CURVEK: THROUGHPUT-CURVE ESTIMATION FOR BATCHED SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference by drafting multiple tokens and verifying them simultaneously, but its efficiency critically depends on selecting an appropriate draft length k. This choice is particularly challenging in batched serving environments, where diverse requests share a common draft length but exhibit different acceptance patterns. Existing methods dynamically adjust k using arm-level rewards, draft-side statistics, or acceptance estimates. However, we theoretically show that these signals remain coarse: they do not directly represent how acceptance changes across draft positions in the current batch. This position-specific view defines a throughput curve over candidate draft lengths, whose maximum gives the draft length selected by our objective. Motivated by this insight, we formulate dynamic draft-length selection as estimating the batch throughput curve and choosing its peak. We propose CurveK, a lightweight and easy-to-implement method that estimates this curve from accepted-token depths *already returned by the verifier*. Importantly, CurveK requires neither additional training nor changes to the verification process. Experiments with 8B and 70B models in vLLM show throughput improvements over baselines across diverse workloads, including mixed-workload settings. Ablation studies validate the design of CurveK, and further analysis shows that its curve estimates consistently select high-throughput draft lengths.

1 INTRODUCTION

Speculative decoding accelerates LLM inference by using a lightweight *draft* model to generate k future tokens, and then using the stronger *target* model to verify these tokens *in parallel*, within a single forward pass (Leviathan et al., 2023; Chen et al., 2023; Xia et al., 2024; Li et al., 2025b). If the target accepts several draft tokens, the system emits them after one target pass instead of generating them one by one. The speedup therefore depends on choosing k: if the draft is too short, the target cannot accept enough tokens during verification because too few were proposed, leaving parallel computation underutilized; if it is too long, draft computation is wasted when tokens are rejected.

This tradeoff is more crucial in practical batch-serving scenarios (Wu et al., 2025; Liu et al., 2025b; Cheng et al., 2026; Liu et al., 2026), where requests from different workloads, such as mathematical reasoning, code generation, open-ended dialogue, and instruction following, are processed together. These requests can exhibit distinct token-acceptance patterns, making a single fixed draft length inefficient as batch composition changes (see Fig.1). Effective speculative decoding therefore requires dynamically adapting the draft length to the changing composition of each batch.

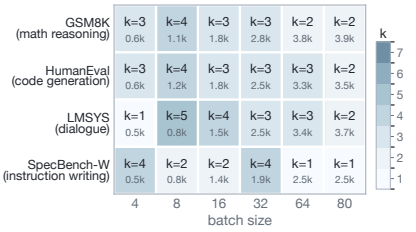

Figure 1: Best fixed draft length from measured end-to-end throughput evaluations across serving settings (**details in App.A.2**).

Existing adaptive methods select the draft length k based on arm-level rewards (Hou et al., 2025; Li et al., 2025a; Sridhar et al., 2025), draft-side statistics (Mamou et al., 2024; Agrawal et al., 2024; Zhang et al., 2025; Huang et al., 2026), or acceptance estimates (Liu et al., 2024b; 2025a; Huang et al., 2025). While these signals facilitate online adjustment, choosing the best shared draft length as batch composition changes remains challenging. This raises a central question:

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

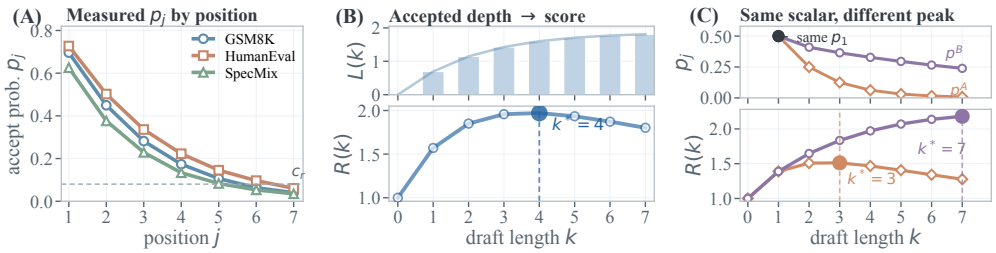

Figure 2: **Acceptance-probability sequences induce the throughput curve.** (A) Measured p_j, the probability that the first j draft tokens are all accepted, for three workloads. (B) Given a p_j sequence, the accepted depth is $L(k) = \sum_{j=1}^{k} p_j$, and the throughput is $R(k) = (1 + L(k))/(1 + kc_r)$, where c_r is the relative draft cost; the peak is k^*. (C) Example: two acceptance sequences share the same first-position acceptance p_1, but their later-position probabilities yield different throughput peaks.

What exactly should an online method estimate to choose the next draft length?

Our key insight is that the per-position acceptance probability for the current batch is the critical quantity to estimate. Let p_j denote the batch-average probability that a request accepts all of its first j draft tokens. The sequence $p_1, \ldots, p_k$ directly determines the expected number of accepted tokens at draft length k. Combining this accepted-token benefit with the draft cost yields a throughput curve whose peak corresponds to the optimal draft length (Fig.2). In contrast, we show that scalar acceptance summaries can **leave different per-position acceptance patterns indistinguishable**, even when those patterns **imply different optimal draft lengths**.

Motivated by this observation, we propose CurveK, a lightweight method that selects the next draft length by identifying the peak of an estimated throughput curve. It uses the accepted depths *already returned* by the verifier for each request to estimate the position-wise acceptance probabilities p_j, and hence the expected number of accepted tokens at each candidate draft-length. Combining this expected benefit with the corresponding draft cost then gives the throughput curve used for selection. Importantly, CurveK requires neither additional training nor changes to the verification procedure, keeping the method simple, efficient, and easy to integrate into existing serving systems.

We implement CurveK in vLLM (Kwon et al., 2023) and evaluate it against prior dynamic draft-length baselines. Our experiments span LLaMA and Qwen models (8B and 70B), greedy and stochastic decoding, multiple batch sizes, and mixed-workload settings. The results show throughput gains across model scales and hardware platforms, including when the workload mixture changes during serving. Ablation studies support both the acceptance estimator and the use of throughput scoring to select draft lengths. Further analyses show that verifier feedback helps predict acceptance at the next step and that the estimated curves guide selection toward high-throughput draft lengths.

2 Related Work

Online draft-length adaptation. Prior work adjusts the draft length using runtime feedback. BanditSpec (Hou et al., 2025) and Nightjar (Li et al., 2025a) treat candidate lengths as arms and update their rewards or losses after execution. TurboSpec (Liu et al., 2024b) and OTF (Liu et al., 2025a) instead estimate acceptance from recent verification outcomes and combine it with execution costs to choose a length. These approaches establish the value of adapting draft length online. Our work focuses on how acceptance varies across draft positions within a heterogeneous batch, and how to use this information to estimate its throughput curve.

Acceptance prediction and throughput-based selection. Another line of work uses predictive signals to guide speculation. DISCO (Mamou et al., 2024) learns when to stop drafting from draft-model statistics, while AdaEDL (Agrawal et al., 2024) uses an entropy-based acceptance bound. SVIP (Zhang et al., 2025) uses draft-side confidence to allocate verification effort; SpecDec++ (Huang et al., 2025) and Speculative Verification (Kim et al., 2025) use an acceptance-prediction head and a companion model, respectively. Closely related, AdaSpec (Huang et al., 2026) estimates acceptance from draft confidences and updates its length decision during drafting using a throughput

objective. DSpark (Cheng et al., 2026) combines a trained confidence head with hardware throughput profiles to choose per-request verification lengths. CurveK estimates the throughput curve from accepted depths returned by earlier verification steps to select the next shared draft length, without training an acceptance predictor. Our draft-confidence ablation compares confidence-based and accepted-depth estimation under the same cost model and selection rule (App.A.5).

Complementary optimizations. Other methods improve speculative decoding by changing how drafts are generated or executed. DFlash (Chen et al., 2026) generates draft blocks in parallel with a lightweight diffusion model. HeteroSpec (Liu et al., 2025b) assigns request-specific draft lengths, TETRIS (Wu et al., 2025) selects draft tokens for verification, and AdaServe (Li et al., 2026) allocates speculative work under latency constraints. DSDE (Yang et al., 2025b) mitigates per-sequence stragglers, PEARL (Liu et al., 2025c) overlaps drafting and verification, and Kangaroo (Liu et al., 2024a) uses early exits for self-speculation. These methods address allocation and execution choices beyond the shared batch-length decision studied here. Theoretical work also analyzes the role of draft–target distributional alignment (Yin et al., 2024). Further discussion appears in App.C.

3 Preliminaries

Speculative decoding uses a draft model to propose k future tokens and a target model to verify these tokens simultaneously within one pass (Leviathan et al., 2023; Chen et al., 2023). Verification proceeds from left to right, accepting the longest draft prefix up to the first rejection. Therefore, the throughput improvement critically depends on two key factors: how many draft tokens are successfully verified and the computational overhead introduced by increasing the draft length k.

To compare candidate draft lengths, we use a normalized cost model with unit verification cost and draft cost kc_r (Leviathan et al., 2023; Chen et al., 2023; Liu et al., 2024b). Let $c_r = c_{\text{draft}}/c_{\text{verify}}$ denote the relative draft cost per token, measured in units of one target-model verification pass. We measure c_r once using forward-time profiling for the given models, hardware, data type, and serving setup, as described in App.G. This model gives a lightweight throughput estimate for online length selection; end-to-end throughput is measured separately in our serving experiments.

To further clarify this relationship, we first consider throughput from the perspective of a single request. Let $Z \in \{0, 1, 2, \ldots\}$ denote the number of consecutive tokens that would be accepted if the draft continued without a length limit. When a finite draft length k is chosen, the number of accepted tokens is $\min(Z, k)$. The verification pass also determines the next token after the accepted prefix, so each speculative step produces, on average, $1 + \mathbb{E}[\min(Z, k)]$ output tokens.

Thus, the normalized throughput for a single request distribution is

$$R_{\text{single}}(k) = \frac{\text{expected tokens per step}}{\text{normalized cost per step}} = \frac{1 + \mathbb{E}[\min(Z, k)]}{1 + c_r k}, \qquad k \in \{0, 1, \ldots, K\},$$

where K is the maximum draft length and $k = 0$ gives standard decoding, so $R_{\text{single}}(0) = 1$.

These definitions capture throughput at the single-request level. In practice, however, serving systems process batches of requests with different acceptance behaviors. This naturally leads to the question: how does batch composition affect the optimal draft length? We explore this next.

4 Acceptance-Probability Sequence and the Throughput Curve

To answer this question, consider B active requests in the current batch, where the method chooses a shared draft length k. The cost side follows the same normalization as in Sec.3: one target-model verification pass plus k draft-token costs gives a normalized cost of $1 + kc_r$. What changes with batch composition is the expected number of accepted draft tokens. Let Z_i denote the accepted-depth random variable for request i. We define the batch-average accepted depth as

$$L(k) = \frac{1}{B} \sum_{i=1}^{B} \mathbb{E}[\min(Z_i, k)] = \sum_{j=1}^{k} p_j,$$

where $p_j := \frac{1}{B} \sum_{i=1}^{B} \mathbb{P}(Z_i \geq j)$ is the batch-average probability of accepting all of the first j draft tokens. The last equality follows from the tail-sum formula for nonnegative integer-valued random

variables (Breiman, 1992). Thus, the sequence $p_1, \ldots, p_K$ determines the expected number of accepted draft tokens per request at each candidate length. Combining this expected count with the normalized cost gives the *batch throughput curve*:

DEFINITION 1 ***(Batch throughput curve)*** *Given a batch acceptance-probability sequence $(p_1, \ldots, p_K)$ and relative draft cost c_r, the batch throughput curve plots throughput against k:*

$$R(k) := \frac{1 + L(k)}{1 + kc_r}.$$

Here, $L(k)$ is the cumulative sum of the acceptance-probability sequence, with $L(0) = 0$, while $1 + kc_r$ is the normalized cost. Thus $R(0) = 1$, corresponding to standard decoding. The curve is defined over $k \in \{0, 1, \ldots, K\}$. Under this normalized throughput objective, the batch's best shared draft length is the peak of this curve.

With this definition, selecting the next batch draft length becomes a finite search over the throughput values. The next proposition records a useful structural property of this target: for any valid acceptance-probability sequence, the maximizers form one contiguous peak region.

PROPOSITION 1 ***(Unimodality of $R(k)$)*** *Under this cost model, $R(k)$ increases to a single peak region and then decreases. The rightmost maximizer is*

$$k^* = \max\big(\{0\} \cup \{k \in [K] : p_k \geq c_r R(k-1)\}\big).$$

Equivalently, the adjacent change is $R(k) - R(k-1) = \big(p_k - c_r R(k-1)\big)/(1 + kc_r)$, so the curve rises exactly while the marginal acceptance probability p_k exceeds the cost threshold $c_r R(k-1)$. The proof is in App.B.1.

Fig.2 also illustrates this result. Panel (A) shows measured acceptance probabilities p_j across draft positions; Panel (B) accumulates these probabilities into $L(k)$, applies cost normalization to obtain the throughput curve $R(k)$, and identifies the peak k^*. Consistent with this analysis, our experiments show nearly unimodal throughput curves, **with gaps of up to 38.8% between the best and worst fixed draft lengths** (see App.A.2 and Fig.4), highlighting the importance of choosing a proper k.

Importantly, the optimal draft length ***depends on how acceptance varies across draft positions***. A single acceptance statistic can *hide these differences and lead to a suboptimal choice*. Fig.2(C) illustrates this limitation: two acceptance sequences with identical first-position acceptance p_1 lead to different throughput peaks. The next proposition further formalizes this ambiguity.

PROPOSITION 2 ***(Limits of scalar acceptance summaries)*** *In general, neither the first-position acceptance probability p_1 nor the mean accepted depth $L(K)$ alone uniquely determines the throughput-maximizing draft length. Consequently, a controller relying only on either scalar summary cannot always select the optimal draft length.*

The proof is in App.B.2. To choose the proper draft length, we therefore turn to the throughput curve, which can be constructed from the position-wise acceptance probabilities and the cost model. If this curve were known, the optimal choice would simply be $\arg\max_k R(k)$. The remaining challenge is how to estimate these probabilities $(p_1, \ldots, p_K)$ online and use them to construct the curve.

5 CURVEK: THE PROPOSED METHOD

At first glance, the throughput curve appears unavailable online: before choosing k, we do not know the current batch acceptance-probability sequence, and thus cannot directly compute $L(k)$ or $R(k)$. Fortunately, each verification step provides a partial but informative view of the same acceptance behavior: after verifying the chosen draft length, the system observes how many tokens were consecutively accepted, or whether the entire proposed prefix was accepted.

Motivated by this, we introduce CurveK, which uses verifier feedback to maintain an accepted-depth estimate for each request and estimate the position-wise acceptance probabilities p_j. Summing these probabilities gives an estimate of $L(k)$ (Sec.5.1). CurveK combines this estimate with the cost model to construct the throughput curve and selects the draft length at its peak (Sec.5.2). The details of the implementation and the pseudocode appear in Sec.5.3 and App.E. Finaly, we analyze the estimation error in Sec.5.4 and evaluate throughput, design choices, and draft-length decisions in Sec.6.

5.1 Estimating the Throughput Curve from Accepted Depths

We first describe how CurveK updates its depth estimates from verifier feedback. At each update-and-decision iteration t, CurveK incorporates feedback from the preceding verification step before selecting the next draft length. For each request i with feedback, let $Z_{t,i}$ denote how many consecutive draft tokens it would have accepted had drafting continued. If the preceding step used draft length $k_{t-1} > 0$, the observed accepted depth is $Y_{t,i} = \min(k_{t-1}, Z_{t,i})$.

When all proposed tokens are accepted and $0 < k_{t-1} < K$, we set $\widetilde{Y}_{t,i} = k_{t-1} + 1$ to account for the possibility that acceptance may continue beyond the proposed prefix; otherwise, we use $\widetilde{Y}_{t,i} = Y_{t,i}$. Since accepted depths fluctuate between steps, CurveK smooths the updates for each request using an exponentially weighted moving average (EWMA):

$$\widehat{Z}_{t,i} = \mu_i^{(t)} = (1-\beta)\mu_i^{(t-1)} + \beta\widetilde{Y}_{t,i}.$$

For each new request, we initialize μ_i to the default draft length $k_{\text{default}} = 7$. We then use $\beta \in (0, 1]$ to control how strongly each update changes the estimate: larger values track changes faster, while smaller values provide stronger smoothing. We evaluate the effect of β in Tab.4 and App.A.7.

To estimate position-wise acceptance probabilities p_j, CurveK uses $\min(k, \widehat{Z}_{t,i})$ as the estimated number of tokens accepted by request i at draft length k. The increase from length $j-1$ to j then estimates the probability that the request accepts all of its first j draft tokens. Averaging these estimates across the B active requests gives

$$\widehat{p}_{t,j} = \frac{1}{B}\sum_{i=1}^{B}\left[\min(j, \widehat{Z}_{t,i}) - \min(j-1, \widehat{Z}_{t,i})\right], \qquad j = 1, \ldots, K.$$

Summing over positions gives the estimated accepted-token count per request at length k:

$$\widehat{L}_t(k) = \sum_{j=1}^{k}\widehat{p}_{t,j} = \frac{1}{B}\sum_{i=1}^{B}\min(k, \widehat{Z}_{t,i}), \qquad k = 0, 1, \ldots, K.$$

The sum telescopes, so CurveK computes $\widehat{L}_t(k)$ directly from the smoothed depths.

5.2 Selecting the Next Draft Length

With $\widehat{L}_t(k)$ in hand, CurveK combines it with the normalized cost model to estimate the throughput curve and selects the draft length at its peak:

$$\widehat{R}_t(k) := \frac{1 + \sum_{j=1}^{k}\widehat{p}_{t,j}}{1 + kc_r} = \frac{1 + \widehat{L}_t(k)}{1 + kc_r}, \quad \widehat{k}_t = \underset{k\in\{0,1,\ldots,K\}}{\arg\max}\ \widehat{R}_t(k).$$

The relative cost c_r is profiled once per serving configuration and held fixed during inference (App.G). The next step uses this length; $k = 0$ denotes standard decoding.

