A HINT FROM THE PAST: GUIDING DIFFUSION LANGUAGE MODELS UNDER CONTEXT UPDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning assistants must accommodate changes to a user's requirements while a solution is being generated. In multi-step problem solving, this requires adapting to new constraints or objectives after reasoning has begun. The partial answer provides potentially useful derivations and solution strategies that can guide the response to these changes. We investigate whether selectively using text from the previous partial answer during generation can improve correctness beyond retaining it in context. To this end, we introduce HiPE, a method that guides diffusion language models under updated conditions by injecting selected passages as revisable hints during generation. Implemented in DiffusionGemma, HiPE uses the current generation state to determine intervention timing, attention to extract candidate passages, and predictive distributions to select their destinations. It requires no additional training and constructs interventions from existing decoder forward passes. On AIME 2024 and 2025, HiPE improves accuracy over regeneration by 2.22 and 3.33 percentage points, respectively, with fewer decoder forward passes on average. It achieves higher accuracy than all evaluated initialization-based reuse methods across mathematics and coding tasks.

1 INTRODUCTION

When a user updates the context during generation, the answer must reflect the updated request even though part of the answer has already been written under the earlier context. For example, while the assistant is solving a problem about a list of integers, the user might correct their total from 36 to 30 and request the sum of squares instead of cubes (Figure 1B). A general strategy for identifying the integers may still apply, but calculations based on the previous total and objective must be reconsidered. Prior work reports overreliance on earlier assumptions and answers across conversation turns (Laban et al., 2026) and shows that strong performance on fixed questions does not ensure reliable adaptation to updates during generation (Wu et al., 2026).

The simplest approach to an update is to generate a new answer from scratch with the original question, the previous partial answer, and the update as context. In this approach, the model alone decides which parts of the previous partial answer are reused in the new answer. As a more direct approach, we consider injecting selected text from the previous partial answer into the generation of the new answer. However, directly reusing such passages may cause the final answer to rely on assumptions that no longer hold under the updated conditions.

To address this risk, injected text must remain revisable during generation. In left-to-right autoregressive decoding, text already written into the output stays fixed while later tokens are generated. A diffusion language model refines a full-length answer over a sequence of denoising steps, and any part of the answer can be revised at a later step when the model's predictions require it (Sahoo et al., 2024; Arriola et al., 2025). Text injected into the answer at one step therefore remains open to revision at later steps. The proposed method is built on diffusion language models.

To guide generation under the updated context, we introduce **Hi**nt from **P**revious answ**E**rs (HiPE), a method that uses selected passages from the previous partial answer as revisable hints during denoising. Implemented in DiffusionGemma (Google DeepMind, 2026), HiPE replaces some random tokens in the answer with the selected text during generation (Figure 1). The choices of when, what, and where to inject depend on the answer taking shape under the new conditions. HiPE uses the current generation state to decide when to inject and attention to find potentially relevant text.

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.

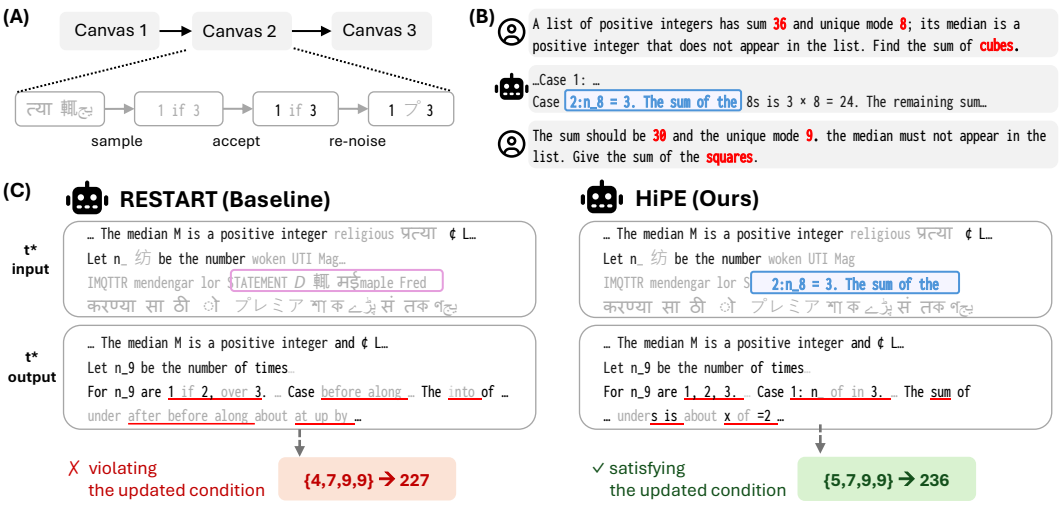


Figure 1: **Guiding denoising after a context update.** (A) Canvas generation alternates token sampling, acceptance, and re-noising. (B) A user changes the problem conditions after a partial answer has been generated; the highlighted passage is selected from that answer. (C) Both RESTART and HiPE generate under the same updated context. HiPE additionally places selected text from the previous answer into the denoising canvas to guide generation under the updated conditions.

Placement considers uncertainty at insertion positions and how well the selected text fits the model's current predictions there. The selection rules require no additional training and use signals from existing decoder forward passes.

We evaluate HiPE on mathematics and coding tasks with context updates, keeping the previous partial answer in the context for every method. HiPE raises accuracy over regeneration on both AIME benchmarks and on both LiveCodeBench subsets while using fewer decoder forward passes on average. The three initialization baselines, which copy the same text into the answer before generation begins, fall below regeneration on every task. The previous answer therefore improves accuracy when injected as revisable hints during generation and reduces accuracy when copied into the answer beforehand.

Our contributions are:

- We pose the question of whether a previous partial answer helps when injected into generation. To answer it, we propose to use the previous partial answer not only as context but also as revisable hints inside diffusion generation under a context update.
- We introduce HiPE, a training-free rule that decides when to inject, which text to select, and where to place it from signals that existing forward passes already produce. No additional forward pass is required, and later denoising steps can revise the injected text.
- HiPE raises accuracy over regeneration on the AIME and LiveCodeBench tasks with fewer forward passes. The ablations reveal that the content, selection, and placement of the hints all contribute to the observed gain.

2 RELATED WORK

Generation under evolving context. Context updates raise the question of how a model treats reasoning produced under earlier assumptions. In conversations where requirements arrive across turns, Laban et al. (2026) find that models rely on early assumptions and answers. Chen et al. (2026) address adaptation across turns through reinforcement learning with reward signals anchored to single-turn responses. In spoken interaction, EchoChain evaluates whether voice assistants incorporate state updates delivered through mid-response interruptions (Modi et al., 2026). Wu et al. (2026) examine updates that arrive during generation and show that performance on fixed questions does not ensure reliable adaptation. The present work addresses the same setting and tests whether

the previous partial answer helps when presented during the denoising of the new answer rather than only in the context.

Diffusion guidance and editing. Earlier diffusion language models support controlled text generation by guiding continuous intermediate representations (Li et al., 2022; Han et al., 2023). For editing with reference text, methods differ in how much of the reference is preserved. DiffusER conditions reconstruction on prototypes or incomplete sequences (Reid et al., 2023). EdiText controls the extent to which reference text is revised (Lee et al., 2025). For evolving contexts, Detect–Remask–Repair repairs selected regions of a summary and keeps the remaining tokens fixed (Zou et al., 2026). In these editing methods the reference either guides a new output or constrains which parts of an existing output are revised. HiPE uses the previous answer to guide a new denoising trajectory. Further denoising revises the injected text without a preservation constraint.

Diffusion decoding and selection. Masked and block diffusion models construct text through intermediate states and therefore choose which positions to generate or revisit (Sahoo et al., 2024; Nie et al., 2025; Arriola et al., 2025). Entropy-Bounded Unmasking and learned unmasking policies select positions with an entropy constraint or a learned policy (Ben-Hamu et al., 2025; Jazbec et al., 2026), and remasking methods such as ReMDM, RemeDi, and CoRe allow generated tokens to be revised (Wang et al., 2025; Huang et al., 2026; Zhai et al., 2026). Whereas these methods decide which positions are generated or revisited, HiPE changes the text presented at positions that DiffusionGemma already re-noises (Google DeepMind, 2026), and the backbone's token sampling, acceptance, and stopping rules are preserved. HiPE admits a span according to its probability and entropy at the insertion positions, a criterion related to locally typical sampling and η-sampling (Meister et al., 2023; Hewitt et al., 2022). Sampling criteria decide which tokens can be sampled, whereas HiPE evaluates a span from the previous answer as a unit.

3 Problem Setup and Background

3.1 Problem setup

We consider a user update that revises the question's conditions or requirements while an assistant is generating an answer. Let q_1 denote the user's initial question and a_1 the partial answer produced before the update. After the update, the assistant generates a new answer a_2 using the updated context, which contains q_1, a_1, and the user's update message.

