RFLASH: RELAXED SPECULATIVE DECODING WITH DRIFT CONSTRAINT

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding losslessly accelerates memory-bound decoding of large language models (LLMs) by using a lightweight draft model to speculate future tokens of the large target model. Relaxed speculative decoding further accelerates decoding by accepting draft tokens that lossless verification would reject. Despite the increase in acceptance length, prior methods often fail to deliver any request-level speedup because they make the output longer. We trace this to a systematic drift toward the weak draft model, which accumulates over the generation and eventually collapses it. To address this, we propose RFlash, a training-free relaxation rule that caps the drift toward the draft, keeping long generations faithful to the target. On long-reasoning benchmarks, RFlash is the only relaxation rule we evaluate that is faster than lossless speculative decoding in every setting, by 6–10%, with output length and accuracy at the lossless level, delivering a practical, accuracy-preserving speedup.

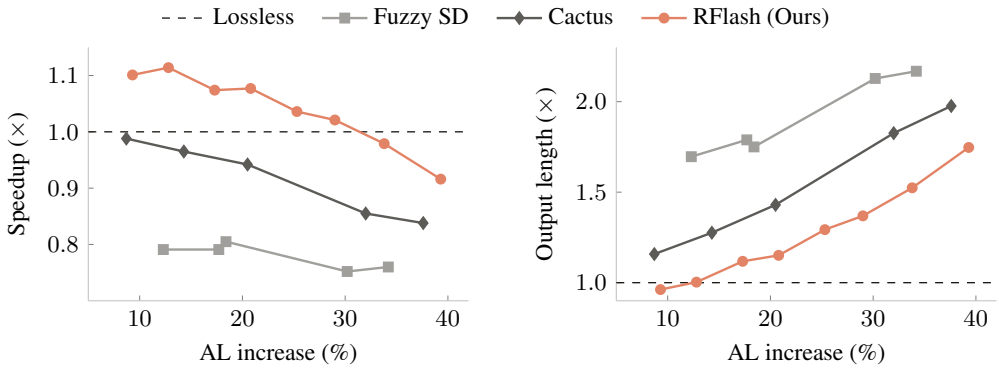
1 INTRODUCTION

Large language models (LLMs) are widely used across a broad range of tasks, including autonomous agents, programming, and conversational chatbots. The growing demand for LLMs necessitates large-scale deployment of LLM inference. However, LLM inference is dominated by memory-bound autoregressive decoding, in which tokens are generated sequentially rather than in parallel, resulting in low GPU utilization. Speculative decoding (Leviathan et al., 2023; Li et al., 2025; Chen et al., 2026; Cheng et al., 2026; Inco AI, 2026) mitigates this bottleneck by using a lightweight draft model to *speculate* the future tokens of the target model and then *verifying* the proposed tokens with the target model in a single forward pass.

Standard speculative decoding is strictly *lossless*: it exactly preserves the output distribution of the target model. However, the acceptance rate is inherently bounded under the lossless constraint, because the tiny draft model has limited capacity compared to the large target model, making it impossible to increase the acceptance rate indefinitely. To further increase decoding speedup beyond the lossless limit, *relaxed* speculative decoding methods loosen the strict verification constraint, either by training a lightweight classifier on the target model's hidden states to accept mismatched draft tokens that are deemed harmless (Bachmann et al., 2025; Garipov et al., 2026), or by reshaping the target distribution to favor draft tokens within a bounded divergence (Hao & Mou, 2026). Either way, they raise the acceptance length by pulling the output distribution toward the draft, thereby accepting additional tokens that lossless verification would have rejected.

Despite the reported per-token speedups, existing relaxation methods largely overlook the generation length, which affects request latency as much as per-token speed does. Even when they do preserve downstream accuracy, these methods make generations longer, and the inflation outweighs the gain in acceptance. As Figure 1 shows, most prior relaxation methods fail to deliver any request-level speedup (below $1\times$), leaving them with no practical advantage over lossless decoding. What makes their outputs longer? Our key insight is that prior methods bound *how far* the distribution may move at each token, but not *in which direction*. The cheapest way to raise acceptance is to shift probability mass toward the draft, so the per-token deviations all point the same way and accumulate with the generation length. We measure this drift with a statistic we establish in Section 4: the score of the optimal classifier that tells whether a text was generated by the draft or by the target, *i.e.*, the log-likelihood ratio of the two, measured against what the target's own sampling would produce, so

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.



(a) Speedup *vs*. AL increase.

(b) Output length *vs*. AL increase.

Figure 1: **(a)** Request-level speedup relative to lossless speculative decoding and **(b)** output length relative to lossless decoding of RFlash, Fuzzy SD (Holsman et al., 2025), and Cactus (Hao & Mou, 2026) versus the increase in acceptance length, on GPQA Diamond (Rein et al., 2023) with Gemma-4 12B (Gemma Team et al., 2026) and a DFlash (Chen et al., 2026) drafter.

that it has zero mean under lossless decoding and grows only as a relaxation rule pushes the output toward the draft. With it, we quantify how this systematic drift affects the output length, and find that the output grows steadily longer with the drift (Figure 2a). This suggests that a practical relaxation rule should constrain the direction of its deviation, not only its size.

Inspired by this insight, we design RFlash, a training-free relaxed speculative decoding method that constrains the drift toward the draft in addition to the per-token divergence, translating its higher acceptance length into practical request-level speedup. RFlash caps how much of each step's divergence may be spent as drift toward the draft, and otherwise takes the most acceptance the budget allows; the cap is solved in closed form at negligible cost. This cap keeps the output faithful to the target, especially in long-horizon reasoning, where the drift of prior rules accumulates. Across two target models and three long-reasoning benchmarks, RFlash is the only rule that is faster than lossless speculative decoding in every setting, by 6–10%, with output length and accuracy at the lossless level, whereas every baseline pays for its acceptance with longer outputs and often ends up slower than lossless decoding. The gain of RFlash also carries over to four drafters of different designs.

Our contributions are as follows: (1) to our knowledge, we are the first to evaluate relaxed speculative decoding by request-level speedup and show that prior methods lose their gain in acceptance to longer outputs; (2) we establish a drift statistic, the log-likelihood ratio of draft to target measured against the target's own sampling, and find that output length grows steadily with it, revealing the failure mode of prior relaxation methods; (3) we propose RFlash, which caps the drift at a multiple of the divergence spent and is solved in closed form per verification block, and which is the only relaxation rule we evaluate that is faster than lossless speculative decoding in every setting.

2 Related Work

Speculative decoding. Speculative decoding mitigates the memory-bound autoregressive decoding in LLM inference by using a lightweight draft model to propose several future tokens, which the target model then verifies in parallel within a single forward pass, using a rejection-sampling rule that provably preserves the target distribution (Leviathan et al., 2023; Chen et al., 2023). The speedup is therefore governed by how closely the draft matches the target, and most work on speculative decoding improves the draft: extra decoding heads (Cai et al., 2024), lightweight autoregressive drafters (Li et al., 2024a;b; 2025), multi-token prediction modules trained with the target (Liu et al., 2024), and parallel drafters that propose all positions at once (Chen et al., 2026; Cheng et al., 2026; Inco AI, 2026). RFlash is orthogonal to these and applies on top of any draft.

Relaxed speculative decoding. Relaxation rules trade exact preservation of the target for acceptance. One line loosens the acceptance test directly: accepting tokens whose target probability exceeds an entropy-dependent threshold (Cai et al., 2024), accepting when the draft and target

distributions are close enough (Leviathan et al., 2023; Holsman et al., 2025), verifying against a mixture of target and draft (Fu et al., 2025; Wang et al., 2026), deciding per token whether to defer to the target (Narasimhan et al., 2025), or maximizing the acceptance probability under a per-token divergence budget (Hao & Mou, 2026). Another line learns the decision, with a classifier that accepts mismatched but harmless tokens (Bachmann et al., 2025; Garipov et al., 2026) or by accepting whole reasoning steps on the verdict of a reward model (Liao et al., 2025) or of the target itself (Pan et al., 2026). Prior rules bound at most how far the distribution may move from the target at each token, but not the direction in which it moves.

