SELF-CONSISTENCY VIA MARGINAL SHARPENING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time compute has become a standard way to improve language-model reasoning without changing model weights, by allocating additional computation to sampling, search, or refinement before committing to an answer. A recent line of inference-time power-sampling methods pushes this idea by amplifying high-probability model outputs, typically doing so by sharpening complete generated sequences. We argue that this is the wrong object for reasoning models: a complete output entangles a latent reasoning trace with a final answer, even though many distinct traces may support the same answer. We introduce *Marginal Sharpening*, which instead sharpens the answer marginal induced by aggregating over latent reasoning traces. This gives an answer-level analogue of self-consistency, but as an inference target rather than a post-hoc voting rule. For integer sharpening strengths, the target has an exact multi-trace decomposition. We use this identity to derive a simple parallel auto-regressive approximation that samples multiple traces and generates one answer whose tokens are jointly supported across them, avoiding both complete-sequence search and exact string aggregation. Across math and code benchmarks, Marginal Sharpening improves over standard sampling and is particularly strong for code generation, where semantically equivalent solutions often differ in surface form. Compared with power sampling, Marginal Sharpening delivers stronger performance with substantially lower runtime, supporting answer-level sharpening as a more effective test-time objective.

1 INTRODUCTION

Modern language models increasingly solve difficult tasks by generating intermediate reasoning before producing a final response. This behavior can be elicited at inference time through chain-of-thought prompting [27, 16], improved through rationale bootstrapping and process supervision [31, 19], and further shaped by reinforcement learning on verifiable domains such as mathematics and code [10, 22]. Rather than mapping a prompt directly to a short answer, these models generate a reasoning trajectory: a derivation, program sketch, or proof outline, and then produce an answer conditioned on it. These intermediate steps can help decompose a problem, explore candidate solutions, and check their consistency before committing to a response. For a given prompt, the model can follow many reasoning paths, each inducing a distribution over answers. Distinct paths may reach the same answer through different arguments, while an error in an individual derivation can lead to a different conclusion. Test-time scaling builds on the premise that exploring these alternatives with additional computation can yield better answers than relying on a single completion.

Existing approaches implement this principle through search and sampling, including beam-search variants for chain-of-thought reasoning [32], tree search [29], Monte Carlo tree search [11], repeated sampling for code generation [6], and strategies for allocating inference-time compute [23]. Search allocates computation across partial reasoning trajectories, whereas repeated sampling explores multiple complete solutions. A simple and widely used approach is self-consistency: sample several complete generations and return the most frequent final answer [26]. Its motivation is that an answer reached through multiple independently sampled reasoning paths has stronger empirical support than one observed only once. As the sampling budget increases, the observed answer frequencies provide increasingly accurate estimates of the model's answer probabilities. Repeated sampling therefore provides a way to estimate answer-level support without committing to any particular derivation.

Self-consistency, however, implements this principle by majority voting, identifying agreement between completed answers. This works well when answers are easy to match or canonicalize, but

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.

becomes restrictive for structured outputs such as code. Functionally equivalent programs may differ in control flow, variable names, or helper functions, causing exact-string voting to fragment their support. Moreover, an answer can receive substantial probability under several reasoning traces without appearing repeatedly in a finite set of sampled completions. These limitations have motivated extensions of self-consistency beyond exact matching [7, 8]. More broadly, they raise the question of how to formulate self-consistency as an explicit inference objective, rather than tying it to a particular rule for aggregating completed outputs.

Power Sampling offers a complementary perspective: it defines an inference target by raising the probabilities of complete generations to a power, thereby favoring sequences that are already likely under the base model [14, 3, 13]. For reasoning models, this sharpens the joint distribution of reasoning traces and final answers. Despite its empirical success [14], we argue that this is not the right inference-time objective for reasoning: joint sharpening can favor an incorrect answer supported by a highly probable but flawed reasoning trace, even when the correct answer receives greater overall support across traces. It can therefore amplify reasoning errors rather than promote self-consistency.