In this setting, some parts of a_1 may remain useful, while others depend on conditions that have changed. For example, suppose the update changes a parameter from $n = 12$ to $n = 15$: a derivation in a_1 expressed in terms of n may still apply, while calculations substituting $n = 12$ are stale. Our goal is to guide the generation of a_2 under the updated context using selected text from a_1.

3.2 Canvas Denoising in DiffusionGemma

DiffusionGemma (Google DeepMind, 2026) is a block diffusion language model (Arriola et al., 2025) that generates an answer as a sequence of *canvases*, each a fixed-length token block denoised jointly while all preceding text remains fixed as context. Each canvas starts with uniformly random tokens. Denoising repeatedly updates this token block using the model's predictions under the given context (Austin et al., 2021; Lou et al., 2024).

Each canvas contains C tokens and is denoised for at most T steps. We denote the input canvas at step t by x_t and refer to the sequence of these states as the denoising trajectory. The token of x_t at position $s \in \{1, \ldots, C\}$ is denoted by $x_{t,s}$. A single decoder forward pass over the context and x_t produces a predictive distribution $p_t(\cdot \mid s)$ over tokens at each position.

Given these predictions, the sampler constructs the input canvas for the next denoising step. It first samples a token $\tilde{x}_{t,s}$ at each position. To retain relatively confident predictions, it selects low-entropy positions as the *accepted set* Ω_t at each step, subject to its entropy bound (Ben-Hamu et al., 2025). Accepted positions receive their sampled tokens, while the remaining positions are *re-noised*

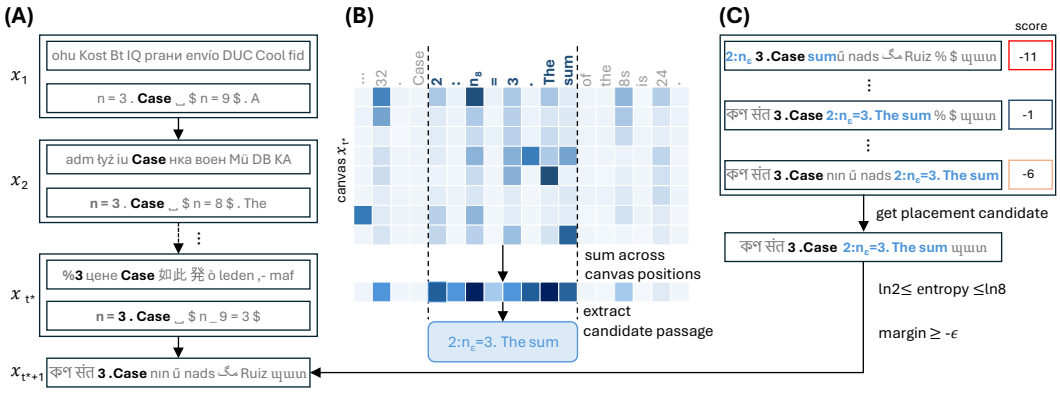


Figure 2: **Overview of HiPE.** (A) Intervention planning begins at step $t^\star$, when accepted tokens provide an initial structure for the answer. (B) Attention from the current canvas to the previous answer is aggregated to extract candidate spans. (C) Candidate placements are scored by mean token log-probability and selected subject to entropy and margin criteria. Selected tokens replace random tokens at re-noised positions.

with fresh uniformly random tokens. The next input canvas is therefore

$$x_{t+1,s} = \begin{cases} \tilde{x}_{t,s} & s \in \Omega_t \quad \text{(accepted)}, \\ z_{t,s} \sim \mathrm{Unif}(\mathcal{V}) & s \notin \Omega_t \quad \text{(re-noised)}, \end{cases} \tag{1}$$

where $\mathcal{V}$ denotes the vocabulary and $z_{t,s}$ is a fresh token sampled uniformly from $\mathcal{V}$ to re-noise position s. Once denoising is complete, the canvas output is fixed as context for the next canvas.

4 HiPE: Selective Text Injection

After a context update, some passages in the previous answer a_1 may remain applicable, whereas others depend on conditions that have changed. To guide the new answer a_2, we select passages using the current denoising state under the updated context. HiPE injects the selected text into the canvas input for one step, allowing subsequent denoising to revise it. The method uses the accepted fraction to determine intervention timing, attention to extract source spans, and predictive distributions to select their destinations. Figure 2 illustrates how HiPE extracts candidate spans, selects their placements, and injects the selected text during denoising.

4.1 Intervention Timing

At initialization, canvas predictions and attention are conditioned on random tokens rather than tokens accepted from previous model predictions. We delay selection until a minimum fraction of the canvas contains accepted tokens. A high accepted fraction, however, leaves fewer positions available for placement.

To determine the planning step, we define the *accepted fraction* as

$$\omega_t = \frac{|\Omega_t|}{C}. \tag{2}$$

At step t, ω_{t-1} measures the fraction of positions in the current input x_t filled with tokens accepted in the preceding transition. Using ω_{t-1} ties the timing criterion to the input used to compute the selection signals.

For each canvas, the planning step $t^\star$ is the first step satisfying

$$\omega_{\mathrm{lo}} \le \omega_{t^\star-1} \le \omega_{\mathrm{hi}}, \qquad t^\star < t_{\mathrm{dec}}. \tag{3}$$

Here, ω_{lo} and ω_{hi} are the accepted-fraction bounds, and t_{dec} is the first excluded planning step. If no step satisfies these conditions, we skip injection on that canvas. Planning is attempted at most once per canvas, even if no span is selected.

4.2 SOURCE SELECTION AND PLACEMENT

Source selection depends on both attention to the previous answer and the predicted token distributions at candidate destinations. We extract candidate spans from the attention profile and evaluate their token log-probabilities at feasible canvas positions.

Candidate extraction. At $t^\star$, let $A[s, i]$ denote attention from canvas position s to source position i in a_1, averaged over decoder layers and heads. We aggregate attention over canvas positions to obtain a score for each source token:

$$\alpha(i) = \sum_{s=1}^{C} A[s, i]. \tag{4}$$

Eligible positions exclude the end-of-answer tail and a source exclusion set U, maintained across canvases to limit repeated selection. We smooth the scores and mark eligible positions at or above their mean. We bridge gaps up to a fixed length and discard spans shorter than the minimum length. A candidate $r = a_1[i{:}i + L]$ contains the source tokens at positions $i, \ldots, i + L - 1$. The extraction details are given in Appendix B.

Placement scoring. Attention identifies source spans but does not determine their destinations in the canvas. We score each candidate placement using the predictive distributions at its destination positions. For a span r starting at canvas position s, we define the mean token log-probability $\ell(r, s)$ and mean predictive entropy $\mathcal{H}(r, s)$ as

$$\begin{aligned} \ell(r, s) &= \frac{1}{L} \sum_{j=0}^{L-1} \log p_{t^\star}(a_1[i + j] \mid s + j), \\ \mathcal{H}(r, s) &= \frac{1}{L} \sum_{j=0}^{L-1} e_{t^\star, s+j}, \end{aligned} \tag{5}$$

where $e_{t,s}$ is the entropy of $p_t(\cdot \mid s)$. Both quantities use the decoder's temperature-one predictive distribution, with natural logarithms and entropy in nats. Both scores are averaged over the L destination positions.

Diffuse predictive distributions provide little discrimination among candidate tokens, whereas concentrated distributions leave little uncertainty at the destination. We therefore restrict placements to an intermediate range of mean predictive entropy:

$$\ln 2 \leq \mathcal{H}(r, s) \leq \ln 8. \tag{6}$$

Among placements satisfying this range and occupying no position in $\Omega_{t^\star - 1}$, we select the one maximizing $\ell(r, s)$. A candidate with no feasible destination is discarded.

Margin criterion. Maximizing $\ell(r, s)$ ranks feasible placements but does not impose a minimum score. We compare $\ell(r, s)$ with $-\mathcal{H}(r, s)$, the expected token log-probability averaged over the same positions. We define the *margin* $m(r, s)$ and require

$$m(r, s) = \ell(r, s) + \mathcal{H}(r, s) \geq -\epsilon. \tag{7}$$

The tolerance permits a deficit of $\epsilon / \ln 2$ bits in mean log-probability relative to this expectation. Table 6 in the appendix gives the dataset-specific values of ϵ. This criterion measures compatibility with current predictions, rather than logical validity under the update.

We process candidates in descending order of their initial placement scores ℓ, adding qualifying spans to the plan. Destinations must not overlap. If a placement overlaps a previously allocated span, we recompute the maximum-ℓ feasible placement and reapply the selection criteria. The resulting plan records source tokens, source indices, and canvas destinations. Appendix B specifies the allocation procedure and numerical checks.

