct` or `Incorrect`. The judge receives no additional context beyond the gold answer and the candidates' final outputs; it does not see the reasoning trace. LiveCodeBench is evaluated via execution-based unit tests and does not require this procedure.

The prompt templates used for each benchmark are shown below.

C.1. GPQA-Diamond

For GPQA-Diamond, the gold answer is one of {A, B, C, D}. The judge checks whether each candidate's response contains the correct option letter.

[colback=gray!5, colframe=gray!50, title=GPQA-Diamond Judge Prompt]

```
Evaluate the correctness of several answers to a
multiple-choice problem. You are given:
1. **gold answer**: The correct ground-truth answer
   (one of A, B, C, D)
2. **candidate answers**: answers provided by different
   candidates

Evaluate and label each candidate answer as `Correct`
or `Incorrect`. If the candidate answer does not
contain the expected value, just mark it as `Incorrect`.
You must output exactly {{ num_candidates }} labels
(one per candidate), in the same order as the
candidates. Output only a comma-separated list of
labels, nothing else.

Example:
- Gold answer: A
Candidates' answers:
- Candidate 1: Therefore the answer is \boxed{A}
- Candidate 2: ### **Final Answer**\n\n$$\n\boxed{B}\n$$
- Candidate 3: **C**

Answer: Correct, Incorrect, Incorrect

Now, here is your turn:
- Gold answer: {{ gold_answer }}

Here are the candidate answers:
{{ list_candidate_answers }}
```

C.2. MATH-500

For MATH-500, the gold answer is a mathematical expression (e.g., `\frac{5}{9}`). The judge must determine whether each candidate's response is mathematically equivalent to the gold answer, accounting for notational variation (e.g., `\dfrac` vs. `\frac`).

[colback=gray!5, colframe=gray!50, title=MATH-500 Judge Prompt]

```
Evaluate the correctness of several answers to a math
problem. You are given:
1. **gold answer**: The correct ground-truth answer
2. **candidate answers**: answers provided by different
   candidates

Evaluate and label each candidate answer as `Correct`
or `Incorrect`. If the candidate answer does not
contain the expected value, just mark it as `Incorrect`.
You must output exactly {{ num_candidates }} labels
(one per candidate), in the same order as the
candidates. Output only a comma-separated list of
```

```
labels, nothing else.

Example:
- Gold answer: "\frac{5}{9}"
Candidates' answers:
- Candidate 1: "Thus, the final result is \dfrac{5}{9}"
- Candidate 2: "**Final Answer:**\n  \n\dfrac{5}{9}"
- Candidate 3: "But when I use the scaled direction
  vectors (3, 2, -6) and (2, -12, -3), their dot"

Answer: Correct, Correct, Incorrect

Now, here is your turn:
- Gold answer: {{ gold_answer }}

Here are the candidate answers:
{{ list_candidate_answers }}
```

Both prompts use Jinja2 template syntax. The variable `num_candidates` controls the expected output length, and `list_candidate_answers` is populated with the formatted candidate responses at evaluation time. We use a single judge call per batch of candidates generated under different steering conditions for the same problem, ensuring that the judge evaluates all conditions for a given problem simultaneously and thus reducing inter-call variance.

D. Use of existing assets

D.1. Models

Table 5. Models used in this work, including their usage roles, source links, and license types.

Model	Usage	Source	License
DeepSeek-R1-Distill-Llama-8B	Experimental subject	HF Hub	MIT License
DeepSeek-R1-Qwen-14B	Experimental subject	HF Hub	Apache 2.0
Qwen3-14B	Experimental subject	HF Hub	Qwen License
Qwen3-32B	Experimental subject	HF Hub	Qwen License
Qwen2.5-32B-Instruct	Evaluation device	HF Hub	Apache 2.0
Gemini-2.5-Flash	Evaluation device	Google AI Studio	Google API Terms of Service

D.2. Datasets

Table 6. Datasets used in the evaluation, including their source repositories and license types.

Dataset	Source	License
MATH	HF Hub	MIT License
MATH-500	HF Hub	MIT License
LiveCodeBench	HF Hub	CC BY 4.0
GPQA-Diamond	GitHub Repository	MIT / CC-BY-4.0