HARD OR STUCK? VIABILITY-CONDITIONED STEERING FOR BUDGETED REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Practical deployments of reasoning models impose token budgets to bound inference cost, but hard limits can truncate trajectories before they reach an answer. Existing control mechanisms rely on fixed schedules or predicted reasoning lengths, frequently failing to distinguish genuinely difficult questions from off-track reasoning. Consequently, they may unnecessarily intervene in trajectories that would otherwise complete naturally. This motivates trajectory-aware intervention adapting timing and strength to individual rollouts. We show that a probe over intermediate hidden states predicts whether a trajectory will finish within budget well before the limit is reached, beyond what generation position alone reveals. Its early prediction is strongly associated with question difficulty and baseline cap propensity, while subsequent increases in predicted cap risk provide a trajectory-relative signal of worsening completion prospects. Motivated by this observation, we introduce Viability-Conditioned Steering (VCS). VCS applies sparse hidden-state updates whose strength adapts online to the increase in cap risk relative to each trajectory's initial level and to deadline pressure as the remaining budget shrinks. The update direction is estimated offline from matched hidden states of finished and capped rollouts. Across diverse reasoning models and benchmarks, VCS outperforms uncontrolled decoding in all 16 settings, boosting accuracy by 3.8–21.7 percentage points, while cutting token consumption by 13.6–34.8% and lowering cap rates by 67.9–92.5%.

1 INTRODUCTION

Longer reasoning and test-time compute can improve answer quality (Wu et al., 2025; Snell et al., 2024; DeepSeek-AI, 2025), but practical deployment must balance these gains against inference cost (Arora & Zanette, 2025; Wen et al., 2025). Token budgets provide a direct way to control generation length (Han et al., 2025; Muennighoff et al., 2025), yet hard limits can truncate reasoning before a response is complete. As shown in Figure 1, 17%–88% of the rollouts reach the 8192-token limit without emitting an end-of-sequence (EOS) token. We call such rollouts *capped*. A capped rollout rarely produces a scorable answer.

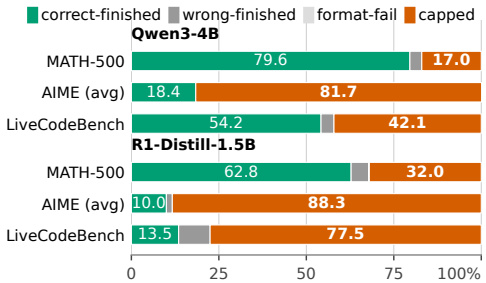


Figure 1: Uncontrolled reasoning under an 8 k-token budget. Token limits constrain generation cost but often yield incomplete reasoning, motivating budget-aware reasoning controls.

Intervention intended to drive a rollout toward completion can mitigate capping. Existing inference-time controls differ in when and how they intervene. Budget Forcing ends the thinking phase at a chosen token limit (Muennighoff et al., 2025), while SEAL applies a fixed steering strength at thought boundaries (Chen et al., 2025). Neither adjusts its timing or strength to a rollout's finishing prospects, so on-track and at-risk rollouts receive identical treatment. Budget Guidance instead predicts the distribution of remaining reasoning length and reweights token probabilities to favor continuations likely to finish within thinking budget (Li et al., 2026a). However, a forecast of remaining reasoning length alone does not distinguish task difficulty from trajectory-level

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

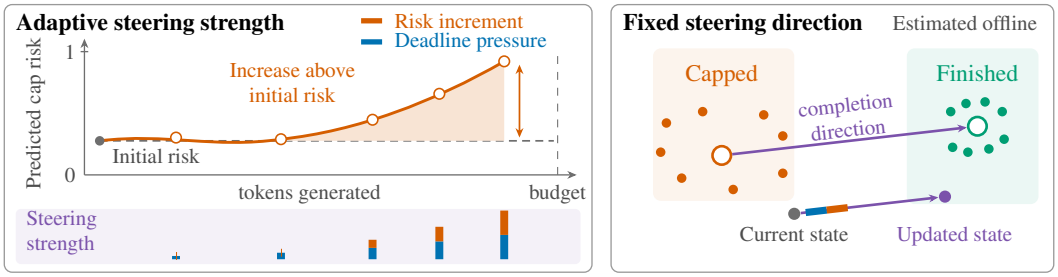


Figure 2: Overview of VCS. Each update is the product of a strength and a direction. *Left:* the strength is set online, combining the relative increment in cap risk above the trajectory's initial risk (orange) with deadline pressure (blue). *Right:* the direction is fixed, estimated offline from the difference between the mean hidden states of finished and capped rollouts.

deterioration. This motivates an intervention rule that responds to signs of an off-track trajectory while allowing difficult questions the reasoning they need.

An intuitive signal for budget-aware intervention is generation progress relative to the token budget. A probe using only this information achieves 0.72–0.82 AUROC in predicting whether a rollout will finish within budget. That number pools states from all positions. Since capped rollouts are by construction the long ones, a state observed late is more likely to come from a capped rollout. Comparing states at matched token positions removes this advantage, and the probe drops to near chance. Adding intermediate hidden states to this probe raises AUROC to 0.92–0.97 across four models in this comparison. Hidden states thus provide a stronger signal than generation progress alone for predicting *viability*, the probability of completion within the remaining budget. We refer to its complement as *cap risk*. Viability is deliberately distinct from correctness: it measures whether the current reasoning trajectory is likely to terminate within budget, not whether its eventual answer will be correct.

An accurate risk forecast alone does not determine how strongly to intervene. High cap risk can arise partly because a question intrinsically requires substantial reasoning under the available budget, or because the particular rollout is becoming less likely to finish than it initially appeared. Intervening solely on the current risk level therefore risks applying the strongest control to difficult questions that may simply need more computation. We instead use each trajectory's early risk estimate as a reference point and track subsequent changes relative to that baseline. Empirically, the initial estimate is strongly associated with question difficulty and per-question cap propensity, while changes from this initial level provide additional trajectory-specific information about eventual capping. In particular, trajectories with similar current risk can have markedly different changes relative to their starting points (Section 3). This motivates conditioning intervention on deterioration relative to a trajectory-specific baseline rather than on current risk alone.

Building on this distinction, we propose Viability-Conditioned Steering (VCS), which applies residual-stream updates during reasoning (Figure 2). Each update combines an adaptive strength with a fixed direction. To set the strength, we design an objective that balances necessary intervention against unnecessary disruption. When viability falls below its initial value, intervention strength increases linearly with the fraction of initial viability lost. Even when viability remains stable, completing the rollout becomes more urgent as the remaining token budget shrinks. We therefore add a signal that grows with generation progress, so the strength has two parts, one from lost viability and one from the budget. This strength scales a fixed completion direction. We estimate this direction offline from the difference between the mean hidden states of finished and capped rollouts at matched generation positions. Estimating the direction offline avoids computing probe gradients at each intervention point, while the adaptive strength allows the intervention to respond to changes in each rollout's viability. The probes and the direction are fitted once per model on MATH training rollouts, and reused unchanged across evaluation benchmarks, including code generation.

Our contributions are as follows.

- **We identify a trajectory-relative signal for budgeted reasoning control.** Intermediate hidden states predict within-budget completion better than position alone. The early pre-

diction is strongly associated with question difficulty and baseline cap propensity, while later changes from this level help predict budget exhaustion.

- **We introduce Viability-Conditioned Steering (VCS).** VCS combines a trajectory-specific signal with a shared deadline signal: intervention increases when viability deteriorates relative to its initial level and as the remaining budget shrinks. The gate scales a fixed completion direction estimated offline from matched finished and capped states, yielding an inexpensive inference-time controller without policy training.
- **We demonstrate consistent accuracy–efficiency gains across models and tasks.** Across four reasoning models and four benchmarks, VCS improves accuracy over uncontrolled decoding in all 16 model–benchmark settings by 3.8–21.7 percentage points while reducing token use by 13.6–34.8% and cap rates by 67.9–92.5%.

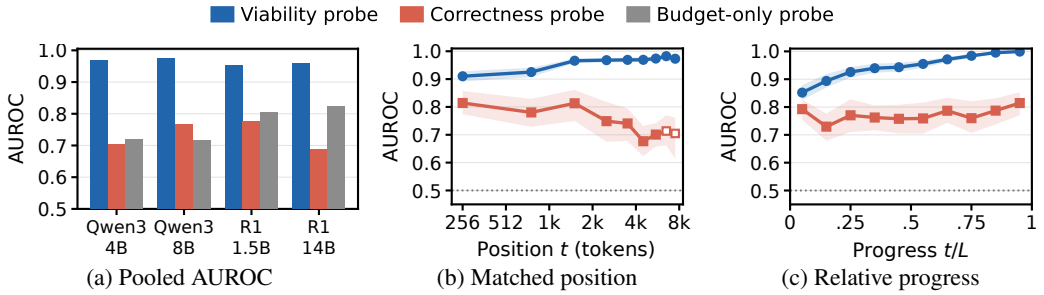
2 Related Work

Training-based length and budget control. Work on length control trains models to meet target word or token counts specified in the prompt. Jie et al. (2024) study several forms of length instruction, and Ruler encodes the requested length with meta length tokens (Li et al., 2024). For reasoning, Arora & Zanette (2025) train policies with an explicit accuracy–cost trade-off, while L1 trains models to match a requested reasoning length (Aggarwal & Welleck, 2025). SelfBudgeter estimates a query-specific budget before budget-guided reinforcement learning (Li et al., 2026b), while DAST assigns difficulty-adaptive length budgets and rewards (Shen et al., 2025). BudgetThinker communicates the remaining budget through periodic control tokens and trains the model to respond to them (Wen et al., 2025). All of these methods update the generation policy.

Inference-time budgeting and termination control. Wu et al. (2025) characterize cost–performance trade-offs of sampling, voting, and search at test time, while Snell et al. (2024) show that the best allocation depends on prompt difficulty. To control a single trajectory, TALE estimates a per-question token budget and includes it in the prompt (Han et al., 2025); s1 either truncates reasoning or appends a continuation cue (Muennighoff et al., 2025); and Chain of Draft asks for just a few words per step to encourage concise reasoning (Xu et al., 2025). Other online methods stop reasoning programs when their answers stabilize (Fu et al., 2025) or prune partial generations by judging whether restarting is worthwhile (Manvi et al., 2024). Methods for terminating individual trajectories use answer confidence (Yang et al., 2026), thought sufficiency (Xiang et al., 2026), or a learned utility balancing accuracy and cost (Ehab et al., 2026). Budget Guidance (Li et al., 2026a) uses hidden states to predict remaining reasoning length. For each candidate next token, it estimates the probability of completion within budget, then reweights the token distribution accordingly.