In this work, we propose *Marginal Sharpening*, which combines the answer-level consensus principle of majority voting with the distribution-sharpening principle of Power Sampling. The key is to aggregate before sharpening: we treat reasoning traces as latent variables, sum their contributions to each answer, and then raise the resulting *answer marginal* to a power. This gives each answer a score determined by its total support across reasoning paths. Finite sharpening provides a soft preference for answers with greater marginal support, while the large-sharpening limit concentrates on maximizers of the answer marginal, following the principle underlying majority voting [26]. Marginal Sharpening expresses self-consistency as a distributional objective rather than a voting procedure, separating the desired answer distribution from the algorithm used to approximate it.

Direct Sampling from the sharpened answer marginal is challenging because evaluating it requires summing over all possible reasoning traces. For integer sharpening strengths, we exploit an exact multi-trace decomposition to construct a parallel sequential importance sampler [9] without explicitly evaluating the marginal or enumerating complete answers. This construction separates two complementary uses of test-time compute: increasing the number of reasoning traces per particle strengthens sharpening, favoring answers with greater marginal support, while increasing the number of particles improves the approximation of the target. We establish a finite-particle error bound showing that the returned answer's distribution converges to the target as the particle budget increases at fixed sharpening strength. We summarize our main contributions below.

Contributions.

- We formalize *Marginal Sharpening*, an inference-time objective that treats reasoning traces as latent variables and sharpens the induced marginal distribution over final answers.
- We exploit an exact multi-trace decomposition at integer sharpening strengths to derive a parallel autoregressive importance sampler with sequence-level corrections. We establish a finite-particle bound on the returned answer's distribution, distinguishing sharpening strength from approximation quality.
- We conduct extensive experiments showing improved accuracy over Power Sampling on mathematics and coding benchmarks, with order-of-magnitude speedups in long-generation settings. Marginal Sharpening achieves the strongest code-generation results among the compared methods and remains competitive with majority voting on mathematical reasoning.

Outline. Section 2 introduces the reasoning–answer decomposition, defines the answer marginal, and distinguishes between marginal and sequence-level sharpening. Section 3 develops Marginal Sharpening, its multi-trace representation, the importance sampler, and the finite-particle guarantee. Section 4 discusses related work. Section 5 evaluates accuracy and runtime across standard and long-generation settings, with ablations on sharpening strength, particle count, and fixed-budget allocation. Section 6 concludes.

2 PRELIMINARIES

Autoregressive language models. Let $T \in \mathbb{N}_1$ be a strictly positive natural number denoting the maximum allowed sequence length. For a prompt $x \in \mathcal{X}$, an autoregressive language model defines

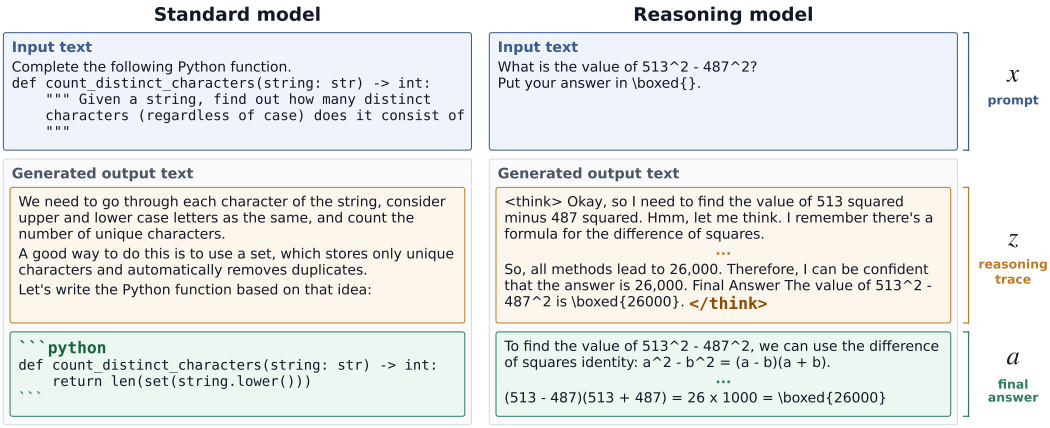


Figure 1: For a prompt x, model completions decompose into a reasoning trace z and a final answer segment a. For standard models, the boundary is identified using task-specific delimiters, such as code fences `'''`, while reasoning models use the task agnostic `</think>` delimiter.

a distribution over token sequences $y = (y_1, \ldots, y_T) \in \mathcal{Y} = \mathcal{V}^T$, where $\mathcal{V}$ is the discrete token vocabulary. The probability of a sequence factorizes token by token as

$$\pi(y \mid x) = \prod_{t=1}^{T} \pi(y_t \mid x, y_{<t}). \tag{1}$$

We include the end-of-sequence token `EOS` in $\mathcal{V}$, so variable-length generations can be represented within the fixed horizon T: the realized sequence is identified by the position of the first `EOS`, with T serving only as a maximum length.