5.3 Implementation Details and Efficiency

In practice, we implement curve estimation and length selection using simple arithmetic on the stored depth estimates, with no additional model calls. Evaluating all candidate lengths takes $O(BK)$ operations, with $O(B)$ storage for the request states and $O(K)$ additional storage for the throughput estimates. The measured decision overhead is small relative to target verification (App.A.12). When $k = 0$ is selected repeatedly, verifier feedback is temporarily unavailable, so we periodically run a short draft to obtain fresh acceptance observations (App.E).

5.4 Estimator Error Decomposition

Next, we analyze how throughput-curve estimation errors affect the throughput of the selected draft length under the normalized cost model. The following bound relates the throughput lost through length selection to three sources of curve-estimation error.

PROPOSITION 3 ***(Estimator error decomposition)*** *For a decision at step t, let $R_t(k)$ be the current batch throughput curve, with $k_t^* = \arg\max_k R_t(k)$ and $\widehat{k}_t = \arg\max_k \widehat{R}_t(k)$. Under the reference concentration condition in App.D.1, with probability at least $1-\delta$,*

$$R_t(k_t^*) - R_t(\widehat{k}_t) \le 2\Big(\underbrace{\varepsilon_t(\delta)}_{\text{finite-sample error}} + \underbrace{J_t}_{\text{Jensen gap}} + \underbrace{\Omega_t}_{\text{online mismatch}} \Big).$$

We interpret each component below; formal definitions are in App.D.1.

Finite-sample error (ε_t). This term captures fluctuations in the estimated curve under the reference concentration condition. The sub-batch study in App.I.4 examines sensitivity to the number of request states: estimates approach the full-batch estimate as more requests are included.

Jensen gap (J_t). This term captures the difference between truncating a request's mean accepted depth and averaging its truncated accepted depths. App.D.1 gives its exact form and a bound that depends on depth variability and the candidate length k.

Online mismatch (Ω_t). This term captures the remaining mismatch between the maintained depth states and the current acceptance distributions, including the effects of smoothing, feedback limited to the chosen draft length, and changing requests. The β sensitivity study in Tab.4 and App.A.7 shows that CurveK maintains stable throughput around the default update rate.

This analysis highlights the decision stability of CurveK. Errors in the estimated curve need not lead to a large throughput loss. When the best length has a clear throughput advantage, estimation errors that preserve this advantage leave the selected length unchanged. When several lengths offer similar throughput, selecting a different length among them incurs little loss. Sec.6 evaluates the resulting length choices, while App.I.4 examines how the curve estimates vary with the number of requests.

6 EXPERIMENTS

In the following, we test CurveK's throughput gains and adaptation to workload changes, and examine how its design affects draft-length selection. Further setup details and results appear in App.A.

We implement CurveK in vLLM (Kwon et al., 2023) and evaluate batched speculative decoding under a common serving path for all methods. This follows recent evidence that speculative-decoding conclusions depend strongly on serving implementation and measurement conditions (Liu et al., 2026). Throughput is measured in generated tokens per second. Unless otherwise stated, 8B evaluations use A100 GPUs, $K = 7$, and $c_r = 0.08$. We evaluate greedy decoding ($T = 0$) and stochastic sampling ($T = 1.0$) across batch sizes B (active requests per batch) on LLaMA-3.1-8B (Grattafiori et al., 2024) and Qwen3-8B (Yang et al., 2025a). App.A.1 details the common serving protocol, model–drafter pairs, workload sizes, and timing. We additionally evaluate LLaMA-3.3-70B with its EAGLE-3 drafter on A100 and H20 using tensor parallelism (TP=4). Each method is tuned independently per platform, with a single selected configuration held fixed across all workloads.

Our primary throughput evaluation uses GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), SpecBench-Mix (a mixed serving workload built from Spec-Bench prompts (Xia et al., 2024); see App.A.1), and AlpacaEval (Dubois et al., 2023). App.A.8 reports results on LMSYS-Chat-1M dialogue (Zheng et al., 2023). We also evaluate a mixed-workload serving trace composed of GSM8K, HumanEval, LMSYS-Chat-1M, CNN/DailyMail (Nallapati et al., 2016), retrieval QA from Natural Questions/DPR (Kwiatkowski et al., 2019; Karpukhin et al., 2020), and instruction-following prompts. Requests from these sources enter the same batched serving stream over time, testing whether a draft-length method can adapt as batch composition changes.

Baselines include vanilla decoding, fixed-k speculative decoding, and four adaptive draft-length controllers: BanditSpec (UCBSpec) (Hou et al., 2025), TurboSpec (Liu et al., 2024b), OTF (Liu et al., 2025a), and Nightjar (Li et al., 2025a). All controllers are integrated into the same vLLM V1 serving pipeline. For each experimental setting, the target–drafter pair, precision, request set, batching configuration, and throughput measurement procedure are held fixed. This setup isolates the effect of draft-length control within the shared runtime. We also tuned the baseline hyperparameters to ensure a fair comparison. App.H describes the baseline decision rules and evaluation settings.

Table 1: **Repeated-run throughput** (mean±std, tok/s). Comparison with Vanilla, Standard SD, and all four dynamic baselines. Top: 8B targets on A100. Bottom: LLaMA-3.3-70B with EAGLE-3 on A100 and H20. Bold marks the highest mean. Mean±std over three seeds is reported.

Model	Workload	Vanilla	Standard SD	OTF	TurboSpec	BanditSpec	Nightjar	CurveK
LLaMA 3.1-8B	GSM8K	1959.8$_{\pm 36.8}$	2471.9$_{\pm 44.6}$	2562.3$_{\pm 47.4}$	2573.1$_{\pm 13.0}$	2431.2$_{\pm 51.4}$	1970.9$_{\pm 34.9}$	**2821.5**$_{\pm 122.6}$
	HumanEval	1688.5$_{\pm 126.9}$	2416.2$_{\pm 70.5}$	2425.6$_{\pm 90.4}$	2385.5$_{\pm 55.7}$	2251.4$_{\pm 200.5}$	1814.7$_{\pm 60.7}$	**2656.0**$_{\pm 135.5}$
	SpecMix	1732.3$_{\pm 31.4}$	2060.1$_{\pm 9.7}$	2194.1$_{\pm 13.9}$	2124.5$_{\pm 54.0}$	2066.5$_{\pm 29.7}$	1989.4$_{\pm 60.2}$	**2275.1**$_{\pm 11.8}$
Qwen3-8B	GSM8K	2048.2$_{\pm 34.5}$	2061.0$_{\pm 45.1}$	2745.2$_{\pm 18.1}$	2743.7$_{\pm 9.2}$	2882.9$_{\pm 18.8}$	3071.0$_{\pm 49.7}$	**3107.5**$_{\pm 17.1}$
	HumanEval	1892.1$_{\pm 39.3}$	1752.4$_{\pm 48.3}$	2493.6$_{\pm 35.2}$	2416.4$_{\pm 42.8}$	2475.7$_{\pm 42.0}$	2513.6$_{\pm 52.2}$	**2672.0**$_{\pm 12.5}$
	SpecMix	1911.3$_{\pm 34.3}$	1723.9$_{\pm 38.9}$	2429.1$_{\pm 65.9}$	2424.1$_{\pm 5.0}$	2501.1$_{\pm 20.2}$	2418.4$_{\pm 133.7}$	**2600.8**$_{\pm 8.7}$

GPU	Workload	Vanilla	Standard SD	OTF	TurboSpec	BanditSpec	Nightjar	CurveK
A100	GSM8K	817.1$_{\pm 21.0}$	1204.1$_{\pm 15.5}$	1222.6$_{\pm 5.0}$	1223.2$_{\pm 8.4}$	1262.3$_{\pm 10.0}$	1229.8$_{\pm 112.4}$	**1377.2**$_{\pm 7.2}$
	HumanEval	712.2$_{\pm 43.8}$	1199.5$_{\pm 17.2}$	1206.9$_{\pm 30.6}$	1205.2$_{\pm 34.8}$	1262.7$_{\pm 14.6}$	1071.8$_{\pm 69.3}$	**1315.6**$_{\pm 31.5}$
	SpecMix	732.2$_{\pm 14.2}$	958.4$_{\pm 2.0}$	1006.2$_{\pm 8.4}$	980.2$_{\pm 10.9}$	1026.9$_{\pm 26.1}$	1002.0$_{\pm 56.7}$	**1088.7**$_{\pm 10.4}$
H20	GSM8K	957.5$_{\pm 4.6}$	1076.0$_{\pm 3.1}$	1372.4$_{\pm 18.5}$	1128.7$_{\pm 26.7}$	1319.2$_{\pm 20.3}$	1292.4$_{\pm 7.4}$	**1526.5**$_{\pm 18.3}$
	HumanEval	897.6$_{\pm 45.5}$	1124.5$_{\pm 16.4}$	1379.3$_{\pm 35.3}$	1275.4$_{\pm 38.4}$	1356.6$_{\pm 52.9}$	1169.1$_{\pm 45.3}$	**1472.5**$_{\pm 65.9}$
	SpecMix	912.6$_{\pm 16.8}$	868.4$_{\pm 3.4}$	1117.9$_{\pm 15.4}$	1087.2$_{\pm 51.2}$	1111.4$_{\pm 48.6}$	1068.2$_{\pm 9.3}$	**1180.8**$_{\pm 64.0}$

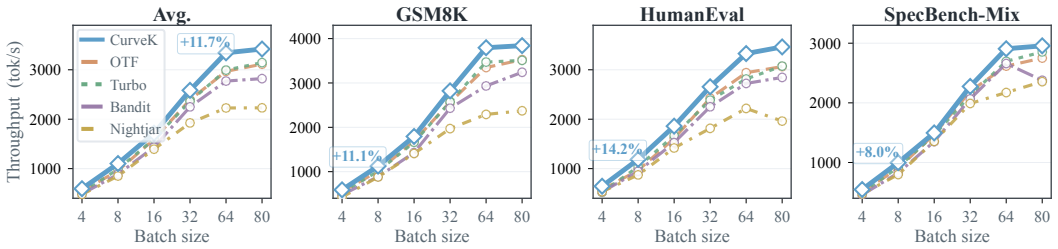

Figure 3: **Dynamic-k scaling under greedy decoding.** Each panel shows throughput as batch size varies on LLaMA-3.1-8B; Avg. averages the three workloads. CurveK uses the same verifier feedback as the baselines and keeps request-level accepted depths to estimate the throughput curve.

6.1 Cross-Workload Throughput Comparison

Dynamic-k comparison. Tab.1 reports repeated-run comparisons for 8B and 70B targets. For the 70B setting, results are obtained on both A100 and H20. CurveK achieves the highest mean throughput in every reported 8B comparison. At the 70B scale, it leads on the three workloads reported in Tab.1 on both A100 and H20. Fig.3 compares throughput across batch sizes on LLaMA-3.1-8B, where our method remains competitive across batch sizes. We also evaluate whether CurveK remains effective with a larger candidate range. For maximum draft lengths $K = 8, \ldots, 12$, it achieves the highest mean throughput among the dynamic methods compared (see App.A.13).

Drafter generalization. We further evaluate a Medusa drafter, where CurveK also achieves higher average throughput than the dynamic baselines. Additional results are provided in App.A.10.

6.2 Mixed-Workload Serving Trace

Single-benchmark evaluations keep the request family fixed. In deployed batched serving, however, math, code, dialogue, summarization, and RAG-style requests can arrive in the same stream. We therefore construct a mixed-workload serving trace, organized into consecutive workload phases: math-heavy, code-heavy, dialogue-heavy, long-context-heavy, balanced, math-code mixed, dialogue-long mixed, and a repeated balanced phase. Over the full trace, the request families are math (20.9%), code (20.9%), dialogue (19.5%), summarization (14.6%), RAG (12.2%), and instruction following (11.9%). Tab.2 reports the results. CurveK achieves higher mean throughput than the dynamic baselines at $B = 32, 64, 128$, whereas TurboSpec has a slightly higher mean at $B = 16$.

To examine how CurveK adapts to changing workload composition, we inspect its selected draft lengths across the trace phases (Fig.7 and App.A.9). It selects longer drafts during math- and code-

Table 2: **Mixed-workload serving trace.** Throughput in tok/s. Bold marks the highest mean. Standard SD follows the official EAGLE-3 evaluation setting.

Batch size (B)	Vanilla	Standard SD	OTF	TurboSpec	BanditSpec	Nightjar	CurveK
16	1100.57$_{\pm 40.23}$	1452.98$_{\pm 31.52}$	1481.36$_{\pm 25.21}$	**1508.05**$_{\pm 20.21}$	1323.87$_{\pm 62.13}$	1337.46$_{\pm 11.64}$	1501.66$_{\pm 6.76}$
32	1820.57$_{\pm 26.12}$	2098.34$_{\pm 52.16}$	2231.58$_{\pm 35.94}$	2185.65$_{\pm 17.23}$	1994.00$_{\pm 31.42}$	2170.43$_{\pm 62.13}$	**2336.40**$_{\pm 16.25}$
64	2569.37$_{\pm 19.52}$	2713.00$_{\pm 17.62}$	2876.12$_{\pm 36.28}$	2790.58$_{\pm 26.18}$	2709.13$_{\pm 19.63}$	2006.01$_{\pm 14.67}$	**3027.86**$_{\pm 35.01}$
128	3246.23$_{\pm 25.61}$	2911.19$_{\pm 11.42}$	3156.26$_{\pm 19.62}$	3202.44$_{\pm 20.44}$	3025.38$_{\pm 27.71}$	3151.06$_{\pm 26.71}$	**3360.04**$_{\pm 12.19}$

heavy phases and shorter drafts during dialogue- and long-context-heavy phases, mirroring the differences in accepted depth across these phases.

6.3 Mechanism and Estimator Evidence

Table 3: **Length-selection and estimator ablations.** (a) Length selection and acceptance estimation; L-/Q- denote LLaMA-3.1-8B/Qwen3-8B. (b) Estimator form ablation at $B = 32$; values are mean$\pm$std over three seeds. The CurveK column is shared by both groups.

Setting	(a) Length selection and acceptance estimation			(b) Estimator form		CurveK
	DirectDepth	Confidence	MeanDepth-K	History seq.	w/o continuation	
L-GSM8K	2627.2$_{\pm 20.8}$	2445.7$_{\pm 123.8}$	2674.2$_{\pm 53.7}$	2478.1$_{\pm 7.0}$	2573.6$_{\pm 83.1}$	**2821.5**$_{\pm 122.6}$
L-HumanEval	2416.6$_{\pm 76.9}$	2118.1$_{\pm 63.2}$	2483.8$_{\pm 68.0}$	2378.0$_{\pm 62.7}$	2403.9$_{\pm 170.2}$	**2656.0**$_{\pm 135.5}$
L-SpecMix	2109.3$_{\pm 29.6}$	1964.0$_{\pm 99.5}$	2139.1$_{\pm 31.3}$	2075.0$_{\pm 7.3}$	2080.2$_{\pm 73.5}$	**2275.1**$_{\pm 11.8}$
Q-GSM8K	2836.6$_{\pm 7.2}$	2735.5$_{\pm 97.9}$	2820.3$_{\pm 9.4}$	2679.8$_{\pm 10.2}$	2839.2$_{\pm 96.8}$	**3107.5**$_{\pm 17.1}$
Q-HumanEval	2538.0$_{\pm 10.1}$	2345.3$_{\pm 54.2}$	2521.1$_{\pm 33.6}$	2380.1$_{\pm 7.6}$	2427.3$_{\pm 19.7}$	**2672.0**$_{\pm 12.5}$
Q-SpecMix	2430.4$_{\pm 7.1}$	2347.3$_{\pm 11.4}$	2456.4$_{\pm 23.5}$	2341.0$_{\pm 3.5}$	2379.9$_{\pm 12.9}$	**2600.8**$_{\pm 8.7}$

Throughput-based selection. We first test whether estimating the throughput curve improves upon selecting directly from the average accepted depth. DirectDepth shares CurveK's depth updates but chooses a length based on their mean, whereas CurveK instead estimates the throughput curve and selects its peak. CurveK achieves higher mean throughput across all six settings in Tab.3(a) and in both additional comparisons in App.A.4, supporting our claim about curve-based length selection.

Acceptance estimation. Tab.3(a) also examines how the information used to construct the throughput curve affects performance. The confidence variant predicts how many draft tokens will be accepted by using draft-side confidence. MeanDepth-K instead uses verifier feedback but summarizes the batch with a single mean accepted depth. Unlike them, CurveK uses each requests accepted depth to estimate the accepted-token count at each candidate length before forming the batch throughput curve. Experiments show that CurveK achieves higher mean throughput than both variants in all six settings, supporting this way of constructing the curve for draft-length selection.

Estimator design. We next examine how CurveK updates the accepted depths used to construct the throughput curve. In Tab.3(b), keeping a history-based acceptance sequence instead of a single smoothed depth does not improve average throughput. Removing the continuation update, which accounts for possible acceptance beyond a fully accepted draft, reduces throughput in all six settings. We also test a geometric model for estimating this additional acceptance, but it achieves lower mean throughput than CurveK on all three workloads (App.A.3). Together, these results support using a simple smoothed depth with the continuation update.

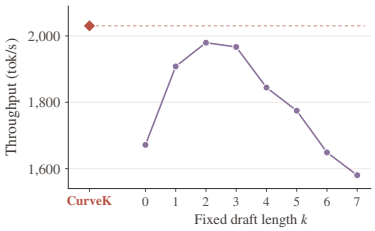

Figure 4: **Mean throughput across the 42 settings in Tabs.7 and 8.** $k = 0$ denotes vanilla decoding.

Next-step prediction. We then test whether past verifier feedback helps predict acceptance in the next step. Before verification, CurveK predicts the average number of draft tokens that will be accepted at the chosen length; we compare this prediction with the subsequently observed count. Figure 5 shows lower mean absolute error than MeanDepth-K on all four workloads, supporting the use of past feedback to predict next-step acceptance (App.A.6).

Smoothing. Finally, we vary the EWMA coefficient β, which controls how strongly each depth update responds to the latest verifier feedback. Tab.4 shows that the best coefficient varies across settings, while the default $\beta = 0.5$ remains competitive.