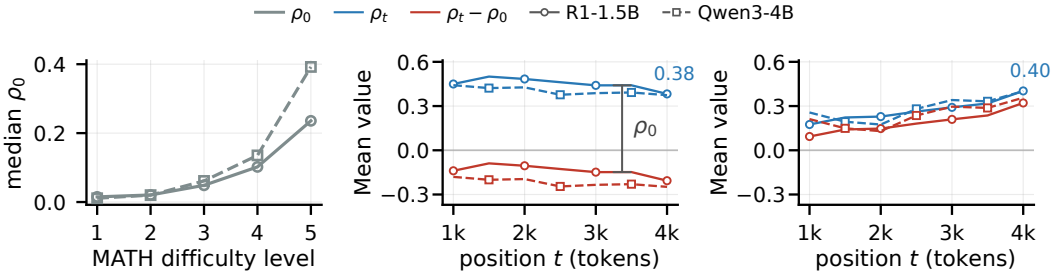
Latent-state analysis and adaptive control of reasoning. Early analyses showed that hidden states encode latent knowledge and truthfulness and can support hallucination detection (Burns et al., 2023; Azaria & Mitchell, 2023; Chen et al., 2024). Latent verifiers estimate reasoning correctness for response selection and self-correction (Piotrowski et al., 2025), or assess intermediate answers to support early exit (Zhang et al., 2025). Activation interventions along probe-derived or contrastive directions change model behavior (Li et al., 2023; Rimsky et al., 2024). For reasoning efficiency, Reasoning Strength Planning identifies a direction that modulates planned effort (Sheng et al., 2025); SEAL steers at thought boundaries to suppress unnecessary reflection (Chen et al., 2025). Activation compression and manifold steering reduce verbose or overextended reasoning (Azizi et al., 2026; Huang et al., 2025). Other methods use feedback to adapt interventions or apply them selectively: PID-steering adjusts strength from a chunk-level redundancy estimate (Bharadwaj, 2025), and ATLAS uses a latent verifier to choose among intervention actions (Nguyen & Le, 2026).

Relation to adaptive reasoning control. VCS differs from prior adaptive steering primarily in the feedback variable it uses. Existing methods adapt intervention based on signals such as redundancy, answer confidence, uncertainty, susceptibility, or learned action utility. VCS instead uses a trajectory-relative completion signal: the same viability forecast is decomposed *operationally* into an early trajectory-specific reference level and subsequent change relative to that level, while a separate position-only term captures deadline pressure. Thus, high baseline risk alone does not determine intervention strength. The ablations in Section 5.2 directly test this distinction against fixed steering, deadline-only control, and gating on the current risk level.



(a) Pooled AUROC (b) Matched position (c) Relative progress

Figure 3: **Viability prediction beyond generation position**. *(a)* Pooled AUROC across four models. *(b)* Qwen3-4B AUROC at matched absolute positions. *(c)* Qwen3-4B AUROC at relative progress t/L. Correctness is measured on naturally finished rollouts. Open markers in (b) denote < 30 wrong finishers. Bands show 95% stratified bootstrap intervals over rollouts.



(a) ρ_0 and difficulty (b) High ρ_0, later finishes (c) Low ρ_0, later caps

Figure 4: **Initial risk ρ_0 increases with question difficulty. Rollouts with similar ρ_t show opposite risk increments.** *(a)* Median per-question mean ρ_0 by MATH difficulty label. *(b,c)* Among rollouts still generating at 4096 tokens, we compare those in the highest initial-risk quartile that later finish with those in the lowest quartile that later cap. These selected groups have similar mean current risk at 4096 tokens but mean risk increments of opposite signs.

3 Diagnosis: What Reasoning States Predict

We first ask whether hidden states predict completion within budget beyond what generation position alone reveals. From base-policy rollouts, we collect residual activations at newlines in the thinking segment, the same positions used for intervention in Section 4. We encode generation progress relative to the token budget as *budget features*. We train a classifier, called a *viability probe*, on these features and the hidden states. Its output estimates the probability that the rollout emits EOS within the remaining budget, corresponding to the viability defined in Section 1. We write ρ_t for the complementary cap risk and ρ_0 for its value at the trajectory's first newline. To measure what generation position alone predicts, we train a budget-only probe on the same budget features. Inspired by Zhang et al. (2025), we also train a correctness probe on naturally finished rollouts to predict whether the final answer is correct. Appendix B.2 provides training and evaluation details.

Hidden states predict capping beyond generation position alone. Capped rollouts reach the token limit, while finished rollouts stop earlier. A state observed late in generation is therefore more likely to come from a rollout that will be capped. Position alone is thus a confound for predicting capping. Without controlling for position, the budget-only probe reaches 0.72–0.82 AUROC without reading a single hidden state (Figure 3a). This predictive advantage disappears once we compare states at matched token positions: the budget-only probe falls to near chance, while the viability probe holds 0.92–0.97 AUROC on all four models and stays high across positions on Qwen3-4B (Figure 3b). The correctness probe does not perform as well. Its AUROC falls at later token positions and recovers only when progress is measured relative to the final rollout length (Figure 3c). But the final rollout length is unknown while the rollout is still being generated. We therefore base the intervention on viability rather than correctness.

Initial risk tracks the question, and rollouts with similar current risk ρ_t can have opposite risk increments. Per-question mean ρ_0 increases with the MATH difficulty label (Figure 4a) and is also associated with a question's empirical cap rate (Figure 8a). Figure 4b,c compares two groups:

rollouts with high initial ρ_0 that eventually finish, and rollouts with low initial ρ_0 that eventually cap. At 4096 tokens the two groups sit at a similar ρ_t but have increments of opposite sign.

High AUROC indicates good discrimination between finished and capped rollouts, but does not specify how strongly to intervene. In an analysis of six continuations for each of 500 MATH-500 questions, the risk increment and current risk give comparable within-question rankings (Appendix A.4). Using initial risk as a reference allows us to assign different intervention strengths to rollouts with similar current risk but different risk increments.

4 VIABILITY-CONDITIONED STEERING

Section 3 motivates measuring deterioration relative to each trajectory's initial viability rather than controlling solely from its current risk level. We now construct VCS in two steps. First, we introduce a local objective that motivates a trajectory-relative feedback term and a state-dependent completion direction. Second, we turn this local prescription into a practical controller by adding deadline pressure and replacing the state-dependent direction with an offline estimate shared across decoding states. These latter steps are design approximations. Let $h_{t,l}$ be the residual state at token position t and layer l. At newline positions C within the thinking segment, VCS applies

$$h'_{t,l} = h_{t,l} + \lambda \underbrace{g_t}_{\text{gate}} \underbrace{d_l}_{\text{completion direction}}, \qquad t \in \mathcal{C},\ l \in \mathcal{L}. \tag{1}$$

where $g_t \in [0, 1]$ is an online gate, d_l is an offline completion direction, and $\lambda > 0$ is a global scale.

Let $H_t = \{h_{t,l}\}_{l\in\mathcal{L}}$ denote the residual states at these layers. The budget features Z_t encode information on the current position t and the token budget B (Appendix B.2). A rollout is finished ($R = 1$) if it emits EOS within budget, and capped ($R = 0$) otherwise. Let $V(H, Z) = \Pr_{\pi_0}(R = 1 \mid H, Z)$ be the probability that generation continuing from state (H, Z) under the base policy π_0 finishes within budget. At position t, viability is $V_t = V(H_t, Z_t)$, estimated from the viability probe $F_\phi(H_t, Z_t)$. We write $\rho_t = 1 - V_t$ for cap risk. [1] At the first newline in the thinking segment, we record the initial viability V_0 and cap risk ρ_0. Each control point's residual state produces the next line's first token, so the intervention operation reads and modifies the same state.

Balancing intervention benefit and disruption We use a simple local decision model to motivate how intervention strength should depend on viability. Suppose that steering a rollout that would otherwise be capped has benefit weight $r > 0$, while perturbing one that would otherwise finish incurs relative cost $c \geq 0$. Because the eventual outcome is unknown at intervention time, the corresponding expected local gain is

$$S_t = r\rho_t - cV_t. \tag{2}$$

The purpose of this model is not to assign an absolute utility to finishing versus capping, but to determine a trajectory-specific intervention threshold. Section 3 suggests that the initial risk ρ_0 contains substantial question-level information, so we want the risk-dependent correction to be inactive at the trajectory's starting level and to become positive for any subsequent increase above it. We therefore set the relative cost so that $S_0 = 0$. For $0 < V_0 < 1$, this gives

$$c = r\frac{1 - V_0}{V_0}, \qquad S_t = r\left(1 - \frac{V_t}{V_0}\right) = r\frac{\rho_t - \rho_0}{1 - \rho_0}. \tag{3}$$

Thus the feedback term is proportional to the fraction of initial viability lost. This normalization follows from choosing the trajectory's own initial viability as the indifference point, and it is not an additional fitted parameter. Positive values indicate deterioration relative to the starting point, whereas non-positive values call for no risk-driven correction.

[1] Importantly, V is defined under the base policy π_0 and is fitted from base-policy rollouts. Once steering perturbs the hidden state, the probe is therefore used as a local control signal rather than as a guaranteed calibrated estimate of the steered policy's future completion probability. We return to this approximation in Appendix A.1.

From the local gain to a state-space direction We next use the decision model above to motivate a local hidden-state update. Let $f = \operatorname{logit} V$. Holding the budget features Z_t fixed, consider the regularized local objective

$$\max_{\delta} \; r \log V(H_t + \delta, Z_t) + c \log\big(1 - V(H_t + \delta, Z_t)\big) - \frac{1}{2\eta}\|\delta\|_2^2. \tag{4}$$

This objective should be viewed as a *local surrogate*, not as a global objective for the steered decoding policy. Its role is to identify how the trajectory-relative gain S_t combines with the local geometry of the viability predictor. Linearizing the log terms at the current base-policy state gives

$$\delta^* = \eta r \quad \underbrace{\frac{\rho_t - \rho_0}{1 - \rho_0}}_{\text{relative increment}} \quad \underbrace{\nabla_H f(H_t, Z_t)}_{\text{local direction}}. \tag{5}$$

Hence the local surrogate separates naturally into a scalar feedback term, which is the relative loss of viability, and a state-dependent direction $\nabla_H f$ that locally increases completion log-odds.

Combining risk increments with deadline pressure. The relative increment is negative when risk sits below ρ_0, where Equation 5 would call for reducing viability. The gate takes its positive part, so it never reverses the completion direction, which also leaves no correction whenever $\rho_t \le \rho_0$, even as the remaining budget shrinks. We therefore add a *deadline schedule* $\rho_{\mathrm{dl}}(Z_t) = 1 - F_{\phi'}(Z_t)$ from the budget-only probe. It estimates cap risk across base-policy rollouts at the same budget position. Because it reads Z_t alone, it is the same for every trajectory there, and it is nonzero even at the first control point, where the relative increment is zero by construction. Taking the positive part, adding the schedule, and clipping the sum at one, we have the gate

$$g_t = \min\Big\{ \rho_{\mathrm{dl}}(Z_t) + \max\Big\{ \frac{\rho_t - \rho_0}{1 - \rho_0},\, 0 \Big\},\; 1 \Big\}. \tag{6}$$

The schedule supplies a baseline strength, the positive relative increment adds feedback when risk rises, and clipping bounds the gate in $[0, 1]$.

Estimating the completion direction. Evaluating $\nabla_H f$ would require a backward pass at every control point, so we replace it with one fixed direction per layer, estimated before decoding. Following contrastive activation steering (Rimsky et al., 2024), we take it from finished and capped base-policy rollouts. We match finished and capped states by generation position to reduce the influence of generation progress on the estimated direction. We then pool the matched states across training questions to compute their respective means, $\bar{h}_l^{\mathrm{fin}}$ and $\bar{h}_l^{\mathrm{cap}}$. Their difference gives

$$\Delta_l = \bar{h}_l^{\mathrm{fin}} - \bar{h}_l^{\mathrm{cap}}, \qquad s_l = \mathbb{E}\big[\|h_{t,l}\|_2\big], \qquad d_l = s_l \, \frac{\Delta_l}{\|\Delta_l\|_2}. \tag{7}$$

Combining this direction with the gate in Equation 6 gives the full update in Equation 1. s_l is the mean residual norm at base-policy control points, and scaling it puts λ in units of a layer's typical residual size, so a single λ serves every layer and model. Appendix A.2 gives sufficient distributional assumptions under which this mean difference is aligned with the gradient it replaces *on average*. These assumptions justify the direction as a computationally convenient surrogate; they do not imply pointwise alignment along every steered trajectory.