4.3 ONE-STEP TEXT INJECTION

At $t^\star$, HiPE replaces fresh noise at planned destinations with source tokens while retaining the sampler's accepted tokens. The plan is applied once. If denoising continues, the modified canvas becomes the next decoder input, and subsequent transitions can revise the inserted tokens. Token sampling, acceptance, and stopping rules remain unchanged. The selection rules are fixed and use signals from existing forward passes. The intervention requires neither training nor an additional decoder call. Additional details are provided in Appendix B.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Tasks and protocol. We use condition-update tasks from Wu et al. (2026), based on GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AIME 2024 and 2025, and LiveCodeBench (Jain et al., 2025). Each task poses a modified version of a problem and the update restores the original, so the original answer or tests grade the result. We introduce the update after the model has generated c canvases, using the text produced up to that point as the partial answer a_1. Additional details are provided in Appendix C.

Model and comparison. We compare HiPE with regeneration and initialization-based reuse to evaluate whether selectively injecting previous-answer text during denoising offers benefits beyond retaining it in context or using it to initialize generation. Every method runs DiffusionGemma-26B-A4B-it with thinking on and decodes with the backbone's own sampler. All methods receive the same updated context, including a_1. RESTART generates a new answer with each canvas initialized to uniform random tokens. The three warm-start baselines instead initialize canvases from the corresponding segments of a_1, adapting the initialization approach of SDEdit (Meng et al., 2022) to token generation. They differ in both how many source tokens they retain and how those positions are selected. *Random warm start* keeps a random half of the available source positions and re-noises the rest. *Full warm start* keeps all available source tokens. *Confidence warm start* keeps the half the decoder rates most likely, which costs one extra forward pass per canvas with source text.

Metrics. We report accuracy using official mathematics answer matching and LiveCodeBench execution pass@1, together with the mean number of function evaluations (NFE). NFE counts decoder forward passes used to generate the new answer after the update. Every reported value averages three generation seeds per problem.

5.2 MAIN RESULTS

Table 1 compares HiPE with regeneration and initialization-based reuse. Relative to RESTART, HiPE increases accuracy on AIME 2024 and 2025 by 2.22 and 3.33 percentage points, respectively, while reducing mean NFE by 3.1 and 6.4. On AtCoder and LeetCode, the gains are smaller, at 0.54 and 0.30 points, with slightly lower mean NFE. These improvements suggest that an answer written under previous conditions can still contain useful guidance for the updated task. Since RESTART also receives that answer in context, the gains further suggest that selecting passages for use during denoising can help the model make use of this guidance. On GSM8K and MATH-500, however, accuracy and mean NFE change little, showing that the benefit is not uniform across tasks.

HiPE achieves higher accuracy than all three warm-start baselines on every evaluated task. On MATH-500, the accuracy loss relative to RESTART ranges from 9.24 to 13.01 percentage points, with an additional 11.5 to 16.3 decoder evaluations on average. On LeetCode, full warm start lowers accuracy by 16.82 points and increases mean NFE by 45.0. Full warm start often yields lower accuracy than random warm start, and confidence warm start offers no consistent advantage over random warm start. These results suggest that text based on previous conditions can interfere with generation under the update. The following ablations examine how HiPE's text selection and placement contribute to its results.

Table 1: Accuracy (%) and mean NFE under matched contexts and generation budgets. AtCoder and LeetCode are subsets of LiveCodeBench. Bold and underlined values indicate the best and second-best results in each column.

Method	GSM8K		MATH-500		AIME 2024		AIME 2025		AtCoder		LeetCode	
	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
RESTART	**84.54**	<u>25.4</u>	**59.99**	**63.1**	<u>51.11</u>	<u>219.6</u>	<u>56.67</u>	<u>346.3</u>	<u>46.20</u>	315.1	<u>66.37</u>	<u>217.5</u>
Random warm start	83.87	30.2	49.96	74.6	46.67	224.8	43.33	351.9	37.14	**304.2**	60.06	238.9
Full warm start	81.85	33.0	46.98	79.4	47.78	238.0	50.00	356.3	34.06	322.0	49.55	262.5
Confidence warm start	83.33	29.8	50.75	76.8	43.33	225.1	44.44	381.7	40.04	<u>310.8</u>	57.36	240.7
HiPE	<u>84.41</u>	**25.0**	<u>59.91</u>	<u>63.4</u>	**53.33**	**216.5**	**60.00**	**339.9**	**46.74**	314.3	**66.67**	**217.3**

Table 2: Ablation study of text selection and placement on the four mathematics tasks and LeetCode. Control results average three randomizations. Bold and underlined values indicate the best and second-best results in each column.

Method	GSM8K		MATH-500		AIME 2024		AIME 2025		LeetCode	
	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
RESTART	**84.54**	25.4	**59.99**	**63.1**	51.11	219.6	56.67	346.3	<u>66.37</u>	<u>217.5</u>
Random tokens, planned positions	**84.54**	<u>25.1</u>	59.89	<u>63.3</u>	<u>52.59</u>	221.9	58.52	<u>344.3</u>	65.67	219.6
Other a_1 spans, planned positions	<u>84.50</u>	**25.0**	<u>59.91</u>	63.5	51.48	219.5	<u>58.89</u>	353.7	65.97	219.8
Selected spans, random positions	<u>84.50</u>	<u>25.1</u>	59.75	63.6	**53.33**	<u>217.4</u>	<u>58.89</u>	349.9	65.57	220.2
HiPE	84.41	**25.0**	<u>59.91</u>	63.4	**53.33**	**216.5**	**60.00**	**339.9**	**66.67**	**217.3**

5.3 ABLATION STUDY

Table 2 examines text selection and placement through three controls. The random-token control preserves the planned positions and span lengths but replaces the selected text with uniformly sampled tokens. The random-position control preserves the selected text but randomizes its destinations and removes the entropy restriction. The alternative-span control preserves the planned positions but substitutes other spans from the previous answer.

On AIME 2024 and AIME 2025, random tokens achieve mean accuracies of 52.59% and 58.52%, respectively, exceeding RESTART but remaining below HiPE's 53.33% and 60.00%. Alternative source spans yield accuracy close to RESTART on AIME 2024 and the highest mean NFE on AIME 2025. Random placement matches HiPE's accuracy on AIME 2024, but on AIME 2025 it reduces accuracy by 1.11 percentage points and increases mean NFE by 10.0. On LeetCode, the three controls fall below RESTART by 0.40 to 0.80 percentage points with higher mean NFE, whereas HiPE remains above RESTART.

These comparisons suggest that the selected content and its placement contribute in complementary ways. The gains with random tokens indicate that perturbations at the planned positions may account for part of the improvement. Selected spans achieve higher mean accuracy, while retaining those spans without the placement constraints does not preserve the full computational benefit. HiPE combines the highest or tied-highest mean accuracy with the lowest mean NFE on both AIME tasks.

5.4 SENSITIVITY ANALYSES

Margin tolerance. A larger margin tolerance ϵ allows candidate spans with lower compatibility scores to be selected. As shown in Figure 3 (left two panels), the effect differs between the two AIME tasks: AIME 2024 achieves higher accuracy at two and three bits, whereas AIME 2025 achieves its highest accuracy at one bit and requires more decoder evaluations at larger tolerances. Relaxing the selection criterion therefore does not consistently improve the results. At one bit, both AIME tasks show higher accuracy and lower mean NFE than RESTART, while MATH-500 remains largely unchanged across the evaluated settings.

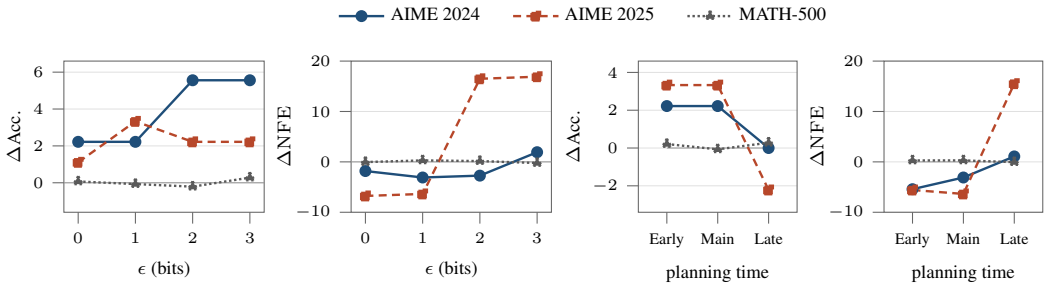


Figure 3: Accuracy and mean NFE changes relative to RESTART as the margin tolerance (left two panels) and planning time (right two panels) vary. Early, Main, and Late use accepted-fraction bounds $(.10, .30)$, $(.25, .45)$, and $(.40, .60)$, respectively.