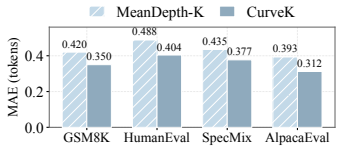

Figure 5: **Next-step mean absolute error (MAE).** Lower is better.

Table 4: **EWMA β sensitivity.** Throughput (mean $\pm$ std, tok/s). L- and Q- denote LLaMA-3.1-8B and Qwen3-8B, respectively.

Setting	$\beta = 0.1$	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.7$	$\beta = 0.9$
L-GSM8K	2794.41$_{\pm 72.5}$	2846.13$_{\pm 78.7}$	2821.5$_{\pm 122.6}$	**2923.30**$_{\pm 118.51}$	2721.04$_{\pm 124.1}$
L-HumanEval	2315.92$_{\pm 109.7}$	**2698.75**$_{\pm 89.2}$	2656.0$_{\pm 135.5}$	2662.25$_{\pm 98.6}$	2535.30$_{\pm 100.9}$
L-SpecMix	2108.46$_{\pm 27.6}$	2136.58$_{\pm 16.1}$	**2275.1**$_{\pm 11.8}$	2154.99$_{\pm 21.3}$	2093.43$_{\pm 16.1}$
Q-GSM8K	2914.53$_{\pm 17.2}$	3009.56$_{\pm 15.0}$	**3107.5**$_{\pm 17.1}$	3048.92$_{\pm 21.3}$	2867.97$_{\pm 19.8}$
Q-HumanEval	2620.71$_{\pm 21.6}$	2656.47$_{\pm 53.1}$	2672.0$_{\pm 12.5}$	**2710.16**$_{\pm 57.8}$	2561.98$_{\pm 36.5}$
Q-SpecMix	2450.93$_{\pm 22.2}$	2491.81$_{\pm 10.1}$	**2600.8**$_{\pm 8.7}$	2581.38$_{\pm 14.2}$	2426.82$_{\pm 36.1}$

6.4 Throughput Quality of Draft-Length Decisions

The next-step analysis shows that past verifier feedback helps predict acceptance, but prediction errors need not lead to poor draft-length choices. We therefore evaluate how much throughput is lost when CurveK selects a length using its estimated curve. At each step, we first record the length selected by CurveK, then verify a draft of the maximum length. The observed accepted depths let us compute throughput across candidate lengths under the profiled cost model and compare CurveK's selection with the best candidate (App.I). Across three workloads and four batch sizes, the selected lengths retain 94.7%–98.1% of the oracle-best throughput on average within each setting (Fig.6).

We also compare measured throughput against the best fixed draft length selected in hindsight (Fig.4). Although this length cannot be known in advance for an unseen serving setting, CurveK still achieves higher average throughput across the evaluated settings.

We then examine two situations that can make length selection more difficult. First, when the best length offers a larger throughput advantage over the alternatives, choosing another length incurs a greater loss. CurveK retains high throughput on these steps, with a larger advantage over a rule based only on the first-position acceptance rate than across all steps (App.I.2). Second, changes in the best length may make past feedback less useful for the next decision. On steps where the best length changes, CurveK retains 96.21% of the oracle-best throughput on average, compared with 97.68% when it stays unchanged (App.I.3). Selection therefore remains effective on these steps, although retained throughput is slightly lower.

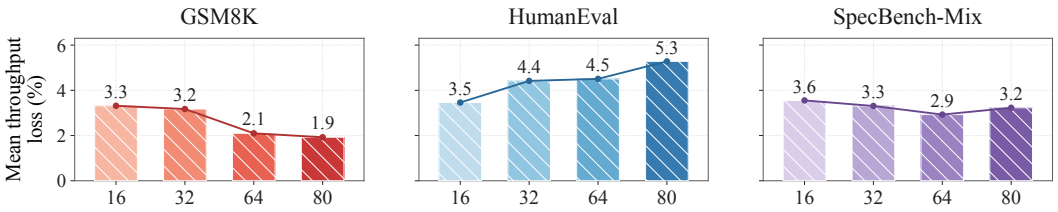

Figure 6: **Throughput quality of CurveK's draft-length decisions.** Oracle-relative throughput loss, $1 - R_{\text{oracle}}(k_{CurveK}) / \max_k R_{\text{oracle}}(k)$, averaged over steps for each workload–batch setting.

7 Conclusion

In this work, we studied online draft-length selection for batched speculative decoding. We showed that scalar acceptance summaries can render different throughput curves and preferred draft lengths indistinguishable, which motivates estimating the batch throughput curve to guide length selection. CurveK estimates this curve from the accepted depths returned by the verifier, thereby preserving request-level information while keeping the drafter and verifier unchanged. Together, throughput comparisons, ablations, and verifier-trace analyses indicate that candidate-specific curve estimation provides an effective basis for online draft-length control.

AI USE STATEMENT

Generative AI tools were used to assist with literature search, drafting and revising portions of the manuscript, and formatting. The authors take responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies inference scheduling using public benchmarks and publicly available models and drafters. It does not involve human-subject studies, private data collection, or decisions about individuals. Improving inference efficiency can reduce serving costs, including the costs of undesirable downstream uses. The work does not introduce new generative capabilities or release a new model or dataset.

REPRODUCIBILITY STATEMENT

The method and evaluation protocol are described in Secs.5 and 6. Formal assumptions and proofs are provided in Apps.B and D. Algorithm pseudocode and implementation details appear in Apps.E and F; the serving setup, cost-ratio calibration, baseline configurations, and oracle-evaluation protocol are documented in Apps.A.1 and G to I.

REFERENCES

Sudhanshu Agrawal, Wonseok Jeon, and Mingu Lee. Adaedl: Early draft stopping for speculative decoding of large language models via an entropy-based lower bound on token acceptance probability. *arXiv preprint arXiv:2410.18351*, 2024.

Leo Breiman. *Probability*. SIAM, 1992.

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.

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.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Oz335dV48X.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. *Advances in Neural Information Processing Systems*, 37:129531–129563, 2024.

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.

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

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent Y. F. Tan, and Zhuoran Yang. Banditspec: Adaptive speculative decoding via bandit algorithms. In *Forty-second International Conference on Machine Learning*, 2025.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. Specdec++: Boosting speculative decoding via adaptive candidate lengths, 2025.

Kaiyu Huang, Hao Wu, Zhubo Shi, Han Zou, Minchen Yu, and Qingjiang Shi. Adaspec: Adaptive speculative decoding for fast, slo-aware large language model serving. In *Proceedings of the 2025 ACM Symposium on Cloud Computing*, SoCC '25, pp. 361–374. Association for Computing Machinery, 2026.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

Sungkyun Kim, Jaemin Kim, Dogyung Yoon, Jiho Shin, Junyeol Lee, and Jiwon Seo. Speculative verification: Exploiting information gain to refine speculative decoding. *arXiv preprint arXiv:2509.24328*, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Rui Li, Zhaoning Zhang, Libo Zhang, Huaimin Wang, Xiang Fu, and Zhiquan Lai. Nightjar: Dynamic adaptive speculative decoding for large language models serving, 2025a. URL https://arxiv.org/abs/2512.22420v5.

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, 2024.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025b.

Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, Yi-Hsiang Lai, Xinhao Cheng, Xupeng Miao, and Zhihao Jia. Adaserve: Accelerating multi-slo llm serving with slo-customized speculative decoding. In *Proceedings of the 21st European Conference on Computer Systems*, EuroSys '26, pp. 36–54. Association for Computing Machinery, 2026.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Jiesong Liu, Brian Park, and Xipeng Shen. A drop-in solution for on-the-fly adaptation of speculative decoding in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9778–9794, 2025a.

Siran Liu, Yang Ye, Qianchao Zhu, Zane Cao, and Yongchao He. Heterospec: Leveraging contextual heterogeneity for efficient speculative decoding. *arXiv preprint arXiv:2505.13254*, 2025b.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. Pearl: Parallel speculative decoding with adaptive draft length. In *International Conference on Learning Representations*, 2025c.

Xiaoxuan Liu, Jongseok Park, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Chen Zhang, Kuntai Du, Xiangxi Mo, Kaichao You, Alvin Cheung, et al. Turbospec: Closed-loop speculation control system for optimizing llm serving goodput. *arXiv preprint arXiv:2406.14066*, 2024b.

Xiaoxuan Liu, Jiaxiang Yu, Jongseok Park, Ion Stoica, and Alvin Cheung. Speculative decoding: Performance or illusion? *Proceedings of Machine Learning and Systems*, 8:1719–1741, 2026.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Dynamic speculation lookahead accelerates speculative decoding of large language models. *arXiv preprint arXiv:2405.04304*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, 2016.

Haifeng Qian, Sujan Kumar Gonugondla, Sungsoo Ha, Mingyue Shang, Sanjay Krishna Gouda, Ramesh Nallapati, Sudipta Sengupta, Xiaofei Ma, and Anoop Deoras. Bass: Batched attention-optimized speculative sampling. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 8214–8224, 2024.

Aditya Sridhar, Nish Sinnadurai, Sean Lie, and Vithursan Thangarasa. Tapout: A bandit-based approach to dynamic speculative decoding. *arXiv preprint arXiv:2511.02017*, 2025.

Zhaoxuan Wu, Zijian Zhou, Arun Verma, Alok Prakash, Daniela Rus, and Bryan Kian Hsiang Low. Tetris: Optimal draft token selection for batch speculative decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33329–33345, 2025.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, 2024.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6460–6473, 2025. doi: 10.18653/v1/2025.naacl-long.328. URL https://aclanthology.org/2025.naacl-long.328/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Mingyu Yang, Jae-Young Choi, Kihyo Moon, Minsung Jang, and Eunjoo Jeon. Dsde: Dynamic speculative decoding with kld stability for real-world serving. *arXiv preprint arXiv:2509.01083*, 2025b.

Ming Yin, Minshuo Chen, Kaixuan Huang, and Mengdi Wang. A theoretical perspective for speculative decoding algorithm. *Advances in Neural Information Processing Systems*, 37:128082–128117, 2024.

Jiebin Zhang, Zhenghan Yu, Liang Wang, Nan Yang, Eugene Yu, Zheng Li, Yifan Song, Dawei Zhu, Xingxing Zhang, Furu Wei, et al. Learning to draft: Adaptive speculative decoding with reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 30798–30819, 2026a.

Situo Zhang, Yifan Zhang, Zichen Zhu, Hankun Wang, Da Ma, Danyang Zhang, Lu Chen, and Kai Yu. Pacer: Blockwise pre-verification for speculative decoding with adaptive length. *IEEE Transactions on Audio, Speech and Language Processing*, 2026b.

Ziyin Zhang, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Rui Wang, and Zhaopeng Tu. Draft model knows when to stop: Self-verification speculative decoding for long-form generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16685–16697. Association for Computational Linguistics, November 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023.

Appendix

Table of Contents

A Additional Empirical Results . . . 16
A.1 Serving and Measurement Setup . . . 16
A.2 Full Main Throughput Tables . . . 16
A.3 Accepted-Depth Estimator Design . . . 17
A.4 Direct Depth-Based Length Selection . . . 19
A.5 Acceptance Estimation and Aggregation Ablation . . . 19
A.6 Next-Step Accepted-Depth Prediction . . . 20
A.7 EWMA Coefficient Sensitivity . . . 21
A.8 Dialogue Workload Results . . . 21
A.9 Mixed-Trace Phase Analysis . . . 21
A.10 Medusa Drafter Analysis . . . 23
A.11 Pareto Method Comparison . . . 23
A.12 Draft-Length Decision Overhead . . . 25
A.13 Sensitivity to the Maximum Draft Length . . . 25
B Proofs and Details for the Acceptance-Probability Sequence and Throughput Curve . . . 26
B.1 Proof of Prop.1 . . . 26
B.2 Details on Summary Collisions . . . 26
C Further Discussion of Related Work . . . 27
D Additional Theory Details for CurveK . . . 28
D.1 Estimator-Error Decomposition . . . 28
E CurveK Algorithm and Implementation Details . . . 30
F Additional Deployment Details . . . 31
F.1 Scope and Portability . . . 31
G Use of c_r in the Implementation . . . 31
G.1 Sensitivity to Cost-Ratio Variation . . . 32
H Baseline Controllers and Evaluation Protocol . . . 32
H.1 BanditSpec / UCBSpec . . . 33
H.2 TurboSpec . . . 33
H.3 OTF . . . 33
H.4 Nightjar . . . 34
I Full Setup for Throughput-Curve Oracle Evaluation . . . 34
I.1 Full-Depth Verifier Comparison . . . 35
I.2 Draft-Length Selection with Separated Oracle Peaks . . . 36
I.3 Tracking Changes in the Preferred Length . . . 36
I.4 Sub-batch Sensitivity of the Throughput-Curve Estimate . . . 37

I.5 Selected-Draft-Length Alignment. 37

A Additional Empirical Results

A.1 Serving and Measurement Setup

All speculative-decoding methods use the same vLLM V1 batched serving path, with the draft-length rule as the changing component. Vanilla uses the target model without speculation. For the references in Tab.1, Standard SD uses a fixed draft length of $k = 5$ for the LLaMA targets and $k = 8$ for Qwen. Tab.5 summarizes the 8B serving setup. The 70B extension uses LLaMA-3.3-70B with its EAGLE-3 drafter, $B = 32$, $K = 7$, greedy decoding, and TP=4 for both target and drafter on A100 and H20. Within each platform, all methods share the target–drafter pair, batch size, maximum draft length, temperature, workloads, seeds, and timing boundary; independently tuned configurations are fixed across workloads.

Table 5: Serving and measurement setup for the 8B evaluations.

Item	Setting
Serving engine	vLLM V1, version 0.11.0, with the same batched serving path for all methods.
Hardware	A single A100/H20 GPU for each evaluation.
Precision and limits	bfloat16; maximum model length 4096; $K = 7$ for dynamic controllers. Single-workload evaluations use maximum generation length 1024; the mixed-workload trace uses 512.
Target and drafter pairs	We use EAGLE-family drafter–target pairs for reproducibility: LLaMA-3.1-8B-Instruct with an EAGLE-3 LLaMA drafter (Li et al., 2025b), and Qwen3-8B with a Qwen EAGLE-3 drafter.
Workload sizes	GSM8K: 1024 prompts; HumanEval: 164 prompts; SpecBench-Mix: 688 prompts; AlpacaEval: 805 prompts; mixed-workload trace: 4,080 prompts.
Decoding	Greedy decoding uses $T = 0$; sampling uses $T = 1.0$.
Warmup and timing	Throughput is the total output-token count divided by the wall-clock duration of the timed llm.generate() call, including prefill, decoding, and runtime overhead within the call. Model initialization and 16 separate warmup prompts are excluded.
Cost profiling	$c_r = 0.08$ is obtained from the forward-time profiling protocol in App.G and is held fixed during online decoding.

SpecBench-Mix construction. SpecBench-Mix is a mixed serving workload built from SpecBench prompts (Xia et al., 2024). We use eight task groups: coding, math reasoning, question answering, dialogue/MT-Bench, retrieval-augmented generation, summarization, translation, and writing. The workload balances these task groups with at most 128 prompts per group, then shuffles requests before batched serving.

We additionally report the 70B results on AlpacaEval in Tab.6, using the same serving setup and configurations as the corresponding comparisons in Tab.1. CurveK achieves the highest mean throughput on H20, while TurboSpec performs best on A100.

A.2 Full Main Throughput Tables

Tabs.7 and 8 provide the full greedy and sampling throughput values behind the static-k reference in Fig.1. The $k^\star$ column is a *hindsight* reference: it picks the best fixed depth after the workload, batch size, and decoding setting are known.

Table 6: **AlpacaEval throughput with LLaMA-3.3-70B and EAGLE-3.** Values are mean±std in tok/s. Bold marks the highest mean.

GPU	BanditSpec	TurboSpec	OTF	Nightjar	CurveK
A100	$1184.61_{\pm 14.79}$	**$1272.96_{\pm 11.72}$**	$1211.99_{\pm 17.93}$	$1211.39_{\pm 103.97}$	$1240.54_{\pm 6.50}$
H20	$1459.80_{\pm 48.10}$	$1351.24_{\pm 17.60}$	$1478.30_{\pm 31.70}$	$1407.35_{\pm 15.35}$	**$1642.00_{\pm 41.20}$**

Table 7: Full LLaMA-3.1-8B greedy-decoding throughput values for the static-k reference.