Offline fitting and online control. We fit both probes and the direction once per model on MATH training rollouts and reuse them across benchmarks. During generation, the gate adapts while the layer band $\mathcal{L}$, direction d_l, and global scale λ remain fixed (Section 5, Appendix B.2). The scale absorbs the constant ηr from the local update, giving an update norm of $\lambda g_t s_l$ at each selected layer. Section 5.2 tests the schedule, risk signal, and direction separately.

5 Experiments

Setup. We evaluate Qwen3-4B and Qwen3-8B (Yang et al., 2025), and DeepSeek-R1-Distill-Qwen-1.5B and 14B (DeepSeek-AI, 2025), with a token budget of $B = 8192$. Unless otherwise

Table 1: **Main results under an 8192-token budget.** AIME pools the 2025 and 2026 exams. Accuracy (Acc.) and cap rate (Cap) are percentages. Tok. is the mean number of generated tokens over all trajectories. Cap counts trajectories that reach the token limit without emitting EOS. Capped trajectories can still be correct, so Acc. and Cap may sum to more than 100%. For each metric within each model–benchmark block, the best value is bold and the second-best is gray.

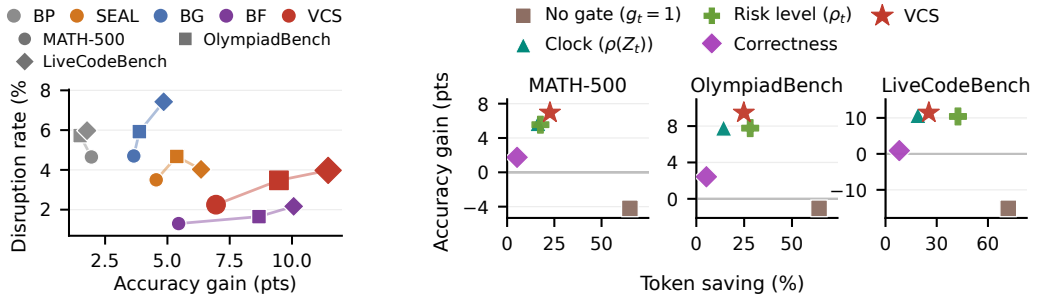
Method	MATH-500			AIME			OlympiadBench			LiveCodeBench		
	Acc. ↑	Cap ↓	Tok. ↓	Acc. ↑	Cap ↓	Tok. ↓	Acc. ↑	Cap ↓	Tok. ↓	Acc. ↑	Cap ↓	Tok. ↓
Qwen3-4B												
CoT	83.2	17.0	4209	21.7	81.7	7749	46.6	47.5	6308	58.9	42.1	5629
Budget Prompt	86.0	13.8	3712	26.7	78.3	7679	47.6	45.8	6108	64.2	33.5	5241
Budget Forcing	86.2	11.4	4189	**45.0**	65.0	7586	51.9	35.3	6258	69.7	31.5	5545
SEAL	84.6	14.0	3606	31.7	65.0	7308	51.3	41.4	5783	67.5	30.5	4892
Budget Guidance	86.0	4.4	**2819**	26.7	36.7	**5723**	53.9	14.8	**4726**	69.3	**11.2**	**3260**
VCS (ours)	**89.8**	**3.4**	3516	43.3	**23.3**	6358	**59.2**	**8.8**	5142	**76.5**	12.9	4864
Qwen3-8B												
CoT	84.4	16.0	4338	21.7	85.0	7769	46.7	50.4	6443	57.9	41.1	5735
Budget Prompt	83.8	17.8	4278	21.7	81.7	7757	47.8	49.3	6428	58.1	41.5	5763
Budget Forcing	89.6	13.4	4315	**43.3**	83.3	7760	**60.4**	43.5	6392	68.7	35.2	5725
SEAL	86.2	12.8	3673	28.3	75.0	7370	51.6	44.7	5880	66.9	31.3	5138
Budget Guidance	89.4	**2.2**	**3473**	31.7	51.7	7167	56.7	**10.1**	5527	71.4	**5.1**	**2782**
VCS (ours)	**90.2**	2.4	3541	38.3	**25.0**	**6272**	57.3	14.4	**5212**	**76.7**	7.4	4635
DeepSeek-R1-Distill-Qwen-1.5B												
CoT	66.8	32.0	4133	10.0	88.3	7618	29.8	62.2	6251	18.0	77.5	6861
Budget Prompt	70.4	22.6	3173	13.3	78.3	7396	31.5	56.8	5777	19.0	77.5	6904
Budget Forcing	75.6	15.0	4017	**21.7**	63.3	7624	36.9	38.4	6203	**26.6**	41.9	6692
SEAL	77.8	14.0	2763	20.0	55.0	6309	38.0	41.8	5047	22.3	63.2	6284
Budget Guidance	77.2	**5.6**	3104	16.7	**16.7**	6982	33.8	13.8	5444	9.2	20.5	6013
VCS (ours)	**78.4**	5.8	**2735**	18.3	28.3	**5495**	**39.3**	**9.3**	**4128**	23.1	**11.0**	**4530**
DeepSeek-R1-Distill-Qwen-14B												
CoT	85.8	10.8	3109	26.7	68.3	7127	50.3	37.5	5196	58.7	39.1	5486
Budget Prompt	87.8	7.4	2475	33.3	65.0	6847	52.5	32.2	4683	59.3	36.8	5401
Budget Forcing	**90.6**	4.4	3027	33.3	35.0	6924	**58.8**	13.4	5178	59.5	35.2	5490
SEAL	89.8	5.0	2449	**45.0**	43.3	6115	54.0	28.2	4448	62.2	31.5	4902
Budget Guidance	82.2	9.0	3043	21.7	56.7	6961	44.5	24.9	4542	**63.0**	9.8	5849
VCS (ours)	89.6	**1.0**	**2425**	35.0	**15.0**	**4849**	55.5	**5.9**	**3754**	**63.0**	**2.9**	**3578**

stated, all runs decode greedily. The evaluation includes MATH-500 (Hendrycks et al., 2021) ($n = 500$), AIME 2025 and 2026 (OpenCompass, 2025; MathArena, 2026) ($n = 60$ in total), OlympiadBench (He et al., 2024) ($n = 674$), and LiveCodeBench (Jain et al., 2025) ($n = 511$). For each model, we fit both probes and the completion direction once using finished and capped rollouts from 1,200 questions in the MATH training split (Appendix B.2). At test time, the steering update in Equation 1 uses $\lambda = 0.2$ and applies only at newline tokens within the thinking segment. Steered runs additionally use a repetition guard that can suspend steering or stop generation (Appendix C). We compare with uncontrolled decoding (CoT) and four baselines that modify inference while leaving the model frozen. *Budget Prompt* states the token limit in the system prompt. *Budget Forcing* forces a transition to the answer at a fixed deadline (Muennighoff et al., 2025). *SEAL* applies a fixed steering vector with constant strength (Chen et al., 2025). *Budget Guidance* uses a predicted distribution over the remaining output length to guide decoding (Li et al., 2026a).

5.1 Improved accuracy and efficiency under a fixed budget

Baselines offer different trade-offs between accuracy and computation. Table 1 shows that across all settings, VCS improves accuracy over uncontrolled CoT by 3.8–21.7 percentage points, reduces token use by 13.6–34.8%, and lowers cap rates by 67.9–92.5%. Budget Prompt remains close to the CoT baseline. SEAL and Budget Forcing can occasionally achieve higher accuracy than VCS, but use more tokens and have higher cap rates wherever they do. Budget Guidance sometimes uses fewer tokens, but never achieves higher accuracy than VCS.

VCS improves accuracy while limiting disruption. We evaluate whether VCS preserves correct answers using the *disruption rate*, the fraction of prompts answered correctly under uncontrolled CoT but incorrectly with VCS, in Figure 5a. SEAL, which steers at a fixed strength, and Budget Guidance, which uses predicted remaining length, disrupt more in every setting.



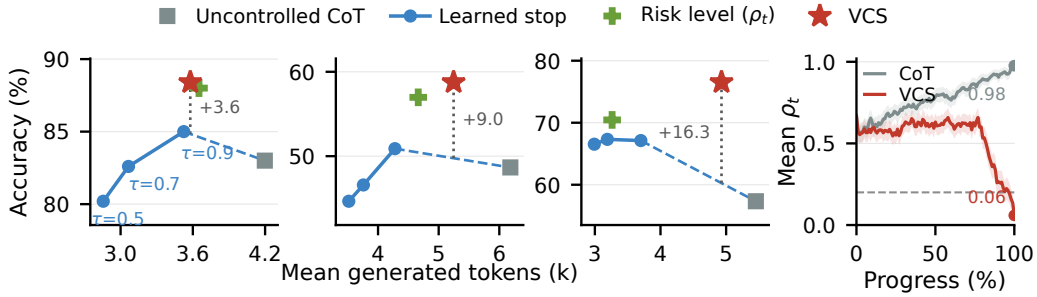
(a) Accuracy gain versus disruption (b) Component ablation as a trade-off

Figure 5: *(a)* Accuracy gain versus disruption. *(b)* Accuracy gain versus token savings relative to each model's uncontrolled CoT baseline, whose position is the origin. Both panels average over four models. Better results lie toward the lower right in (a) and upper right in (b). *No gate* steers at full strength at every control point. *Deadline* keeps the schedule alone. *Risk level* uses ρ_t alone. *Correctness* replaces the viability probe and completion direction with correctness counterparts.

VCS transfers from mathematics to code. The probes and the completion direction see only MATH training rollouts, yet we still observe considerable gains on LiveCodeBench (Figure 5a). Their training labels describe completion within budget, a property of the generation process that requires no task specification or correctness judgement. Each component shows this separately. The initial risk estimate ρ_0 predicts capped LiveCodeBench rollouts with 0.87–0.92 AUROC, outperforming the benchmark's difficulty annotation (Table 4). Reversing the MATH-trained direction brings LiveCodeBench accuracy back close to the uncontrolled baseline in all four models (Table 7).

5.2 Ablations

Control strength, direction, and prediction target each contribute to performance. Pushing at full strength at every newline saves 66% of tokens but lowers accuracy by 6.5 points on average, with model-averaged accuracy falling on every benchmark (Figure 5b). *Deadline* keeps only the schedule, while *Risk level* uses ρ_t alone. Across all 16 combinations, these variants gain 8.7 and 8.0 accuracy points over CoT, respectively, compared with 10.4 for the full gate. Risk level ρ_t saves more tokens, while the full gate has the highest model-averaged accuracy on every benchmark. Appendix D.2 further compares their outcomes on Qwen3-4B by initial risk and risk increment. The direction and prediction target also matter. Replacing the viability probe and completion direction with correctness counterparts preserves only part of the full gate's accuracy gain.