Table 3: Removing one end of the entropy range at a time, with every other constant fixed. Within each column the best value is in bold and the second best is underlined.

	GSM8K		MATH-500		AIME 2024		AIME 2025	
Entropy range (bits)	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
RESTART	<u>84.54</u>	25.4	<u>59.99</u>	**63.1**	<u>51.11</u>	219.6	56.67	346.3
$[0, 3]$, no lower bound	84.41	<u>25.0</u>	<u>59.99</u>	<u>63.2</u>	<u>51.11</u>	**212.8**	<u>58.89</u>	<u>342.9</u>
$[1, \infty)$, no upper bound	**84.95**	**24.4**	**60.41**	63.7	50.00	226.3	57.78	358.7
$[1, 3]$, both bounds	84.41	<u>25.0</u>	59.91	63.4	**53.33**	<u>216.5</u>	**60.00**	**339.9**

Planning time. As shown in Figure 3 (right two panels), the identical AIME accuracy gains in the early and main windows suggest that the benefit is not confined to one early planning window. Later planning removes the AIME 2024 gain, reduces AIME 2025 accuracy below RESTART, and increases mean NFE on both tasks despite lower plan frequencies. On AIME 2025, the share of rollouts with a plan decreases from 41.1% to 34.4%, yet the NFE difference relative to RESTART changes from -6.36 to $+15.52$ (Table 10). This pattern suggests that fewer interventions need not incur less computation. Their effect depends on the generation state and the source spans selected under that state.

Entropy bounds. Table 3 evaluates the lower and upper entropy bounds in Eq. (6) by removing each bound separately. On GSM8K and MATH-500 no setting produces a meaningful difference, although removing the upper bound raises the share of pairs with a plan to 38% and 77%. The bounds matter on the two AIME tasks. Without the lower bound, positions the decoder has already settled become eligible, so the injected text lands where predictions are confident and accuracy stays at or slightly above RESTART. Removing the upper bound admits placements with more diffuse predictive distributions. On both AIME tasks, this setting yields lower accuracy and higher mean NFE than with both bounds.

5.5 SELECTIVE REUSE DURING ANSWER GENERATION

Table 4 reports accuracy on rollouts in which HiPE forms a plan. Within this subset, accuracy improves by 5.13 and 8.11 percentage points on AIME 2024 and AIME 2025, and by 1.99 and 1.64 points on AtCoder and LeetCode, respectively. GSM8K and MATH-500 show decreases of 1.09 and 0.29 points. These interventions yield net accuracy gains on the AIME and coding tasks, but do not consistently improve accuracy across tasks.

Figure 4 (left) shows that plan frequency is highest on the first canvas and generally lower thereafter. On AIME 2024 and AIME 2025, it decreases from 17.8% and 13.3% on the first canvas to 5.6% and 4.4% on the third, respectively. The within-rollout AIME analysis in Appendix F.3 also finds that the best candidate log-probability tends to decline across canvases. One interpretation is that, as the new answer develops under the updated conditions, the remaining previous-answer text becomes less compatible with its continuation.

Table 4: Accuracy (%) of RESTART and HiPE on rollouts grouped by whether HiPE forms a plan. For rollouts with a plan, Δ reports the accuracy difference between HiPE and RESTART in percentage points. For rollouts without a plan, the two methods have the same accuracy.

	rollouts with a plan				rollouts without a plan	
Dataset	Rollouts	RESTART Acc.	HiPE Acc.	Δ	Rollouts	Acc.
GSM8K	92	83.70	82.61	-1.09	652	84.66
MATH-500	343	56.85	56.56	-0.29	1064	61.00
AIME 2024	39	61.54	66.67	+5.13	51	43.14
AIME 2025	37	45.95	54.05	+8.11	53	64.15
AtCoder	151	29.14	31.13	+1.99	401	52.62
LeetCode	61	44.26	45.90	+1.64	272	71.32

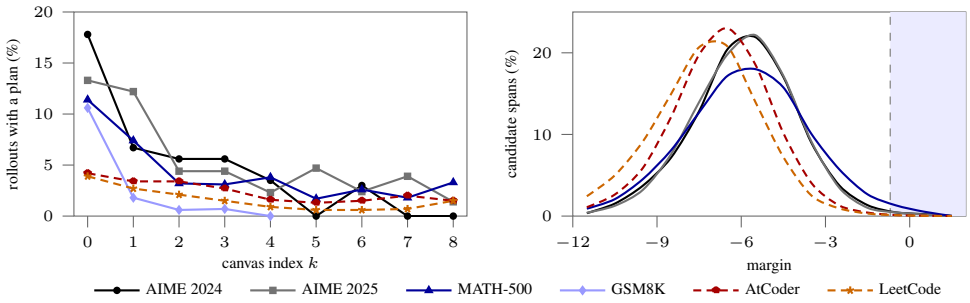


Figure 4: Left: percentage of rollouts in which HiPE forms a plan at each canvas of the new answer. Right: distribution of candidate margins $m = \ell + \mathcal{H}$, which measure compatibility with the current canvas predictions. The dashed line marks the margin threshold for mathematics, with qualifying margins shaded, and the dotted line marks the threshold for code.

Figure 4 (right) shows the margin-based filtering of candidate spans. The margin distributions peak near -5.5 nats on the mathematics tasks and -6.5 nats on the coding tasks, well below their respective thresholds of approximately -0.69 and -1.39 nats. Most attention-derived candidates therefore do not satisfy the compatibility criterion. Attention to a source passage thus does not by itself establish suitability for insertion into the current canvas. Placement-based screening addresses this distinction by evaluating candidate spans against the current destination predictions.

Taken together, these observations suggest that the suitability of previous reasoning for reuse depends on the developing answer, rather than on the source text alone. Attention identifies candidate spans, but their placement requires a separate compatibility assessment. The AIME and coding results show that interventions selected in this way can improve final accuracy, while the results on GSM8K and MATH-500 indicate that local compatibility alone does not guarantee a better answer.

6 CONCLUSION

We introduced HiPE, a method that guides a denoising trajectory by placing selected text from the assistant's previous partial answer into the new answer's denoising input for one step. The selection rules read signals that existing decoder forward passes already produce, add no training and no additional forward pass, and leave the backbone's sampling, acceptance, and stopping rules unchanged. Implemented for DiffusionGemma, HiPE raises accuracy on both AIME update tasks by 2.22 and 3.33 percentage points over RESTART while using fewer decoder forward passes per pair, and it is the most accurate of the five evaluated methods on those tasks and on both code subsets. The three initialization baselines, which place the same text in the canvas before denoising rather than during it, are below RESTART on every subset, so how previous text is presented matters more than whether it is available.

AI USE STATEMENT

In this work, we used generative AI tools to implement the method, to obtain feedback on the design of the control experiments, and to draft summaries of result tables. The implementation was verified by the authors against the procedure in Appendix B, and the summaries were checked against the recorded runs. We have not used generative AI tools to generate synthetic data sets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims or assist in proofs, to propose or refine hypotheses, or to clean and reformat data sets, and translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to polish the writing and to search for and verify the publication venues of cited work. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses public benchmarks (GSM8K, MATH-500, AIME 2024 and 2025, and LiveCodeBench) and the condition-update variants released by Wu et al. (2026). No human subjects were involved and no new data were collected. The method operates at inference time on an open-weight model and adds no training. We are not aware of other ethical concerns beyond those shared by language model research in general.

REPRODUCIBILITY STATEMENT

The evaluation protocol, the compared methods, and the metrics are specified in Section 5.1. All constants of the method, including the tolerance per dataset, are listed in Table 6 (Appendix A), and the intervention procedure is given as pseudocode in Appendix B. Data preparation, the definition of the partial answer, the scoring of mathematics and code outputs, and the uncertainty estimates are described in Appendix C. Per-cell results for every sensitivity setting and control seed are reported in Appendices D and E. Source code, the evaluation records of every reported run, and the scripts that regenerate the tables and figures from those records will be made available in an anonymous repository.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S. Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, volume 38, pp. 55981–56007. Curran Associates, Inc., 2025. doi: 10.52202/085713-1874.

Xingwu Chen, Zhanqiu Zhang, Steven Y. Guo, and Difan Zou. Breaking contextual inertia: Reinforcement learning with single-turn anchors for stable multi-turn interaction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6302–6318, 2026. doi: 10.18653/v1/2026.findings-acl.313.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Google DeepMind. DiffusionGemma. Model overview, 2026. URL https://deepmind.google/models/gemma/diffusiongemma/.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023. doi: 10.18653/v1/2023.acl-long.647.

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.

John Hewitt, Christopher D. Manning, and Percy Liang. Truncation sampling as language model desmoothing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 3414–3427, 2022. doi: 10.18653/v1/2022.findings-emnlp.249.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=BsZeTuB5fD.