3 Preliminaries

Lossless verification. A generation is a token sequence $y = y_1, y_2, \ldots$; at step t the target and the draft define next-token distributions p_t and q_t given the prefix $y_{<t}$. Decoding proceeds in blocks of M positions; we index positions by j within a block and by t along the generation, and drop the index for a single position. The draft proposes $x_1, \ldots, x_M$ from distribution $q_1, \ldots, q_M$; one target forward pass yields distribution $p_1, \ldots, p_M$, and the proposals are verified in order by rejection sampling: position j is accepted with probability

$$a_j \;=\; \sum_y \min\big(p_j(y), q_j(y)\big), \tag{1}$$

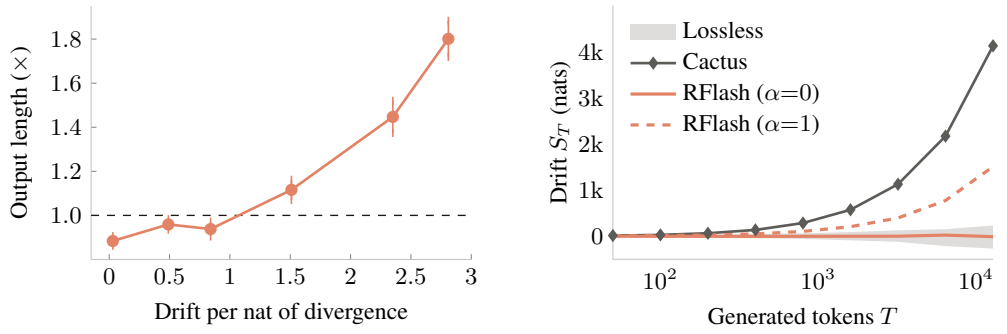
on average over its proposal $x_j \sim q_j$, is reached with probability $w_j = \prod_{i<j} a_i$ when positions are treated as independent, and on the first rejection a correction token is drawn from the residual of p_j against q_j and the block ends. The expected acceptance length per block is $\sum_j w_j a_j$, and every emitted token is an exact sample from the target.

Relaxed verification as a modified target. Every token-level relaxation rule discussed in Section 2 can be written the same way: it replaces p_j by a modified target $\hat{p}_j$ and runs the standard test, so the emitted token at a reached position is a sample from $\hat{p}_j$ and the acceptance probability is $a_j(\hat{p}_j) = \sum_y \min(\hat{p}_j, q_j)$; any accept-or-resample rule has such a $\hat{p}_j$, with the same output distribution and at least the same acceptance (Lemma 2). A relaxation rule is thus a choice of $\hat{p}_j$ at each position. Every lemma and proposition cited in Sections 3–5 is proved in Appendix A.1.

4 Drift Toward the Draft

By Equation (1), acceptance at a position can rise only if $\hat{p}_j$ moves mass toward tokens the draft favors over the target (Lemma 3). Every gain in acceptance is thus a step toward the draft. The steps are small. But they all point the same way, and they lead to the draft's own regime. The draft is a much weaker model. What it does confidently is the low-entropy part of the target's behavior, *e.g.*, repeating a phrase or a whole paragraph, restating the problem, or enumerating test cases instead of solving the problem. Once the text enters such a mode, draft and target agree on continuing it. Every proposal is accepted, and the mode sustains itself. The output has collapsed onto a narrow, sharply peaked slice of the target distribution that the draft can reproduce. This is the length inflation of Figure 1. It is paid directly in the time to finish the request: every extra token costs a target forward pass, whatever the acceptance rate. Figure 2a isolates the cause. At equal divergence per token, output length rises from $0.88\times$ to $1.8\times$ that of lossless decoding as the share of the divergence spent on drift grows. Prior rules bound the *size* of each step. This cannot prevent the collapse, because the direction is what matters.

This section makes two claims. Section 4.1: the collapse has a per-step rate, δ_t. It is exactly what a relaxation rule adds to the log-likelihood ratio of the emitted text at each step. Section 4.2: a divergence budget cannot control that rate. On real states, the direction that buys acceptance is exactly the drift direction.



(a) Output length *vs*. drift.

(b) Drift S_T *vs*. generated tokens.

Figure 2: **(a)** Output length relative to lossless decoding versus the realized drift per nat of divergence, for RFlash runs on GPQA Diamond with Gemma-4 12B and a DFlash drafter. Length rises from $0.88\times$ lossless at zero drift to $1.8\times$ when nearly all of the divergence is drift. The length inflation is set by the direction of the deviation, not by its size. **(b)** Drift (Equation (4)) of the generated text toward the draft, on LiveCodeBench with Gemma-4 12B and a DFlash drafter. Under lossless decoding it is a martingale, mean zero and spread $\propto \sqrt{T}$. Cactus leaves the band within a hundred tokens and drifts linearly; RFlash with drift allowance $\alpha = 0$ keeps the drift at zero mean; $\alpha > 0$ lets it grow at the divergence it spends.

4.1 The drift and its per-step rate

At each step, a relaxation rule shifts the expected increment of the log-likelihood ratio of the emitted text by a single number, the *drift rate*

$$\delta_t \;=\; \mathbb{E}_{\hat{p}_t}[z_t] - \mathbb{E}_{p_t}[z_t], \qquad z_t(y) = \log \frac{q_t(y)}{p_t(y)}. \tag{2}$$

Here is why this is the right number. "How draft-like has the text become" asks whether a generation $y = y_1, \ldots, y_T$ was written by the target or by the draft. This is a binary classification. By the Neyman–Pearson lemma, no statistic separates the two better than the log-likelihood ratio of the two generation distributions.* By the chain rule, it is

$$\log \frac{Q(y)}{P(y)} \;=\; \sum_{t \le T} z_t(y_t). \tag{3}$$

This statistic moves even when nothing is wrong. Under lossless decoding $\mathbb{E}_{p_t}[z_t] = -D_{\mathrm{KL}}(p_t \,\|\, q_t) \le 0$, so the target's own text looks more and more like the target's as it gets longer. The drift is what is left after we subtract, at every step, the increment the target's own sampling would produce:

$$S_T \;=\; \sum_{t \le T} \Big[z_t(y_t) \;-\; \mathbb{E}_{p_t}\big[z_t\big] \Big]. \tag{4}$$

Let $\mathcal{F}_t = \sigma(y_1, \ldots, y_t)$. The functions p_t, q_t, z_t depend only on the prefix, so they are $\mathcal{F}_{t-1}$-measurable. The increment of S_T at step t therefore has conditional mean $\mathbb{E}[z_t(y_t) \mid \mathcal{F}_{t-1}] - \mathbb{E}_{p_t}[z_t]$. This is zero if $y_t \sim p_t$, and δ_t if $y_t \sim \hat{p}_t$. Hence the Doob decomposition

$$S_T \;=\; M_T + \sum_{t \le T} \delta_t, \qquad \mathbb{E}\big[M_t - M_{t-1} \,\big|\, \mathcal{F}_{t-1}\big] = 0. \tag{5}$$

The martingale M_T has mean zero and is of order $\sqrt{T}$ under every rule, including lossless (Proposition 1). Everything a relaxation rule adds is the second term. If δ_t keeps one sign, the mean of S_T grows linearly in T. If $\delta_t = 0$ at every step, S_T is a martingale, and the log-likelihood ratio grows at exactly the rate the target's own sampling would produce on the same prefix. Figure 2b plots S_T.

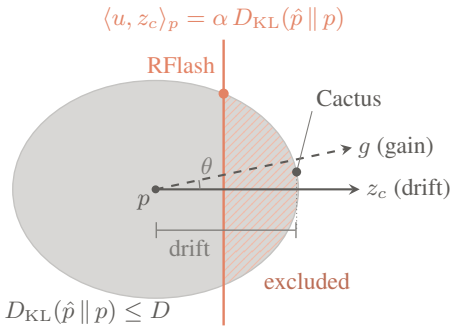


Figure 3: The geometry of a single decoding step in $L^2(p)$: the target p at the origin, the drift direction z_c horizontal. A deviation u raises acceptance by its component along g and adds its component along z_c to the drift; a divergence budget only confines u to a convex set around p (gray). As g and z_c are nearly parallel ($\cos\theta \approx 0.97$), Cactus, taking the point with the largest acceptance gain, converts almost the whole budget into drift. On top of the budget, RFlash caps the drift at α times the divergence (coral line; hatched part excluded) and takes the best remaining point.

4.2 A DIVERGENCE BUDGET ALONE DOES NOT CONTROL THE DRIFT

Prior rules bound how far each step may move, measured by the KL divergence $D_{\mathrm{KL}}(\hat{p}_t\,\|\,p_t)$ of the modified target from the target. Such a budget cannot control δ_t. The reason is that on real verification states, the direction that buys acceptance is, up to a small angle, the drift direction. To see this, we put the deviation, the drift and the acceptance gain into one coordinate system. Everything below concerns a single position. We therefore drop the step index t and write p, q, $\hat{p}$, z for p_t, q_t, $\hat{p}_t$, z_t. The deviation relative to the target is

$$u(y) \;=\; \frac{\hat{p}(y)}{p(y)} - 1, \qquad \hat{p} = p\,(1+u), \quad \mathbb{E}_p[u] = 0. \tag{6}$$

We view every function of the token, u and z included, as a random variable in $L^2(p)$, with

$$\langle f, h\rangle_p = \mathbb{E}_p[f\,h], \qquad \|f\|_p = \langle f, f\rangle_p^{1/2}, \qquad \mathbb{E}_{\hat{p}}[f] \;=\; \mathbb{E}_p[f] + \langle u, f\rangle_p. \tag{7}$$

The last identity is $\mathbb{E}_{\hat{p}}[f] = \mathbb{E}_p[(1+u)f]$. It says that moving the target to $\hat{p}$ shifts every expectation by the projection of u on f, *i.e.*, by the covariance of u and f under the target. Inner products in $L^2(p)$ are covariances, and angles are correlations. We use both readings.