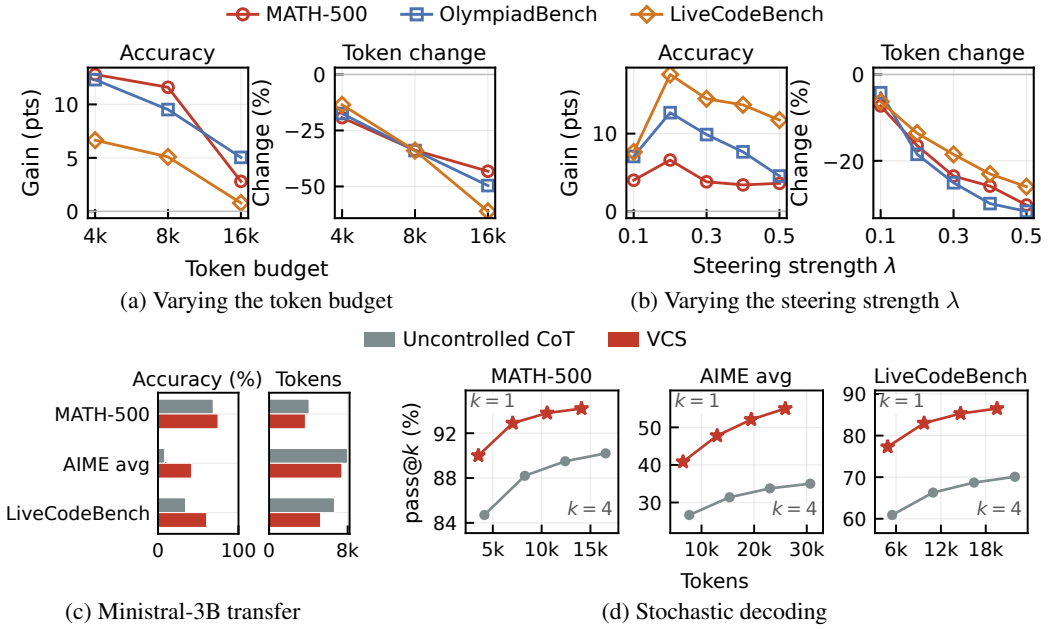
(a) MATH-500 (b) OlympiadBench (c) LiveCodeBench (d) Paired rescues

Figure 6: **Steering versus stopping on Qwen3-4B.** *Learned stop* forces the final-answer transition. Uncontrolled CoT is the $\tau \to 1$ endpoint. *Risk level* steers with strength based on ρ_t alone. Panel (d) plots mean predicted risk for rescued OlympiadBench trajectories and their uncontrolled counterparts. Rescues are correct, uncapped steered rollouts whose uncontrolled counterparts hit the cap.

Steering outperforms stopping at matched token counts. *Learned stop* uses the same probe at newline control point, forcing a transition to the final answer when $\rho_t \geq \tau$. Sweeping that threshold traces its accuracy–token curve (Figure 6). On Qwen3-4B, VCS exceeds the interpolated stopping curve by 3.6–16.3 accuracy points at equal mean token counts across MATH-500, OlympiadBench,

and LiveCodeBench. The comparison supports using the forecast to guide continued generation, giving at-risk trajectories a chance to recover before producing answers.

Steering rescues trajectories that otherwise exhaust the budget. On OlympiadBench, steering produces correct, uncapped answers on prompts whose uncontrolled rollouts hit the cap. Among the rescued trajectories analyzed in Figure 6d, 92% end with a continuous span of low predicted risk, with a median length of 586 tokens. This pattern is consistent with recovery before termination.



(a) Varying the token budget

(b) Varying the steering strength λ

(c) Ministral-3B transfer

(d) Stochastic decoding

Figure 7: **Robustness beyond the default backbone and inference setting. (a)** Budget sweep on R1-1.5B. **(b)** Steering-strength sweep on Qwen3-4B. **(c)** Raw Ministral-3B accuracy and token use on MATH-500, AIME, and LiveCodeBench. **(d)** pass@k against token cost on Qwen3-4B.

5.3 VCS EXTENDS ACROSS TASKS, BACKBONES, AND INFERENCE SETTINGS

The benefits persist across budgets and steering strengths. On R1-1.5B, increasing the budget from 4k to 16k, with both probes and the direction refitted at each budget, narrows accuracy gains from 6.7–12.8 to 0.8–5.0 points but raises token savings from 13.5–19.3% to 43.2–61.1% (Figure 7a). On Qwen3-4B, all tested strengths improve accuracy over uncontrolled CoT.

VCS remains effective across backbones, tasks, and decoding settings. On Ministral-3B, we fit the probes and completion direction on MATH rollouts and retain the default inference settings. VCS gains 6.0 accuracy points on MATH-500 while reducing cap rate and token use (Figure 7c). On GPQA Diamond (Rein et al., 2024), we reuse the probes and completion directions and obtain accuracy gains of 2.0–7.1 points across the four main backbones, again reducing cap rate and token use. Under stochastic decoding with four samples per prompt on Qwen3-4B, VCS improves pass@4 by 4.0 points on MATH-500 and 16.4 points on LiveCodeBench at lower token cost (Figure 7d).

6 CONCLUSION

Budgeted reasoning is not only about how much compute remains, but whether a particular trajectory is still viable. We show that hidden states reveal within-budget completion prospects well before the cap, and that deterioration relative to each rollout's own baseline provides a useful feedback signal without requiring a correctness predictor. VCS turns this signal into adaptive latent steering, combining trajectory-relative risk with deadline pressure and a fixed offline completion direction. Across models and tasks, this simple controller improves accuracy while reducing token use and cap rates. *More broadly, our results show that a reasoning model need not know whether it is correct to control its computation effectively: knowing whether it is likely to finish can already provide a powerful signal for test-time reasoning.*

AI USE STATEMENT

We used generative AI tools to assist with code implementation and literature review. We also used them to review the objective's derivation and assumptions, interpret diagnostic results, and improve the manuscript's organization and clarity. We have not used generative AI tools to generate synthetic datasets, propose hypotheses, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, and the remaining required-disclosure tasks are not applicable to this work. The authors take responsibility for the final content, including all text, claims, results, and code produced with AI assistance.

ETHICS STATEMENT

This work evaluates inference-time steering of existing language models on reasoning benchmarks. Steering can change an otherwise correct answer into an incorrect one, and we report this disruption alongside accuracy and efficiency (Figure 5a). Our evaluation does not assess changes in harmful outputs, bias, or privacy, so the reported gains should not be interpreted as evidence that steering preserves or improves model safety.

REPRODUCIBILITY STATEMENT

Section 4 specifies the controller and completion-direction estimator, and Appendix A gives the assumptions and derivations. Appendix B.2 documents the probe architecture, data splits, training settings, and calibration. Section 5 and Appendix C describe the models, benchmarks, decoding settings, evaluation procedures, baselines, and repetition guard. The rollout sampling and comparisons underlying the risk-increment analysis are detailed in Appendix A.4.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025. doi: 10.48550/arXiv.2503.04697. URL https://arxiv.org/abs/2503.04697.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 60770–60808. Curran Associates, Inc., 2025. doi: 10.52202/085713-2032. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/579b5b84311e122584dedab1d3b7613f-Paper-Conference.pdf.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Seyedarmin Azizi, Erfan Baghaei Potraghloo, Souvik Kundu, and Massoud Pedram. Activation steering for chain-of-thought compression. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36676–36687, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1828. URL https://aclanthology.org/2026.findings-acl.1828/.

Aryasomayajula Ram Bharadwaj. Adaptive activation steering for efficient llm reasoning via closed-loop pid control. *arXiv preprint arXiv:2506.18831*, 2025. doi: 10.48550/arXiv.2506.18831. URL https://arxiv.org/abs/2506.18831.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. INSIDE: LLMs' internal states retain the power of hallucination detection. In *International Conference on Learning Representations*, volume 2024, pp. 3056–3076,

2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/0d1986a61e30e5fa408c81216a616e20-Paper-Conference.pdf.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. SEAL: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*, 2025. doi: 10.48550/arXiv.2504.07986. URL https://arxiv.org/abs/2504.07986.

DeepSeek-AI. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Mohammed Ehab, Aymane El Gadarri, Vivek F. Farias, Adam Jozefiak, and Ciamac C. Moallemi. OS-Pruner: Pruning chains-of-thought of reasoning models via optimal stopping. *arXiv preprint arXiv:2607.11089*, 2026. doi: 10.48550/arXiv.2607.11089. URL https://arxiv.org/abs/2607.11089.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Simunic Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling LLM reasoning programs with certaindex. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 141987–142025. Curran Associates, Inc., 2025. doi: 10.52202/085713-4751. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/d037fd021c9aace128b8ce25001cdb6c-Paper-Conference.pdf.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. Mitigating overthinking in large reasoning models via manifold steering. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 102543–102568. Curran Associates, Inc., 2025. doi: 10.52202/085713-3430. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/946827aadb9f7dbdb27f4a0c77dd1859-Paper-Conference.pdf.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/94074dd5a072d28ff75a76dabed43767-Paper-Conference.pdf.

Renlong Jie, Xiaojun Meng, Lifeng Shang, Xin Jiang, and Qun Liu. Prompt-based length controlled generation with multiple control types. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1067–1085, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.63. URL https://aclanthology.org/2024.findings-acl.63/.

Jiaming Li, Lei Zhang, Yunshui Li, Ziqiang Liu, Yuelin Bai, Run Luo, Longze Chen, and Min Yang. Ruler: A model-agnostic method to control generated length for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3042–3059, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.172. URL https://aclanthology.org/2024.findings-emnlp.172/.

Junyan Li, Wenshuo Zhao, Yang Zhang, and Chuang Gan. Steering LLM thinking with budget guidance. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37437–37451, San Diego, California, United States, 2026a. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1866. URL https://aclanthology.org/2026.findings-acl.1866/.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41451–41530. Curran Associates, Inc., 2023. doi: 10.52202/075280-1797. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/81b8390039b7302c909cb769f8b6cd93-Paper-Conference.pdf.

Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. SelfBudgeter: Adaptive token allocation for efficient LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21135–21156, San Diego, California, United States, 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1063. URL https://aclanthology.org/2026.findings-acl.1063/.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: LLMs can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024. doi: 10.48550/arXiv.2410.02725. URL https://arxiv.org/abs/2410.02725.

MathArena. AIME 2026 dataset. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/MathArena/aime_2026.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Tuc Nguyen and Thai Le. ATLAS: Verifier-guided adaptive latent activation steering for efficient llm reasoning. *arXiv preprint arXiv:2601.03093*, 2026. doi: 10.48550/arXiv.2601.03093. URL https://arxiv.org/abs/2601.03093.

OpenCompass. AIME 2025 dataset. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

Bartosz Piotrowski, Witold Drzewakowski, Konrad Staniszewski, and Piotr Miłoś. Lightweight latent verifiers for efficient meta-generation strategies. *arXiv preprint arXiv:2504.16760*, 2025. doi: 10.48550/arXiv.2504.16760. URL https://arxiv.org/abs/2504.16760.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-Proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. DAST: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2322–2331, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-industry.160. URL https://aclanthology.org/2025.emnlp-industry.160/.

Leheng Sheng, An Zhang, Zijian Wu, Weixiang Zhao, Changshuo Shen, Yi Zhang, Xiang Wang, and Tat-Seng Chua. On reasoning strength planning in large reasoning models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 62978–63012. Curran Associates, Inc., 2025. doi: 10.52202/085713-2115. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5b26b9e634ba10f6c51c6db7365c4c28-Paper-Conference.pdf.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. doi: 10.48550/arXiv.2408.03314. URL https://arxiv.org/abs/2408.03314.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Budgetthinker: Empowering budget-aware llm reasoning with control tokens. *arXiv preprint arXiv:2508.17196*, 2025. doi: 10.48550/arXiv.2508.17196. URL https://arxiv.org/abs/2508.17196.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, volume 2025, pp. 55824–55845, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/8c3caae2f725c8e2a55ecd600563d172-Paper-Conference.pdf.