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 (ICLR)*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. In *International Conference on Machine Learning (ICML)*, 2026.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

Che Hyun Lee, Heeseung Kim, Jiheum Yeom, and Sungroh Yoon. EdiText: Controllable coarse-to-fine text editing with diffusion language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22798–22815, 2025. doi: 10.18653/v1/2025.acl-long.1111.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343. Curran Associates, Inc., 2022. doi: 10.52202/068431-0313.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848. PMLR, 2024.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2023. doi: 10.1162/tacl_a_00536.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=aBsCjCPu_tE.

Smit Nautambhai Modi, Gandharv Mahajan, Marc Wetter, and Randall Welles. EchoChain: A full-duplex benchmark for state-update reasoning under interruptions. *arXiv preprint arXiv:2604.16456*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 50608–50646. Curran Associates, Inc., 2025. doi: 10.52202/085713-1689.

Machel Reid, Vincent J. Hellendoorn, and Graham Neubig. DiffusER: Diffusion via edit-based reconstruction. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=nG9RF9z1yy3.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 130136–130184. Curran Associates, Inc., 2024. doi: 10.52202/079017-4135.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. doi: 10.52202/085713-4927.

Tsung-Han Wu, Mihran Miroyan, David M. Chan, Trevor Darrell, Narges Norouzi, and Joseph E. Gonzalez. Are large reasoning models interruptible? In *International Conference on Machine Learning (ICML)*, 2026.

Kevin Zhai, Sabbir Mollah, Zhenyi Wang, and Mubarak Shah. CORE: Context-robust remasking for diffusion language models. In *International Conference on Machine Learning (ICML)*, 2026.

Hao Zou, Zachary Horvitz, Chandhru Karthick, Zhou Yu, and Kathleen McKeown. Detect, remask, repair: Diffusion editing for faithful summarization of evolving contexts. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026.

A NOTATION AND EXPERIMENTAL SETTINGS

Table 5 indexes the notation used in the main method. All logarithms in the method are natural; division by $\ln 2$ converts nats to bits.

Table 5: Notation used in the method and supplementary evaluation.

Symbol	Meaning
q_1, a_1, a_2, c	Initial question, partial answer, new answer (§3.1); interruption depth (§5.1)
$C, k, s, t, T, \mathcal{V}$	Canvas size, zero-based canvas index, one-based position within a canvas, one-based denoising step, step limit, vocabulary
$x_t, x_{t,s}, \hat{x}$	Input canvas, its token at position s, final argmax canvas
$\tilde{x}_{t,s}, z_{t,s}$	Token sampled from the model; fresh uniform random token
$p_t, \tilde{p}_t, \kappa_t$	Temperature-one predictions, sampler distribution, sampler temperature
$e_{t,s}, e_t$	Predictive entropy at position s (§4.2); entropies across the canvas
Ω_t, ω_t	Accepted positions and their fraction (§3.2)
$t^\star, \omega_{\text{lo}}, \omega_{\text{hi}}, t_{\text{dec}}$	Decision step for a canvas, lower/upper accepted-fraction bounds, first excluded decision step
$A_t, A[s,i], \alpha(i)$	Layer/head-averaged attention at step t, its entries at $t^\star$, total canvas attention to source token i
U	Source positions in a_1 excluded from the candidate mask before gap bridging
r, i, L, j	Candidate span, start in a_1, length, offset within the span
$I(r,s)$	Canvas positions a span r would occupy starting at s
$\ell, \mathcal{H}, m, \epsilon$	Mean token log-probability, mean placement entropy, signed difference from expected log-probability, permitted mean deficit
$B, \mathcal{S}(r;B), s^\star(r)$	Blocked canvas positions, allowed starts, start with maximum ℓ
P, h_s, i_s	Planned canvas positions, token assigned to s, its source position in a_1
ρ	Fraction of available source positions noised at initialization (Appendix E.1)
NFE, ΔAcc.	Decoder forward count; accuracy difference in percentage points, relative to RESTART

Table 6: Main HiPE hyperparameters and shared generation settings.

Setting	Value
Canvas size	$C = 256$ tokens
Maximum denoising steps per canvas	$T = 48$
Accepted-fraction bounds for planning	$[\omega_{\text{lo}}, \omega_{\text{hi}}] = [0.25, 0.45]$
First excluded planning step	$t_{\text{dec}} = 12$
Attention smoothing width	16 tokens
Maximum gap bridged during candidate extraction	2 tokens
Minimum span length	8 tokens
Placement entropy range	$[\ln 2, \ln 8]$ nats
Margin tolerance (GSM8K, MATH-500, AIME)	$\epsilon = \ln 2$ nats (1 bit)
Margin tolerance (LiveCodeBench)	$\epsilon = \ln 4$ nats (2 bits)
Injection duration	1 denoising step

The planning bounds and entropy range in Table 6 are shared across the main evaluation tasks. The margin tolerance differs between mathematics and code to account for their different token log-probability distributions (Appendix D). These settings are design heuristics; their numerical values are not uniquely determined by the signals they use. The comparisons in §5.4 vary the tolerance and planning bounds while retaining the same entropy range.

Backbone sampling details. The predictive distribution $p_t(\cdot \mid s)$ in §3.2 is the softmax of the decoder logits at temperature one. The backbone draws tokens from $\tilde{p}_t(\cdot \mid s) \propto p_t(\cdot \mid s)^{1/\kappa_t}$, where

the sampler temperature κ_t decreases linearly over the denoising schedule. It selects low-entropy positions under $\tilde{p}_t$ using the entropy-bound acceptance rule. The previous step's logits provide an additional self-conditioning input. Injection does not directly modify that signal, although later predictions and self-conditioning values can change.

A canvas terminates when its argmax is stable and the mean entropy under $\tilde{p}_t$ is below the stopping threshold, or when it reaches T steps. Its output is the final argmax canvas. Run-specific temperature endpoints, acceptance and stopping thresholds, and stability settings are not documented in the available evaluation manifests.

B Detailed intervention procedure

Span extraction. At the decision step, attention is averaged over layers and heads and summed over canvas queries. We smooth the source scores with a moving average of width 16, averaging only in-range positions at the source boundaries. The candidate mask marks positions outside U and the end-of-answer tail whose smoothed scores are at least the mean over these eligible positions. We bridge gaps of at most two tokens between marked positions and keep spans of at least eight tokens.

Gap bridging treats low-attention positions and positions in U alike, so an excluded source position can reappear inside a span. The end-of-answer tail remains excluded because no marked position lies beyond it.

Scoring distribution. Both ℓ and $\mathcal{H}$ use the temperature-one distribution $p_{t^\star}$, keeping the scoring convention independent of the sampler's temperature schedule. The score ℓ is a mean of token log-probabilities, and $-\mathcal{H}$ is their expectation under the same predictive distributions. Unlike the tokenwise absolute deviation used in locally typical sampling (Meister et al., 2023), our margin averages signed deviations over a source span and applies a lower bound, allowing opposing deviations to cancel.

Greedy allocation. Let $I(r, s) = \{s, \ldots, s+L-1\}$ be the canvas positions occupied by a length-L span r starting at s, and B the positions unavailable for placement. The allowed start positions are

$$\mathcal{S}(r; B) = \{s : 1 \le s \le C - L + 1,\ I(r, s) \cap B = \varnothing,\ \ln 2 \le \mathcal{H}(r, s) \le \ln 8\}. \tag{8}$$

Initially, $B = \Omega_{t^\star - 1}$. Algorithm 1 orders candidates by their initial maximum-ℓ placements and allocates qualifying spans without overlap. If an earlier allocation blocks a placement, it recomputes the best feasible placement and reapplies the checks, retaining the initial candidate order. Plan size follows from the qualifying spans and available positions; the evaluated implementation's additional cap is documented in Appendix C.

Placement searches use unrounded mean entropies. At each chosen placement, including replacement, entropy is rounded to four decimal places for the entropy-range and margin checks; this can affect eligibility near a threshold.

Plan representation and injection. Let P be the planned canvas positions, with each $s \in P$ assigned a source index i_s and token $h_s = a_1[i_s]$. Planning uses the preceding accepted set $\Omega_{t^\star - 1}$, but the sampler can accept additional planned positions at step $t^\star$. To retain these newly accepted predictions, we write only at $P \setminus \Omega_{t^\star}$. The next input becomes

$$x_{t^\star+1,s} = \begin{cases} \tilde{x}_{t^\star,s}, & s \in \Omega_{t^\star}, \\ h_s, & s \in P \setminus \Omega_{t^\star}, \\ z_{t^\star,s} \sim \mathrm{Unif}(\mathcal{V}), & \text{otherwise.} \end{cases} \tag{9}$$

The plan is applied once. If denoising continues, the next decoder call reads the modified canvas; later transitions follow the original sampler. Removing newly accepted positions can leave holes in a span, so the written subset need not retain the planned span's average entropy or margin.