		Throughput (tokens/sec)										
Workload	B	Vanilla	k=1	k=2	k=3	k=4	k=5	k=6	k=7	**CurveK**	$k^\star$	Δ%
GSM8K	4	343.0$_{\pm 0.3}$	504.9$_{\pm 1.9}$	554.4$_{\pm 1.9}$	**586.8**$_{\pm 0.6}$	560.6$_{\pm 1.8}$	555.2$_{\pm 2.4}$	541.6$_{\pm 2.4}$	516.6$_{\pm 2.4}$	**590.4**$_{\pm 2.1}$	3	+0.6%
GSM8K	8	665.2$_{\pm 0.2}$	947.1$_{\pm 2.7}$	988.9$_{\pm 7.7}$	1045.3$_{\pm 3.3}$	**1095.3**$_{\pm 9.1}$	988.0$_{\pm 12.8}$	954.3$_{\pm 5.8}$	925.1$_{\pm 7.0}$	**1125.3**$_{\pm 5.9}$	4	+2.7%
GSM8K	16	1192.3$_{\pm 32.0}$	1583.8$_{\pm 12.6}$	1684.7$_{\pm 8.6}$	**1761.2**$_{\pm 18.9}$	1713.3$_{\pm 33.0}$	1705.2$_{\pm 15.1}$	1666.1$_{\pm 7.9}$	1617.5$_{\pm 17.5}$	**1793.8**$_{\pm 29.2}$	3	+1.9%
GSM8K	32	1959.8$_{\pm 36.8}$	2451.4$_{\pm 71.7}$	2752.8$_{\pm 44.2}$	**2802.6**$_{\pm 72.4}$	2632.6$_{\pm 39.9}$	2471.9$_{\pm 44.6}$	2380.3$_{\pm 46.7}$	2305.9$_{\pm 63.8}$	**2821.5**$_{\pm 122.6}$	3	+0.7%
GSM8K	64	3019.3$_{\pm 193.0}$	3654.7$_{\pm 94.7}$	**3771.3**$_{\pm 11.9}$	3641.3$_{\pm 141.3}$	3428.0$_{\pm 66.3}$	3203.4$_{\pm 52.0}$	2869.9$_{\pm 66.4}$	2669.8$_{\pm 7.5}$	**3794.6**$_{\pm 22.0}$	2	+0.6%
GSM8K	80	3384.0$_{\pm 141.3}$	3788.2$_{\pm 190.6}$	**3859.0**$_{\pm 95.8}$	3693.5$_{\pm 86.7}$	3470.9$_{\pm 106.4}$	3290.6$_{\pm 64.4}$	2925.8$_{\pm 38.9}$	2799.0$_{\pm 74.5}$	**3841.6**$_{\pm 159.0}$	2	-0.5%
HumanEval	4	339.6$_{\pm 2.0}$	508.0$_{\pm 2.6}$	571.6$_{\pm 1.9}$	**616.7**$_{\pm 5.8}$	598.0$_{\pm 0.9}$	595.9$_{\pm 9.8}$	588.3$_{\pm 3.1}$	570.3$_{\pm 1.7}$	**643.7**$_{\pm 4.1}$	3	+4.4%
HumanEval	8	639.9$_{\pm 6.7}$	940.9$_{\pm 7.8}$	1005.1$_{\pm 14.1}$	1079.0$_{\pm 10.6}$	**1159.5**$_{\pm 8.8}$	1057.7$_{\pm 6.4}$	1035.5$_{\pm 7.2}$	1016.4$_{\pm 13.8}$	**1185.9**$_{\pm 10.8}$	4	+2.3%
HumanEval	16	1097.6$_{\pm 35.5}$	1482.4$_{\pm 47.1}$	1658.4$_{\pm 43.8}$	**1808.2**$_{\pm 7.7}$	1747.1$_{\pm 17.1}$	1787.1$_{\pm 11.4}$	1736.1$_{\pm 20.7}$	1668.3$_{\pm 65.8}$	**1859.2**$_{\pm 49.4}$	3	+2.8%
HumanEval	32	1688.5$_{\pm 126.9}$	2105.3$_{\pm 25.6}$	2465.9$_{\pm 155.4}$	**2549.6**$_{\pm 83.8}$	2497.7$_{\pm 58.2}$	2416.2$_{\pm 70.5}$	2412.1$_{\pm 118.2}$	2400.1$_{\pm 49.1}$	**2656.0**$_{\pm 135.5}$	3	+4.2%
HumanEval	64	2416.7$_{\pm 216.9}$	3112.0$_{\pm 62.5}$	3228.6$_{\pm 270.3}$	**3319.3**$_{\pm 75.8}$	3067.9$_{\pm 84.8}$	3155.1$_{\pm 63.5}$	2795.8$_{\pm 65.2}$	2670.0$_{\pm 56.2}$	**3327.9**$_{\pm 50.0}$	3	+0.3%
HumanEval	80	2578.1$_{\pm 185.7}$	2970.3$_{\pm 101.2}$	**3474.8**$_{\pm 221.7}$	3355.4$_{\pm 105.3}$	3296.8$_{\pm 58.7}$	3232.3$_{\pm 103.0}$	2856.8$_{\pm 36.1}$	2778.9$_{\pm 174.9}$	**3462.4**$_{\pm 415.0}$	2	-0.4%
SpecBench-Mix	4	336.8$_{\pm 0.7}$	468.1$_{\pm 2.0}$	506.5$_{\pm 1.5}$	**531.9**$_{\pm 2.4}$	510.9$_{\pm 2.1}$	504.6$_{\pm 3.3}$	493.7$_{\pm 3.0}$	473.0$_{\pm 3.2}$	**546.7**$_{\pm 0.7}$	3	+2.8%
SpecBench-Mix	8	635.4$_{\pm 3.5}$	860.4$_{\pm 3.7}$	888.1$_{\pm 5.3}$	929.5$_{\pm 4.5}$	**977.0**$_{\pm 5.1}$	886.3$_{\pm 3.3}$	856.7$_{\pm 2.7}$	840.2$_{\pm 3.0}$	**997.5**$_{\pm 2.4}$	4	+2.1%
SpecBench-Mix	16	1128.9$_{\pm 13.6}$	1384.1$_{\pm 5.3}$	1449.0$_{\pm 6.3}$	**1523.4**$_{\pm 7.8}$	1483.5$_{\pm 8.2}$	1479.1$_{\pm 2.4}$	1438.0$_{\pm 8.9}$	1399.3$_{\pm 3.6}$	**1496.2**$_{\pm 9.7}$	3	-1.8%
SpecBench-Mix	32	1732.3$_{\pm 31.4}$	2080.0$_{\pm 37.8}$	2227.5$_{\pm 47.7}$	**2281.8**$_{\pm 23.9}$	2136.9$_{\pm 23.0}$	2060.1$_{\pm 9.7}$	2034.3$_{\pm 19.7}$	1963.1$_{\pm 13.0}$	**2275.1**$_{\pm 11.8}$	3	-0.3%
SpecBench-Mix	64	2462.6$_{\pm 57.1}$	2819.4$_{\pm 47.0}$	**2932.2**$_{\pm 41.8}$	2906.9$_{\pm 39.3}$	2721.7$_{\pm 44.0}$	2600.7$_{\pm 24.2}$	2393.4$_{\pm 20.6}$	2271.5$_{\pm 17.1}$	**2906.7**$_{\pm 35.4}$	2	-0.9%
AlpacaEval	16	1233.2$_{\pm 7.2}$	1510.1$_{\pm 8.1}$	1540.5$_{\pm 5.2}$	**1609.1**$_{\pm 10.1}$	1542.8$_{\pm 3.6}$	1528.4$_{\pm 9.8}$	1476.4$_{\pm 6.8}$	1434.0$_{\pm 6.3}$	**1579.8**$_{\pm 11.7}$	3	-1.8%
AlpacaEval	32	2058.1$_{\pm 14.6}$	2411.7$_{\pm 13.0}$	2541.0$_{\pm 15.9}$	**2641.6**$_{\pm 5.8}$	2333.4$_{\pm 19.8}$	2296.3$_{\pm 11.7}$	2212.4$_{\pm 3.4}$	2126.1$_{\pm 8.9}$	**2657.2**$_{\pm 30.3}$	3	+0.6%
AlpacaEval	64	3191.0$_{\pm 11.4}$	3565.7$_{\pm 10.1}$	**3578.6**$_{\pm 3.2}$	3529.0$_{\pm 33.8}$	3160.9$_{\pm 40.1}$	3002.0$_{\pm 33.6}$	2681.0$_{\pm 18.9}$	2507.7$_{\pm 22.3}$	**3596.4**$_{\pm 7.4}$	2	+0.5%
AlpacaEval	80	3530.4$_{\pm 24.4}$	3700.8$_{\pm 17.7}$	**3874.0**$_{\pm 53.5}$	3647.6$_{\pm 17.2}$	3222.2$_{\pm 23.0}$	3147.5$_{\pm 8.2}$	2696.0$_{\pm 23.6}$	2625.0$_{\pm 40.4}$	**3786.8**$_{\pm 47.8}$	2	-2.3%

After evaluating all candidate draft lengths, we identify the fixed length with the highest average throughput across the evaluated settings: $k = 3$ under greedy decoding and $k = 2$ under sampling. CurveK improves average throughput by 2.37% and 1.97%, respectively, over these hindsight-selected fixed lengths, without evaluating every candidate length for each new serving configuration. The best fixed length also varies across individual settings, motivating adaptive length selection as serving conditions change.

These evaluations also **show the throughput cost of an unsuitable draft length**. Relative to the best fixed length in each setting, the worst fixed length reduces throughput by a median of 17.7% under greedy decoding and 18.5% under sampling. Even the hindsight-selected fixed lengths above incur losses of up to 6.9% and 9.3%, respectively, relative to selecting the best fixed length separately for each setting.

A.3 Accepted-Depth Estimator Design

We examine two choices in CurveK's estimator: keeping one smoothed depth per request and accounting for possible continuation when a draft is fully accepted. The first comparison replaces the depth state or removes the continuation update. The second compares CurveK with a geometric model that explicitly fits the unobserved continuation.

Depth history and continuation update. Tab.9 tests two variants of the accepted-depth estimator while keeping the same throughput-curve scoring rule, candidate set, and serving path as CurveK. *History seq.* maintains a smoothed acceptance probability at each draft position for each request, instead of using a single smoothed depth to estimate the throughput curve. *w/o continuation* keeps CurveK's depth smoothing and throughput scoring unchanged. When all k draft tokens are accepted and $k < K$, it records k instead of $k + 1$.

Table 8: Full LLaMA-3.1-8B sampling-decoding throughput values for the static-k reference.

		Throughput (tokens/sec, T=1.0)											
Workload	B	Vanilla	k=1	k=2	k=3	k=4	k=5	k=6	k=7	**CurveK**	$k^\star$	Δ%
GSM8K	4	340.6$_{\pm 0.9}$	453.3$_{\pm 9.4}$	467.7$_{\pm 6.5}$	**478.6**$_{\pm 15.4}$	452.5$_{\pm 8.7}$	445.1$_{\pm 6.9}$	425.6$_{\pm 13.8}$	405.8$_{\pm 9.2}$	**469.3**$_{\pm 12.1}$	3	-1.9%
GSM8K	8	657.8$_{\pm 0.9}$	844.9$_{\pm 19.6}$	834.2$_{\pm 6.3}$	851.2$_{\pm 25.8}$	**868.6**$_{\pm 23.0}$	789.3$_{\pm 19.2}$	746.1$_{\pm 7.1}$	703.6$_{\pm 21.6}$	**860.7**$_{\pm 17.6}$	4	-0.9%
GSM8K	16	1199.7$_{\pm 22.3}$	1373.8$_{\pm 51.7}$	1388.0$_{\pm 58.8}$	**1435.2**$_{\pm 68.6}$	1349.1$_{\pm 71.7}$	1342.5$_{\pm 50.7}$	1294.5$_{\pm 53.8}$	1219.0$_{\pm 70.8}$	**1474.9**$_{\pm 19.0}$	3	+2.8%
GSM8K	32	1886.9$_{\pm 28.8}$	2177.0$_{\pm 91.4}$	**2336.5**$_{\pm 61.6}$	2315.1$_{\pm 70.4}$	1921.9$_{\pm 63.3}$	1949.4$_{\pm 65.6}$	1770.7$_{\pm 43.7}$	1729.7$_{\pm 17.6}$	**2275.3**$_{\pm 72.8}$	2	-2.6%
GSM8K	64	2808.6$_{\pm 26.0}$	2915.0$_{\pm 26.1}$	**2944.7**$_{\pm 254.6}$	2799.6$_{\pm 79.8}$	2483.1$_{\pm 64.0}$	2448.6$_{\pm 185.4}$	2074.3$_{\pm 42.5}$	1975.3$_{\pm 79.1}$	**2984.8**$_{\pm 5.6}$	2	+1.4%
GSM8K	80	3079.4$_{\pm 82.4}$	**3207.0**$_{\pm 219.4}$	2987.8$_{\pm 79.0}$	2893.8$_{\pm 243.6}$	2487.2$_{\pm 66.1}$	2335.4$_{\pm 62.8}$	2095.1$_{\pm 74.2}$	2050.8$_{\pm 110.9}$	**3105.1**$_{\pm 89.7}$	1	-3.2%
HumanEval	4	337.8$_{\pm 0.7}$	472.1$_{\pm 4.2}$	521.3$_{\pm 4.5}$	**547.1**$_{\pm 5.0}$	528.3$_{\pm 10.1}$	520.7$_{\pm 7.4}$	505.8$_{\pm 6.9}$	490.8$_{\pm 10.4}$	**535.7**$_{\pm 4.9}$	3	-2.1%
HumanEval	8	636.6$_{\pm 2.8}$	855.0$_{\pm 19.1}$	908.0$_{\pm 6.3}$	957.0$_{\pm 24.4}$	**1000.6**$_{\pm 27.4}$	912.7$_{\pm 22.0}$	885.5$_{\pm 22.9}$	860.5$_{\pm 16.3}$	**1053.2**$_{\pm 23.7}$	4	+5.3%
HumanEval	16	1102.1$_{\pm 61.5}$	1366.0$_{\pm 47.3}$	1512.7$_{\pm 30.0}$	**1547.3**$_{\pm 49.3}$	1525.0$_{\pm 31.6}$	1544.7$_{\pm 13.4}$	1509.0$_{\pm 42.7}$	1393.1$_{\pm 140.5}$	**1552.5**$_{\pm 37.0}$	3	+0.3%
HumanEval	32	1743.2$_{\pm 56.7}$	2166.5$_{\pm 45.2}$	**2180.2**$_{\pm 96.5}$	2138.4$_{\pm 317.2}$	2115.5$_{\pm 174.6}$	2046.1$_{\pm 142.3}$	2083.5$_{\pm 10.7}$	1971.5$_{\pm 106.9}$	**2351.5**$_{\pm 90.1}$	2	+7.9%
HumanEval	64	2295.1$_{\pm 83.5}$	2756.0$_{\pm 94.1}$	2843.6$_{\pm 56.2}$	**2945.6**$_{\pm 85.6}$	2733.4$_{\pm 197.7}$	2594.4$_{\pm 187.9}$	2292.2$_{\pm 75.9}$	2069.3$_{\pm 67.8}$	**2833.6**$_{\pm 257.6}$	3	-3.8%
HumanEval	80	2532.7$_{\pm 137.0}$	2835.0$_{\pm 80.5}$	**3179.0**$_{\pm 112.2}$	2723.6$_{\pm 343.1}$	2864.7$_{\pm 153.3}$	2488.5$_{\pm 496.3}$	2440.9$_{\pm 86.9}$	2295.3$_{\pm 54.0}$	**3051.8**$_{\pm 133.9}$	2	-4.0%
SpecBench-Mix	4	333.9$_{\pm 0.3}$	422.2$_{\pm 5.4}$	434.3$_{\pm 5.9}$	**438.3**$_{\pm 4.5}$	413.6$_{\pm 14.8}$	406.0$_{\pm 3.3}$	396.7$_{\pm 10.5}$	379.3$_{\pm 3.1}$	**435.2**$_{\pm 9.6}$	3	-0.7%
SpecBench-Mix	8	629.1$_{\pm 3.9}$	781.5$_{\pm 10.3}$	760.5$_{\pm 9.1}$	**783.8**$_{\pm 13.8}$	783.8$_{\pm 25.2}$	717.1$_{\pm 11.2}$	679.9$_{\pm 9.1}$	672.4$_{\pm 24.1}$	**769.6**$_{\pm 14.8}$	3	-1.8%
SpecBench-Mix	16	1112.5$_{\pm 14.3}$	1238.3$_{\pm 6.2}$	1241.3$_{\pm 28.1}$	**1271.3**$_{\pm 31.7}$	1207.5$_{\pm 26.4}$	1208.2$_{\pm 26.6}$	1115.6$_{\pm 24.3}$	1125.5$_{\pm 31.5}$	**1285.4**$_{\pm 14.3}$	3	+1.1%
SpecBench-Mix	32	1751.0$_{\pm 11.2}$	1849.2$_{\pm 50.4}$	**1881.3**$_{\pm 40.8}$	1865.0$_{\pm 75.2}$	1719.8$_{\pm 84.7}$	1639.3$_{\pm 77.1}$	1645.4$_{\pm 69.8}$	1534.0$_{\pm 76.8}$	**1863.2**$_{\pm 17.6}$	2	-1.0%
SpecBench-Mix	64	2468.3$_{\pm 49.4}$	**2429.9**$_{\pm 76.7}$	2331.3$_{\pm 36.1}$	2305.5$_{\pm 57.0}$	2165.8$_{\pm 128.6}$	2007.7$_{\pm 26.6}$	1818.2$_{\pm 34.1}$	1733.8$_{\pm 69.2}$	**2455.3**$_{\pm 81.1}$	1	+1.0%
AlpacaEval	16	1211.4$_{\pm 2.5}$	**1267.7**$_{\pm 19.7}$	1218.4$_{\pm 27.4}$	1224.3$_{\pm 26.7}$	1147.8$_{\pm 37.8}$	1116.6$_{\pm 42.8}$	1077.0$_{\pm 26.8}$	1040.9$_{\pm 27.2}$	**1216.0**$_{\pm 33.3}$	1	-4.1%
AlpacaEval	32	2003.2$_{\pm 6.2}$	**2000.5**$_{\pm 49.3}$	1971.8$_{\pm 54.7}$	1955.0$_{\pm 53.1}$	1724.3$_{\pm 14.9}$	1659.3$_{\pm 48.4}$	1567.1$_{\pm 60.8}$	1516.3$_{\pm 32.6}$	**1969.4**$_{\pm 36.8}$	1	-1.6%
AlpacaEval	64	3056.2$_{\pm 17.4}$	**2885.0**$_{\pm 51.5}$	2710.2$_{\pm 52.1}$	2580.6$_{\pm 25.6}$	2301.0$_{\pm 18.4}$	2166.6$_{\pm 31.2}$	1885.8$_{\pm 64.9}$	1792.1$_{\pm 41.8}$	**2793.1**$_{\pm 41.8}$	1	-3.2%
AlpacaEval	80	3394.6$_{\pm 14.3}$	**2994.8**$_{\pm 63.7}$	2939.0$_{\pm 74.0}$	2685.2$_{\pm 73.1}$	2316.0$_{\pm 89.3}$	2237.6$_{\pm 57.7}$	1898.5$_{\pm 50.0}$	1832.6$_{\pm 66.2}$	**2995.4**$_{\pm 74.4}$	1	0.0%

Table 9: **Estimator-form ablation on LLaMA-3.1-8B and Qwen3-8B.** Throughput (mean$\pm$std over three seeds, tok/s). L-/Q- denote the two targets with their EAGLE-3 drafters. All settings use $B = 32$, greedy decoding, $K = 7$, $c_r = 0.08$, and $\beta = 0.5$. Gain compares CurveK with the variant without the continuation update; Avg. is the mean of the per-setting gains.