Yang Xiang, Yixin Ji, Ruotao Xu, Dan Qiao, Zheming Yang, Juntao Li, and Min Zhang. When is thinking enough? early exit via sufficiency assessment for efficient reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23541–23556, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1080. URL https://aclanthology.org/2026.acl-long.1080/.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025. doi: 10.48550/arXiv.2502.18600. URL https://arxiv.org/abs/2502.18600.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 88170–88210, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8df90a1440ce782d1f5607b7a38f2531-Paper-Conference.pdf.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025. doi: 10.48550/arXiv.2504.05419. URL https://arxiv.org/abs/2504.05419.

A Derivation details

Throughout this appendix, $R = 1$ marks a rollout that emits EOS within the token budget B under the base policy π_0, and $R = 0$ one that is capped, so that viability is $V(H_t, Z_t) = \Pr_{\pi_0}(R = 1 \mid H_t, Z_t)$.

Algorithm 1 summarizes the steering rule. The probes and completion directions are fitted offline and remain fixed during generation. Implementation settings are described in Appendix C.

Algorithm 1 Viability-Conditioned Steering

Require: Prompt x, token budget B, scale λ, selected layers $\mathcal{L}$
Require: Viability probe F_ϕ, budget-only probe $F_{\phi'}$, completion directions $\{d_l\}_{l\in\mathcal{L}}$ from Equation 7
Generate the first token y_1 without steering.
$y \leftarrow [y_1]$; $\rho_0 \leftarrow$ unset
while $|y| < B$ and the last token is not EOS **do**
  $t \leftarrow |y|$; $g_t \leftarrow 0$
  if t is a newline position inside the thinking segment **then**
    Obtain probe states H_t and budget features Z_t.
    $\rho_t \leftarrow 1 - F_\phi(H_t, Z_t)$
    if ρ_0 is unset **then**
      $\rho_0 \leftarrow \rho_t$
    end if
    $a_t \leftarrow \max\left\{\dfrac{\rho_t - \rho_0}{1-\rho_0},\, 0\right\}$
    $\rho_{\mathrm{dl}} \leftarrow 1 - F_{\phi'}(Z_t)$
    $g_t \leftarrow \min\{\rho_{\mathrm{dl}} + a_t,\, 1\}$
  end if
  Decode the next token, applying $h_{t,l} \leftarrow h_{t,l} + \lambda g_t d_l$ at each $l \in \mathcal{L}$.
  Append the generated token to y.
end while
return y

A.1 Construction and first-order solution of the local objective

We analyze Equation 4 at (H_t, Z_t) with Z_t fixed. Assume that V is locally differentiable and lies in $(0, 1)$, and write $f = \operatorname{logit} V$. Using the base-policy viability function at perturbed states is a local modeling assumption. This does not guarantee that viability predictions remain calibrated under intervention.

Setting the cost. The net gain $S_t = r\rho_t - cV_t = (r + c)\rho_t - c$ is positive above $\rho^* = c/(r + c)$. For $V_0 \in (0, 1)$, requiring no risk-based correction initially gives only $S_0 \le 0$, or $c \ge r\rho_0/V_0$. Requiring correction for every increase above initial risk also sets $\rho^* = \rho_0$, hence $c = r\rho_0/V_0$. Substitution gives $S_t = r(\rho_t - \rho_0)/(1 - \rho_0)$ as in Equation 3. Within this model, the threshold fixes the normalization when $r > 0$ is shared across trajectories.

From net gain to a local update. For $L(V) = r \log V + c \log(1 - V)$, holding r and c fixed,

$$\frac{\partial L}{\partial f} = \left(\frac{r}{V} - \frac{c}{1-V}\right) V(1-V) = r(1-V) - cV. \tag{8}$$

The logarithms cancel the sigmoid factor, so the derivative at the current state equals S_t. With $q_t = \nabla_H f(H_t, Z_t)$, the hidden-state gradient is $\nabla_H L = S_t q_t$. Linearizing L while retaining the quadratic penalty gives, up to a constant,

$$J_{\mathrm{lin}}(\delta) = S_t q_t^\top \delta - \frac{1}{2\eta}\|\delta\|_2^2, \qquad \delta^* = \eta S_t q_t. \tag{9}$$

For $\eta > 0$, this quadratic has the unique maximizer shown above, which is the signed update in Equation 5. Setting $c = 0$ gives $S_t = r\rho_t$, the risk-level alternative.

The gate. To avoid reversing the local completion direction when $S_t < 0$, restrict the linearized update to $\delta = aq_t$ with $a \ge 0$. For $q_t \ne 0$, the optimum is $a^* = \eta \max\{S_t, 0\}$. If $q_t = 0$, the update is zero. This gives the positive relative increment used in Equation 6. The budget-only probe adds a shared estimate of cap risk at the current budget position, including when $\rho_t \le \rho_0$. Adding this schedule and clipping the sum at one are design choices beyond the local solution.

The fixed direction. Replacing q_t with a fixed direction is an approximation. For a nonzero fixed vector $u = \lambda(d_l)_{l\in\mathcal{L}}$ and $\delta = au$ with $a \ge 0$, the same linearized objective instead gives

$$a^* = \eta \max\left\{S_t \frac{q_t^\top u}{\|u\|_2^2},\, 0\right\}. \tag{10}$$

The projection factor depends on the current state. VCS uses a global scale and the gate rather than solving this optimization at each control point. Under the assumptions in Appendix A.2, the fixed direction is aligned with the gradient on average. This need not hold at every state.

A.2 Alignment of the completion direction

We give sufficient conditions for alignment with the joint-state gradient $\nabla_H f$ used in Equation 5. Treat $H = (h_l)_{l\in\mathcal{L}}$ as the concatenated residual states. At fixed budget features $z = Z_t$, assume that $H \mid R = r, Z = z$ is Gaussian with mean $\mu_r(z)$ and covariance $\Sigma(z) = \mathrm{diag}(\Sigma_l(z))_{l\in\mathcal{L}}$, shared by finished and capped rollouts, with each block positive definite. The block-diagonal assumption makes the layers conditionally independent given the outcome and budget features. Let $\Delta_l(z) = \mu_{1,l}(z) - \mu_{0,l}(z)$. Applying Bayes' rule to the joint Gaussian densities and differentiating gives

$$w_l(z) = \nabla_{h_l} f(H, z) = \Sigma_l(z)^{-1}\Delta_l(z). \tag{11}$$

Positive definiteness gives $\Delta_l(z)^\top w_l(z) > 0$ whenever $\Delta_l(z) \neq 0$. Under these assumptions, each mean difference aligns with the corresponding block of the joint gradient.

VCS uses one direction per layer across all positions. Let q be the distribution of generation positions shared by the matched finished and capped states, so that $\Delta_l = \mathbb{E}_{z\sim q}[\Delta_l(z)]$. If the covariance is also constant across these positions, $\Sigma_l(z) = \Sigma_l$, then $\mathbb{E}_{z\sim q}[w_l(z)] = \Sigma_l^{-1}\Delta_l$. For nonzero Δ_l at each selected layer and $s_l > 0$, the scaled direction $d = (d_l)_{l\in\mathcal{L}}$ in Equation 7 therefore satisfies

$$d^\top \mathbb{E}_{z\sim q}[\nabla_H f(H, z)] = \sum_{l\in\mathcal{L}} \frac{s_l}{\|\Delta_l\|_2}\Delta_l^\top \Sigma_l^{-1}\Delta_l > 0. \tag{12}$$

This is a population alignment result under the stated assumptions. It does not guarantee alignment at every position, for finite-sample estimates, or under intervention. With cross-layer covariance, each block of the joint gradient also depends on mean differences at other layers, so separate per-layer Gaussian assumptions alone are insufficient. Position-dependent covariance likewise need not preserve the pooled alignment.

Estimation. We split rollouts by prompt and estimate the direction on the training split. Each residual state at a newline inside the thinking segment is labeled by its rollout's finished-or-capped outcome. We match states from the two outcomes by the number of generated tokens, then compute the class means in Equation 7.

A.3 Dependence on initial risk and budget

The relative increment remains associated with initial risk. Across six model–benchmark pairs at $t = 2048$, its correlation with ρ_0 ranges from 0.21 to 0.56, compared with $|r| \leq 0.30$ for the logit difference $\mathrm{logit}\,\rho_t - \mathrm{logit}\,\rho_0$. For the same positive risk increment, dividing by $1 - \rho_0$ gives a larger contribution when initial risk is higher. This normalization follows from the benefit–cost model and threshold in Appendix A.1. It measures the share of initial viability lost, but does not remove all dependence on initial risk.

A shrinking budget can affect both the relative increment and the deadline schedule $\rho_{\mathrm{dl}}(Z_t)$. The gate retains both contributions. Subtracting a shared budget component from the increment could change when it becomes positive and when the gate reaches one, so its effect would not be a simple rescaling of intervention strength.

A.4 Same-prompt continuations behind Figure 4

Sampling and initial risk. We sample $K = 6$ base-policy continuations for each of the 500 MATH-500 questions at temperature 1.0, top-p 0.95, and top-k 20, with an 8192-token budget and the same prompt template as in evaluation. We recompute risk along the saved continuations at the newline positions used for intervention.

For Qwen3, Figure 4b,c and the within-question ranking use the second control point as the initial-risk reference. R1 uses the first control point throughout, as do both models in Figure 4a and Figure 8a. Table 3 reports both Qwen3 readings.

Groups in Figure 4b,c. Among rollouts still generating at 4096 tokens, we select those in the top initial-risk quartile that later finish and those in the bottom quartile that later cap. All have a risk reading by 1024 tokens, so group membership stays fixed across plotted positions. Table 2 gives the sample counts and mean risks. Using Qwen3's deployed first control point instead gives mean current risks of 0.430 and 0.462 and increments of −0.295 and 0.338. The same contrast holds. These groups are selected by outcome and do not measure intervention effects.

Table 2: Groups in Figure 4b,c at 4096 tokens. Q4 and Q1 denote the top and bottom initial-risk quartiles. Risk values are means over rollouts.

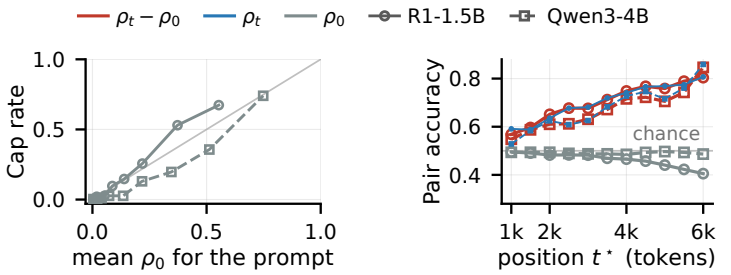
Model	Group	Questions	Rollouts	ρ_t	$\rho_t - \rho_0$
R1-1.5B	Q4, later finishes	43	78	0.382	−0.206
	Q1, later caps	50	68	0.403	0.321
Qwen3-4B	Q4, later finishes	45	102	0.374	−0.248
	Q1, later caps	18	31	0.401	0.356

Associations across questions. Table 3 summarizes the 500×6 continuations. For rollout risks r_{qk}, question means $\bar{r}_q$, and overall mean $\bar{r}$, the between-question variance share is SS_B/SS_T, with $SS_B = 6\sum_q(\bar{r}_q - \bar{r})^2$ and $SS_T = \sum_{q,k}(r_{qk} - \bar{r})^2$. The categorical effect size η^2 is the fraction of variance in question means explained by MATH difficulty groups. The two linear R^2 values use numeric difficulty levels 1–5 or empirical cap rates from the six continuations, with an intercept and all 500 unbinned question means. They describe in-sample associations, not held-out prediction or probability calibration. Cap rates are not corrected for sampling noise.