State and empty plans. The exclusion set U starts empty and is carried across canvases. A canvas with no eligible step or no qualifying span receives no injection; planning is attempted at most once. At completion, source indices whose planned tokens match the final argmax canvas $\hat{x}$ are added to

Algorithm 1 Detailed pseudocode for one canvas. U is carried between canvases; the plan is local to this canvas.

Require: Updated context, source a_1, source exclusion set U, settings in Table 6
Ensure: Final canvas $\hat{x}$ and updated U

Initialize x_1 with uniform random tokens; $\Omega_0 \leftarrow \varnothing$; $\omega_0 \leftarrow 0$
pending $\leftarrow$ TRUE; $P \leftarrow \varnothing$ {planned canvas positions}
for $t = 1, \ldots, T$ **do**
  Read context and x_t with the decoder to obtain p_t, e_t, and A_t; set *write* $\leftarrow$ FALSE
  if *pending* **then**
    if $t \geq t_{\text{dec}}$ **or** $\omega_{t-1} > \omega_{\text{hi}}$ **then**
      pending $\leftarrow$ FALSE {no injection on this canvas}
    else if $\omega_{t-1} \geq \omega_{\text{lo}}$ **then**
      $t^\star \leftarrow t$; *pending* $\leftarrow$ FALSE
      $B \leftarrow \Omega_{t-1}$ {blocked canvas positions}
      Extract candidate spans using the mask and gap bridging in §4.2
      Discard spans with $\mathcal{S}(r; B) = \varnothing$
      For each remaining span r, set $s \leftarrow \arg\max_{u \in \mathcal{S}(r;B)} \ell(r, u)$
      for placed candidates $(r = a_1[i{:}i + L], s)$ in descending initial ℓ **do**
        if the placement passes the entropy and margin checks **then**
          if $I(r, s) \cap B \neq \varnothing$ **then**
            If $\mathcal{S}(r; B) = \varnothing$, **continue** to the next candidate
            $s \leftarrow \arg\max_{u \in \mathcal{S}(r;B)} \ell(r, u)$
            If the new position fails either check, **continue** to the next candidate
          end if
          For $j = 0, \ldots, L - 1$, set $h_{s+j} \leftarrow a_1[i + j]$, $i_{s+j} \leftarrow i + j$
          Add $I(r, s)$ to P, B
        end if
      end for
      write $\leftarrow (P \neq \varnothing)$
    end if
  end if
  Original sampler forms x_{t+1} and Ω_t by Eq. 1; $\omega_t \leftarrow |\Omega_t|/C$
  if *write* **then**
    $x_{t+1,s} \leftarrow h_s$ for each $s \in P \setminus \Omega_t$
  end if
  if original stopping rule is satisfied **then**
    break
  end if
end for
Set $\hat{x}$ to the final model prediction's argmax canvas
$U \leftarrow U \cup \{i_s : s \in P,\ \hat{x}_s = h_s\}$ {including skipped positions}
return $(\hat{x}, U)$

U, including positions skipped during writing. Algorithm 1 uses layer/head-averaged attention A_t and positional entropies e_t; token sampling, self-conditioning, and stopping follow the backbone.

C Data preparation and evaluation details

Task records. The mathematics records contain an original problem and answer, a modified problem, and an update message. The runner uses the modified problem as q_1, the update as the new user message, and the original answer as the corrected reference. Code records additionally contain original and modified starter code; the runner prepends the appropriate starter code to each problem. Functional ground truth is supplied by the original LiveCodeBench tests. The available records do not document source-dataset versions, the update-authoring and validation procedure, or complete exclusion accounting.

Table 7: Main evaluation subsets, interruption depths, and generation budgets.

Dataset	Depth c	Problems	Pairs	Token limit
GSM8K	1	248	744	4,096
MATH-500	2	469	1,407	8,192
AIME 2024	3	30	90	8,192
AIME 2025	3	30	90	16,384
LCB, AtCoder	5	184	552	8,192
LCB, LeetCode	5	111	333	8,192

The AIME evaluation covers all 30 source problems from each year. For MATH-500, the exclusion list accounts for 12 of the 31 omitted problems; the reduction from 500 available GSM8K records to 248 evaluated problems is also incompletely documented. Methods use identical pairs within each comparison, but pairing does not resolve the remaining uncertainty about dataset coverage.

Preparing the partial answer. The maximum number of canvases equals the token limit in Table 7 divided by $C = 256$. Thinking is enabled for both generations. The cache pipeline generates partial-answer caches and slices their tokens to each cell's fixed interruption depth. The prompt text and the source tokens consulted by HiPE use the same truncation. The modified initial problem, update, and truncated a_1 then condition a fresh a_2; internal generation states of a_1 are not resumed. The dataset-specific interruption depths in Table 7 are fixed independently of the thinking boundary. Validated fractions interrupted inside versus after thinking are unavailable.

Code scoring. Generated programs are evaluated with the official LiveCodeBench tests and a 20-second timeout. Python uses the harness; C++ is compiled with g++ and run on the same tests and output comparisons. Outputs without extracted code are incorrect. At the 8,192-token budget, 262 of the 885 RESTART generations reach the limit.

Pairing and uncertainty. The main evaluation uses generation seeds 42, 43, and 44 for every source. In the main comparison, a pair compares RESTART and HiPE for the same cell, source/example identifier, and seed, sharing the truncated-answer cache, decoding path, and sampler random numbers. The arms coincide before their first planned injection. Accuracy uses official correctness; NFE counts all decoder calls for a_2 after interruption and can change through both answer length and steps per canvas. Aggregate differences are taken before display rounding. Pooled AIME results weight both years equally, using 90 paired rollouts each. The two years retain their respective generation budgets.

Repeated generation seeds from one source are dependent observations. We first average the three paired differences within each source problem, then resample problems with replacement 20,000 times to obtain percentile 95% confidence intervals. We also apply two-sided Wilcoxon signed-rank tests to nonzero problem-level differences. These tests are exploratory and are not adjusted for multiple comparisons. Standard deviations across control randomizations characterize that variation only; they do not quantify uncertainty across source problems.

All reported intervals in Table 8 include zero. Only three AIME24 and two AIME25 problems have nonzero accuracy differences. The pooled AIME estimate combines years with different budgets; within-year comparisons take precedence. The intervals do not account for undocumented dataset exclusions.

Execution environment. GPU and memory specifications, resolved computation precision and sampler settings, and exact model/library/compiler revisions are incompletely documented. Wall-clock time and peak memory were not measured. Selection adds processing overhead despite using existing decoder calls, so NFE differences alone do not establish a runtime advantage.

Implementation revision. The reported HiPE runs were collected with an additional 128-token plan cap per canvas. A retrospective audit reported no candidates rejected by this cap in the evaluated

Table 8: Paired differences from RESTART with problem-level 95% intervals and exploratory Wilcoxon p-values (Appendix C). Accuracy differences are percentage points. Mathematics uses one-bit tolerance and LiveCodeBench two bits. Dashes indicate unavailable intervals and p-values. Pooled AIME combines years with different budgets.

Dataset	Problems	ΔAcc. [95% CI]	$p_{\text{Acc.}}$	ΔNFE [95% CI]	p_{NFE}
GSM8K	248	-0.13 $[-0.40, +0.00]$	1.000	-0.37 $[-1.08, +0.04]$	0.122
MATH-500	469	-0.07 $[-0.50, +0.36]$	1.000	$+0.29$ $[-0.30, +0.93]$	0.325
AIME 2024	30	$+2.22$ $[-2.22, +7.78]$	0.750	-3.10 $[-9.30, +2.92]$	0.171
AIME 2025	30	$+3.33$ $[+0.00, +8.89]$	0.500	-6.36 $[-32.63, +16.20]$	0.888
LiveCodeBench	295	$+0.45$ [—]	—	-0.52 [—]	—
AIME pooled	60	$+2.78$ $[-0.56, +6.67]$	0.250	-4.73 $[-18.53, +7.08]$	0.364

conditions. The current implementation and Algorithm 1 omit it; the reported results are retained from those runs. This observation does not bound plan sizes on future inputs.

Plan-frequency measurements. A rollout is counted as having a plan when at least one canvas has a nonempty plan. These counts do not verify actual writes: newly accepted positions are skipped, and stopping can prevent a subsequent decoder call from reading the modified canvas.

Control procedures. The random-token control replaces selected text with uniform random tokens at planned destinations. The random-position control retains the selected spans but randomizes destinations and removes the entropy-range restriction. The alternative-span control keeps the planned destinations and lengths and writes other spans of a_1, drawn from outside the selected ones. Later predictions, source exclusions, and plans may consequently differ between arms. Actual-write masks are unavailable in the evaluation export, so verification is limited to the recorded configurations and outputs.