Drift, gain and divergence in $L^2(p)$. Applying Equation (7) to the drift rate and the acceptance gain makes each a projection of u on a fixed direction. The divergence is the squared length of u (Figure 3).

Drift. Take $f = z$. Centering z costs nothing, since $\mathbb{E}_p[u] = 0$, so

$$\delta \;=\; \langle u,\, z_c\rangle_p, \qquad z_c = z - \mathbb{E}_p[z]. \tag{8}$$

Acceptance gain. Mass added to a token with $p(y) < q(y)$ is accepted. A token with $p(y) \ge q(y)$ already has all of its draft mass accepted and gains nothing. So, as long as no token is pushed past the draft,

$$a(\hat{p}) - a(p) \;=\; \langle u,\, g\rangle_p, \qquad g = \mathbb{1}[p < q] - \mathbb{E}_p\big[\mathbb{1}[p < q]\big]. \tag{9}$$

Otherwise the right-hand side is an upper bound (Lemma 3). g is the acceptance-gain direction: the deviation that buys the most acceptance per unit of length.

Divergence. The KL divergence spent is, to second order (Lemma 4),

$$D_{\mathrm{KL}}(\hat{p}\,\|\,p) \;=\; \tfrac{1}{2}\,\|u\|_p^2 + O\big(\mathbb{E}_p|u|^3\big). \tag{10}$$

The drift and the gain are first order in u. The divergence is second order. The drift and the gain are read off the same u along two directions, z_c and g. What remains is the angle between them. It is the correlation, under the target, between the log-likelihood ratio and the indicator of the tokens the draft favors.

*The draft proposes from a small candidate set and puts no mass elsewhere, where $\log q_t/p_t$ would be $-\infty$. Throughout we use the smoothed draft $(1-\epsilon)\,q_t + \epsilon\,p_t$, $\epsilon = 10^{-3}$, so that every z_t is finite. Our results are insensitive to the value of ϵ.

A budget-only rule drifts by construction. A divergence budget bounds the *length* of u. By Cauchy–Schwarz, it bounds the drift only as

$$|\delta| \;\le\; \|u\|_p \, \|z_c\|_p \;\approx\; \sqrt{2\, D_{\mathrm{KL}}(\hat{p}\|p)}\; \|z_c\|_p, \tag{11}$$

with equality when u is a multiple of z_c. Here $\|z_c\|_p$ is the standard deviation of z under the target, a property of the state. Call a rule that maximizes acceptance under a divergence budget and nothing else a *budget-only* rule. Cactus is one. A budget-only rule moves along g. And g is the centered indicator of $\{p < q\} = \{z > 0\}$, *i.e.*, the sign of z. A random variable and its sign are positively correlated under any distribution. The correlation falls short of one only to the extent that $|z|$ is spread unevenly within each sign (Lemma 5). So the angle θ between g and z_c is small by construction. A small angle means that the direction that buys acceptance is nearly the drift direction. A budget-only rule therefore spends almost its entire divergence budget on drift. At every position it reaches a fraction $\cos\theta$ of the largest drift the budget allows (Lemma 6). The increments share a sign, so by Equation (5) S_T grows linearly in T, however small the budget.

Empirically, the angle is even smaller. On real verification states where the draft and target disagree, the correlation $\cos\theta$ between g and z_c has median 0.97 for Qwen3.8 27B. At the median state, a budget-only rule attains 97% of the largest drift its budget allows. Controlling the drift therefore requires constraining the component of u along z_c directly. This is exactly what RFlash implements.

5 Method

5.1 The drift constraint

Prior rules bound only the length of the deviation u. RFlash also caps its component along z_c, *i.e.*, the drift it adds. The cap is a multiple of the KL divergence the position spends:

$$\langle u, z_c\rangle_p \;\le\; \alpha\, D_{\mathrm{KL}}(\hat{p}\,\|\,p), \qquad \alpha \ge 0. \tag{12}$$

We call α the *drift allowance*. It is the drift a position may add per nat of divergence it spends. At $\alpha = 0$ no drift is allowed. By Equation (5), S_T then cannot grow in expectation, and the log-likelihood ratio of the emitted text grows no faster than under the target's own sampling. As $\alpha \to \infty$ the constraint disappears and the rule becomes budget-only: it maximizes acceptance under the KL budget alone, as Cactus does. In between, the drift is tied to the divergence. It grows at most at α times the rate at which divergence is spent. That rate is second order in the deviation, whereas the drift of a budget-only rule grows at the first-order rate of Equation (11). In the geometry of Section 4.2, by Equation (10) the constraint reads $\cos\theta(u, z_c) \le \alpha\, \|u\|_p/(2\|z_c\|_p)$. The allowed angle shrinks with the deviation: at the small per-token deviations of practice, u must be nearly orthogonal to z_c for any α, whereas a budget-only rule has $\cos\theta \approx 1$.

Because g and z_c are nearly parallel, imposing the constraint one position at a time wastes most of the divergence budget on nothing. A position that must stay neutral by itself can only use the part of g orthogonal to z_c, which has length $\|g\|_p \sin\theta$. It keeps only a fraction $\sin\theta$ of the gain per unit length. For the same acceptance it pays $1/\sin^2\theta$ times the divergence (Lemma 6), an order of magnitude at the measured angles. The room is across the positions of a block.

5.2 The block program

The constraint is imposed per verification block, on the reach-weighted sum over its positions. This sum is what the walk actually accumulates. Throughout, the divergence a position spends is $D_{\mathrm{KL}}(\hat{p}_j \,\|\, p_j)$, the KL divergence of the modified target from the target. For a block with target distributions $p_1, \ldots, p_M$, RFlash solves

$$\begin{aligned} \max_{\hat{p}_1,\ldots,\hat{p}_M} \quad & \sum_j w_j\, a_j(\hat{p}_j) \;-\; \lambda \sum_j w_j\, D_{\mathrm{KL}}(\hat{p}_j \,\|\, p_j) \\ \text{s.t.} \quad & \sum_j w_j \left[\langle u_j, z_j\rangle_{p_j} - \alpha\, D_{\mathrm{KL}}(\hat{p}_j \,\|\, p_j)\right] \le 0, \qquad w_j = \prod_{i<j} a_i(\hat{p}_i), \end{aligned} \tag{13}$$

with $u_j = \hat{p}_j/p_j - 1$. The objective is the expected accepted length minus the KL divergence spent along the walk, priced at λ. The constraint is the drift condition Equation (12), accumulated with the

rule's own reach weights. The method has two hyperparameters, set once and held fixed: the price λ and the allowance α. With these weights, the block's expected drift is at most α times its expected divergence. The guarantees of Section 5.1 therefore hold at block resolution. S_T sampled at block boundaries, minus α times the divergence spent, is a supermartingale (Proposition 9). At $\alpha = 0$ the constraint always binds, so the drift sampled at block boundaries is a martingale, exactly as under lossless decoding (Proposition 8).

Closed form. The block has one drift constraint, so it has one multiplier $\mu \geq 0$: the price the block pays per nat of drift. Unlike λ and α, it is not a hyperparameter. It is solved for at decode time, once per block, from the constraint itself. Position j sees both prices scaled by its expected remaining run $R_j = 1 + a_{j+1}R_{j+1}$ ($R_M = 1$), because its acceptance is multiplied by everything that follows: $\lambda_j = \lambda/R_j$ and $\mu_j = \mu/R_j$. We write $\Lambda_j = \lambda_j - \alpha\mu_j$ for the net price of divergence once the allowance is credited. Holding the reach weights fixed, the stationary point of Equation (13) is (Proposition 7)

$$\hat{p}_j(y) \;=\; p_j(y)\, r_j(y), \qquad r_j(y) \;=\; \min\Big\{ \max\Big(\tfrac{q_j(y)}{p_j(y)},\; A_j(y)\, x_j\Big),\; A_j(y)\, x_j\, e^{1/\Lambda_j} \Big\}, \tag{14}$$

with $A_j(y) = e^{-\mu_j z_j(y)/\Lambda_j}$ and $x_j > 0$ fixed by $\sum_y p_j r_j = 1$. The rule has two parts. The tilt A_j scales every token and leans the whole row against the draft: down on the tokens the draft favors ($z_j > 0$), up on those it disfavors. The raise e^{1/Λ_j} then lifts every token that still has acceptance to gain, up to the draft's own probability. At $\mu = 0$ the tilt is flat and only the raise remains. This is the budget-only rule. The multiplier μ is what makes the drift of the raises cancel across the block.