Table 3: Effect sizes from the same 500×6 continuations as Figure 4. Index 0 denotes the first control point and index 1 the second. η^2 uses categorical MATH labels. R^2_{label} relates numeric labels to per-question mean ρ_0, and R^2_{cap} relates that mean to empirical cap rate.

Model	Index	SS_B/SS_T	η^2	R^2_{label}	R^2_{cap}
R1-1.5B	0	0.9191	0.229	0.207	0.569
Qwen3-4B	0	0.9999	0.273	0.254	0.508
Qwen3-4B	1	0.9434	0.219	0.197	0.520

Figure 8a shows that Qwen3's initial risk overpredicts capping in the middle of the range. Its probe was trained on MATH levels 3–5, where capping is more common than in MATH-500 overall.



(a) ρ_0 and cap rate (b) Within-question ranking

Figure 8: **Initial risk is associated with capping across questions, and increments distinguish outcomes within a question.** *(a)* Per-question mean ρ_0 against empirical cap rate, averaged within deciles of mean ρ_0. The diagonal marks equality. *(b)* Fraction of pairs from the same question in which a signal ranks a capped rollout above a finished one, among rollouts still generating at $t^\star$.

Ranking within questions. Figure 8b compares capped and finished continuations of the same question that are still generating at $t^\star$. We use the latest risk reading at or before that position. At 1024, 2048, and 4096 tokens, the risk increment and current risk give similar rankings. Their concordance scores differ by at most 0.022 across the two models. This test does not establish an advantage

for the increment or assess its normalization. For Qwen3, using the deployed token-2 reading instead makes ρ_0 concordance fall to 0.271–0.327. Those rankings depend on tiny differences before the continuations diverge.

Human difficulty labels. Table 4 uses greedy evaluation rollouts, separately from the $K = 6$ samples. Initial risk correlates with the benchmark's difficulty labels and predicts capping more accurately in all six model–benchmark pairs.

Table 4: ρ_0 against the benchmark's own difficulty annotation, on the greedy evaluation rollouts. AUROC is for predicting that the greedy run hits the 8192-token cap; the label is scored the same way.

Model	Benchmark	Spearman(ρ_0, label)	AUROC ρ_0	AUROC label
R1-1.5B	MATH-500	$+0.51$	0.829	0.687
	LiveCodeBench	$+0.74$	0.871	0.812
Qwen3-4B	MATH-500	$+0.58$	0.910	0.744
	LiveCodeBench	$+0.83$	0.924	0.846
Qwen3-8B	MATH-500	$+0.58$	0.909	0.725
	LiveCodeBench	$+0.82$	0.910	0.828

B Diagnosis and probe details

Viability versus correctness, by position. Figure 3b,c resolve the two prediction targets over position on Qwen3-4B. At matched absolute position the viability probe is at 0.91 in the first 0–512-token bin and rises to 0.97–0.98 beyond $1{,}500$ tokens; finished-only correctness peaks at 0.81 and falls to about 0.70 beyond $4{,}000$ tokens. The two targets have very different base rates, capping being near-balanced while wrong finishers are a $\sim 2\%$ minority (Appendix B.3), so the gap says nothing about which target is intrinsically easier. In panels (b) and (c) finish uses the probe trained on all sampled positions and correctness the augmented-label probe; the correctness bands are wider because wrong finishers become sparse at late positions.

Comparability with Zhang et al. (2025). They predict the correctness of an upcoming intermediate answer from judged answer- or paragraph-level states within the reasoning chunk that leads to it; we predict rollout termination, and the final correctness of a naturally finished rollout, from generic newline states. The targets and readout positions differ, so the numbers are not comparable, and the late decline of our correctness probe in Figure 3b does not contradict their result. Figure 3c shows the qualitative parallel: indexed by relative progress t/L, our correctness signal also rises near completion. For budgeted control the absolute position is the relevant index, since it fixes the remaining budget while L is unknown online, and there finish prediction remains high as final correctness becomes harder to read.

B.1 Across-question risk curves

Figure 9 reports the across-question measurement on Qwen3-4B greedy OlympiadBench trajectories. At each absolute position it compares the AUROC of ρ_t with that of its raw rise $\rho_t - \rho_0$ among trajectories still generating. The full forecast is already predictive in the first bin, while the rise gains discrimination later. This is not a within-question ranking; Figure 8b makes that comparison.

Figure 10 extends this measurement to Qwen3-8B and R1-1.5B, with risk recomputed teacher-forced along saved greedy trajectories by each model's frozen probe. These AUROCs pool different questions and therefore also reflect question difficulty. At the first control point the probe discriminates eventual capping at 0.92 and 0.91 AUROC on Qwen3-8B and at 0.84 and 0.83 on R1-1.5B. The trajectory term sits between 0.42 and 0.60 in the first bin across the four panels, with bootstrap intervals that exclude chance from above on OlympiadBench for Qwen3-8B and from below on MATH-500 for R1-1.5B, and rises to 0.76–0.85 beyond $4{,}000$ tokens. R1-1.5B reaches its first newline at a median of 70 generated tokens against 2 for the Qwen3 models, so the first reading is not uniformly a pre-generation quantity.

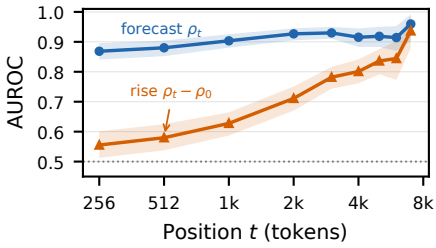


Figure 9: Across-question cap-risk AUROC on Qwen3-4B (OlympiadBench) for the forecast ρ_t and its raw rise $\rho_t - \rho_0$, among greedy trajectories still generating at each position. Bands are 95% bootstrap intervals over trajectories.

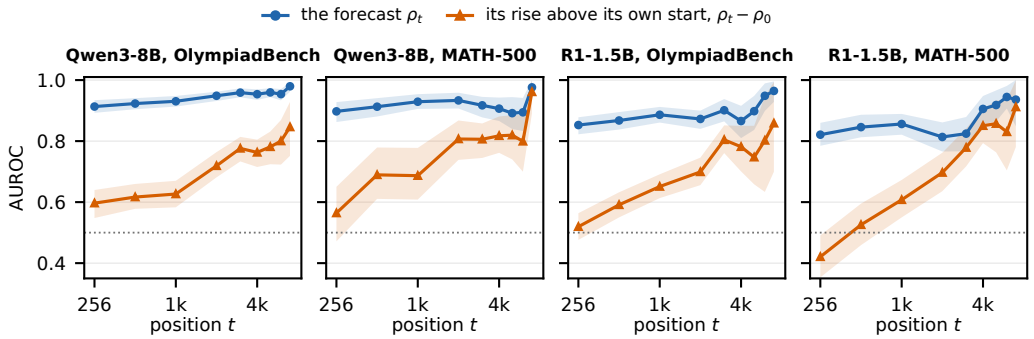


Figure 10: Across-question risk curves on Qwen3-8B and R1-1.5B. Among the greedy trajectories still generating at t, AUROC against eventual capping of the forecast ρ_t and of its raw rise $\rho_t - \rho_0$. Bands are 95% bootstrap intervals over trajectories.

B.2 Probe Training Details

Probe and inputs. The viability, budget-only and correctness probes share one architecture and differ only in target and eligible rollouts. VCS uses the same viability probe the diagnosis evaluates; no separate model is trained for deployment. At a control point we read the residual state $h_{t,l}$ at each layer of a per-model band $\mathcal{L}$, apply a parameter-free LayerNorm, average across the band to $\bar{h}_t \in \mathbb{R}^d$, and append the budget features $Z_t = \big(t/B,\ (B-t)/B,\ \log(B-t+1)\big)$ with $B = 8192$. F_ϕ is a two-layer MLP, $\mathrm{Linear}(d{+}3 \to 512) \to \mathrm{GELU} \to \mathrm{Dropout}(0.1) \to \mathrm{Linear}(512 \to 1)$, read through a sigmoid as viability; the budget-only probe $F_{\phi'}$ is the same architecture trained on the same states with $\bar{h}_t$ withheld. Activations are extracted with one causal forward per rollout at every newline inside the thinking segment, the candidate intervention locations. We choose a band of four consecutive layers in the middle to later part of each model, motivated by evidence that reasoning steering is effective at these depths (Chen et al., 2025). The bands remain fixed across benchmarks. Table 5 lists the band and width per model, with the matched-position discrimination of the probes.

Table 5: Controlled band $\mathcal{L}$, width d, and matched-position AUROC (matched-t, window 64) of the viability and finished-only correctness probes, with the budget-only probe as a baseline. Matching removes the pooled position confound of Figure 3a, collapsing that baseline to chance. The correctness column is the standard finished-only probe; the denser augmented-label variant of Appendix B.3 backs the per-position curve in Figure 3b.

Model	Band $\mathcal{L}$	d	Viability	Correctness	Budget-only
Qwen3-4B	22–25	2560	.958	.650	.500
Qwen3-8B	22–25	4096	.968	.763	.500
R1-1.5B	17–20	1536	.921	.727	.498
R1-14B	30–33	5120	.929	.655	.498

Splits, optimization and calibration. Rollouts are split by problem identifier, $70/15/15$, with a fixed seed, so no problem crosses splits and per-problem difficulty cannot leak from train to test. We train with AdamW (lr $= 10^{-4}$, weight decay 10^{-4}), batch 256, for at most 10 epochs, early-stopping on validation AUROC with patience 3, under plain binary cross-entropy: the gate reads ρ_t and ρ_0 as probabilities, so we keep the natural class prior rather than rebalance the loss. After training we fit a single temperature on the validation split; for the Qwen3-4B viability probe this reduces test ECE from 0.023 to 0.010 at $T = 1.33$.

B.3 Correctness probe analyses

The correctness probe is trained only on rollouts that finish before the cap, with the evaluated final answer as label; including capped rollouts would turn the label into a near-copy of the cap indicator, since under a budget most wrong answers come from trajectories that never finish. Wrong finishers are a 2–3% minority, so the $1{,}200$-problem set used for the viability probe leaves too few of them for per-position estimates. For the correctness curves we therefore relabel the whole level-3–5 pool ($4{,}677$ problems, K=6, $28{,}062$ rollouts), which yields 570 distinct wrong finishers, 69 of them in the held-out test split against a 0.974 finished-correct rate. The negatives remain few and concentrated in a few hundred trajectories, which is why the correctness bands in Figure 3b widen with position: beyond about $6{,}000$ tokens fewer than 30 distinct wrong finishers survive, and none reach the cap, since finishing means stopping before it.

The augmented correctness probe is used only for the position-resolved diagnostic curves in Figure 3b,c. The correctness arm in Table 7 instead uses a separate probe and orthogonalized direction fitted on naturally finished rollouts from the same $1{,}200$ training prompts used for the viability artifacts. Thus, the two analyses use different correctness-training pools.

Matched-position AUROC under the problem-identifier split shows discrimination across questions, not within them: a within-question test would rank a finished-correct against a finished-wrong rollout of the same problem at the same budget position, and almost no problem contributes both at this scale. The early correctness signal may therefore partly reflect question difficulty.

C Evaluation protocol

Benchmarks and judges. MATH-500, AIME, and OlympiadBench are scored by exact match of the final boxed answer against the reference; AIME pools the 2025 and 2026 sets ($n = 60$), and OlympiadBench uses the 674 problems valid in every run. LiveCodeBench ($n = 511$) is scored with the official checker for every arm, including the paired disruption computations. The cap rate counts trajectories that reach the 8192-token limit without emitting end-of-sequence; correctness is judged on whatever answer the trajectory contains, so a capped trajectory can be correct.