D CELL-LEVEL SENSITIVITY RESULTS

Table 9 gives the per-task values underlying the tolerance sweep in Figure 3. Zero tolerance requires a nonnegative margin. The main setting uses one bit for mathematics and two bits for code. On code the rule intervenes on 14.2% of pairs at one bit, 24.0% at two, and 39.7% at three, against 24 to 43% on the mathematics tasks at one bit, because token log-probabilities on code are lower and fewer spans pass the margin (Figure 4, right). Wider tolerance does not uniformly improve accuracy or NFE.

Planning-time sensitivity. Table 10 combines the per-task timing results, plan frequencies, and problem-level intervals. Conditions share source–generation-seed pairs and matched RESTART outputs. Changing the bounds affects both planning eligibility before the fixed $t < 12$ cutoff and the state at selection, so the comparison does not isolate planning time for an otherwise identical intervention.

All intervals include zero and compare each condition with RESTART, not with other timing conditions. The degenerate late-AIME24 accuracy interval reflects no observed correctness changes, rather than certainty on new problems.

Entropy-bound removal. The results appear in Table 3. Recorded main-run thresholds are 0.693 and 2.079 nats, approximating $\ln 2$ and $\ln 8$. The boundary-removal report lists the same retained thresholds, but full commands are unavailable for independent configuration verification.

Interruption depth. Table 11 compares $c = 3, 4, 5$ on the same 26 AIME24 and 28 AIME25 problems with generation seeds 42–44 (78 and 84 pairs), using a matched RESTART at each depth. Plan frequency declines with depth, while accuracy and NFE show no consistent trend. This comparison is descriptive: additional-depth intervals are unavailable, and incompletely documented budgets and cache construction prevent attributing cross-depth differences to interruption timing alone.

Table 9: Margin-tolerance sensitivity relative to matched RESTART. Correctness changes count paired rollouts; dashes indicate unavailable results.

		Tolerance $\epsilon/\ln 2$ (bits)			
Dataset	Quantity	0	1	2	3
AIME 2024	ΔAcc. (pp)	$+2.22$	$+2.22$	$+5.56$	$+5.56$
	ΔNFE	-1.82	-3.10	-2.73	$+1.92$
	Wrong to correct / correct to wrong	2 / 0	3 / 1	7 / 2	9 / 4
AIME 2025	ΔAcc. (pp)	$+1.11$	$+3.33$	$+2.22$	$+2.22$
	ΔNFE	-6.76	-6.36	$+16.49$	$+16.90$
	Wrong to correct / correct to wrong	2 / 1	3 / 0	5 / 3	7 / 5
MATH-500	ΔAcc. (pp)	$+0.07$	-0.07	-0.21	$+0.28$
	ΔNFE	-0.04	$+0.29$	$+0.16$	-0.23
	Wrong to correct / correct to wrong	2 / 1	4 / 5	7 / 10	11 / 7
GSM8K	ΔAcc. (pp)	—	-0.13	—	-0.13
	ΔNFE	—	-0.37	—	-0.39
	Wrong to correct / correct to wrong	—	0 / 1	—	2 / 3
AtCoder	ΔAcc. (pp)	—	$+0.36$	$+0.54$	$+1.45$
	ΔNFE	—	-0.87	-0.73	-0.02
	Wrong to correct / correct to wrong	—	4 / 2	7 / 4	15 / 7
LeetCode	ΔAcc. (pp)	—	-0.30	$+0.30$	-1.80
	ΔNFE	—	-1.24	-0.18	-1.27
	Wrong to correct / correct to wrong	—	1 / 2	3 / 2	4 / 10

Table 10: Planning-time sensitivity relative to RESTART, with problem-level 95% bootstrap intervals. Bounds are $(\omega_{\text{lo}}, \omega_{\text{hi}})$, with $t < 12$; $(.25, .45)$ is the main setting.

Dataset	Timing bounds	ΔAcc. (pp) [95% CI]	ΔNFE [95% CI]	Rollouts with a plan (%)
AIME 2024	$(.10, .30)$	$+2.22$ $[-4.44, +10.00]$	-5.42 $[-23.58, +10.74]$	52.2
	$(.25, .45)$	$+2.22$ $[-2.22, +7.78]$	-3.10 $[-9.30, +2.92]$	43.3
	$(.40, .60)$	$+0.00$ $[+0.00, +0.00]$	$+1.08$ $[-0.40, +2.82]$	23.3
AIME 2025	$(.10, .30)$	$+3.33$ $[+0.00, +7.78]$	-5.58 $[-30.26, +19.41]$	36.7
	$(.25, .45)$	$+3.33$ $[+0.00, +8.89]$	-6.36 $[-32.63, +16.20]$	41.1
	$(.40, .60)$	-2.22 $[-5.56, +0.00]$	$+15.52$ $[-0.54, +39.03]$	34.4
GSM8K	$(.10, .30)$	$+0.13$ $[-0.27, +0.67]$	$+0.01$ $[-0.13, +0.15]$	27.3
	$(.25, .45)$	-0.13 $[-0.40, +0.00]$	-0.37 $[-1.08, +0.04]$	12.4
	$(.40, .60)$	$+0.27$ $[+0.00, +0.67]$	-0.00 $[-0.08, +0.09]$	5.8
MATH-500	$(.10, .30)$	$+0.21$ $[-0.28, +0.71]$	$+0.28$ $[-0.99, +1.72]$	37.0
	$(.25, .45)$	-0.07 $[-0.50, +0.36]$	$+0.29$ $[-0.30, +0.93]$	24.4
	$(.40, .60)$	$+0.28$ $[+0.00, +0.64]$	-0.05 $[-0.96, +0.85]$	14.5

Table 11: Interruption-depth comparisons on common subsets of n paired rollouts. Depth c counts completed canvases in the previous answer. Differences use matched RESTART outputs at each depth; the $c = 3$ subsets differ from the full 90-pair main evaluation.

			Accuracy (%)					
Dataset	c	n	RESTART	HiPE	ΔAcc. (pp)	ΔNFE	Wrong to correct / correct to wrong	With a plan (%)
AIME 2024	3	78	52.56	55.13	+2.56	−2.41	3 / 1	43.6
AIME 2024	4	78	57.69	57.69	+0.00	−3.63	0 / 0	25.6
AIME 2024	5	78	46.15	48.72	+2.56	−8.96	2 / 0	14.1
AIME 2025	3	84	54.76	58.33	+3.57	−6.79	3 / 0	42.9
AIME 2025	4	84	51.19	55.95	+4.76	−19.45	4 / 0	34.5
AIME 2025	5	84	51.19	51.19	+0.00	+14.32	0 / 0	17.9

E ADDITIONAL CONTROLS AND INITIALIZATION COMPARISON

E.1 SDEDIT ADAPTATION AND CLEAN-TEXT INITIALIZATION

Our adaptation of SDEdit (Meng et al., 2022) applies its noise-then-denoise principle to token canvases. It changes the initial canvas while keeping the updated context, including a_1, and the original sampler. For canvas k, it uses the corresponding segment of a_1, excluding the end-of-answer tail. With noise ratio $\rho = 0.5$ — *random warm start* in §5.1 — it retains source tokens at a randomly selected half of the available positions and leaves the others at their original uniform random draws. Positions beyond the source text also remain random. Here ρ is the fraction of source positions left as uniform token noise; the original decoding schedule and step budget are retained. Source tokens are not fixed during subsequent denoising.

The clean-text initialization control, *full warm start*, uses $\rho = 0$, retaining all available source tokens at initialization. The corruption mask uses a separate random generator, preserving the sampler's random draws. For a partial segment of length n, the number of noised positions is $\mathrm{round}(\rho n)$.

Both variants use thinking-enabled generation and the matched contexts and pairs of the main evaluation. Table 1 reports mathematics and coding results; Table 12 adds the available problem-level intervals against RESTART. These intervals do not estimate differences between initialization variants or against HiPE.

Table 12: Random ($\rho = 0.5$) and full ($\rho = 0$) warm starts relative to RESTART, with problem-level 95% bootstrap intervals. Accuracy differences are percentage points. LiveCodeBench intervals cover only full warm start.

Dataset	ρ	ΔAcc. [95% CI]	ΔNFE [95% CI]
GSM8K	0	−2.69 [−5.51, +0.00]	+7.63 [+5.73, +9.47]
	0.5	−0.67 [−3.49, +2.15]	+4.79 [+3.03, +6.57]
MATH-500	0	−13.01 [−16.35, −9.81]	+16.33 [+11.25, +21.63]
	0.5	−10.02 [−12.94, −7.11]	+11.49 [+5.96, +17.16]
AIME 2024	0	−3.33 [−16.67, +10.00]	+18.41 [−21.83, +62.79]
	0.5	−4.44 [−17.78, +8.89]	+5.21 [−23.74, +34.28]
AIME 2025	0	−6.67 [−21.11, +6.67]	+10.08 [−50.17, +68.67]
	0.5	−13.33 [−27.78, +0.00]	+5.61 [−53.95, +67.27]
LiveCodeBench	0	−13.90 [−17.74, −10.17]	+21.29 [+11.44, +31.97]

Initialization and intermediate injection differ in timing, location, and amount of source text, so their comparison does not isolate individual selection rules.