Setting	History seq.	w/o continuation	CurveK	Gain
L-GSM8K	2478.1$_{\pm 7.0}$	2573.6$_{\pm 83.1}$	**2821.5**$_{\pm 122.6}$	+9.6%
L-HumanEval	2378.0$_{\pm 62.7}$	2403.9$_{\pm 170.2}$	**2656.0**$_{\pm 135.5}$	+10.5%
L-SpecMix	2075.0$_{\pm 7.3}$	2080.2$_{\pm 73.5}$	**2275.1**$_{\pm 11.8}$	+9.4%
Q-GSM8K	2679.8$_{\pm 10.2}$	2839.2$_{\pm 96.8}$	**3107.5**$_{\pm 17.1}$	+9.4%
Q-HumanEval	2380.1$_{\pm 7.6}$	2427.3$_{\pm 19.7}$	**2672.0**$_{\pm 12.5}$	+10.1%
Q-SpecMix	2341.0$_{\pm 3.5}$	2379.9$_{\pm 12.9}$	**2600.8**$_{\pm 8.7}$	+9.3%
Avg.	2388.7	2450.7	**2688.8**	+9.7%

CurveK achieves higher mean throughput than both variants in all six settings, improving over the variant without the continuation update by 9.7% on average. This supports the scalar EWMA estimator with the one-step continuation update used in CurveK.

Censored geometric alternative. We compare CurveK with a per-request censored geometric estimator that models the unobserved continuation after a fully accepted draft. For each request, the estimator keeps its most recent 32 accepted-depth observations. Let A_i be the sum of accepted depths and D_i the number of observations ending in rejection. The fitted acceptance parameter and batch position-wise estimate are

$$a_i = \frac{A_i}{A_i + D_i}, \qquad \widehat{p}_j = \frac{1}{B}\sum_{i=1}^{B} a_i^j.$$

Fully accepted drafts contribute to A_i without adding a rejection to D_i. The estimator sums $\widehat{p}_j$ to obtain the accepted-token benefit and selects the highest-scoring length in $\{1, \ldots, K\}$ under the same throughput objective as CurveK.

Table 10: Censored geometric estimation versus CurveK on LLaMA-3.1-8B with EAGLE-3. All settings use $B = 32$, $K = 7$, greedy decoding, and $c_r = 0.08$. Values are mean±std throughput in tok/s.

Workload	Censored geometric MLE	CurveK
GSM8K	$2511.39_{\pm 34.91}$	**2821.5**$_{\pm 122.6}$
HumanEval	$2354.70_{\pm 133.42}$	**2656.0**$_{\pm 135.5}$
SpecMix	$2092.39_{\pm 23.03}$	**2275.1**$_{\pm 11.8}$

Tab.10 reports the comparison using the CurveK results from Tab.1. CurveK achieves higher mean throughput on all three workloads. Together, the comparisons show that the history-based sequence and geometric model do not improve average throughput over CurveK in these settings, while the simple continuation update improves average throughput over omitting it.

A.4 Direct Depth-Based Length Selection

We test whether selecting a draft length from the average accepted depth is sufficient, or whether throughput scoring improves the choice. DirectDepth uses the same depth updates as CurveK, but selects the next draft length directly from the mean of the per-request depth estimates. CurveK instead uses these estimates to construct a throughput curve and selects the draft length at its peak. The results in Tab.11 show that CurveK achieves higher mean throughput in all eight comparisons, supporting the benefit of throughput-based selection.

Table 11: Direct depth-based selection versus CurveK. All settings use $B = 32$, $K = 7$, and greedy decoding. Values are mean±std throughput in tok/s.

Model	Workload	DirectDepth	CurveK
LLaMA-3.1-8B	GSM8K	$2627.2_{\pm 20.8}$	**2821.5**$_{\pm 122.6}$
	HumanEval	$2416.6_{\pm 76.9}$	**2656.0**$_{\pm 135.5}$
	SpecMix	$2109.3_{\pm 29.6}$	**2275.1**$_{\pm 11.8}$
	AlpacaEval	$2381.1_{\pm 7.6}$	**2657.2**$_{\pm 30.3}$
Qwen3-8B	GSM8K	$2836.6_{\pm 7.2}$	**3107.5**$_{\pm 17.1}$
	HumanEval	$2538.0_{\pm 10.1}$	**2672.0**$_{\pm 12.5}$
	SpecMix	$2430.4_{\pm 7.1}$	**2600.8**$_{\pm 8.7}$
	AlpacaEval	$2571.8_{\pm 2.2}$	**2623.9**$_{\pm 64.2}$

A.5 Acceptance Estimation and Aggregation Ablation

The acceptance-estimator ablation in Sec.6 keeps the same throughput-scoring rule

$$R(k) = \frac{1 + L(k)}{1 + kc_r}$$

and changes how $L(k)$ is estimated. We compare two ways of estimating acceptance under the same cost model and length-selection rule: draft-confidence products (Huang et al., 2026) and verifier-returned accepted depths. Let $\mathcal{B}_t$ be the active batch and $B_t = |\mathcal{B}_t|$. MeanDepth-K uses the same accepted-depth EWMA update rule as CurveK, but averages the request depths before candidate-specific truncation:

$$\bar{\mu}_t = \frac{1}{B_t} \sum_{i \in \mathcal{B}_t} \mu_{t,i}, \qquad \widehat{L}_{\text{mean}}(k) = \min(k, \bar{\mu}_t).$$

CurveK instead uses $\widehat{L}_{\text{CurveK}}(k) = B_t^{-1} \sum_{i \in \mathcal{B}_t} \min(k, \mu_{t,i})$. This comparison tests whether truncating each request's depth before averaging improves draft-length selection.

AdaSpec (Huang et al., 2026) writes the per-position acceptance probability as a cumulative product of drafter softmax confidences. For request i, let

$$c_{i,l} = \max_v \text{softmax}(\text{logits}_{i,l})_v$$

be the drafter's maximum softmax confidence at draft position l. We compute

$$\widehat{p}_j^{(i)} = \prod_{l=1}^{j} c_{i,l}, \qquad \widehat{p}_j = \frac{1}{B_t} \sum_{i \in \mathcal{B}_t} \widehat{p}_j^{(i)}.$$

This follows the original estimator form: multiply along each request first, then average over the batch. We then form the cumulative acceptance estimate and use the same $R(k)$ scoring formula as CurveK. Before the first observed draft, unobserved per-position estimates are initialized to 1.0. From the second step onward, we apply EMA smoothing

$$\widehat{p}_j^{(t+1)} \leftarrow (1-\alpha)\widehat{p}_j^{(t)} + \alpha\, \widehat{p}_j^{\mathrm{batch},(t)},$$

with $\alpha = 0.5$, matching the CurveK setting in the primary evaluation. The selection formula is unchanged:

$$k_{t+1} = \operatorname*{arg\,max}_{k \in \{0,\ldots,K\}} R(k).$$

This ablation fixes $c_r = 0.08$ and uses the same candidate lengths and throughput-scoring rule as CurveK. It compares confidence-based and accepted-depth estimators; MeanDepth-K separately tests the effect of averaging request depths before evaluating candidate lengths.

AdaSpec estimates acceptance from draft-side confidence and searches a quasi-concave objective. For this estimator comparison, we feed the confidence-derived estimates into the same explicit argmax over $\{0, \ldots, K\}$ used by CurveK. This keeps the search rule fixed across the compared estimators.

In the vLLM batched speculative-decoding path, the drafter proposes k tokens for all requests in the batch at the same step. After computing drafter logits, we record the maximum softmax value at each draft position. For each position j, the values $\{c_{i,j}\}_{i=1}^{B}$ are stored during drafting to form one confidence sequence per request. The AdaSpec-style estimator uses these sequences over the same draft-length action space as the other methods; CurveK and the four primary baselines use accepted-depth feedback. If no draft token is proposed in a step, the estimator leaves the confidence sequence unchanged until the next completed draft.

We use the following estimator settings. The EMA coefficient is $\alpha = 0.5$, matching CurveK in the main table. Max-softmax values are computed in fp32. If the recorded confidence sequence is shorter than k, the cumulative product is truncated to the available length; missing positions keep the previous EMA value. The EMA variables persist across batch-size changes. This estimator comparison uses the same batched speculative-decoding path as the other evaluated methods.

Results. Tab.12 gives the full results for the three estimators summarized in Tab.3. All settings use $B = 32$, $K = 7$, greedy decoding, and $c_r = 0.08$.

Table 12: Draft-confidence and batch-mean accepted-depth variants versus CurveK. L- and Q- denote LLaMA-3.1-8B and Qwen3-8B; SpecMix denotes SpecBench-Mix. Values are mean±std throughput in tokens/sec.

Setting	Confidence	MeanDepth-K	CurveK
L-GSM8K	$2445.7_{\pm 123.8}$	$2674.2_{\pm 53.7}$	$\mathbf{2821.5}_{\pm 122.6}$
L-HumanEval	$2118.1_{\pm 63.2}$	$2483.8_{\pm 68.0}$	$\mathbf{2656.0}_{\pm 135.5}$
L-SpecMix	$1964.0_{\pm 99.5}$	$2139.1_{\pm 31.3}$	$\mathbf{2275.1}_{\pm 11.8}$
Q-GSM8K	$2735.5_{\pm 97.9}$	$2820.3_{\pm 9.4}$	$\mathbf{3107.5}_{\pm 17.1}$
Q-HumanEval	$2345.3_{\pm 54.2}$	$2521.1_{\pm 33.6}$	$\mathbf{2672.0}_{\pm 12.5}$
Q-SpecMix	$2347.3_{\pm 11.4}$	$2456.4_{\pm 23.5}$	$\mathbf{2600.8}_{\pm 8.7}$
Avg.	2326.0	2515.8	**2688.8**

In Tab.12, MeanDepth-K achieves higher mean throughput than the confidence variant in all six settings, and CurveK improves further in every setting.

A.6 Next-Step Accepted-Depth Prediction

We evaluate predictions formed before each verification step against the batch-average accepted depth observed in that step. The analysis uses LLaMA-3.3-70B on A100 with $B = 32$, $c_r = 0.1$,

and $\beta = 0.5$. We evaluate decode steps for which all active requests have earlier verifier feedback. CurveK predicts the accepted-token count at the chosen length using $\widehat{L}_t(k_t)$. The batch-mean alternative first averages the depth states and then truncates this mean at k_t.

Table 13: Next-step accepted-depth prediction on LLaMA-3.3-70B at $B = 32$. Correlation is measured against the observed batch-average accepted depth. Mean absolute error (MAE) is in tokens; lower is better.

Workload	CurveK correlation	CurveK MAE	Batch-mean MAE
GSM8K	0.422	**0.350**	0.420
HumanEval	0.406	**0.404**	0.488
SpecMix	0.419	**0.377**	0.435
AlpacaEval	0.300	**0.312**	0.393

The positive correlations in Tab.13 show that recent history carries information about the next verification step. CurveK's predictions have lower mean absolute error than the batch-mean alternative on all four workloads. This analysis evaluates the predictive signal, while the phase traces in App.A.9 show how selected lengths change with the workload.

A.7 EWMA Coefficient Sensitivity

Tab.14 reports the EWMA sensitivity results shown in Tab.4. The highest-throughput coefficient varies across settings, while the default $\beta = 0.5$ remains competitive with nearby values.

Table 14: EWMA β sensitivity. All settings use $B = 32$ and greedy decoding. Values are throughput in tokens/sec.

Setting	$\beta = 0.1$	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.7$	$\beta = 0.9$	Best
LLaMA-GSM8K	2794.41$_{\pm 72.5}$	2846.13$_{\pm 78.7}$	2821.5$_{\pm 122.6}$	**2923.30**$_{\pm 118.51}$	2721.04$_{\pm 124.1}$	0.7
LLaMA-HumanEval	2315.92$_{\pm 109.7}$	**2698.75**$_{\pm 89.2}$	2656.0$_{\pm 135.5}$	2662.25$_{\pm 98.6}$	2535.30$_{\pm 100.9}$	0.3
LLaMA-SpecMix	2108.46$_{\pm 27.6}$	2136.58$_{\pm 16.1}$	**2275.1**$_{\pm 11.8}$	2154.99$_{\pm 21.3}$	2093.43$_{\pm 16.1}$	0.5
Qwen3-GSM8K	2914.53$_{\pm 17.2}$	3009.56$_{\pm 15.0}$	**3107.5**$_{\pm 17.1}$	3048.92$_{\pm 21.3}$	2867.97$_{\pm 19.8}$	0.5
Qwen3-HumanEval	2620.71$_{\pm 21.6}$	2656.47$_{\pm 53.1}$	2672.0$_{\pm 12.5}$	**2710.16**$_{\pm 57.8}$	2561.98$_{\pm 36.5}$	0.7
Qwen3-SpecMix	2450.93$_{\pm 22.2}$	2491.81$_{\pm 10.1}$	**2600.8**$_{\pm 8.7}$	2581.38$_{\pm 14.2}$	2426.82$_{\pm 36.1}$	0.5

A.8 Dialogue Workload Results

LMSYS-Chat-1M (Zheng et al., 2023) adds a real-world dialogue workload to the evaluation. Under the same LLaMA-3.1-8B serving setup, CurveK achieves the highest mean throughput at both evaluated batch sizes (Tab.15).

A.9 Mixed-Trace Phase Analysis

Fig.7 examines the $B = 32, T = 0$ mixed-workload trace at the batch-step level. For each step, requests are grouped by workload phase, and CurveK's selected draft length is compared with the accepted depth returned by the verifier. The selected length increases in phases with higher accepted depths, such as code-heavy and math-code mixed phases, and decreases in dialogue-heavy and long-context-heavy phases. This analysis is consistent with the aggregate dynamic-method result in Tab.2: CurveK adapts to changing batch composition using verifier feedback. Phase labels are applied after the run solely to aggregate the plotted statistics.

Table 15: **LMSYS-Chat-1M dialogue workload results.** Throughput tokens/sec on LLaMA-3.1-8B under greedy decoding, using the same dynamic-k baseline set and serving path as the main comparison.

B	OTF	Turbo	Bandit	Nightjar	CurveK	vs. best prior
64	3262.11	3287.89	2948.59	3195.52	**3375.17**	+2.65%
80	3371.87	3475.10	3044.36	3364.13	**3611.88**	+3.94%

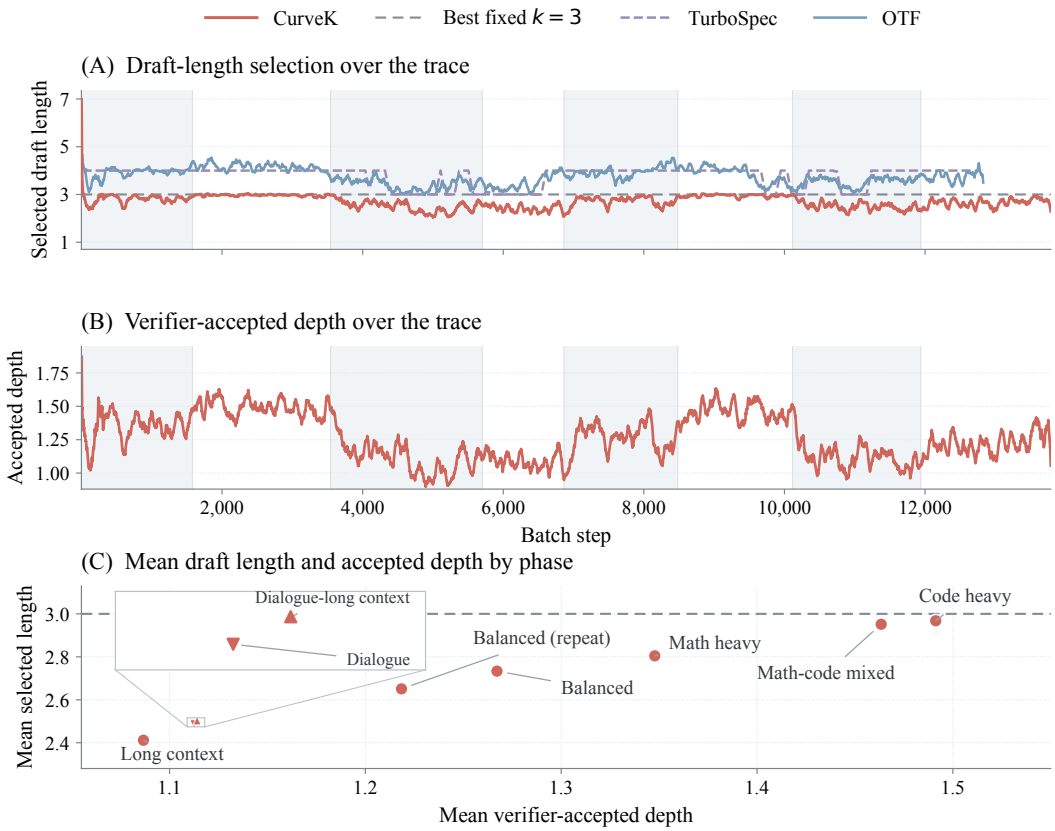

Figure 7: **Mixed-workload trace phase analysis.** LLaMA-3.1-8B + EAGLE-3 on the mixed-workload trace at $B = 32, T = 0$. (A) Rolling selected draft length over batch steps; the best fixed depth $k = 3$ is shown as a reference, with TurboSpec and OTF traces from the same setting. (B) Rolling accepted depth returned by the verifier under CurveK. (C) Per-phase mean CurveK selected length versus per-phase mean accepted depth returned by the verifier.