Solving for μ at decode time. Fix μ. Then Equation (14) gives every $\hat{p}_j$ of the block in closed form. Here x_j is exact, because $\sum_y p_j r_j$ is piecewise linear in x_j with known breakpoints. This yields the block's constraint slack,

$$D(\mu) \;=\; \sum_j w_j \left[\langle u_j, z_j \rangle_{p_j} - \alpha\, D_{\mathrm{KL}}(\hat{p}_j \,\|\, p_j) \right] \quad \text{evaluated at the solution of Equation (14).} \tag{15}$$

D is continuous, and $D(0)$ is the drift excess of the budget-only rule. If $D(0) \leq 0$, the constraint is slack and $\mu = 0$. Otherwise μ is the root of $D(\mu) = 0$. We find it by a one-dimensional search on $[0, \lambda/\alpha)$. Each step of the search is one closed-form evaluation (Proposition 8). The reach weights w_j and runs R_j depend on the solution too. They are initialized from the lossless acceptances, and the whole solve is repeated two or three times to a fixed point.

5.3 Verification and cost

Verification is the standard test with $\hat{p}_j$ in place of p_j, residual sampling included. Nothing else in the decoding loop changes. Equation (14) is solved in one fused kernel per verification step. On an NVIDIA GB300 GPU with Qwen3.8 27B in FP8 and a DFlash 2 drafter, the kernel costs below $0.5\,\%$ of the target's forward pass at every batch size we measured.

6 Evaluation

6.1 Setup

Models and baselines. We evaluate three target models with four drafter designs: Qwen3.8 27B (Qwen Team, 2026b) with a DFlash 2 (Inco AI, 2026) drafter (block size 8), Gemma-4 12B (Gemma Team et al., 2026) with a DFlash (Chen et al., 2026) drafter (block size 16), and Qwen3.5 4B (Qwen Team, 2026a) with all four: its native multi-token prediction (MTP) head (3 draft tokens), and DFlash, DSpark (Cheng et al., 2026) and DFlash 2 drafters (block size 16). We compare with Cactus (Hao & Mou, 2026), Fuzzy Speculative Decoding (FSD) (Holsman et al., 2025), and the token-level rule of Speculative Cascades (Narasimhan et al., 2025), each run at five operating points of $+5$ to $+25\%$ acceptance length over lossless decoding. The lossless reference is standard speculative sampling with the same drafter.

Method	AIME25				GPQA Diamond				LiveCodeBench			
	AL	Len	Spd.	Acc	AL	Len	Spd.	Acc	AL	Len	Spd.	Acc
Qwen3.8 27B DFlash 2	4.32	1×	4.05×	98.0	3.78	1×	3.49×	89.7	3.79	1×	3.40×	90.7
+ Fuzzy SD	+6.1%	1.71×	2.59×	−6.6	+4.8%	1.67×	2.27×	−1.6	+6.4%	1.57×	2.38×	−13.5
+ Spec. Cascades	+2.9%	0.94×	**4.50**×	+0.1	+2.4%	0.99×	3.66×	+0.1	+4.9%	1.28×	3.09×	−3.1
+ Cactus	+7.0%	1.08×	4.05×	−0.1	+6.0%	1.05×	3.56×	+0.1	+7.0%	1.07×	3.43×	−0.6
+ RFlash	+11.0%	1.00×	4.46×	+0.9	+8.7%	1.02×	**3.73**×	0.0	+10.5%	1.01×	**3.71**×	+0.2
Gemma-4 12B DFlash	7.56	1×	7.53×	81.4	5.58	1×	5.79×	73.5	6.21	1×	6.40×	85.3
+ Fuzzy SD	+10.1%	1.60×	5.72×	−4.1	+17.7%	1.79×	4.57×	−6.3	+8.1%	1.66×	4.67×	−4.1
+ Spec. Cascades	+33.5%	2.16×	5.12×	−11.1	+67.5%	3.48×	3.42×	−17.6	+66.3%	4.23×	2.88×	−32.7
+ Cactus	+5.7%	1.15×	6.93×	+0.3	+8.7%	1.16×	5.73×	−1.4	+6.6%	1.10×	6.34×	−0.7
+ RFlash	+5.6%	0.93×	**8.28**×	+2.0	+5.1%	0.94×	**6.31**×	+0.9	+8.6%	1.02×	**6.78**×	0.0

Table 1: Comparison with relaxed speculative decoding baselines. *AL* is the acceptance length (tokens per target forward pass). *Len* is the mean output length relative to lossless decoding. *Spd.* is the request-level speedup over autoregressive decoding, counted in target forward passes. The best value among relaxation rules per benchmark is in **bold**, and values below lossless speculative decoding are shown in red. *Acc* is task accuracy. For each method, we select the hyperparameters with the highest worst-case speedup among those losing less than one point of accuracy on every benchmark, or overall if none does (Appendix A.2).

Benchmarks and metrics. We evaluate on three long-reasoning benchmarks: AIME25 (16 seeds), GPQA Diamond (Rein et al., 2023) (10 seeds), and LiveCodeBench (Jain et al., 2025) (5 seeds). Besides accuracy, acceptance length, and output length, we report the request-level speedup over autoregressive decoding: the number of target forward passes autoregressive decoding would need for the lossless output (one per token) divided by the number the rule uses, both summed over all requests of a benchmark. Lossless speculative decoding is reported on the same scale and serves as the reference; unlike acceptance length, this speedup accounts for changes in output length, so a rule that generates more tokens can fall below lossless speculative decoding despite accepting more per step. Appendix A.2 gives the full setup.

6.2 Comparison with baselines

As shown in Table 1, RFlash is the only rule that is faster than lossless speculative decoding on every model and benchmark, by 1.06–1.10×, and it is the fastest rule in five of the six settings. Thanks to the drift constraint, it also stays closest to lossless decoding: its output length is within 7% of lossless, and its accuracy never drops. The baselines, in contrast, lose their gain in acceptance to longer outputs. On Qwen3.8 27B, this leaves Cactus barely faster than lossless speculative decoding, while Speculative Cascades on LiveCodeBench and Fuzzy SD everywhere are even slower than lossless; on Gemma-4 12B, every baseline is slower than lossless speculative decoding on every benchmark, and Fuzzy SD and Speculative Cascades also lose accuracy substantially. On AIME25 with Gemma-4 12B, for instance, Cactus and RFlash raise the acceptance length by the same amount, yet Cactus ends 8% slower than lossless speculative decoding and RFlash 10% faster.

6.3 Frontier comparison

Figure 4 sweeps every rule over its operating points on Gemma-4 12B. For the baselines, a larger acceptance length comes with longer outputs and mostly lower accuracy, so pushing them further does not pay off: none of their operating points is faster than lossless speculative decoding on any benchmark, and they mostly get slower as the acceptance length grows. Every operating point of RFlash shown, in contrast, is faster than lossless speculative decoding, with output length and accuracy close to lossless. On Qwen3.8 27B (Figure A1 in Appendix A.3), the outputs of the baselines likewise grow with the acceptance length.