Reasoning traces and answer segments. Reasoning-model completions naturally separate into two parts. A prefix contains intermediate computation, such as scratch work, a derivation, a proof outline, or a program sketch, while the suffix contains the final response produced from that computation. We write the parsed completion as $y = (z, a)$, where z denotes the reasoning trace and a denotes the final answer segment. This decomposition is only a parsing convention: the model still assigns probability to the full sequence, and $\pi(z, a \mid x)$ denotes the probability of the sequence whose parsed components are (z, a). Although both components are represented as token sequences, we distinguish their roles by introducing separate typed spaces $\mathcal{Z}$ and $\mathcal{A}$ for reasoning traces and answer segments, respectively. Formally, one may take $\mathcal{Z}, \mathcal{A} \subseteq \mathcal{V}^{\leq T}$, with parsed completions supported on pairs (z, a) whose concatenation, together with the answer-boundary delimiter, has length at most T.

In any large language model, this split can be identified by clear delimiters `</del>` such as boxing the answers for math problems (`boxed{}`) or fences `'''` for coding problems. In modern reasoning models [28, 4], this split is made more explicit, is task agnostic, and is generally identified by the delimiter `</think>`, which marks the end of the reasoning segment[1]. Figure 1 illustrates the decomposition with two examples. Generally, the answer segment a may itself be a long structured sequence, such as a program, a proof, or a natural-language final response.

With this distinction, the model induces a natural answer marginal by summing over all reasoning traces that lead to the same answer string:

$$m(a \mid x) = \sum_{z \in \mathcal{Z}} \pi(z, a \mid x) = \sum_{z \in \mathcal{Z}} \pi(z \mid x)\pi(a \mid x, z). \tag{2}$$

Power sampling. Power sampling [14] is a recent inference-time sharpening principle: instead of sampling directly from the model distribution, it samples from a powered version that amplifies high-probability generations. This gives the joint sharpening target

$$p_\alpha^{\text{joint}}(z, a \mid x) = \frac{\pi(z, a \mid x)^\alpha}{\sum_{z', a'} \pi(z', a' \mid x)^\alpha}. \tag{3}$$

[1] More generally, the framework only requires a separation to identify the end of reasoning.

For $\alpha > 1$, this distribution concentrates mass on complete trajectories that are already likely under the base model. Sampling from this objective was shown to achieve performance comparable to RL fine-tuning [14]. However, a complete trajectory contains both a reasoning trace and the answer produced from it. Full sequence-level power sampling therefore sharpens the joint object (z, a): it favors reasoning traces and answers that are likely together.

Reasoning traces are better viewed as latent computations that should be aggregated over [26]. The same answer may be supported by many different derivations, implementations, or proof strategies. Joint sharpening treats these alternatives as competing complete sequences rather than as shared support for a common answer-level belief. For example, suppose the correct answer $a^\star$ is reached through several distinct derivations, while an incorrect answer $\widetilde{a}$ is produced by a single, especially likely but flawed shortcut as depicted in the table.

z	a	$\pi(z, a \mid x)$
z_1	$a^\star$	0.20
z_2	$a^\star$	0.20
z_3	$a^\star$	0.20
z_4	$\widetilde{a}$	0.40

The base model prefers $a^\star$ since $m(a^\star \mid x) > m(\tilde{a} \mid x)$. But joint power sampling with, say, $\alpha = 4$, favors the wrong answer as $3 \times 0.2^4 \ll 0.4^4$. Thus, **(1)** joint sharpening can favor a single high-probability trajectory over several distinct reasoning traces that collectively support another answer. In such cases, it moves probability away from the answer preferred by the base model's marginal distribution and may therefore hurt performance. **(2)** Sampling from the joint sharpened distribution is also computationally challenging and requires additional approximation. The original method relies on block MCMC [14], while later approaches [13, 3] use future-token lookahead or more elaborate Monte Carlo corrections. These approximations become less accurate and increasingly costly as the generation length grows. Together, these two limitations motivate sharpening the model's answer-level belief directly, and building a simpler and more scalable sampler.