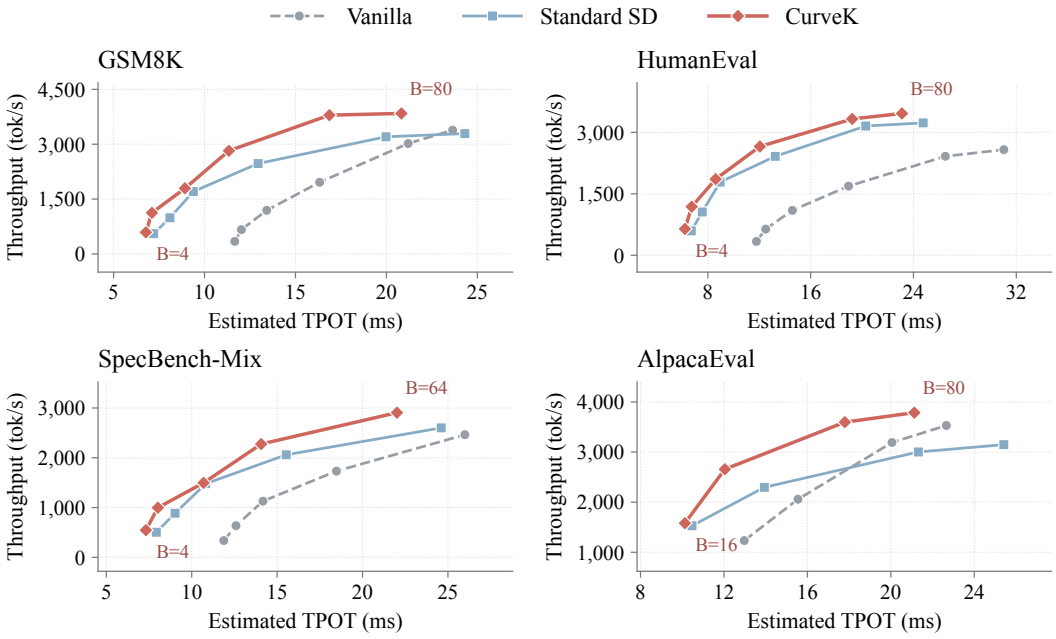

Figure 8: Throughput–TPOT Pareto frontier under greedy decoding. Vanilla is no speculative decoding; Standard SD is speculative decoding with fixed draft length $k = 5$; CurveK is the proposed dynamic-k method. Each point is a batch size; TPOT is approximated as B divided by throughput.

A.10 Medusa Drafter Analysis

In addition to the EAGLE-family drafter used in the primary experiments, we evaluate a Medusa drafter (Cai et al., 2024) with LLaMA-3.1-8B-Instruct. This evaluation uses the same verifier, batching policy, timing path, and dynamic-k baselines as the primary experiments.

Across the Medusa-drafter results in Tabs.16 and 17, CurveK has the highest average throughput among the dynamic-k methods in the single-benchmark sampling results and in both mixed-arrival panels. This suggests that the accepted-depth throughput-curve mechanism also carries over to a Medusa-style drafter.

A.11 Pareto Method Comparison

To assess throughput and latency together, we compare throughput and estimated time per output token (TPOT) across batch sizes in Figs.8 and 9. CurveK achieves higher throughput than fixed-length speculative decoding at most evaluated batch sizes under both greedy decoding and sampling. The AlpacaEval panels use the same 805-prompt workload as the throughput tables.

Table 16: **Medusa-drafter dynamic-k throughput on the single-benchmark sampling results.** Tokens/sec under sampling ($T = 1.0$). All settings use $B = 32$. The Avg. row averages the three workloads shown. Blue marks the highest value in each row; peach marks the second highest.

Setting	OTF	Turbo	Bandit	Nightjar	CurveK
GSM8K	1798.50$_{\pm 86.5}$	1741.04$_{\pm 121.3}$	1724.31$_{\pm 100.1}$	1587.38$_{\pm 96.2}$	**1944.67**$_{\pm 93.9}$
HumanEval	1877.48$_{\pm 72.7}$	1833.57$_{\pm 64.7}$	1913.66$_{\pm 25.5}$	1852.09$_{\pm 32.4}$	**1977.12**$_{\pm 74.7}$
SpecBench-Mix	1467.02$_{\pm 73.6}$	1562.17$_{\pm 86.0}$	1592.13$_{\pm 49.9}$	1519.57$_{\pm 71.8}$	**1608.94**$_{\pm 20.2}$
Avg.	1714.33	1712.26	1743.37	1653.01	**1843.58**

Table 17: **Medusa-drafter dynamic-k throughput on the mixed-arrival serving region.** Tokens/sec on the full mixed-arrival trace. Each panel includes the same dynamic-baseline set, and the Avg. row averages the four batch sizes shown.

Greedy decoding ($T = 0$)

B	OTF	Turbo	Bandit	Nightjar	CurveK
16	1261.40	**1264.15**	1206.76	1235.24	1233.80
32	1955.84	1963.48	1973.74	1906.84	**1986.98**
64	2683.10	2632.50	2677.10	2672.17	**2686.31**
128	3033.77	3020.47	3106.13	3194.13	**3234.75**
Avg.	2233.53	2220.15	2240.93	2252.10	**2285.46**

Sampling ($T = 1.0$)

B	OTF	Turbo	Bandit	Nightjar	CurveK
16	1053.40	1049.10	1046.31	1012.05	**1059.00**
32	1673.83	**1687.71**	1666.49	1665.85	1686.21
64	2310.34	2290.43	2290.80	2289.11	**2385.88**
128	2737.85	2726.78	2764.01	2772.68	**2790.99**
Avg.	1943.86	1938.51	1941.90	1934.92	**1980.52**

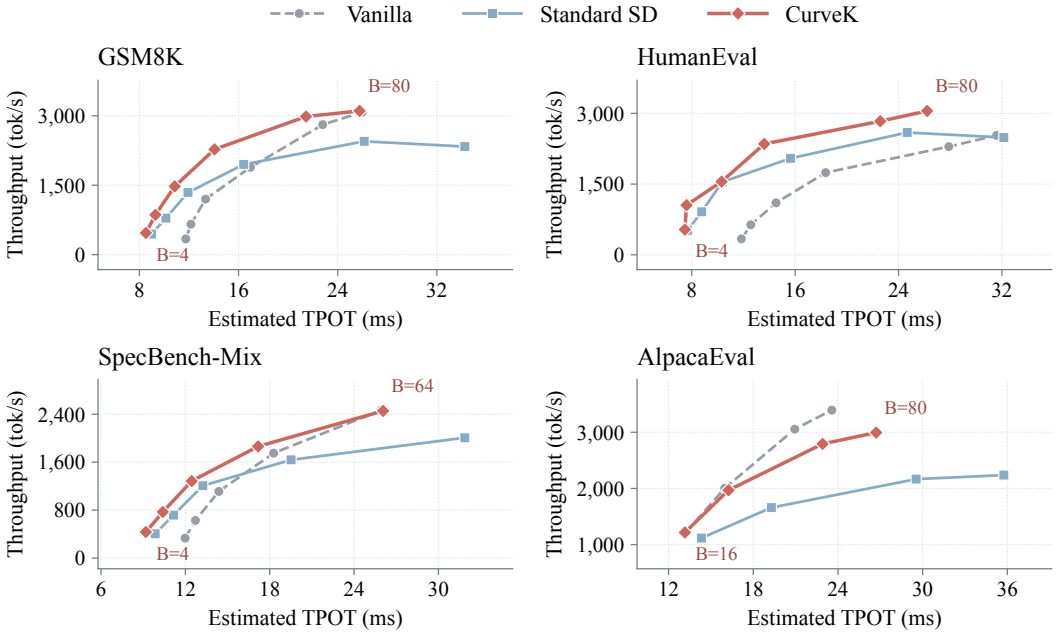

Figure 9: Throughput–TPOT Pareto frontier under sampling ($T = 1.0$). The layout matches Fig.8.

A.12 Draft-Length Decision Overhead

Tab.18 reports the per-call draft-length decision cost used to compare the decision logic with target verification.

Table 18: Draft-length decision overhead. Values are mean ± standard deviation in μs.

Method	$B = 8$	$B = 32$	$B = 128$
CurveK	48.9$_{\pm 1.5}$	53.7$_{\pm 0.9}$	75.8$_{\pm 1.3}$
BanditSpec	8.2$_{\pm 0.3}$	8.0$_{\pm 0.4}$	8.0$_{\pm 0.3}$
TurboSpec	3.3$_{\pm 0.2}$	3.3$_{\pm 0.1}$	3.3$_{\pm 0.3}$
OTF	4.2$_{\pm 0.1}$	4.1$_{\pm 0.2}$	4.2$_{\pm 0.3}$
Nightjar	2.0$_{\pm 0.4}$	2.0$_{\pm 0.4}$	2.0$_{\pm 0.4}$

Tab.18 shows that CurveK's decision cost is small relative to target verification. Its main computation is batch reduction and argmax, with complexity $O(BK)$. At $B = 128$, one draft-length decision takes about 76 μs. Scalar baselines are $O(K)$ and independent of B, taking about 2–8 μs. At $B = 32$, the measured 53.7 μs decision overhead is approximately 0.18% of a 30 ms A100 decode step.

A.13 Sensitivity to the Maximum Draft Length

Tab.19 evaluates sensitivity to the maximum candidate draft length $K \in \{3, 4, 5, 6, 7\}$. We set $K = 7$ following the speculation depth used in EAGLE-3 drafter training. This convention fixes the candidate range before workload-level evaluation. The default achieves the highest reported throughput on all four workloads. Overall, the standard EAGLE-3 candidate range is a strong default for CurveK, and throughput remains stable across nearby caps.

Table 19: **Maximum-draft-length sensitivity.** Throughput tokens/sec for CurveK with the LLaMA-3.1-8B target and EAGLE-3 drafter. All settings use $B = 32$, greedy decoding, and $c_r = 0.08$.

Workload	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$	Best
GSM8K	2531.6	2698.9	2552.6	2736.6	**2821.5**$_{\pm 122.6}$	7
HumanEval	2541.2	2549.8	2563.2	2577.3	**2656.0**$_{\pm 135.5}$	7
SpecBench-Mix	2159.7	2205.6	2093.7	2190.1	**2275.1**$_{\pm 11.8}$	7
AlpacaEval	2509.7	2534.7	2462.8	2430.9	**2657.2**$_{\pm 30.3}$	7

We further expand the candidate range to $K = 8, \ldots, 12$, keeping the controller hyperparameters fixed. On GSM8K, CurveK achieves the highest mean throughput among all compared dynamic methods at every tested maximum draft length (Tab.20). These experiments use LLaMA-3.1-8B with EAGLE-3, $B = 32$, greedy decoding, and $c_r = 0.08$.

Table 20: Throughput with larger maximum draft lengths on GSM8K. Controller hyperparameters are held fixed across K. Values are mean±std in tok/s.

K	OTF	TurboSpec	BanditSpec	Nightjar	CurveK
8	2668.2$_{\pm 46.1}$	2614.1$_{\pm 66.9}$	2629.0$_{\pm 277.1}$	2360.6$_{\pm 212.1}$	**2815.4**$_{\pm 79.2}$
9	2652.3$_{\pm 44.1}$	2624.0$_{\pm 45.5}$	2805.0$_{\pm 102.7}$	2340.1$_{\pm 269.7}$	**2844.5**$_{\pm 52.1}$
10	2643.3$_{\pm 44.0}$	2617.9$_{\pm 67.2}$	2636.8$_{\pm 258.9}$	2382.0$_{\pm 379.4}$	**2826.8**$_{\pm 47.8}$
11	2649.1$_{\pm 54.7}$	2621.1$_{\pm 61.1}$	2625.2$_{\pm 214.2}$	2366.6$_{\pm 317.0}$	**2835.7**$_{\pm 48.6}$
12	2653.9$_{\pm 60.7}$	2604.9$_{\pm 70.8}$	2541.0$_{\pm 206.6}$	2571.3$_{\pm 191.3}$	**2786.6**$_{\pm 67.6}$

B Proofs and Details for the Acceptance-Probability Sequence and Throughput Curve

B.1 Proof of Prop. 1

For adjacent draft lengths,

$$R(k) - R(k-1) = \frac{p_k - c_r R(k-1)}{1 + kc_r}.$$

Rewrite $R(k)$ as

$$R(k) = \frac{(1 + (k-1)c_r)R(k-1) + c_r(p_k/c_r)}{1 + kc_r}.$$

Thus $R(k)$ is a convex combination of $R(k-1)$ and p_k/c_r:

$$\min\big(R(k-1), p_k/c_r\big) \le R(k) \le \max\big(R(k-1), p_k/c_r\big). \tag{$*$}$$

If $R(k)$ is nondecreasing for every k, the result is immediate. Otherwise, let k_0 be the first step where $R(k_0) < R(k_0 - 1)$. By $(*)$, $p_{k_0}/c_r \le R(k_0) < R(k_0 - 1)$; equality $p_{k_0}/c_r = R(k_0)$ would force $R(k_0) = R(k_0 - 1)$ in the convex-combination representation, so in fact $p_{k_0}/c_r < R(k_0)$. Since the sequence p_k is nonincreasing, $p_{k_0+1}/c_r \le p_{k_0}/c_r < R(k_0)$, and App.B.1 gives $R(k_0 + 1) < R(k_0)$. Repeating the same argument inductively shows that once the throughput decreases, it decreases strictly thereafter. Hence the sequence is unimodal, and the last nondecreasing step is a maximizer.

B.2 Details on Summary Collisions

This appendix gives the collision calculation used by Prop.2 and two concrete construction families. Let $\mathcal{P}_K = \{p \in [0,1]^K : 1 \ge p_1 \ge \cdots \ge p_K \ge 0\}$ denote the set of valid acceptance-probability sequences.

Collision gap. For two regimes p and q, the throughput gap is the difference between the oracle value that chooses the best draft length separately for each regime and the best value achievable by a method that must use the same draft-length distribution for both regimes. If $S(p) = S(q) = z$, an S-only method uses one draft-length distribution $\pi(\cdot \mid z)$ for both regimes. Under an equal mixture over p and q, its expected value is

$$\sum_{k=0}^{K} \pi(k \mid z) \frac{R_p(k) + R_q(k)}{2} \le \max_{k \in \{0,\dots,K\}} \frac{R_p(k) + R_q(k)}{2}.$$

This upper bound is attainable by choosing the best common draft length. A regime-aware oracle chooses the best k separately after observing the regime, giving

$$\frac{\max_k R_p(k) + \max_k R_q(k)}{2}.$$

This difference is therefore the unavoidable throughput gap from merging the two regimes. If the maximizer sets of R_p and R_q are disjoint, every common draft length is suboptimal for at least one regime, so the gap is strictly positive.

p_1-only collision. Assume $c_r < a \le 1$. Define

$$p^{\text{front}} = (a, 0, \dots, 0), \qquad p^{\text{tail}} = (a, a, \dots, a).$$

Both sequences are valid elements of $\mathcal{P}_K$, and both have the same next-token acceptance probability $p_1 = a$. For the front-loaded sequence,

$$R_{p^{\text{front}}}(k) = \frac{1 + a}{1 + kc_r}, \qquad k \ge 1,$$

so the maximizer is $k = 1$. For the tail-persistent sequence,

$$R_{p^{\text{tail}}}(k) = \frac{1 + ka}{1 + kc_r},$$

which increases with k when $a > c_r$, so the maximizer is K. Under an equal mixture, every p_1-only method incurs a throughput gap of at least

$$\Delta_{p_1} = \frac{1}{2}\left(\frac{1+a}{1+c_r} + \frac{1+Ka}{1+Kc_r}\right) - \max_{k\in\{1,\dots,K\}} \frac{2+a(1+k)}{2(1+kc_r)}.$$

For $K = 7$, $c_r = 0.08$, and $a = 0.5$, the oracle value is 2.1368, the best same-action value is 1.9231, and the oracle-relative gap is 10.0%. The gap is positive on an open range of a values. For the same K and c_r, maximizing this expression over $a \in (c_r, 1]$ gives a 13.3% relative gap at $a \approx 0.174$.

Mean-depth-only collision. Choose an integer $m \in \{3, \dots, K\}$ and

$$\max\left\{mc_r, \frac{1+3c_r}{1+c_r}\right\} < \ell \le 2.$$

Define

$$p^{\text{front}} = (1, \ell-1, 0, \dots, 0), \qquad p^{\text{spread}} = (\underbrace{\ell/m, \dots, \ell/m}_{m \text{ positions}}, 0, \dots, 0).$$

Both sequences have the same maximum-depth mean $L(K) = \ell$. For p^{front}, the condition $\ell > (1+3c_r)/(1+c_r)$ gives $R(2) > R(1)$, and the throughput decreases after $k = 2$; hence $k = 2$ is optimal. For p^{spread}, the condition $\ell > mc_r$ makes

$$R_{p^{\text{spread}}}(k) = \frac{1+k\ell/m}{1+kc_r}, \qquad k \le m,$$

increase with k; for $k > m$, the numerator remains $1+\ell$ while the denominator increases. Thus m is optimal. Under an equal mixture, every $L(K)$-only method incurs a throughput gap of at least

$$\Delta_L = \frac{1}{2}\left(\frac{1+\ell}{1+2c_r} + \frac{1+\ell}{1+mc_r}\right) - \max_{k\in\{1,\dots,K\}} \frac{R_{p^{\text{front}}}(k) + R_{p^{\text{spread}}}(k)}{2}.$$

For $K = m = 7$, $c_r = 0.08$, and $\ell = 1.5$, the oracle value is 1.8789, the best same-action value is 1.6934, and the oracle-relative gap is approximately 9.9%. This value is one representative point in the feasible interval $\ell \in (\max\{mc_r, (1+3c_r)/(1+c_r)\}, 2]$; throughout this interval the two regimes have the same mean depth and different optimal draft lengths.

C Further Discussion of Related Work

Online feedback and length selection. BanditSpec (Hou et al., 2025) and Nightjar (Li et al., 2025a) learn from the performance of executed draft lengths, maintaining rewards or losses for the candidate arms. TurboSpec (Liu et al., 2024b) and OTF (Liu et al., 2025a) instead use recent verification outcomes to estimate acceptance and combine these estimates with execution costs. CurveK also adapts from verification feedback, but maintains accepted-depth states for the active requests and uses them to construct a throughput curve over candidate lengths. Our comparison therefore concerns how runtime feedback is represented and used for the next decision. The baseline decision rules and evaluation configurations are described in App.H.

Acceptance prediction and verification scheduling. Acceptance can also be predicted before target verification. DISCO (Mamou et al., 2024) and AdaEDL (Agrawal et al., 2024) use draft-model statistics to decide when to stop drafting, while PACER (Zhang et al., 2026b) uses a trained layer to pre-verify draft blocks. SpecDec++ (Huang et al., 2025) and Speculative Verification (Kim et al., 2025) obtain predictive signals from an acceptance head and a companion model, respectively.