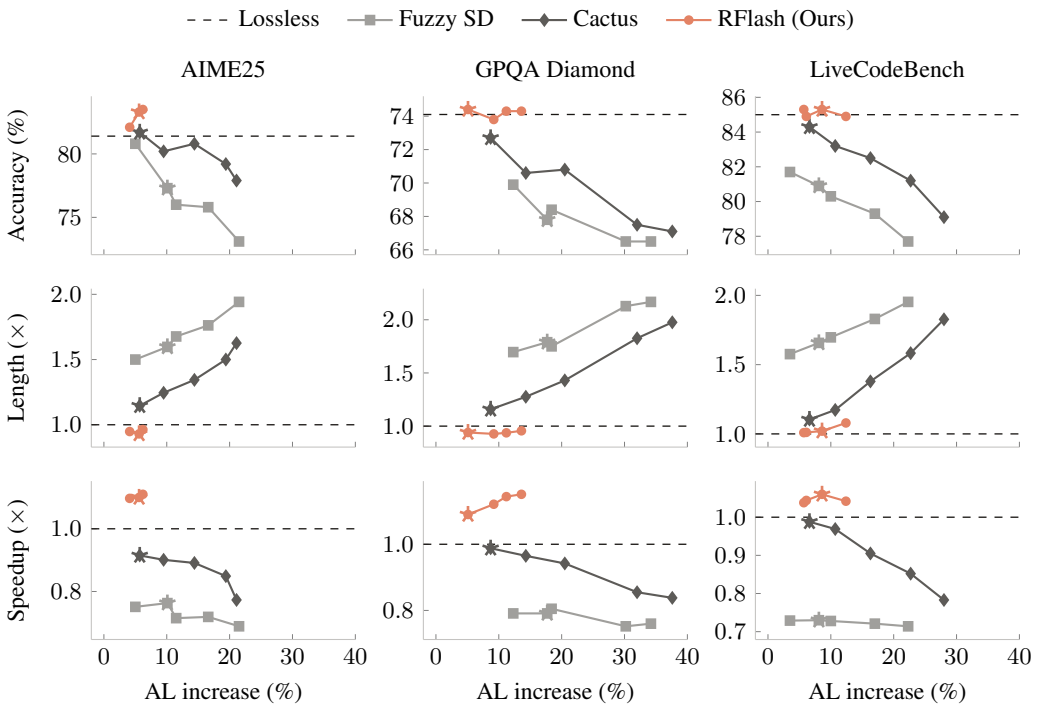


Figure 4: Accuracy, output length relative to lossless decoding, and request-level speedup relative to lossless speculative decoding (top to bottom) versus the increase in acceptance length on Gemma-4 12B with a DFlash drafter. Each baseline is swept over its operating points; the dashed line marks lossless decoding. Stars ($\star$) mark the operating point of each rule in Table 1.

Drafter	Lossless		+ RFlash			
	AL	Spd.	AL	Len	Spd.	Acc
MTP	2.94	2.93×	+10.7%	1.01×	3.21×	−0.7
DFlash	3.57	3.40×	+7.7%	1.04×	3.53×	−0.5
DSpark	4.27	4.06×	+11.1%	1.02×	4.46×	−0.3
DFlash 2	4.02	3.86×	+12.8%	1.05×	4.20×	+0.3

Table 2: RFlash on top of different drafters, on GPQA Diamond with Qwen3.5 4B.

6.4 Ablation Study

Different drafters. Table 2 applies RFlash on top of four drafters on Qwen3.5 4B, from the model's native MTP head to parallel drafters. RFlash is faster than lossless speculative decoding with every drafter, by 1.09–1.10× on MTP, DSpark and DFlash 2 and 1.04× on DFlash, with output length within 5% of lossless and accuracy within one point. Its gain is thus largely orthogonal to the drafter: moving from MTP to DSpark raises the lossless speedup from 2.93× to 4.06×, and RFlash adds a further 10% on top. At a similar relative gain, the absolute gain grows with the drafter: +0.28× on MTP, +0.34× on DFlash 2, and +0.40× on DSpark.

7 Conclusion

In this paper, we proposed RFlash, a training-free relaxed speculative decoding method that translates higher acceptance length into practical request-level speedup. Based on the insight that prior relaxation rules bound only the size of each per-token deviation, not its direction, and therefore drift systematically toward the draft, we designed a drift constraint that caps the drift of each verification block at a multiple of the divergence it spends. RFlash is the only relaxation rule we evaluate that is faster than lossless speculative decoding in every setting, with output length and accuracy at the lossless level, whereas prior rules largely lose their gain in acceptance to longer outputs as the drift accumulates over long reasoning chains.

AI USE STATEMENT

We used generative AI tools to formulate mathematical claims, to help prove them and write the proofs (Appendix A.1), and to implement the methods; we did not use them for any other task that requires disclosure. We also used them to find related work, draft and polish the text, and make figures. The authors checked every proof step by step, tested all AI-written code, verified every reported number against the raw outputs, added and checked all references themselves, and reviewed and revised all AI-assisted text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *International Conference on Learning Representations*, volume 2025, pp. 40894–40914, 2025. 1, 3

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024. 2

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. 2

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block Diffusion for Flash Speculative Decoding. *arXiv preprint arXiv:2602.06036*, 2026. 1, 2, 7

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026. 1, 2, 7

Jiale Fu, Yuchu Jiang, Junkai Chen, Jiaming Fan, Xin Geng, and Xu Yang. Fast large language model collaborative decoding via speculation. *arXiv preprint arXiv:2502.01662*, 2025. 3

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. Autojudge: Judge decoding without manual annotation. *Advances in Neural Information Processing Systems*, 38:94605–94642, 2026. 1, 3

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. 2, 7

Yongchang Hao and Lili Mou. Cactus: Accelerating auto-regressive decoding with constrained acceptance speculative sampling. *arXiv preprint arXiv:2604.04987*, 2026. 1, 2, 3, 7, 15

Maximilian Holsman, Yukun Huang, and Bhuwan Dhingra. Fuzzy speculative decoding for a tunable accuracy-runtime tradeoff. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26257–26273, 2025. 2, 3, 7, 15

Inco AI. Dflash 2: Keep drafting parallel, August 2026. URL https://inco.ai/blog/dflash2/. 1, 2, 7

Naman Jain, 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. 8

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192. 1, 2, 3, 12

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a. 2

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b. 2

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025. URL https://arxiv.org/abs/2503.01840. 1, 2

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025. 3

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 2

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 44949–44987, 2025. 3, 7, 15

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *Advances in Neural Information Processing Systems*, 38:12730–12749, 2026. 3

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5. 7

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8. 7

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. 2, 8

Ziyi Wang, Siva Rajesh Kasa, Santhosh Kumar Kasa, Jiaru Zou, Sumit Negi, Ruqi Zhang, Nan Jiang, Qifan Song, et al. Diversed: Relaxed speculative decoding via dynamic ensemble verification. *arXiv preprint arXiv:2604.07622*, 2026. 3

A.1 PROOFS

Throughout, the vocabulary is finite and the draft is smoothed as in the footnote of Section 4.1, so $\tilde{q}_t(y) \geq \epsilon\, p_t(y)$ and every $z_t(y)$ with $p_t(y) > 0$ is finite; a token with $p_t(y) = 0$ is never emitted by the target and, since $D_{\mathrm{KL}}(\hat{p}_t \,\|\, p_t) < \infty$ forces $\hat{p}_t \ll p_t$, never by a relaxation rule either. All expectations below therefore exist. We write $\tilde{q}$ for the smoothed draft where the distinction matters and q otherwise.

A.1.1 THE DRIFT STATISTIC

Proposition 1 (Drift). *Let $\mathcal{F}_t = \sigma(y_1, \ldots, y_t)$ and let S_T be as in Equation (4).*

(i) Under lossless decoding, $(S_t)_{t\geq 0}$ is an $(\mathcal{F}_t)$-martingale with $\mathbb{E}[S_T] = 0$ and $\mathrm{Var}[S_T] = \sum_{t\leq T} \mathbb{E}\big[\mathrm{Var}_{p_t}(z_t)\big]$.

(ii) Under a relaxation rule that emits $y_t \sim \hat{p}_t$, with $\hat{p}_t$ $\mathcal{F}_{t-1}$-measurable, $S_T = M_T + \sum_{t\leq T} \delta_t$ where $\delta_t = \mathbb{E}_{\hat{p}_t}[z_t] - \mathbb{E}_{p_t}[z_t]$ is $\mathcal{F}_{t-1}$-measurable and (M_t) is an $(\mathcal{F}_t)$-martingale with $M_0 = 0$. In particular $\mathbb{E}[S_T] = \mathbb{E}\big[\sum_{t\leq T} \delta_t\big]$, which is at least ℓT if $\delta_t \geq \ell$ for all t.

Proof. p_t, q_t and hence z_t are functions of $y_{<t}$, so they are $\mathcal{F}_{t-1}$-measurable, and $\mathbb{E}[z_t(y_t) \mid \mathcal{F}_{t-1}] = \mathbb{E}_{\pi_t}[z_t]$ for whichever distribution π_t the token is drawn from. (i) With $\pi_t = p_t$ the increment $S_t - S_{t-1} = z_t(y_t) - \mathbb{E}_{p_t}[z_t]$ has conditional mean zero, which is the martingale property; $\mathbb{E}[S_T] = 0$ follows by induction, and the variance formula because martingale increments are uncorrelated and the conditional variance of the increment is $\mathrm{Var}_{p_t}(z_t)$. (ii) With $\pi_t = \hat{p}_t$ the conditional mean of the increment is δ_t. Set $M_t = \sum_{s\leq t} \big[z_s(y_s) - \mathbb{E}_{\hat{p}_s}[z_s]\big]$; its increments have conditional mean zero, and $S_T - M_T = \sum_{t\leq T} \delta_t$ by definition. This is the Doob decomposition of S_T, which is unique. □

A.1.2 RELAXED VERIFICATION AS A MODIFIED TARGET

A verification rule at one position is a pair (β, ρ): the draft's proposal $x \sim q$ is accepted with probability $\beta(x) \in [0, 1]$, and otherwise a token is drawn from a distribution ρ (which may depend on x). Standard verification against a distribution π is the rule $\beta(x) = \min\{1, \pi(x)/q(x)\}$, $\rho = (\pi - q)^+ / \sum_y (\pi - q)^+$, and it emits a sample from π (Leviathan et al., 2023).