3 Marginal Sharpening

We introduce *Marginal Sharpening*, a test time objective that defines a power distribution over the answer marginal. For a prompt x and sharpening strength $\alpha > 1$, the target is

$$p_\alpha^{\mathrm{marg}}(a \mid x) = \frac{m(a \mid x)^\alpha}{Z_\alpha(x)}, \qquad Z_\alpha(x) = \sum_{a' \in \mathcal{A}} m(a' \mid x)^\alpha. \tag{4}$$

The effect is to amplify answers with large total support across latent reasoning traces. An answer can therefore receive high probability because it is supported collectively by many plausible traces, even if no single complete trajectory dominates. This is the answer-level analogue of power sampling: the powered object is the marginal answer distribution, not the joint distribution over traces and answers.

In the large-sharpening limit, the distribution concentrates on the most supported answer:

$$a \sim p_\alpha^{\mathrm{marg}}(\cdot \mid x) \xrightarrow[\alpha \to \infty]{} \arg\max_{a'} m(a' \mid x). \tag{5}$$

Thus, marginal sharpening is a soft version of self-consistency [26]: finite α samples from a sharpened answer distribution, while the limit recovers the answer with maximum marginal support. The target is well-defined as a powered answer marginal, but sampling from it is not straightforward.

A useful simplification occurs when the sharpening strength is an integer, $\alpha = K \in \mathbb{N}_1$. In this case, the powered answer marginal has an exact representation using K copies of the reasoning trace variable. Let $\pi_z(\cdot \mid x)$ denote the distribution over reasoning traces induced by the base model: operationally, we sample from $\pi(\cdot \mid x)$ but stop when the answer-boundary delimiter `</del>` is emitted. After a trace z is fixed, $\pi(a \mid x, z)$ is the model probability of generating answer segment a as the continuation. Expanding the Kth power gives

$$m(a \mid x)^K = \left(\sum_{z \in \mathcal{Z}} \pi_z(z \mid x) \pi(a \mid x, z) \right)^K = \sum_{z_{1:K} \in \mathcal{Z}^K} \prod_{i=1}^K \pi_z(z_i \mid x) \pi(a \mid x, z_i). \tag{6}$$

After normalization with $Z_K(x) = \sum_{a \in \mathcal{A}} m(a \mid x)^K$, we obtain the sharpened answer distribution

$$p_K^{\mathrm{marg}}(a \mid x) = \frac{\sum_{z_{1:K}} \prod_{i=1}^K \pi_z(z_i \mid x) \pi(a \mid x, z_i)}{Z_K(x)} = \frac{\mathbb{E}_{z_{1:K} \sim \pi_z^{\otimes K}(\cdot \mid x)} \left[\prod_{i=1}^K \pi(a \mid x, z_i) \right]}{Z_K(x)}. \tag{7}$$

Algorithm 1 Marginal Sharpening with Sequential Importance Sampling

Require: Prompt x, sharpening strength K, number of particles P, maximum output length T

1: Generate $M = K \cdot P$ independent reasoning traces for x, i.e. for all $m \in [M]$, $z_m \sim \pi_z(\cdot \mid x)$.
2: Form P particles with groups $z_{1:K}^{(p)}$, each containing K traces and mark them as *active*.
3: Initialize the answers $a_{\leq 0}^{(p)} \leftarrow \emptyset$ and log-normalizers $\ell_p \leftarrow 0$ for all $p \in [P]$.
4: Compute remaining budget $L \leftarrow T - \sum_{p,k} |z_k^{(p)}|/M$.
5: **for** $t = 1, \ldots, L$ **do**
6: For every *active* particle $p \in [P]$ compute the local normalizer and the next token distribution

$$\rho_t^{(p)} = \sum_{u \in \mathcal{V}} \exp\left(\sum_{k=1}^{K} \log \pi\left(u \mid x, z_k^{(p)}, a_{\leq t-1}^{(p)}\right)\right),$$

$$q_t^{(p)}(v \mid x, a_{\leq t-1}^{(p)}) = \frac{\exp\left(\sum_{k=1}^{K} \log \pi(v \mid x, z_k^{(p)}, a_{\leq t-1}^{(p)})\right)}{\rho_t^{(p)}}$$