Intervention settings. All VCS rows use the steering update in Equation 1 with $\lambda = 0.2$, applied only at newlines inside the thinking segment and subject to the repetition guard below. The probes and the direction are trained once per model on rollouts from $1{,}200$ MATH training prompts (levels 3–5), disjoint from the evaluation sets, and reused unchanged on every benchmark. Disruption is computed against prompt-paired greedy reruns from the same pipeline.

Repetition guard. We use a repetition guard to limit persistent repetition in steered runs. The guard suspends steering when repetition is detected and stops generation if it persists. Table 6 reports EOS completion, guard stops, and budget exhaustion for Qwen3-4B across all four benchmarks. Guard stops are counted separately from budget exhaustion. Even when both are counted as unfinished, the resulting rates remain below the reported CoT cap rates across all four benchmarks.

Baselines. *Budget Prompt* states the 8192-token budget in the system prompt and otherwise decodes greedily. *Budget Guidance* uses our native remaining-length predictor, trained for three epochs, for Qwen3-4B, R1-1.5B and R1-14B, and the authors' released predictor for Qwen3-8B. Qwen3-4B uses the released gate threshold; R1-1.5B and R1-14B use a lower one selected on a calibration sweep. All are evaluated under our common prompts, budgets, generation limit and judges. *Budget Forcing* decodes greedily and, once 512 tokens of budget remain, closes the thinking segment and appends the cue `Final Answer:`.

Table 6: **Termination outcomes for Qwen3-4B.** All values are percentages of evaluated questions. EOS, Guard, and Cap denote EOS completion, repetition-guard stopping without EOS, and budget exhaustion without EOS, respectively. The categories are mutually exclusive and sum to 100% before rounding.

Benchmark	n	EOS	Guard	Cap	Guard + Cap	CoT Cap
MATH-500	500	93.8	2.8	3.4	6.2	17.0
AIME	60	66.7	10.0	23.3	33.3	81.7
OlympiadBench	674	81.9	9.3	8.8	18.1	47.5
LiveCodeBench	511	84.9	2.2	12.9	15.1	42.1

D Additional results

D.1 Per-model component ablation

Table 7 gives the per-model accuracy, cap rate, and token counts behind Figure 5b, which plots each arm relative to its own model's greedy run and then averages over the four models.

Table 7: **Component ablation.** Every ablation arm shares the model, layers, newline control points, guards, $\lambda = 0.2$, and the 8192-token budget, and alters ours in one component. *No gate* steers at full strength at every newline. *Deadline* keeps only the schedule, using the budget-only probe. *Risk level* uses current risk rather than the relative increment, corresponding to $c = 0$ in Equation 4. *Reverse direction* keeps the gate and negates d_l. *Correctness* ρ_t gates on a probe of answer correctness and steers along a correctness direction orthogonalised against d_l, so the two signals cannot leak into each other. LiveCodeBench is scored with the official checker. Best value per column within each model block in bold.

Arm	MATH-500			AIME			OlympiadBench			LiveCodeBench		
	Acc.↑	Cap↓	Tok.↓	Acc.↑	Cap↓	Tok.↓	Acc.↑	Cap↓	Tok.↓	Acc.↑	Cap↓	Tok.↓
Qwen3-4B												
CoT	83.2	17.0	4209	21.7	81.7	7749	46.6	47.5	6308	58.9	42.1	5629
No gate ($g_t = 1$)	79.2	**0.6**	**1154**	15.0	**1.7**	**2175**	47.9	**5.0**	**1943**	47.2	5.9	**1744**
Deadline (ρ_{dl})	88.4	10.4	3810	40.0	63.3	7561	56.5	35.8	5943	73.0	21.1	5026
Risk level (ρ_t)	89.0	1.6	3558	36.7	10.0	4660	55.8	9.8	4713	74.0	**4.7**	3250
Reverse direction ($-d_l$)	83.0	14.6	4081	25.0	70.0	7567	48.5	46.0	6223	58.3	38.0	5405
Correctness ρ_t (orth.)	85.2	16.0	4168	25.0	78.3	7673	48.1	45.5	6207	59.3	36.8	5519
VCS (ours)	**89.8**	3.4	3516	**43.3**	23.3	6358	**59.2**	8.8	5142	**76.5**	12.9	4864
Qwen3-8B												
CoT	84.4	16.0	4338	21.7	85.0	7769	46.7	50.4	6443	57.9	41.1	5735
No gate ($g_t = 1$)	74.0	**1.2**	**793**	11.7	**6.7**	**1918**	40.9	**6.1**	**1487**	33.7	**2.3**	**1236**
Deadline (ρ_{dl})	87.6	10.2	4012	33.3	63.3	7463	56.1	31.8	6013	74.8	18.4	5192
Risk level (ρ_t)	89.0	2.6	3793	30.0	10.0	4635	56.1	8.0	4538	73.0	4.1	4160
Reverse direction ($-d_l$)	82.8	17.2	4337	21.7	80.0	7654	48.1	50.0	6437	58.5	38.4	5674
Correctness ρ_t (orth.)	82.8	18.4	4381	21.7	76.7	7682	47.3	49.9	6419	59.1	36.4	5565
VCS (ours)	**90.2**	2.4	3541	**38.3**	25.0	6272	**57.3**	14.4	5212	**76.7**	7.4	4635
DeepSeek-R1-Distill-Qwen-1.5B												
CoT	66.8	32.0	4133	10.0	88.3	7618	29.8	62.2	6251	18.0	77.5	6861
No gate ($g_t = 1$)	71.4	9.4	**1840**	13.3	20.0	**3801**	35.6	13.6	**2802**	16.4	14.5	**2243**
Deadline (ρ_{dl})	77.0	**4.6**	2825	18.3	28.3	6034	36.9	13.1	4625	23.3	11.7	4328
Risk level (ρ_t)	75.4	**4.6**	2996	20.0	**15.0**	4850	38.1	10.7	4318	**24.9**	11.9	3082
Reverse direction ($-d_l$)	72.0	12.6	3358	16.7	38.3	5964	31.3	32.6	5235	18.2	15.9	4284
Correctness ρ_t (orth.)	72.8	13.2	3323	**21.7**	53.3	6633	34.4	32.6	5174	22.3	29.4	5343
VCS (ours)	**78.4**	5.8	2735	18.3	28.3	5495	**39.3**	**9.3**	4128	23.1	**11.0**	4530
DeepSeek-R1-Distill-Qwen-14B												
CoT	85.8	10.8	3109	26.7	68.3	7127	50.3	37.5	5196	58.7	39.1	5486
No gate ($g_t = 1$)	78.8	3.6	**1546**	18.3	18.3	**3459**	44.8	8.0	**2321**	35.6	**2.2**	**1490**
Deadline (ρ_{dl})	**90.0**	2.0	2591	30.0	21.7	5897	55.0	9.3	4232	**65.3**	5.1	4459
Risk level (ρ_t)	89.0	1.8	2644	26.7	**10.0**	4587	54.5	**5.9**	3801	63.2	2.9	3020
Reverse direction ($-d_l$)	86.2	9.0	3050	30.0	58.3	6936	50.7	34.0	5094	58.1	24.5	5088
Correctness ρ_t (orth.)	86.4	10.4	3063	33.3	65.0	6936	53.3	32.9	5098	56.4	30.1	5195
VCS (ours)	89.6	**1.0**	2425	**35.0**	15.0	4849	**55.5**	**5.9**	3754	63.0	2.9	3578

D.2 Intervention outcomes by initial risk and risk increment

We compare VCS with risk-level steering on Qwen3-4B and OlympiadBench, pairing outcomes by question. Uncontrolled risk traces are available for 670 of 674 questions; 653 uncontrolled rollouts are still generating at 2048 tokens. We group these questions by initial risk (split at the median, 0.5667) and whether $\rho_t - \rho_0 > 0$, using the latest risk reading at or before 2048 from replay of the uncontrolled text. Both methods use the same question groups.

Table 8: Paired outcomes within groups defined by uncontrolled risk. Disruption measures the percentage of answers changed from correct to incorrect, among questions answered correctly without intervention. Cap-to-correct measures the percentage answered correctly among questions whose uncontrolled rollouts hit the cap. N is the total number of questions in each group.

Initial risk	Risk change	N	Disruption ↓		Cap-to-correct ↑	
			Risk level	VCS	Risk level	VCS
Low	≤ 0	246	2.6%	3.1%	50.0%	80.0%
Low	> 0	81	5.6%	5.6%	47.5%	55.0%
High	≤ 0	181	12.7%	5.5%	25.8%	32.3%
High	> 0	145	37.5%	29.2%	23.8%	22.1%

VCS disrupts fewer correct answers in the high-initial-risk, non-increasing-risk group and answers more previously capped questions correctly in the low-initial-risk, increasing-risk group (Table 8). These differences are small (exact paired $p = 0.125$ and 0.375, respectively), though their directions persist at 1024 and 4096 tokens. In the former group, VCS also uses more tokens (6432 versus 4942) and caps more often (16.0% versus 12.2%).

These exploratory results do not establish the proposed mechanism. The methods generate different trajectories from the start, and their gates differ in both the risk reference and the deadline schedule. The observed benefits are also not confined to the two highlighted groups.

D.3 Transfer to Ministral-3B

We additionally apply VCS to `mistralai/Ministral-3-3B-Reasoning-2512`, whose text decoder has 26 layers. The controlled band is layers 15–18 and the finish probe is trained from 300×6 sampled MATH training rollouts. Its held-out AUROC is 0.945. We otherwise retain the 8192-token budget, newline-only intervention, and $\lambda = 0.2$ used in the main experiments. Figure 7c shows the measured MATH-500, AIME, and LiveCodeBench comparisons. Table 9 reports these results together with OlympiadBench results. AIME pools the 2025 and 2026 sets ($n = 60$).

Table 9: Ministral-3B transfer results. Accuracy and cap rate are percentages; Tok. is the mean number of generated tokens.

Benchmark	Arm	Acc. ↑	Cap ↓	Tok. ↓
MATH-500	Greedy	68.6	25.4	4076
	VCS ($\lambda = 0.2$)	74.6	11.0	3697
AIME 2025+2026	Greedy	8.3	90.0	8007
	VCS ($\lambda = 0.2$)	41.7	65.0	7413
OlympiadBench	Greedy	25.2	59.3	6490
	VCS ($\lambda = 0.2$)	37.1	34.6	5809
LiveCodeBench	Greedy	34.2	60.7	6658
	VCS ($\lambda = 0.2$)	60.5	27.0	5242

D.4 Transfer to GPQA

We evaluate the same MATH-trained probes and completion directions on GPQA Diamond (Rein et al., 2024) ($n = 198$), a multiple-choice benchmark of graduate-level biology, physics, and chemistry questions. We retain the 8192-token budget, newline-only intervention, and $\lambda = 0.2$. Table 10

shows that VCS improves accuracy for all four backbones while reducing both cap rate and mean generated tokens. The accuracy gain ranges from 2.0 to 7.1 points, cap rate falls by 5.6–53.0 points, and token use falls by 8.4%–33.6%.

Table 10: Transfer to GPQA Diamond ($n = 198$). Accuracy and cap rate are percentages; Tok. is the mean number of generated tokens.