Lemma 2 (Modified target). *Let $m(y) = q(y)\beta(y) + \big(1 - \sum_x q(x)\beta(x)\big)\,\bar{\rho}(y)$, with $\bar{\rho}$ the residual averaged over the rejected proposal, be the distribution of the token a rule (β, ρ) emits. Standard verification against $\hat{p} = m$ emits the same distribution and accepts with probability $\sum_y \min(m, q) \geq \sum_y q\,\beta$.*

Proof. The first claim is the correctness of standard verification. For the second, $m(y) \geq q(y)\beta(y)$ pointwise and $\beta \leq 1$, so $\min(m(y), q(y)) \geq \min(q(y)\beta(y), q(y)) = q(y)\beta(y)$. □

So for any rule there is a modified target that produces the same output distribution with at least the same acceptance, which is why Section 3 describes a rule by $\hat{p}_j$ alone.

A.1.3 THE GEOMETRY OF A DEVIATION

Fix a position and write $\hat{p} = p(1 + u)$ with $\mathbb{E}_p[u] = 0$, $u \geq -1$.

Lemma 3 (Acceptance gain). *With $S = \{y : p(y) < q(y)\}$,*

$$a(\hat{p}) - a(p) \;=\; \sum_{y\in S} \min\big(p(y)u(y),\, q(y) - p(y)\big) \;-\; \sum_{y\notin S} \big(p(y) - q(y) - p(y)u(y)\big)^+ \;\leq\; \langle u, g\rangle_p, \tag{16}$$

with equality iff no token crosses the draft, i.e., *$\hat{p} \leq q$ on S and $\hat{p} \geq q$ off S. In particular $a(\hat{p}) > a(p)$ requires $u(y) > 0$ for some $y \in S$.*

Proof. On S, $\min(\hat{p},q)-\min(p,q)=\min(p+pu,q)-p=\min(pu,q-p)$; off S, $\min(p+pu,q)-q=-(p-q-pu)^+$. Summing gives the identity. The first sum is at most $\sum_S pu=\langle u,\mathbb{1}_S\rangle_p=\langle u,g\rangle_p$ (centering is free since $\mathbb{E}_p[u]=0$), with equality iff $pu\le q-p$ on S; the second is ≤ 0 with equality iff $pu\ge -(p-q)$ off S. If $u\le 0$ on S the first sum is ≤ 0 and the gain is ≤ 0. □

Lemma 4 (Divergence to second order). *$D_{\mathrm{KL}}(\hat{p}\,\|\,p)=\frac{1}{2}\|u\|_p^2+R$ with $|R|\le\frac{1}{6}\,\mathbb{E}_p\big[|u|^3/(1+\min(u,0))^2\big]$; in particular $|R|\le\frac{2}{3}\mathbb{E}_p|u|^3$ when $u\ge-\frac{1}{2}$.*

Proof. $D_{\mathrm{KL}}(\hat{p}\|p)=\mathbb{E}_p[(1+u)\log(1+u)]=\mathbb{E}_p[f(u)]+\mathbb{E}_p[u]$ with $f(u)=(1+u)\log(1+u)-u$, and $\mathbb{E}_p[u]=0$. Now $f(0)=f'(0)=0$, $f''(u)=1/(1+u)$ and $f'''(u)=-1/(1+u)^2$, so by Taylor's theorem $f(u)=u^2/2-u^3/(6(1+\xi)^2)$ for some ξ between 0 and u, and $|1+\xi|\ge 1+\min(u,0)$. □

Lemma 5 (The angle between g and z_c). *With the smoothed draft, $\{p<q\}=\{z>0\}$, so g is the centered indicator of $\{z>0\}$. Then $\langle g,z_c\rangle_p=\mathbb{E}_p[z^+]\,P_p(z\le 0)+\mathbb{E}_p[z^-]\,P_p(z>0)\ge 0$, with equality iff z does not change sign on the support of p; and $\cos\theta=1$ iff z takes a single value on $\{z>0\}$ and a single value on $\{z\le 0\}$.*

Proof. $\tilde{q}>p\iff(1-\epsilon)q+\epsilon p>p\iff q>p$ for $\epsilon<1$. Write $\pi=P_p(z>0)$ and $\mathbb{1}=\mathbb{1}[z>0]$. Then $\langle g,z_c\rangle_p=\mathrm{Cov}_p(\mathbb{1},z)=\mathbb{E}_p[z\mathbb{1}]-\pi\,\mathbb{E}_p[z]=\mathbb{E}_p[z^+]-\pi(\mathbb{E}_p[z^+]-\mathbb{E}_p[z^-])=(1-\pi)\mathbb{E}_p[z^+]+\pi\mathbb{E}_p[z^-]$, where $z^\pm\ge 0$ are the positive and negative parts. This vanishes iff $\pi(1-\pi)=0$ or both parts vanish on their sets, *i.e.*, iff z has one sign. Cauchy–Schwarz is an equality iff z_c is a multiple of g, *i.e.*, z is an affine function of $\mathbb{1}[z>0]$, *i.e.*, two-valued by sign. □

Lemma 6 (Budget-only and neutral deviations). *Among deviations with $\|u\|_p\le r$, the linearized gain $\langle u,g\rangle_p$ is maximized by $u=r\,g/\|g\|_p$, which has drift $\delta=\cos\theta\cdot r\|z_c\|_p$,* i.e.*, a fraction $\cos\theta$ of the Cauchy–Schwarz bound Equation (11). Among deviations with $\|u\|_p\le r$ and $\langle u,z_c\rangle_p=0$, the maximal linearized gain is $r\|g\|_p\sin\theta$; to match the unconstrained gain a neutral deviation needs length $r/\sin\theta$, hence, by Lemma 4, $1/\sin^2\theta$ times the divergence to second order.*

Proof. Both are Cauchy–Schwarz: in $L^2(p)$ for the first claim, and in the orthogonal complement of z_c for the second: writing $g=\|g\|_p\cos\theta\,\hat{z}_c+g_\perp$ with $\hat{z}_c=z_c/\|z_c\|_p$ and $\langle g_\perp,z_c\rangle_p=0$, a neutral u satisfies $\langle u,g\rangle_p=\langle u,g_\perp\rangle_p\le\|u\|_p\|g_\perp\|_p$, and $\|g_\perp\|_p^2=\|g\|_p^2-\|g\|_p^2\cos^2\theta$. □

A.1.4 THE CLOSED FORM

Fix a block and let $a_j=a_j(\hat{p}_j)$ be the acceptance probabilities of the rule, $w_j=\prod_{i<j}a_i$ its reach weights, and $R_j=1+a_{j+1}R_{j+1}$, $R_M=1$, its expected remaining runs. The objective of Equation (13) is the expected accepted length, $\sum_k w_k a_k=\sum_k\prod_{i\le k}a_i=\sum_k P(\text{at least } k \text{ accepted})$, and differentiating it with respect to a_j through every later reach weight gives

$$\frac{\partial}{\partial a_j}\sum_k w_k a_k \;=\; w_j\Big[1+a_{j+1}+a_{j+1}a_{j+2}+\cdots\Big] \;=\; w_j R_j. \tag{17}$$

Equation (13) is solved as a fixed point on the reach weights: within a pass, the weights that multiply the divergence and the constraint are held at their current values, while the objective's dependence on $\hat{p}_j$ through the later reach weights is kept, which is where R_j enters; the weights are then recomputed from the solution and the pass repeated. The terms this drops from the exact stationarity conditions (the dependence of w_k, $k>j$, on $\hat{p}_j$ in the divergence and constraint sums) are one order higher in the deviation than the terms kept, since $D_{\mathrm{KL}}(\hat{p}_k\|p_k)$ and $\langle u_k,z_k\rangle$ are themselves of second and first order in u_k. With the weights given, the Lagrangian separates over positions, and position j solves, after division by w_jR_j,

$$\max_{\hat{p}}\;\Phi_j(\hat{p}) \;=\; a(\hat{p})-\lambda_j\,D_{\mathrm{KL}}(\hat{p}\,\|\,p)-\mu_j\Big[\langle u,z\rangle_p-\alpha\,D_{\mathrm{KL}}(\hat{p}\,\|\,p)\Big],\qquad \lambda_j=\lambda/R_j,\;\;\mu_j=\mu/R_j, \tag{18}$$

over distributions $\hat{p}\ll p$, with $\mu\ge 0$ the multiplier of the block inequality constraint ($\mu=0$ when it is slack, by complementary slackness). We write $\Lambda_j=\lambda_j-\alpha\mu_j$ and drop the index j inside the proof.