7: Sample $a_t^{(p)} \sim q_t^{(p)}(\cdot \mid x, a_{\leq t-1}^{(p)})$ for every *active* particle p.
8: Append new tokens for every *active* particle p $a_{\leq t}^{(p)} \leftarrow (a_{\leq t-1}^{(p)}, a_t^{(p)})$.
9: Update particle log-weights for every *active* particle p:

$$\ell_p \leftarrow \ell_p + \log \rho_t^{(p)}$$

10: Mark particle p as *inactive* if $a_t^{(p)} = \texttt{EOS}$.
11: **If** all particles are inactive, **break**.
12: **end for**
13: Sample $p^* \in [P]$ with $\Pr(p^* = p) \propto \exp(\ell_p)$, and set $a^* \leftarrow a^{(p^*)}$.
14: **return** reasoning traces $z_{1:K}^{(p^*)}$ and final answer a^*.

For sharpening K, Equation (7) shows that the sharpened marginal favors answers that are consistently supported by K independent reasoning traces. In addition, it shows that $p_K^{\mathrm{marg}}(\cdot \mid x)$ is also the answer marginal of an augmented joint distribution

$$p_K(z_{1:K}, a \mid x) = \frac{1}{Z_K(x)} \pi_z^{\otimes K}(z_{1:K} \mid x) \prod_{k=1}^{K} \pi(a \mid x, z_k). \tag{8}$$

Sampling exactly from the augmented joint distribution is generally intractable, but its structure leads to an efficient sequential importance sampler [9]. We first sample $z_{1:K} \sim \pi_z^{\otimes K}(\cdot \mid x)$, then generate the answer autoregressively by locally normalizing the product of the K next-token distributions. This proposal requires no lookahead and has the desired property of being locally variance-optimal [9] as detailed in Appendix A, Proposition A.1. Specifically, we sample from

$$q_t(v \mid x, z_{1:K}, a_{<t}) = \frac{\prod_{k=1}^{K} \pi(v \mid x, z_k, a_{<t})}{\rho_t(x, z_{1:K}, a_{<t})},$$

where $\rho_t(x, z_{1:K}, a_{<t}) = \sum_{u \in \mathcal{V}} \prod_{k=1}^{K} \pi(u \mid x, z_k, a_{<t})$. The resulting joint proposal is therefore

$$Q(z_{1:K}, a \mid x) = \pi_z^{\otimes K}(z_{1:K} \mid x) Q(a \mid x, z_{1:K}),$$

with $Q(a \mid x, z_{1:K}) = \prod_{t=1}^{|a|} q_t(a_t \mid x, z_{1:K}, a_{<t})$.

With this proposal, the unnormalized importance weight for the joint target is simplified to

$$w(z_{1:K}, a) = \frac{\pi_z^{\otimes K}(z_{1:K} \mid x) \prod_{k=1}^{K} \pi(a \mid x, z_k)}{Q(z_{1:K}, a \mid x)} = \prod_{t=1}^{|a|} \rho_t(x, z_{1:K}, a_{<t}). \tag{9}$$

This construction motivates Algorithm 1. It draws P independent joint particles $(z_{1:K}^{(p)}, a^{(p)})$, computes their weights of Equation (9), and resamples one particle proportionally to these weights. It

is therefore a self-normalized importance sampler for the joint distribution $p_K(z_{1:K}, a \mid x)$. Discarding the trace component of the selected particle yields an approximation of its answer marginal $p_K^{\mathrm{marg}}(\cdot \mid x)$. We denote the resulting distribution of the returned answer by $\mathcal{L}_{K,P}(\cdot \mid x)$; the following proposition quantifies its error and shows that it converges to $p_K^{\mathrm{marg}}(\cdot \mid x)$ as $P \to \infty$.