AdaSpec (Huang et al., 2026) and DSpark (Cheng et al., 2026) are particularly relevant because they connect predicted acceptance with throughput-based decisions. AdaSpec updates confidence-based acceptance estimates during drafting. DSpark uses a trained confidence head and hardware throughput profiles to select a verification prefix for each request. CurveK uses accepted depths from earlier verification steps to select the next shared draft length. The confidence-based ablation in App.A.5 compares draft-confidence and accepted-depth estimates under the same cost model and length-selection rule, isolating how the source of acceptance information affects throughput.

Draft generation and serving optimizations. A separate line of work changes how draft candidates are generated (Miao et al., 2024). Medusa (Cai et al., 2024) uses multiple prediction heads, DFlash (Chen et al., 2026) generates draft blocks with a lightweight diffusion model, and Sequoia (Chen et al., 2024) and EAGLE-2 (Li et al., 2024) organize candidates into trees. Kangaroo (Liu et al., 2024a) and LayerSkip (Elhoushi et al., 2024) explore self-speculation through early exiting.

Serving methods address how speculative work is allocated and executed (Qian et al., 2024; Zhang et al., 2026a): HeteroSpec (Liu et al., 2025b) assigns request-specific lengths, TETRIS (Wu et al., 2025) selects tokens for verification, and AdaServe (Li et al., 2026) accounts for latency constraints. DSDE (Yang et al., 2025b) mitigates per-sequence stragglers, while PEARL (Liu et al., 2025c) overlaps drafting and verification. These designs change the available candidates or their execution costs. CurveK studies the shared draft-length decision within a given serving configuration.

D Additional Theory Details for CurveK

This appendix gives the proof details for the estimator-error decomposition in Sec.5.4.

D.1 Estimator-Error Decomposition

The decomposition separates three effects: finite-sample error, Jensen gap, and online mismatch.

Fix a decision step t, after the verifier feedback used to update $\mu_{t,i}$ has been incorporated. Let $\mathcal{B}_t$ be the active batch and $B_t = |\mathcal{B}_t|$. For this reference analysis, fix the batch and its current acceptance distributions; all expectations use this same reference setting. For proof bookkeeping, cap the latent accepted depth at the maximum candidate length: $X_{t,i} := \min(Z_{t,i}, K)$, and write $m_{t,i} = \mathbb{E}[X_{t,i}]$. This cap leaves all candidate-depth terms unchanged, since every candidate satisfies $k \le K$ and hence $\min(k, X_{t,i}) = \min(k, Z_{t,i})$. The target curve over the active batch is

$$R_t(k) = \frac{1 + \frac{1}{B_t}\sum_{i\in\mathcal{B}_t} \mathbb{E}\min(k, X_{t,i})}{1 + kc_r}, \qquad k = 0, \dots, K.$$

The decomposition stated in Sec.5.4 is

$$\max_k |\widehat{R}_t(k) - R_t(k)| \le \Delta_t = \varepsilon_t(\delta) + J_t + \Omega_t$$

with probability at least $1 - \delta$ under the reference concentration condition below.

Finite-sample error. Assume that the actual estimated accepted-token benefit satisfies, for every candidate k and $u > 0$,

$$\Pr\left(\left|\widehat{L}_t(k) - \mathbb{E}\widehat{L}_t(k)\right| > u\right) \le 2\exp\left(-\frac{2N_t^{\text{eff}}u^2}{K^2}\right).$$

Here N_t^{eff} specifies the strength of the assumed concentration. This condition applies to the actual estimator, which smooths each request's depth before truncating it at a candidate length. Define

$$\varepsilon_t(\delta) = K\sqrt{\frac{\log(2(K+1)/\delta)}{2N_t^{\text{eff}}}}.$$

Approximation terms. The Jensen-gap term is

$$J_t = \max_k \frac{1}{1+kc_r}\frac{1}{B_t}\sum_{i\in\mathcal{B}_t} b_{t,i}(k), \quad b_{t,i}(k) = \min(k, m_{t,i}) - \mathbb{E}\min(k, X_{t,i}).$$

The online mismatch term is

$$\Omega_t = \max_k \frac{1}{1+kc_r}\frac{1}{B_t}\sum_{i\in\mathcal{B}_t} \left|\mathbb{E}\min(k, \mu_{t,i}) - \min(k, m_{t,i})\right|.$$

This term measures the remaining mismatch between the maintained depth states and the current acceptance distributions, including the effects of smoothing, censored feedback, and changing requests. The bound does not assume that this mismatch vanishes over time.

The Jensen term has a useful exact form:

$$b_{t,i}(k) = \frac{1}{2}\left(\mathbb{E}|X_{t,i} - k| - |m_{t,i} - k|\right) \geq 0.$$

If $\sigma_{t,i}^2 = \mathrm{Var}(X_{t,i})$, then

$$b_{t,i}(k) \leq \min\left\{ \frac{\sqrt{(m_{t,i}-k)^2 + \sigma_{t,i}^2} - |m_{t,i} - k|}{2},\ \min(m_{t,i}, k)\left(1 - \frac{\max(m_{t,i}, k)}{K}\right) \right\}.$$

Proof. Define the mean-depth curve

$$R_t^m(k) = \frac{1 + \frac{1}{B_t}\sum_{i\in\mathcal{B}_t} \min(k, m_{t,i})}{1 + kc_r}.$$

For every candidate k,

$$\begin{aligned}|\widehat{R}_t(k) - R_t(k)| \leq\ & |\widehat{R}_t(k) - \mathbb{E}\widehat{R}_t(k)| \\ & + |\mathbb{E}\widehat{R}_t(k) - R_t^m(k)| + |R_t^m(k) - R_t(k)|.\end{aligned}$$

We bound these three terms separately.

Jensen gap. The last term is exactly the Jensen gap averaged over requests:

$$R_t^m(k) - R_t(k) = \frac{1}{1 + kc_r}\frac{1}{B_t}\sum_{i\in\mathcal{B}_t} b_{t,i}(k).$$

Thus $\max_k |R_t^m(k) - R_t(k)| \leq J_t$. The identity for $b_{t,i}(k)$ follows from

$$\min(k, x) = \frac{x + k - |x - k|}{2}.$$

Substituting $x = m_{t,i}$ and subtracting the expectation of the same identity with $x = X_{t,i}$ gives the exact form above. Jensen's inequality gives $b_{t,i}(k) \geq 0$. Cauchy–Schwarz gives the variance-sensitive bound. For the support-only bound, the chord inequality $\min(k, x) \geq kx/K$ for $x \in [0, K]$ implies $\mathbb{E}\min(k, X_{t,i}) \geq k m_{t,i}/K$, yielding the two cases $m_{t,i} \leq k$ and $m_{t,i} \geq k$.

$$\begin{aligned}\frac{1}{B_t}\sum_{i\in\mathcal{B}_t} \mathbb{E}[\min(k, X_{t,i})] &\leq \frac{1}{B_t}\sum_{i\in\mathcal{B}_t} \min(k, m_{t,i}) \\ &\leq \min\left(k, \frac{1}{B_t}\sum_{i\in\mathcal{B}_t} m_{t,i}\right).\end{aligned}$$

With the reference means $m_{t,i}$, the first inequality captures compression within each request, while the second captures the additional effect of averaging requests before truncation. Request-level aggregation avoids the second gap in this reference setting. The online estimator's fluctuations and mismatch are accounted for separately in the error decomposition.

Online mismatch. For the fixed active batch, linearity of expectation gives

$$\mathbb{E}\widehat{R}_t(k) = \frac{1 + B_t^{-1}\sum_{i\in\mathcal{B}_t} \mathbb{E}\min(k, \mu_{t,i})}{1 + kc_r}.$$

The triangle inequality therefore bounds the second term by Ω_t. Together with the Jensen term,

$$\max_k |\mathbb{E}\widehat{R}_t(k) - R_t(k)| \leq J_t + \Omega_t.$$

Finite-sample error. The reference concentration condition and a union bound over $k = 0, \ldots, K$ give

$$\max_k |\widehat{L}_t(k) - \mathbb{E}\widehat{L}_t(k)| \leq \varepsilon_t(\delta)$$

with probability at least $1-\delta$. Since $1+kc_r \geq 1$, the same term bounds the stochastic fluctuation of the throughput curve:

$$\max_k |\widehat{R}_t(k) - \mathbb{E}\widehat{R}_t(k)| \leq \varepsilon_t(\delta).$$

Combining these terms proves the uniform curve-error bound. To obtain the selection-loss bound, use $\widehat{R}_t(\widehat{k}_t) \geq \widehat{R}_t(k_t^*)$:

$$\begin{aligned} R_t(k_t^*) - R_t(\widehat{k}_t) &\leq |R_t(k_t^*) - \widehat{R}_t(k_t^*)| + |\widehat{R}_t(\widehat{k}_t) - R_t(\widehat{k}_t)| \\ &\leq 2\max_k |\widehat{R}_t(k) - R_t(k)|. \end{aligned}$$

Thus selecting the peak of the estimated curve loses at most twice the maximum curve error.

The reference bound isolates the finite-sample error term under a concentration model. The full-depth diagnostics in Sec.6.4 examine how the estimated curves guide length selection, while the serving experiments measure end-to-end throughput.

E CurveK Algorithm and Implementation Details

Alg. 1 CurveK decision loop

Require: Active batch $\mathcal{B}_t$, accepted-depth estimates $\{\mu_i\}$, previous draft length k_{t-1}, verifier feedback $\{Y_{t,i}\}$, maximum draft length K, cost ratio c_r, smoothing coefficient β, default draft length k_{default}, probe interval P, probe length k_{probe}, consecutive-zero counter d (initially 0)

if no verifier feedback has been observed **then**
  return k_{default}
end if
for all $i \in \mathcal{B}_t$ with μ_i undefined **do**
  $\mu_i \leftarrow k_{\text{default}}$
end for
if $k_{t-1} > 0$ **then**
  for all $i \in \mathcal{B}_t$ with verifier feedback **do**
    Apply the one-step continuation update

$$\widetilde{Y}_{t,i} = \begin{cases} Y_{t,i}, & Y_{t,i} < k_{t-1} \text{ or } k_{t-1} = K, \\ k_{t-1}+1, & Y_{t,i} = k_{t-1} < K. \end{cases}$$

    $\mu_i \leftarrow (1-\beta)\mu_i + \beta\widetilde{Y}_{t,i}$
  end for
end if
for $k = 0, 1, \ldots, K$ **do**
  $\widehat{L}_t(k) \leftarrow |\mathcal{B}_t|^{-1} \sum_{i\in\mathcal{B}_t} \min(k, \mu_i)$
  $\widehat{R}_t(k) \leftarrow (1+\widehat{L}_t(k))/(1+kc_r)$
end for
$\widehat{k}_t \leftarrow \arg\max_{0\leq k\leq K} \widehat{R}_t(k)$
$k_t \leftarrow \widehat{k}_t$
if $\widehat{k}_t = 0$ **then**
  $d \leftarrow d+1$
  if $d \geq P$ **then**
    $k_t \leftarrow k_{\text{probe}}$, $d \leftarrow 0$
  end if
else
  $d \leftarrow 0$
end if
Delete estimates for requests that leave the current batch
return k_t

Alg. 1 gives the draft-length selection algorithm used throughout the evaluation. The implementation details are as follows. First, newly arrived requests are initialized at the configured default depth,

$k_{\text{default}} = 7$ in our experiments, and then updated once verifier feedback arrives. Second, the one-step continuation update is applied only when a request fully accepts a nonmaximum draft, because the previous chosen length k_{t-1} only reveals that the observed prefix was accepted. Third, accepted-depth estimates are stored by the method: they are created when a request enters the batch and deleted as soon as the request leaves. CurveK therefore stores one value per request currently in the batch and performs $O(|\mathcal{B}_t|K)$ arithmetic per selection.

If the curve repeatedly recommends zero, the serving controller periodically executes a short draft to obtain fresh acceptance feedback. After $P = 10$ consecutive zero recommendations, it drafts $k_{\text{probe}} = 1$ token and resets the counter. The resulting feedback updates the same depth estimates, allowing speculation to resume when acceptance improves. The curve-error bound concerns the maximizing recommendation $\widehat{k}_t$; probe steps use $k_t = k_{\text{probe}}$ to refresh feedback.

F Additional Deployment Details

F.1 Scope and Portability

CurveK selects a shared draft length from a configured candidate set in batched serving systems that return each request's accepted prefix length after verification. It reuses this feedback without an auxiliary predictor or offline training stage. The one-step continuation update accounts for the possible benefit of extending a fully accepted draft.

Cost calibration. The cost ratio c_r is profiled per serving configuration. When the hardware, target–drafter pair, or batch-size range changes, the calibration is repeated using the procedure in App.G.

Operating conditions. Short requests provide less feedback, while abrupt changes in acceptance can make earlier observations less informative. Our next-step predictions have lower mean absolute error than the batch-mean alternative (App.A.6), and the mixed-workload experiments show that selected lengths change with workload composition (App.A.9).

Baseline details. We evaluate the dynamic controllers in Tab.1 within the same vLLM batched speculative-decoding path. Model execution, verification, batching, and throughput measurement are shared. We follow the baseline decision rules in the original papers, as summarized in App.H.

Possible extensions. To account explicitly for estimation uncertainty, a future controller could attach uncertainty estimates to the throughput curve and compare candidate lengths using lower confidence bounds. When the candidates are difficult to distinguish, it could retain the previous length; when no speculative length offers a sufficiently reliable benefit over standard decoding, it could temporarily disable speculation and periodically probe to refresh feedback.

Another extension is to choose a separate draft length for each request. This would allow longer drafts for requests with higher acceptance, but efficient execution would require variable-length batching, padding, or regrouping requests. The throughput objective would also need to account for the resulting batch execution costs. We leave these extensions to future work.

G Use of c_r in the Implementation

The ratio c_r expresses the cost of one draft token in units of one target verification pass (Yan et al., 2025). It determines the additional cost kc_r used to compare candidate draft lengths.

When moving to a new GPU or a new target/drafter pair, c_r is determined in advance by the same rule. We first fix the dtype, tensor-parallel setting, GPU memory configuration, batch-size range, and context-length range used by the serving setup; if these settings change, we profile again. We then run a short calibration through the same vLLM speculative-decoding path: build KV cache from warmup prompts, discard warmup steps, and record CUDA-event timings over a stable window. For each calibration step t, let $T_{\text{verify},t}$ be the time of one target verify pass and $T_{\text{draft},t}$ be the total time

for the drafter to generate k_{cal} tokens. The step gives

$$r_t = \frac{T_{\text{draft},t}/k_{\text{cal}}}{T_{\text{verify},t}}.$$

This is the cost of one draft token in units of one target verify pass. We take the median of r_t over a stable window at representative batch sizes and context lengths. If a deployment spends most of its time in several distinct batch-size ranges, the same procedure can be repeated per range and the value for the dominant serving range can be used. CurveK receives c_r as a configuration value. In the main A100 setup, this profiling protocol gives a ratio close to 0.08, and we use the rounded value $c_r = 0.08$.

At runtime, CurveK estimates $L(k)$ from the current batch's accepted-depth estimates and then computes

$$\widehat{R}(k) = \frac{1 + \widehat{L}(k)}{1 + kc_r}, \qquad \widehat{k} = \arg\max_k \widehat{R}(k).$$

The profiled ratio c_r sets the cost of extending the draft, while verifier feedback updates the expected accepted-token benefit.

G.1 Sensitivity to Cost-Ratio Variation

The selected draft length changes only when the profiled cost crosses a throughput-curve margin. For a given acceptance curve $L(k)$, adjacent draft lengths switch order when $p_k = c_r R_{c_r}(k-1)$; equivalently, the sign of $p_k - c_r R_{c_r}(k-1)$ determines whether $R_{c_r}(k)$ exceeds $R_{c_r}(k-1)$. Fig.10 evaluates these margins using the verifier-revealed acceptance curves from the oracle evaluations. For each workload, we vary c_r around the nominal 0.08 and measure the throughput obtained when k is selected with the perturbed cost value. In the evaluated traces, selecting draft lengths with a cost ratio perturbed by up to 25% retains at least 99.1% of the score obtained with the nominal ratio.

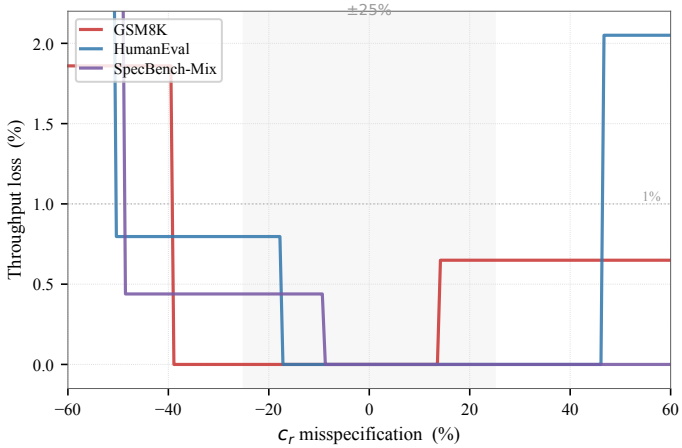

Figure 10: **Throughput under c_r variation.** For each workload, we compute the mean verifier-revealed acceptance curve from the B=80 oracle evaluations, then measure throughput when k is selected using a perturbed c_r relative to the nominal 0.08. The gray band marks $\pm 25\%$; the dotted line marks a 1% change from the nominal-cost selection.

H Baseline Controllers and Evaluation Protocol

This section documents the implementations of BanditSpec (Hou et al., 2025), TurboSpec (Liu et al., 2024b), OTF (Liu et al., 2025a), and Nightjar (Li et al., 2025a) evaluated in the common vLLM V1 runtime. We tune each controller for the evaluation setting and compare the selected configurations within the same vLLM serving pipeline. For the 70B experiments, each method uses one configuration per hardware platform, held fixed across all four workloads. The following subsections describe how each method uses runtime feedback to choose the next draft length.