Proposition 7 (Closed form). *If $\Lambda_j > 0$, Equation (18) has a unique maximizer, $\hat{p}_j = p_j r_j$ with r_j given by Equation (14), where $x_j > 0$ is any solution of $\sum_y p_j(y)\, r_j(y) = 1$; such an x_j exists, and the resulting $\hat{p}_j$ does not depend on which solution is taken.*

Proof. Write $\hat{p} = p\, r$ with $r \geq 0$ on the support of p. In terms of r, $a(\hat{p}) = \sum_y \min(pr, q)$ is concave and piecewise linear, $D_{\mathrm{KL}}(\hat{p}\|p) = \sum_y p\, r \log r$ is strictly convex, and $\langle u, z\rangle_p = \sum_y p\,(r-1)\, z$ is linear. Hence $\Phi = a - \Lambda\, D_{\mathrm{KL}} - \mu\, \langle u, z\rangle_p$ is strictly concave for $\Lambda > 0$, the feasible set $\{r \geq 0,\ \sum_y pr = 1\}$ is compact and convex, so the maximizer exists and is unique, and the KKT conditions are necessary and sufficient. Since $\partial(r \log r)/\partial r \to -\infty$ as $r \to 0$, the maximizer has $r(y) > 0$ wherever $p(y) > 0$, and only the equality constraint carries a multiplier, c. The stationarity condition in $r(y)$, divided by $p(y)$, reads

$$s(y) \;-\; \Lambda\,\big(\log r(y) + 1\big) \;-\; \mu\, z(y) \;-\; c \;=\; 0, \qquad s(y) \in \begin{cases} \{1\} & p(y)r(y) < q(y), \\ [0,1] & p(y)r(y) = q(y), \\ \{0\} & p(y)r(y) > q(y), \end{cases} \tag{19}$$

where $s(y)$ is a subgradient of $\min(pr, q)$ in pr. Solving for r, $\log r(y) = \log A(y) + \log x + s(y)/\Lambda$ with $A(y) = e^{-\mu z(y)/\Lambda}$ and $x = e^{-c/\Lambda - 1}$, *i.e.*, $r(y) = A(y)\, x\, e^{s(y)/\Lambda}$. The three cases of Equation (19) are mutually exclusive and exhaustive because $e^{1/\Lambda} > 1$: if $A(y)\, x\, e^{1/\Lambda} < q(y)/p(y)$ then $s = 1$ is consistent and $r = Axe^{1/\Lambda}$; if $A(y)\, x > q(y)/p(y)$ then $s = 0$ is consistent and $r = Ax$; otherwise $r = q/p$ with $s = \Lambda \log\big(q/(pAx)\big) \in [0, 1]$. In all three cases $r(y) = \min\{\max(q/p,\, Ax),\, Axe^{1/\Lambda}\}$, which is Equation (14).

It remains to choose x. Each $r(y; x)$ is continuous and nondecreasing in x, so $F(x) = \sum_y p(y)\, r(y; x)$ is continuous and nondecreasing, with $F(x) \to 0$ as $x \to 0$ and $F(x) \to \infty$ as $x \to \infty$; hence $F(x) = 1$ has a solution. F is constant on an interval only where every token sits at its kink, $\hat{p} = q$ on the support of p, so any two solutions give the same $\hat{p}$. Finally, the maximizer of Equation (18) satisfies Equation (19) for some c, hence is of this form with $F(x) = 1$, and by uniqueness it is the only such point. □

Two remarks. At $\mu = 0$, where $A \equiv 1$, $F(x) = 1$ forces $x \leq 1 \leq xe^{1/\Lambda}$ (if $x > 1$ every $r > 1$; if $xe^{1/\Lambda} < 1$ every $r < 1$), so a token is raised only if $q > p$, lowered only if $q < p$, and never past the draft: the budget-only rule of Section 5.2. And a token outside the draft's candidate set has $q(y) = 0$, hence $\tilde{q}(y)/p(y) = \epsilon$, $z(y) = \log \epsilon$ and kink 0, so every such token receives the same factor $r = e^{-\mu \log \epsilon/\Lambda}\, x$. Our implementation therefore solves Equation (14) on the candidates plus one bucket for the rest of the vocabulary.

A.1.5 The block multiplier

For a pass with the reach weights given, let $\hat{p}_j(\mu)$ be the solution of Equation (18) and $D(\mu) = \sum_j w_j \big[\langle u_j, z_j\rangle_{p_j} - \alpha\, D_{\mathrm{KL}}(\hat{p}_j(\mu)\|p_j)\big]$ the constraint of Equation (13).

Proposition 8 (Block multiplier). *D is continuous on $\{\mu \geq 0 : \Lambda_j > 0\ \forall j\}$. At $\mu = 0$ the solution is the budget-only rule and every position has $\langle u_j, z_j\rangle_{p_j} \geq 0$, with equality only if $\hat{p}_j(0) = p_j$. If $D(0) \leq 0$ the constraint is slack and $\mu^\star = 0$. Otherwise the constraint binds at the optimum: for $\alpha = 0$, $D(0) \geq 0$ always, $D(\mu) < 0$ for all large μ unless every z_j is constant on the support of p_j, and a root $\mu^\star \geq 0$ exists; for $\alpha > 0$ a root exists in $[0, \lambda/\alpha)$ whenever D changes sign there. In general $D(0) = \sum_j w_j \big[\langle u_j, z_j\rangle_{p_j} - \alpha\, D_{\mathrm{KL}}(\hat{p}_j(0)\|p_j)\big]$, and for $\alpha = 1$ this is $\sum_j w_j \big[D_{\mathrm{KL}}(p_j\|\tilde{q}_j) - D_{\mathrm{KL}}(\hat{p}_j(0)\|\tilde{q}_j)\big]$.*

Proof. Continuity: Φ_j is strictly concave and depends continuously on μ, so its maximizer over the compact feasible set is continuous in μ (Berge's theorem), and D is a continuous function of the maximizers. At $\mu = 0$ the rule is $r = \min\{\max(q/p, x), xe^{1/\lambda_j}\}$ with $x \leq 1 \leq xe^{1/\lambda_j}$, so $u = r - 1 \geq 0$ only on tokens with $q > p$, *i.e.*, $z > 0$ (Lemma 5), and $u < 0$ only where $z < 0$; thus $u\, z \geq 0$ pointwise and $\langle u_j, z_j\rangle_{p_j} \geq 0$, with equality iff $u_j = 0$. This gives $D(0) \geq 0$ for $\alpha = 0$, and for $\alpha = 1$ the stated form follows from the identity $\langle u, z\rangle_p - D_{\mathrm{KL}}(\hat{p}\|p) = \sum_y \hat{p} \log(\tilde{q}/p) - \sum_y p \log(\tilde{q}/p) - \sum_y \hat{p} \log(\hat{p}/p) = D_{\mathrm{KL}}(p\|\tilde{q}) - D_{\mathrm{KL}}(\hat{p}\|\tilde{q})$. For $\alpha = 0$ and

$\mu \to \infty$, $A_j(y) = e^{-\mu z_j(y)/\lambda_j}$ concentrates $\hat{p}_j$ on the tokens minimizing z_j (the lowered branch Ax of Equation (14) is bounded only from below, so nothing caps their mass), so $\langle u_j, z_j \rangle_{p_j} \to \min_y z_j(y) - \mathbb{E}_{p_j}[z_j] < 0$ unless z_j is constant; a reach-weighted sum of such terms is eventually negative. The intermediate value theorem gives the root. □

For $\alpha > 0$ the sign change at the upper end is not guaranteed in general; we solve for μ on $[0, 0.99\lambda/\alpha)$ and accept the residual of the constraint if no root is found there.

A.1.6 The block constraint and the drift

The constraint of Equation (13) is imposed per block with the rule's reach weights, so the statements of Section 5.1 hold at block resolution. Let block b start at step τ_b, let $\mathcal{G}_b = \mathcal{F}_{\tau_b}$, and let $S_{(b)} = S_{\tau_{b+1}} - S_{\tau_b}$ be the increment of the drift over the block.