Proposition 3.1 (Finite-particle guarantee)**.** *Fix a prompt x and an integer $K \geq 1$. Let $\mathcal{L}_{K,P}(\cdot \mid x)$ denote the distribution of the final answer $a^\star$ returned by Algorithm 1 using $P \geq 1$ independent particles and $\|\cdot\|_{\mathrm{TV}}$ the total variation distance. Then*

$$\|\mathcal{L}_{K,P}(\cdot \mid x) - p_K^{\mathrm{marg}}(\cdot \mid x)\|_{\mathrm{TV}} \leq \frac{\Delta_K(x)}{2P}.$$

with $\Delta_K(x) = \left(1 - Z_K(x) + \sqrt{1 - Z_K(x)}\right)/Z_K(x)$ an increasing function with K.

In particular, $\mathcal{L}_{K,P}(\cdot \mid x) \xrightarrow[P\to\infty]{} p_K^{\mathrm{marg}}(\cdot \mid x)$ and $\mathcal{L}_{1,P}(\cdot \mid x) = m(\cdot|x)$ for all $P \geq 1$.

In addition of characterizing convergence, this bound makes a tradeoff between K and P explicit. Increasing K concentrates the target on answers with stronger marginal support, thereby enforcing greater self-consistency, but it also increases $\Delta_K(x)$ making the target distribution harder to approximate. Increasing P compensates for this difficulty by reducing the approximation error. Hence, K controls the strength of the target, with larger values generally yielding better but harder-to-approximate distributions, while P controls the quality of the approximation. We provide the proof in Appendix B, together with a discussion of its statistical implication: marginal sharpening defines a more favorable sampling problem than Power Sampling by operating on the smaller answer space.

Cost and implementation. Algorithm 1 has two main parameters: the sharpening strength K, which determines the target distribution, and the number of particles P, which controls approximation quality. The P particles are processed in parallel. Each decoding step requires K trace-conditioned model evaluations per particle, for a total of $\mathcal{O}(KP)$ evaluations. These evaluations are batched across particles, while the cached states associated with each trace are reused throughout generation.

4 Related Work

Chain-of-thought and reasoning models. Chain-of-thought prompting showed that language models can improve on multi-step tasks by generating intermediate reasoning before the final answer [27], and similar behavior can be elicited with simple zero-shot prompts [16]. Recent reasoning models make this trace–answer structure central, often using post-training or reinforcement learning to produce longer reasoning traces for mathematics and code [10, 15]. Our framework uses this structure directly, while remaining agnostic to how the traces are produced.

Self-consistency. Self-consistency estimates answer-level support by repeatedly sampling answers and returning the most frequent one [26]. This works well when answers can be exactly matched or canonicalized, but becomes brittle when equivalent solutions have different surface forms. To address this limitation, universal self-consistency uses model-based response selection [7], while integrative decoding and ThinkMerge generate a shared answer by aggregating next-token log-probabilities conditioned on sampled responses or reasoning traces, respectively [8, 25]. Marginal sharpening instead defines the sharpened answer marginal as the inference target, which our algorithm approximates with convergence guarantees. Finite sharpening yields a soft answer-level distribution, while the large-sharpening limit concentrates on answers sought by self-consistency.

Inference-time sharpening. Power-sampling methods sharpen model distributions at inference time without additional training. Power Sampling uses Metropolis–Hastings to sample from sequence-level power distributions [14]; Scalable Power Sampling uses an autoregressive proposal with future-continuation corrections [13]; and Power-SMC uses sequential Monte Carlo to sample from the same trajectory-level target [3]. These methods sharpen complete generations. Marginal sharpening instead changes the object of sharpening: it sharpens the answer marginal induced by latent reasoning traces.

5 Experiments

We evaluate on five reasoning benchmarks spanning mathematics and code generation. MATH-500 is a 500-problem subset of MATH for competition-style mathematical problem solving [12], while