Common setup. All four baselines and CurveK choose from $\gamma \in \{0, 1, \ldots, K\}$, with $K = 7$ unless a larger candidate range is specified. At each step, before choosing the next draft length, the method receives the previous verifier feedback: observed per-request accepted depths $\{Y_i\}$, the previous k, the previous step wall-clock time, and the current batch size. Each method first updates its stored statistics from completed verification feedback and then chooses the next draft length. Here $\gamma = 0$ denotes vanilla decoding, and $\gamma \geq 1$ means that the drafter proposes γ draft tokens before target verification. Each controller summarizes batched feedback using the reward, loss, or acceptance statistics specified in its subsection below.

H.1 BANDITSPEC / UCBSPEC

BanditSpec (Hou et al., 2025) treats each candidate draft length as a bandit arm. We use UCBSpec, following the speculation-length serving experiment in the original paper. Its reward is measured throughput: the number of generated tokens divided by elapsed time. After trying each candidate once, UCBSpec chooses the length with the largest upper confidence bound,

$$\gamma_{t+1} = \arg\max_{\gamma \in \{0,\ldots,K\}} \left(\widehat{r}_{\gamma,t} + b_{\gamma,t}\right),$$

where $\widehat{r}_{\gamma,t}$ is the empirical mean reward and $b_{\gamma,t}$ is the UCBSpec confidence bonus defined by Hou et al. (2025). After execution, only the selected arm's reward statistics and observation count are updated. The $\gamma = 0$ arm corresponds to vanilla decoding.

H.2 TURBOSPEC

TurboSpec writes batched serving configuration selection as a throughput maximization problem:

$$G(k) = \frac{\text{accept_len}(k)}{T_{\text{batch}}(k)}, \qquad k^* = \arg\max_k G(k).$$

The numerator $\text{accept_len}(k) = (1 - \alpha^{k+1})/(1 - \alpha)$ comes from the truncated-geometric assumption. The denominator $T_{\text{batch}}(k) = T_{\text{target}} + kT_{\text{draft}}$ is obtained from a forward-time regression in the original method. The algorithm maintains a global α through the paper's global acceptance update routine, estimated online as a weighted average of recent acceptance rates. When $k^* = 0$ wins, the original algorithm disables speculation for that batch size and re-evaluates speculation after a fixed reset interval.

The configuration used in our experiments is as follows. The α estimator uses a sliding window of the most recent 200 steps with a right-censored geometric maximum-likelihood estimate, accumulating accepted-depth totals and termination-count totals per step. We use the runtime per-arm step-time sliding mean with a window of 50 observations for $T_{\text{batch}}(k)$. Unobserved arms are filled by a linear extrapolation $a + bk$. We set the reset interval to 10 steps and visit all eight arms once in round-robin order to initialize $T(k)$.

H.3 OTF

With γ denoting the candidate draft length, Liu et al. (2025a) give the closed-form selection formula

$$\gamma^* = \arg\max_{\gamma} \frac{1 - \beta_{\text{OTF}}^{\gamma+1}}{(1 - \beta_{\text{OTF}})\,(\gamma a_q + b_p(\gamma))}.$$

The denominator $\gamma a_q + b_p(\gamma)$ is the total latency of the next speculation step under window γ: a_q is the draft single-token time, and $b_p(\gamma)$ is the target verification time, which may depend on γ. The paper estimates a_q and b_p from the most recent steps. The acceptance parameter β_{OTF} follows the paper's right-censored geometric MLE, using the ratio between accepted-depth totals and termination counts over the latest h verification steps, with the cap $\beta_{\text{OTF}} \leq \text{Acc}_{\max}$ to avoid degeneracy.

The configuration used in our experiments is as follows. We use $h = 5$, one of the $\{5, 6, 7\}$ sensitivity settings reported in the original paper, and $\text{Acc}_{\max} = 0.98$, the example value given in the method section. The joint latency estimate uses a runtime per-γ step-time sliding mean with a window of 50 observations. Unobserved γ's are filled by linear extrapolation $a + b\gamma$.

H.4 NIGHTJAR

We follow Nightjar's draft-length controller in arXiv:2512.22420v5 (Li et al., 2025a). It maintains a separate bandit state for each batch size and organizes decisions into blocks of length $H_B = 2^{j_B-1}$, divided into bins of duration $\sqrt{H_B}$. At the start of bin b_B, it explores with probability $1/b_B$ by sampling a draft length uniformly; otherwise, it selects the length that minimizes

$$\widetilde{\ell}_{B,\gamma} + s_t(\gamma),$$

where $\widetilde{\ell}_{B,\gamma}$ is the empirical mean latency per generated token. The switching penalty $s_t(\gamma)$ is the draft-model prefill cost divided by γ when restarting speculation from 0 to a positive length, and zero otherwise. The chosen length is held fixed throughout the bin. After each step, Nightjar updates the selected arm's latency statistics and advances the counters for the current batch size, following Algorithm 1 of the original paper.

I FULL SETUP FOR THROUGHPUT-CURVE ORACLE EVALUATION

The verifier-defined oracle in Fig.6 and Tab.21 is obtained from full-depth diagnostic runs with $K = 7$. Before each verification step, CurveK records its estimated curve and recommended length $\widehat{k}_t$ using feedback from preceding steps. The diagnostic execution then drafts and verifies all K tokens, independently of that recommendation, exposing the accepted prefix depth $z_{t,i}^{(K)} = \min(Z_{t,i}, K)$ for each observed request. This observation determines the accepted prefix for every candidate $k \le K$, including when all K tokens are accepted.

For the B_t requests observed at step t, we compute the empirical position-wise acceptance and oracle curve as

$$\begin{aligned}
p_{t,j}^{\text{oracle}} &= \frac{1}{B_t}\sum_{i=1}^{B_t} \mathbf{1}\{z_{t,i}^{(K)} \ge j\}, \\
L_t^{\text{oracle}}(k) &= \sum_{j=1}^{k} p_{t,j}^{\text{oracle}} = \frac{1}{B_t}\sum_{i=1}^{B_t} \min(k, z_{t,i}^{(K)}), \\
R_t^{\text{oracle}}(k) &= \frac{1 + L_t^{\text{oracle}}(k)}{1 + kc_r}, \qquad k = 0, \dots, K.
\end{aligned}$$

We evaluate the previously recorded recommendation on this newly observed curve, retaining steps with both an estimated curve and a full-depth observation. The retained throughput is $R_t^{\text{oracle}}(\widehat{k}_t)/\max_k R_t^{\text{oracle}}(k)$, averaged over steps within each setting. This measures decision quality on the full-depth diagnostic trajectory under the profiled cost model, with $c_r = 0.08$. The oracle curves compare candidate lengths on a shared full-depth verification trace, whereas the fixed-length experiments measure end-to-end throughput from separate runs at each length. The end-to-end throughput comparisons are reported in Tabs.1 and 2.

I.1 Full-Depth Verifier Comparison

Table 21: Per-setting throughput-curve oracle evaluations used by Fig.6. Each bar in the figure corresponds to one workload–batch pair in this table; its mean throughput loss is one minus the retained oracle-best throughput. Sequence agreement is the per-step correlation between CurveK's estimated $L(k)$ and the oracle $L(k)$ computed from verifier-curve traces.

Workload	B	Records	Sequence agreement	Within 3% best throughput	Retained oracle-best throughput
GSM8K	16	4220	0.955	60.6%	96.7%
GSM8K	32	2385	0.961	66.7%	96.8%
GSM8K	64	1189	0.967	81.4%	97.9%
GSM8K	80	999	0.966	84.3%	98.1%
HumanEval	16	1311	0.960	61.1%	96.5%
HumanEval	32	852	0.960	62.3%	95.6%
HumanEval	64	603	0.956	61.5%	95.5%
HumanEval	80	531	0.957	62.7%	94.7%
SpecBench-Mix	16	4316	0.952	59.9%	96.4%
SpecBench-Mix	32	2371	0.962	67.9%	96.7%
SpecBench-Mix	64	1314	0.966	72.8%	97.1%
SpecBench-Mix	80	1163	0.960	73.3%	96.8%

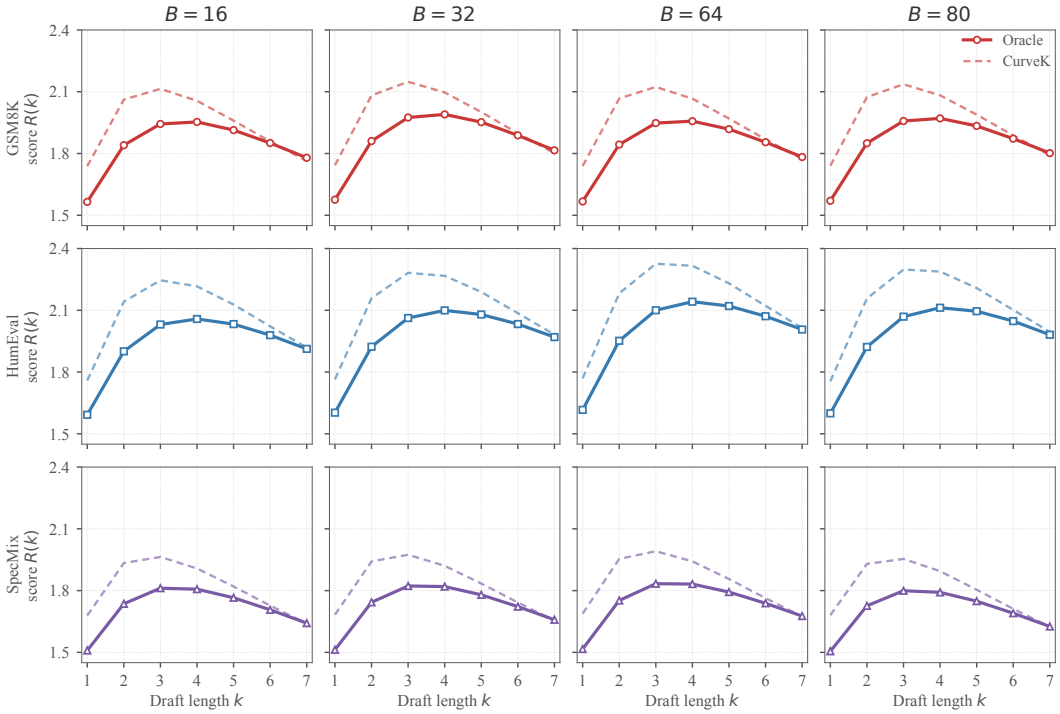

Figure 11: Per-setting throughput-curve oracle evaluations for all twelve workload–batch pairs in Tab.21. Solid lines are oracle throughput curves and dashed lines are CurveK's online estimates.

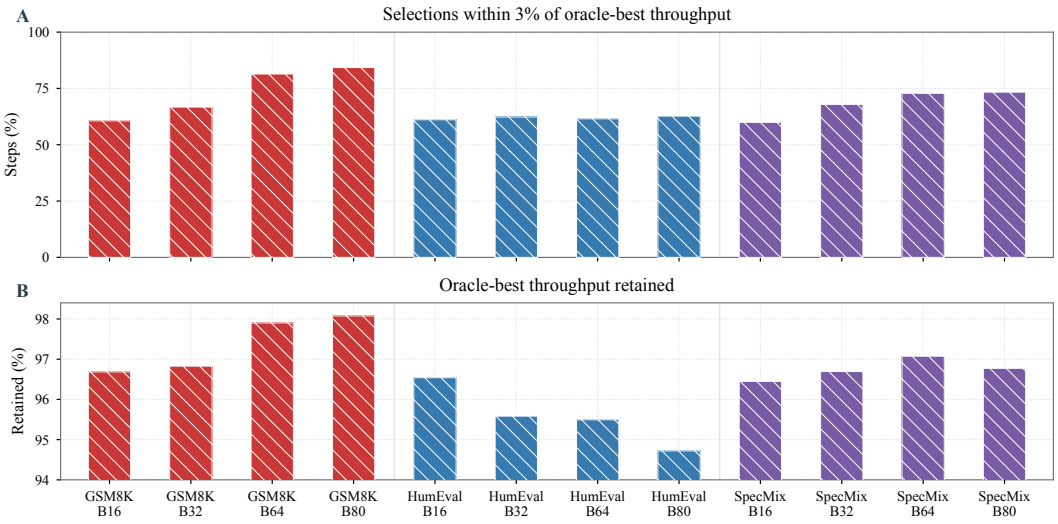

Figure 12: Per-setting selection metrics for the same twelve oracle-evaluation settings. A shows the fraction of steps within 3% of the oracle-best throughput. B shows the retained oracle-best throughput.

I.2 Draft-Length Selection with Separated Oracle Peaks

To assess how much draft-length selection matters on these traces, we first compare the best and worst throughputs among positive draft lengths $k = 1, \ldots, 7$. Their median relative gap is 20.51%. We then examine the gap between the best and second-best candidates to assess selection when deviations from the optimum are more costly. For each full-depth verifier record, let $R^{\text{oracle}}_{t,(1)}$ and $R^{\text{oracle}}_{t,(2)}$ be the largest and second-largest candidate throughput values. We group steps by the relative gap

$$g_t = \frac{R^{\text{oracle}}_{t,(1)} - R^{\text{oracle}}_{t,(2)}}{R^{\text{oracle}}_{t,(1)}}.$$

Tab.22 reports all steps and the subsets with $g_t \geq 1\%$ and $g_t \geq 2\%$. Each retained-throughput value is the mean ratio of the selected-length score to the oracle-best score within that subset.

The scalar rule uses the observed first-position acceptance rate to construct a geometric acceptance curve and select its throughput-maximizing length. Specifically, it sets $p^{\text{scalar}}_{t,j} = (p^{\text{oracle}}_{t,1})^j$ and uses the same cost model and candidate range as CurveK. This reference tests what a single acceptance rate captures about the full curve.

Table 22: Retained oracle throughput on steps with separated peaks. The gap compares the best and second-best candidate lengths. Step share is relative to all evaluated steps; retention is averaged within each subset.

Step subset	Step share	CurveK retention	Scalar-rule retention
All steps	100%	**96.68%**	96.25%
Gap $\geq 1\%$	51.62%	**95.62%**	94.80%
Gap $\geq 2\%$	23.56%	**93.74%**	92.50%

CurveK's advantage over the scalar rule is larger in the subsets with more separated peaks. These results complement the aggregate retained-throughput results by evaluating decisions when the top candidate lengths are less interchangeable.

I.3 Tracking Changes in the Preferred Length

We further examine steps where the oracle-best draft length changes from the preceding step in the same full-depth diagnostic trace. CurveK retains 96.21% of oracle throughput on these steps, compared with 97.68% when the best length remains unchanged. The lower retention around changes

is consistent with a delay in tracking new acceptance patterns, although this comparison does not isolate tracking lag from other sources of estimation error.

Fixed-offset diagnostic. We also test whether simply recommending a longer draft improves selection, by adding one or two tokens to each recorded recommendation and evaluating it on the oracle curve. Retained oracle throughput rises from 96.68% to 97.55% with a one-token shift, but falls to 95.84% with a two-token shift. Thus, a small offset helps on these traces, while a larger shift worsens selection.

I.4 Sub-batch Sensitivity of the Throughput-Curve Estimate

We test sensitivity to the number of request states by sampling b requests from each recorded $B = 64$ step and recomputing the throughput curve. The reference uses all request states from that step, with the cost model and stored depth estimates unchanged. Tab.23 reports curve discrepancies, draft-length changes, score loss, and the reference-curve margin over 200 repeated subsamples. When several lengths have similar throughput, the selected length may change with little loss.

Table 23: Sub-batch sensitivity of the throughput-curve estimate. Curve gaps, score loss, and Margin use the full-batch estimated curve as the reference.

Workload	b	Steps	Max L gap	Max R gap	k change	Score loss	Margin
GSM8K	4	1278	0.4058	0.3160	45.17%	0.03921	0.0585
GSM8K	8	1259	0.2746	0.2140	31.18%	0.01901	0.0586
GSM8K	16	1210	0.1817	0.1417	17.50%	0.00852	0.0590
GSM8K	32	1142	0.1048	0.0818	6.99%	0.00227	0.0591
GSM8K	48	1121	0.0599	0.0467	2.61%	0.00048	0.0590
HumanEval	4	577	0.3967	0.3072	39.68%	0.03444	0.0591
HumanEval	8	508	0.2582	0.2002	24.47%	0.01644	0.0598
HumanEval	16	381	0.1810	0.1404	11.35%	0.00632	0.0605
HumanEval	32	309	0.1052	0.0816	3.44%	0.00105	0.0620
HumanEval	48	276	0.0617	0.0479	1.17%	0.00019	0.0621
SpecBench-Mix	4	1396	0.4069	0.3184	48.37%	0.03802	0.0416
SpecBench-Mix	8	1290	0.2742	0.2150	37.29%	0.01700	0.0408
SpecBench-Mix	16	1180	0.1830	0.1436	26.30%	0.00818	0.0400
SpecBench-Mix	32	1127	0.1051	0.0825	15.51%	0.00314	0.0398
SpecBench-Mix	48	1084	0.0608	0.0477	8.84%	0.00102	0.0397

Across all three workloads, using more request states reduces both the curve discrepancy and the selection-score loss relative to the full-batch estimate.

I.5 Selected-Draft-Length Alignment

We compare CurveK's selected lengths with the best fixed draft lengths from the complete static-k grid under the same workload, batch size, and decoding temperature. Tab.24 summarizes this comparison; the per-step verifier-oracle comparison is reported in Tab.21.

Table 24: Selected-depth alignment on standard throughput evaluations. The final column counts settings where CurveK selects $k \leq 2$ for most steps while the hindsight best static depth is $k \geq 5$ in the same setting.

Setting group	Settings	Median low-k rate	Best static $k \leq 3$	Best static $k \geq 5$	High-depth cases
LMSYS-Chat-1M	6	55.2%	4	1	1
Mixed-arrival trace	2	14.7%	2	0	0

Tab.24 shows that four of the six dialogue settings and both mixed-arrival settings favor fixed lengths of $k \leq 3$. In the dialogue setting at $B = 8$, however, the best fixed length is $k = 5$, while CurveK selects $k \leq 2$ on 58.5% of recorded steps.