Proposition 9 (Block supermartingale)**.** *If the block constraint holds with $w_j = \prod_{i<j} a_i(\hat{p}_i)$, then*

$$\mathbb{E}\big[S_{(b)} \,\big|\, \mathcal{G}_b\big] \;\le\; \alpha\, \mathbb{E}\Big[\sum_{t \text{ emitted in block } b} D_{\mathrm{KL}}(\hat{p}_t \,\|\, p_t) \,\Big|\, \mathcal{G}_b \Big], \tag{20}$$

with equality when the constraint binds. Hence $S_{\tau_b} - \alpha \sum_{t<\tau_b} D_{\mathrm{KL}}(\hat{p}_t \| p_t)$ is a $(\mathcal{G}_b)$-supermartingale, and a martingale when the constraint binds in every block; at $\alpha = 0$ the drift itself, sampled at block boundaries, cannot grow in expectation.

Proof. Condition on $\mathcal{G}_b$ and on the draft's M proposals for the block, which are drawn before verification. Then every p_j, q_j, $\hat{p}_j$ and z_j in the block is determined (they are functions of the drafted prefix), and the walk is a sequence of independent verifications: position j is reached iff positions $1, \dots, j-1$ were accepted, which has probability $w_j = \prod_{i<j} a_i(\hat{p}_i)$, and if it is reached the emitted prefix equals the drafted prefix, so p_j is the target's conditional distribution at that step and the emitted token is a sample from $\hat{p}_j$ (Lemma 2); after a rejection the block ends. Therefore $S_{(b)} = \sum_j \mathbb{1}[j \text{ reached}] \big(z_j(y_j) - \mathbb{E}_{p_j}[z_j]\big)$ and, taking the conditional expectation, $\mathbb{E}[S_{(b)} \mid \mathcal{G}_b, \text{proposals}] = \sum_j w_j\, \delta_j = \sum_j w_j \langle u_j, z_j \rangle_{p_j} \le \alpha \sum_j w_j\, D_{\mathrm{KL}}(\hat{p}_j \| p_j)$ by the constraint, with equality when it binds; the right-hand side is likewise $\mathbb{E}[\sum_{j \text{ reached}} D_{\mathrm{KL}}(\hat{p}_j \| p_j) \mid \mathcal{G}_b, \text{proposals}]$. Averaging over the proposals gives Equation (20), and the (super)martingale statements follow by summing over blocks. □

The reach weights in the solver are those of the converged fixed point, so Equation (20) holds up to the pass tolerance.

A.2 Evaluation Details

Models and drafters. Qwen3.8 27B (FP8) uses a DFlash 2 drafter with block size 8; Qwen3.5 4B (BF16) uses a DFlash 2 drafter and Gemma-4 12B (BF16) a DFlash drafter, both with block size 16. DFlash 2 proposes from an explicit per-position draft distribution q_j; DFlash proposes the greedy token, and for the rules that need a draft distribution we use the softmax of its draft logits.

Benchmarks and sampling. GPQA Diamond has 198 questions (10 seeds), AIME25 30 problems (16 seeds), and LiveCodeBench 1,055 problems (5 seeds). The generation budget is 131,072 tokens for GPQA Diamond and AIME25 and 196,608 tokens for LiveCodeBench. GPQA Diamond is graded by the selected option, AIME25 by exact integer match, and LiveCodeBench by its official test harness. Each model is sampled with the thinking-mode configuration of its model card: temperature 1.0 and top-p 0.95 for all models, top-k 20 for the Qwen models and 64 for Gemma, and a presence penalty of 1.5 for Qwen3.5 4B.

Baselines. Cactus (Hao & Mou, 2026) maximizes the acceptance of the drafted token under a per-token KL budget δ. FSD (Holsman et al., 2025) accepts a position whenever the Jensen–Shannon divergence between the target and draft distributions is below a threshold T. The token-level rule of Speculative Cascades (Narasimhan et al., 2025) verifies the draft against a mixture of the draft and target distributions controlled by a deferral parameter α. Each rule is re-implemented following its official code, and all rules share the same verification loop, differing only in the acceptance rule.

Operating points. Each baseline trades acceptance for fidelity through a single hyperparameter (δ, T, or α), and RFlash through two (λ and α). For every model and baseline, we place five operating points at $+5$, $+10$, $+15$, $+20$, and $+25\%$ acceptance length over lossless decoding, calibrated on a subset of the prompts (15 per task for Qwen3.8 27B and 40 per task for Gemma-4 12B) and confirmed to within ± 1 percentage point before the full runs. Speculative Cascades cannot reach the lower operating points on Gemma-4 12B: with a confident drafter, its token-level rule accepts whenever the drafted token is the target's most likely token, so the gain does not vanish as $\alpha \to 0$ ($+52\%$ at the smallest α); we report this minimum instead. RFlash is swept over a grid of (λ, α). Table 1 uses one operating point per model and rule for all benchmarks, chosen by the same criterion for every rule: among the operating points that lose less than one point of accuracy on every benchmark, the one with the highest worst-case speedup across benchmarks, or, if none qualifies, the one with the highest worst-case speedup. On Qwen3.8 27B this selects $\lambda = 1.5$, $\alpha = 0.5$ for RFlash and the $+5\%$ point for all three baselines; on Gemma-4 12B, it selects $\lambda = 0.7$, $\alpha = 0.25$ for RFlash, the $+5\%$ point for Cactus, the $+10\%$ point for FSD, and the minimum for Speculative Cascades.

Metrics. Every request is paired with the lossless run of the same problem and seed. The acceptance length gain is the ratio of the mean per-request acceptance length to that of lossless decoding, averaged over seeds; the output length is relative to lossless decoding. The request-level speedup of a rule is the number of target forward passes autoregressive decoding would need for the lossless output, one per generated token, divided by the number of target forward passes the rule uses, both summed over all requests of a benchmark (all problems and seeds, each run of a relaxation rule paired with the lossless run of the same seed). For lossless speculative decoding this is the token-weighted acceptance length; for a relaxation rule it is the lossless value times the ratio of lossless forward passes to those of the rule, so a rule that generates more tokens than lossless decoding can fall below it.

A.3 Additional Results

Frontier on Qwen3.8 27B. Figure A1 repeats the frontier comparison of Figure 4 on Qwen3.8 27B with a DFlash 2 drafter.

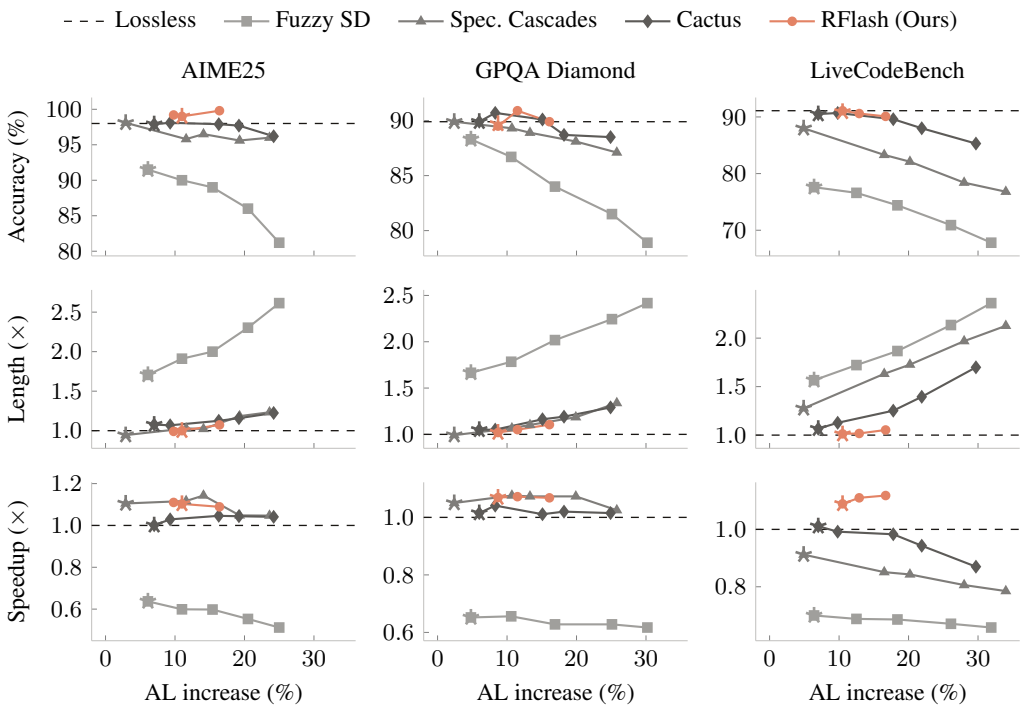


Figure A1: Accuracy, output length relative to lossless decoding, and request-level speedup relative to lossless speculative decoding (top to bottom) versus the increase in acceptance length on Qwen3.8 27B with a DFlash 2 drafter. Each baseline is swept over its operating points; the dashed line marks lossless decoding. Stars ($\star$) mark the operating point of each rule in Table 1.