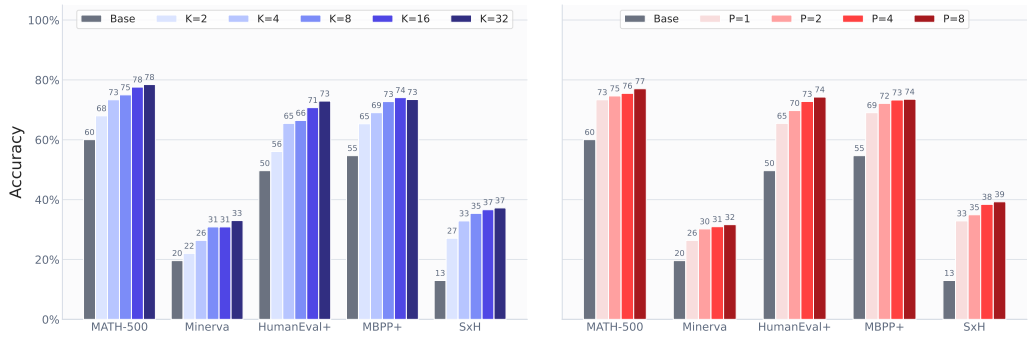


Figure 2: Test-time scaling for Marginal Sharpening with Qwen2.5 14B. Accuracy improves as either K increases (at fixed $P = 1$) or P increases (at fixed $K = 4$).

Minerva [18] contains 272 challenging quantitative problems from scientific domains. For both, final answers are extracted from boxed expressions and checked symbolically with Math-Verify [17]. HumanEval+ and MBPP+ contain 164 and 378 Python code synthesis problems specified through natural-language docstrings and descriptions, respectively. Both augment the original benchmarks with more rigorous test suites [5, 21]. Finally, the Spider Extra-Hard (SxH) dataset consists of 166 text-to-SQL problems and is evaluated by comparing the predicted and reference query results across SQLite test databases [30]. Together, the five datasets provide a diverse testbed for reasoning.

We use a base sampling temperature of $\tau = 1.0$ and refer to $\tau = 0.25$ as low-temperature sampling, following [14]. Our main experiments use Phi-3.5 [1] and Qwen2.5 [24] models, with a maximum generation length of $T = 4096$. Since these models do not expose an explicit reasoning–answer boundary, we define the answer segment using task-specific delimiters: `\boxed{` for mathematics, ``` for code and `Final answer` for Text-to-SQL. The answer segment naturally comes after the delimiter. Reported results are averaged over five runs and *standard errors do not exceed* 0.5%.

5.1 Test-Time Scaling

Marginal Sharpening scales with compute. We use Qwen2.5-14B to study how Marginal Sharpening scales with its two main compute knobs, K and P. The parameter K controls the sharpening strength, while P controls the approximation quality of the sharpened marginal. Figure 2 shows that increasing either parameter generally improves accuracy over standard sampling. For fixed $P = 1$, increasing K concentrates probability on self-consistent answers and generally improves performance. For fixed $K = 4$, increasing P further improves accuracy by more faithfully approximating the target distribution, highlighting the practical benefit of the sequence-level importance correction. In both cases, gains gradually saturate as more compute is allocated. Overall, Marginal Sharpening converts additional test-time compute into higher accuracy in two complementary ways: increasing K defines a sharper, often better target, while increasing P improves the approximation of that target.

Budget allocation between K and P. We study how to allocate a fixed budget of reasoning traces between sharpening strength K and the number of particles P, with $KP = 32$. Figure 3 shows that all allocations substantially outperform temperature sampling, with relatively stable performance across configurations. On mathematical benchmarks, shifting compute toward larger K generally helps: MATH-500 improves up to $K = 16$, while Minerva benefits up to $K = 16$–32. Coding benchmarks favor more balanced allocations, with $K = 4$–8 performing best on HumanEval+, MBPP+ and SxH. Overall, once $K \geq 4$, the precise allocation has only a modest effect, indicating robustness to the choice of K and P. Appendix C.1 confirms this trend across the other models. A practical guideline is therefore to sharpen strongly to define a self consistent target, then devote the remaining budget to increasing P and improving its approximation. This is consistent with the statistical tradeoff in Proposition 3.1.

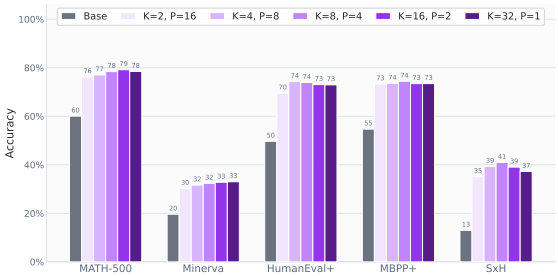


Figure 3: Qwen2.5-14B: Comparison of allocation configurations with the budget $K \cdot P = 32$.