E.2 CONFIDENCE-SELECTED INITIALIZATION

This baseline, *confidence warm start*, tests whether token confidence improves the choice of source positions used at initialization. For each canvas with an available segment of a_1, it constructs a separate probe canvas containing that segment at its original positions and runs one decoder forward pass under the updated context and the current a_2 prefix. It ranks the source tokens by their token log-probability and retains the highest-scoring $\lfloor n/2 \rfloor$ of the n available tokens in the actual initial canvas, leaving the other positions at uniform random draws. The sampler then proceeds without further reinsertion. Probe calls are included in NFE, one for each canvas with a nonempty reference segment; they do not advance the generation random-number state.

Results are reported in Table 1. The probe evaluates a clean reference canvas, whereas HiPE selects spans against the evolving denoising state.

E.3 PER-SEED CONTROL RESULTS

Table 13 reports AIME random-token and random-position results on 90 paired rollouts per year. Some randomizations exceed HiPE in accuracy, so the aggregate comparison does not imply an advantage for every seed. Control-seed standard deviations measure randomization variability, separately from uncertainty across source problems.

Table 13: Per-seed AIME results for random-token and random-position controls. SD denotes the sample standard deviation across control seeds.

		AIME 2024		AIME 2025	
Control	Seed	Acc. (%)	Mean NFE	Acc. (%)	Mean NFE
Random tokens	1	54.44	218.7	58.89	345.4
	2	51.11	221.5	57.78	342.7
	3	52.22	225.4	58.89	344.8
	mean	52.59	221.9	58.52	344.3
	SD	1.70	3.4	0.64	1.4
Random positions	0	53.33	215.9	57.78	353.3
	1	54.44	215.4	61.11	353.5
	2	52.22	220.8	57.78	343.0
	mean	53.33	217.4	58.89	349.9
	SD	1.11	3.0	1.92	6.0
HiPE	—	53.33	216.5	60.00	339.9

Random-token controls use noise seeds 1–3 in all mathematics tasks. Random-position controls use position seeds 0–2 in all mathematics tasks. These are distinct from generation seeds 42–44.

F ADDITIONAL ANALYSES

These analyses use the main setting of each task. Plan frequencies count nonempty plans as defined in Appendix C.

F.1 SELECTION PATTERNS

The mathematics candidate-margin distributions in Figure 4 use 78 AIME24, 84 AIME25, and 1,407 MATH-500 rollouts, compared with 90 per AIME year in the main evaluation. The code distributions use the 859 of 885 LiveCodeBench rollouts with candidate records at two bits. Records pool candidates across canvases and rollouts; upper-tail selection reflects the margin filter and does not establish correctness under the update.

F.2 Correctness and NFE changes

Across 180 paired AIME rollouts, HiPE changes six incorrect answers to correct and one correct answer to incorrect. The six improvements occur on four distinct problems.

Problem-level median NFE differences are zero on all four mathematics tasks, and the corresponding confidence intervals include zero (Table 8). On AIME25, problem 9 alone contributes -9.26 forward passes to the overall mean difference of -6.36, partly offset by increases elsewhere. In its seed-44 rollout, HiPE corrects the answer while reducing NFE from 1,101 to 268. Conversely, on AIME24 problem 15, seed 44, it corrects the answer while increasing NFE from 178 to 290. Mean savings therefore do not describe a typical problem, and correctness gains need not reduce computation.

F.3 Plan frequency by canvas index

Table 14 reports, for each canvas index k within an answer, the share of rollouts in which a plan was formed and the change in the best candidate's mean log-probability relative to the first canvas. Plan frequencies in this table and Figure 4 are computed among rollouts that reach the corresponding canvas. The change in log-probability is computed within rollouts that reach at least nine canvases and have candidates on at least five of them, so the comparison is between canvases of the same answer. The best fit declines with k in 84% of AIME 2024 rollouts and 80% of AIME 2025 rollouts (median slope -0.045 and -0.014 nats per canvas, Wilcoxon $p < 10^{-4}$ for both). Figure 4 in the main text plots the plan frequency for the four mathematics tasks and the two code subsets. The decline is small per canvas, but admission is a threshold on the margin, so the share of admitted candidates falls faster than the fit.

Table 14: Plan frequency and change in the best candidate's mean log-probability by canvas index k. $\Delta\ell$ is measured in nats relative to $k = 0$.

	AIME 2024		AIME 2025	
k	with a plan (%)	$\Delta\ell$ of best candidate	with a plan (%)	$\Delta\ell$ of best candidate
0	17.8	—	13.3	—
1	6.7	-0.29	12.2	-0.47
2	5.6	-0.74	4.4	-0.41
3	5.6	-0.73	4.4	-0.89
4	3.5	-0.86	2.3	-0.88
5	0.0	-0.59	4.7	-0.86
6	3.0	-1.05	2.4	-1.30
7	0.0	-1.40	3.9	-0.79
8	0.0	-1.98	1.4	-0.74

G Qualitative example

Figure 5 shows one rollout in which HiPE forms a plan. The selected span comes from the intersection-point derivation of the interrupted answer and is written into the new answer during the second canvas. The final answer keeps the structure of that derivation while the coordinates follow the updated plane. RESTART reaches an incorrect answer after 1,101 forward passes, and HiPE reaches the reference answer after 268. The logs record planned positions and final outputs, not the intermediate states the decoder read, so the example shows that the final output changes after injection and does not show which injected tokens survived.

User

```
The set of points in $3$-dimensional coordinate space that
lie in the plane $x+y+z=72$ whose coordinates satisfy the
inequalities\[x-yz<z-xy<y-zx\]forms four disjoint convex regions.  ...
Find $b.$
```

Assistant (interrupted) — the highlighted span is selected and written into canvas position 151

```
...  The boundaries are the lines:  $l_1:  2x+y=72 \implies y =
-2x+72$ $l_2:  y=-1$ $l_3:  x+2y=72 \implies y = -1/2x+36$ $l_4:
x=-1$ Let's find the intersection points of these four lines:  $
l_1 \cap l_2$:  $y=-1, 2 x-1=72 \implies 2x=73 \implies x=36.5$.  Point
$(36.5, -1)$.  $l_1 \cap l_3$:  $2x+y=72, x+2y=72$.  Subtracting gives
$x-y=0 \implies x=y=24$.  Point $(24, 24)$.  ...
```

User (update)

```
Use the plane x + y + z = 75.  Change the inequality chain to x - yz
< y - zx < z - xy.  The set now forms three disjoint convex regions.
Keep that exactly one region has finite area and its area is of the
form a\sqrt b with b squarefree, but now compute a + b instead of b.
```

RESTART — answer 1, incorrect, 1,101 NFE

```
Let the plane be $x+y+z=75$.  The inequalities are $x-yz < y-zx <
z-xy$.  ...  Testing the regions, we find that the finite region in
the $xy$-plane is the triangle with vertices $B(25, 25)$, $D(77, -1)$,
and $C(-1, -1)$.  ...  $$\text{Area}_{xy} = \frac{1}{2} |25(-1 - (-1)) +
77(-1 - 25) + (-1)(25 - (-1))|$$ $$\text{Area}_{xy} = \frac{1}{
```

HiPE — answer 510, correct, 268 NFE

```
Let the plane be $x+y+z=75$.  The inequalities are $x-yz < y-zx <
z-xy$.  ...  Testing the regions, we find that the finite region is the
triangle with vertices $B(25, 25)$, $F(-1, 38)$, and $C(-1, -1)$.  ...
Thus, $a=507$ and $b=3$.  The value $a+b$ is $507+3 = 510$.
```

Figure 5: A rollout from AIME 2025 with a plan. The update moves the plane, reorders the inequality chain, and asks for $a + b$ instead of b. The highlighted span of the interrupted answer is written into the new answer during the second canvas. Both answers are generated under the same updated context, which includes the interrupted answer. Ellipses mark omitted text.

H LIMITATIONS

Our evaluation is limited to one backbone and one family of condition-update tasks. HiPE relies on the backbone's acceptance and re-noising operations, and its effectiveness with other models and samplers remains untested. Our experiments use predefined update messages and interruption points. Performance under naturally occurring user updates, including ambiguous or incomplete revisions, remains unexplored. Finally, our evaluation considers a single context update per answer; adaptation to repeated updates remains unexplored.