Model	Arm	Acc.↑	Cap↓	Tok.↓
Qwen3-4B	CoT	45.5	49.0	6277
	VCS ($\lambda = 0.2$)	51.0	7.6	5131
Qwen3-8B	CoT	52.0	57.6	6731
	VCS ($\lambda = 0.2$)	57.6	4.5	5190
R1-Distill-Qwen-1.5B	CoT	15.7	29.8	5023
	VCS ($\lambda = 0.2$)	22.7	24.2	4599
R1-Distill-Qwen-14B	CoT	53.5	25.8	5278
	VCS ($\lambda = 0.2$)	55.6	4.5	3507

D.5 Under sampling rather than greedy decoding

The main results in Table 1 use greedy decoding. Because a deployed reasoning model usually samples, we repeat the comparison on Qwen3-4B at Qwen3's own recommended setting, temperature 0.6 with top-p 0.95 and top-k 20, drawing $k = 4$ continuations per prompt. Both arms are generated at that temperature. The gate keeps $\lambda = 0.2$ and every other setting of Table 1.

The comparison covers MATH-500, AIME, and LiveCodeBench, as shown in Figure 7d. AIME pools the 2025 and 2026 sets ($n = 60$), giving 240 sampled continuations per arm.

Table 11 reports both arms at every k. Per sample, VCS gains 5.4 points on MATH-500 and 16.4 on LiveCodeBench, against 6.6 and 17.6 under greedy decoding: the advantage is not an artifact of the arg-max policy. Paired by prompt, VCS is ahead on 68 prompts and behind on 17 for MATH-500, and ahead on 172 against 11 for LiveCodeBench. On AIME, VCS improves pass@4 from 35.0% to 55.0% while reducing token use by 15.5%.

Table 11: Sampling at temperature 0.6 on Qwen3-4B, $k = 4$ continuations per prompt. pass@k is the unbiased estimator over the four draws; tokens are the cumulative generated tokens for those k samples. LiveCodeBench is scored by the official checker.

	Base policy		VCS		Difference	
k	pass@k	tokens	pass@k	tokens	acc.	tokens
MATH-500						
1	84.7	4,139	90.0	3,521	+5.4	−14.9%
2	88.2	8,278	92.9	7,042	+4.7	−14.9%
3	89.5	12,417	93.8	10,563	+4.3	−14.9%
4	90.2	16,555	94.2	14,084	+4.0	−14.9%
AIME 2025+2026						
1	26.7	7,668	40.8	6,481	+14.2	−15.5%
2	31.4	15,336	47.8	12,963	+16.4	−15.5%
3	33.8	23,004	52.1	19,444	+18.3	−15.5%
4	35.0	30,672	55.0	25,925	+20.0	−15.5%
LiveCodeBench						
1	60.9	5,477	77.3	4,877	+16.4	−10.9%
2	66.3	10,954	83.0	9,755	+16.7	−10.9%
3	68.7	16,431	85.3	14,632	+16.6	−10.9%
4	70.1	21,908	86.5	19,509	+16.4	−10.9%

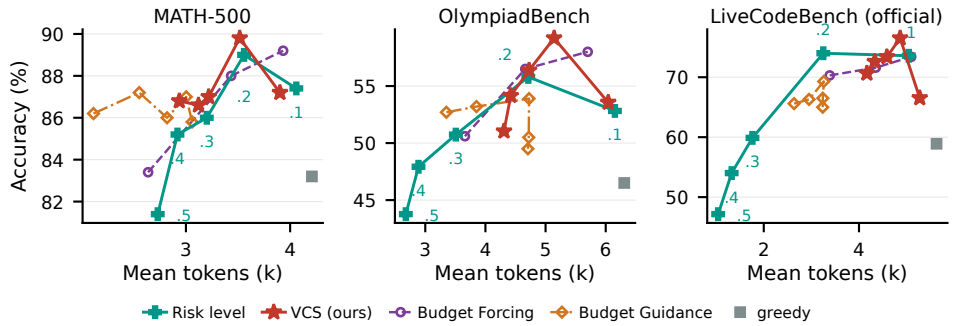


Figure 11: Accuracy against mean generated tokens on Qwen3-4B as the steering strength varies over $\lambda \in \{0.1, \ldots, 0.5\}$ along each solid curve, for VCS and the risk-level gate; the point labels are carried by the risk-level curve, whose points spread across the whole token range. Each gate is swept separately here, whereas all arms of Table 7 share $\lambda = 0.2$. The purple dashed curve is Budget Forcing at thinking deadlines of 2048, 4096, and 6144 tokens; the orange dash-dotted curve is Budget Guidance over five target budgets. For Budget Guidance, the target-4096 MATH-500, target-6000 OlympiadBench, and target-5000 LiveCodeBench points are the published batch-1 results; the remaining points use batch 8. LiveCodeBench accuracy is scored by the official checker throughout.

D.6 Computational efficiency

Token count is only a proxy for compute. At each control point, VCS evaluates both the viability probe and the budget-only probe on the selected-layer states, then applies residual updates at four layers. Because these operations are not reflected in token counts, we directly measure their overhead. Table 12 reports back-to-back measurements of both methods on a single RTX 3090, using batch size 1 and greedy decoding on the first 20 MATH-500 problems. We reset the peak-memory counter before each run.

VCS reduces decoding throughput by only 2.7% and adds 0.40 GiB of peak memory (4.9%), while reducing generated tokens by 10% on the same problems. This reduction translates into a 7.2% decrease in end-to-end latency, with VCS finishing faster on 13 of the 20 problems. Thus, the measured runtime savings remain substantial after accounting for the intervention overhead, confirming that the token reduction is not merely an accounting artifact.

Table 12: Runtime overhead of VCS. Qwen3-4B, batch 1, greedy decoding, one RTX 3090, 20 MATH-500 problems, both arms timed back to back on the same card. Peak memory is `torch.cuda.max_memory_allocated`.

	Seconds	Tokens	Tokens/s	Peak memory	Cap rate
Greedy	199.4	4,362	22.71	8.19 GiB	20%
VCS	185.1	3,942	22.10	8.59 GiB	5%
Ratio	0.93×	0.90×	0.973×	1.049×	

D.7 Strength and deadline sweeps

Figure 11 sweeps the steering strength λ for both VCS and the risk-level gate from §5.2, following the protocol of Table 7. LiveCodeBench is evaluated using its official checker, and the $\lambda = 0.2$ points correspond to the results reported in Tables 1 and 7. As λ increases, each gate traces a distinct accuracy–token trade-off curve. The main-text comparison therefore represents one operating point on each curve, rather than two isolated configurations.

D.8 QUALITATIVE EVALUATION

We select three cases that make the contrast between the policies explicit. In each case, uncontrolled CoT exhausts the 8192-token budget without a validated answer, learned stopping returns a wrong answer, and the deployed VCS steering update from Equation 1 completes the missing reasoning and closes the thinking segment on its own. The boxes contain condensed excerpts from the saved outputs. Mathematical symbols are normalized for typesetting, and `[...]` indicates omitted text.

Example 1 (MATH-500 #390; answer 12). *Find the smallest distance from the origin to the graph* $y = (x^2 - 3)/\sqrt{2}$*, write it as* $\sqrt{a}/b$*, and compute* $a + b$*.* Uncontrolled CoT repeatedly obtains two derivatives for the same expression and caps while trying to reconcile them:

```
... the correct derivative is 2x^3 - 4x. [...] Wait, but according to the expansion,
    derivative is x^3 - 4x. [...] Therefore, there is still a contradiction. [the analysis
    continues to the token limit]
```

On the uncontrolled trajectory, learned stopping does not fire until token 5440 ($\rho_t = 0.95$), after VCS has already finished at token 4535. It preserves the differentiation error and answers incorrectly:

```
... D(x) = (1/2)x^4 - 2x^2 + 9/2. Then derivative is 2*(1/2)x^3 - 4x = x^3 - 4x. But
    according to the other method, derivative is 2x^3 - 4x. Therefore, there is a
    contradiction here.
</think>
Final Answer:  \boxed{11}
```

VCS instead rewrites the objective in $z = x^2$, which removes the ambiguous step and yields the minimum directly:

```
... Let me set z = x^2. Then D = z + (1/2)(z - 3)^2.

D = (1/2)z^2 - 2z + 9/2. [...] Taking derivative with respect to z: z - 2. Setting to zero
    gives z = 2. Therefore, x^2 = 2 [...]. Therefore, the minimal distance squared is 5/2,
    so distance is sqrt(5/2) = sqrt(10)/2. Therefore, answer is 12.
[...]
</think>
[...]
Therefore, the final answer is:
$$
a + b = 10 + 2 = \boxed{12}.
$$
```

Example 2 (OlympiadBench #503; answer $\frac{15-8\sqrt{3}}{33}$). *If* $6\tan^{-1} x + 4\tan^{-1}(3x) = \pi$*, compute* x^2*.* Uncontrolled CoT derives several algebraic candidates but caps before deciding which one satisfies the original inverse-trigonometric equation:

```
... the solutions for y are: From the first quadratic: y = [23 +/- 8sqrt(7)]/9. From the
    second quadratic: y = [15 +/- 8sqrt(3)]/33. Now, since y = x^2, and x is real, y must be
     positive. Let me check if all these solutions are positive. [the analysis continues to
    the token limit]
```

Learned stopping fires much earlier, at token 1405 ($\rho_t = 0.97$), while the multiple-angle expansion is still incomplete, and returns 1:

```
... tan(4B) = 12x(1 - 9x^2) / (81x^4 - 54x^2 + 1).

Okay, that's tan(4B). Now, let's compute tan(6A). Similarly, A = arctan(x), so tan(A) = x.
    Let me compute tan(6A). Let me use multiple-angle formulas.
</think>
Final Answer:  \boxed{1}
```

VCS carries the candidate check through and rejects the extraneous root:

```
... x^2 is either [15 + 8sqrt(3)]/33 or [15 - 8sqrt(3)]/33. Now, we need to check which of
    these solutions is valid [...].

For x approximately 0.935, the left-hand side is approximately 9.438, so this solution is
    invalid. For x approximately 0.186, it is approximately 3.146, which is approximately pi
    .

Therefore, the valid solution is y = [15 - 8sqrt(3)]/33.
[...]
```

```
</think>
[...]
### Final Answer
$$
\boxed{\dfrac{15 - 8\sqrt{3}}{33}}
$$
```

Example 3 (OlympiadBench #311; answer $\frac{4}{7}$). *Eight people, including triplets Barry, Carrie, and Mary, are randomly paired into four two-person canoes. What is the probability that no two triplets share a canoe?* Uncontrolled CoT conditions on Barry and Carrie being separated, then loses track of how their partners affect Mary's placement and caps:

```
... the probability that Barry and Carrie are not in the same canoe is 6/7. Then, given that,
     the probability that Mary is not in the same canoe as Barry or Carrie. [...] Mary is
    one of [the four people left]. Now, the question is, what is the probability [the
    analysis continues to the token limit]
```

Learned stopping fires at token 2955 ($\rho_t = 0.92$) while the same conditional argument remains unresolved, and takes the incomplete factor $6/7$ as its answer:

```
... if we have already fixed Barry and Carrie in their pairs, then Mary is in the remaining 4
     people, so she can't be with Barry or Carrie. Therefore, the probability is 1? [...]
    Therefore, the total probability would be 6/7 * 1 = 6/7? But that seems too high.
</think>
Final Answer:  \boxed{\dfrac{6}{7}}
```

VCS avoids the conditional dependence by counting pairings directly:

```
The total number of pairings is 8!/(2^4 * 4!) = 105.
For any specified pair among Barry, Carrie, and Mary, the other six people can be paired in
    15 ways. These three events are disjoint, so the number of invalid pairings is 3 * 15 =
    45.
favorable pairings = 105 - 45 = 60
60/105 = 4/7
[...]
</think>
[...]
### Final Answer
\boxed{\dfrac{4}{7}}
```