Table 1: Results for standard models, averaged over 5 seeds. Best is bolded; second best is underlined.

Model / Method	MATH-500	Minerva	HumanEval+	MBPP+	SxH
Phi3.5-Mini-Instruct (4B)					
Base	0.373	0.147	0.531	0.378	0.180
Low Temperature	0.355	0.178	0.579	0.411	0.184
Power Sampling	0.509	0.251	0.573	0.613	0.182
Majority Voting ($k = 32$)	**0.554**	0.261	0.537	0.370	<u>0.210</u>
Marginal Sharpening ($K = 8,\ P = 4$)	**0.553**	**0.315**	**0.635**	**0.655**	0.206
Marginal Sharpening ($K = 16,\ P = 2$)	<u>0.541</u>	<u>0.313</u>	<u>0.618</u>	**0.656**	**0.218**
Qwen2.5 (7B)					
Base	0.490	0.121	0.417	0.350	0.117
Low Temperature	0.691	0.201	0.633	0.243	0.280
Power Sampling	0.730	0.223	0.669	0.255	0.321
Majority Voting ($k = 32$)	**0.780**	**0.324**	0.451	0.222	0.159
Marginal Sharpening ($K = 8,\ P = 4$)	0.730	0.246	**0.677**	<u>0.664</u>	**0.400**
Marginal Sharpening ($K = 16,\ P = 2$)	<u>0.734</u>	<u>0.259</u>	**0.677**	**0.667**	<u>0.390</u>
Qwen2.5 (14B)					
Base	0.600	0.196	0.497	0.547	0.130
Low Temperature	0.733	0.275	0.659	0.481	0.304
Power Sampling	0.776	0.288	0.674	0.450	0.361
Majority Voting ($k = 32$)	**0.826**	**0.408**	0.561	0.500	0.199
Marginal Sharpening ($K = 8,\ P = 4$)	0.784	0.324	**0.739**	**0.744**	**0.410**
Marginal Sharpening ($K = 16,\ P = 2$)	<u>0.792</u>	<u>0.328</u>	<u>0.731</u>	<u>0.734</u>	<u>0.390</u>

5.2 Performance Comparison

For our first comparison, we consider three models of increasing scale: Phi-3.5-Mini-Instruct (4B) and the 7B and 14B models from the Qwen2.5 family [24]. We compare Marginal Sharpening with Power Sampling [14] and Majority Voting [26]. Power Sampling uses its recommended hyperparameters, $\alpha = 4$, $B = 16$, and $N = 10$. Majority Voting uses 32 samples, while Marginal Sharpening uses the same total trace budget, $KP = 32$, with $(K, P) = (8, 4)$ and $(16, 2)$. At this budget, Marginal Sharpening is at least $\sim 3\times$ faster than Power Sampling. Table 1 reports accuracies averaged over five seeds. Marginal Sharpening performs best on the three coding benchmarks for all three models, with particularly large gains on MBPP+. This supports sharpening the answer marginal rather than the full sequence distribution: in code generation, probability mass for a correct solution can be spread across many syntactically distinct programs. Strikingly, on MBPP+, Power Sampling underperforms the base model for both Qwen2.5 models, illustrating the pathological behavior described in Section 2: joint sharpening can amplify a high-probability trajectory even when it moves probability away from the answer favored by the model's marginal distribution. On mathematics, Marginal Sharpening remains competitive, consistently achieving the second-best results and attaining the best performances for Phi-3.5. Majority Voting remains strong on these more canonical tasks, where exact-answer aggregation is particularly effective. Overall, Marginal Sharpening is strongest on coding, competitive on mathematics, and consistently outperforms Power Sampling across all three models. Importantly, it also improves over the base model in every setting, unlike the competing test-time scaling methods that sometimes deteriorate the base performance. Additional details about Power Sampling's pathological behavior and runtime is provided in Appendix C.2.

5.3 Long Context Reasoning Models

Recent reasoning models allocate substantially more tokens to intermediate reasoning before producing a final answer [28, 4], making long-generation settings increasingly relevant for test-time scaling. We therefore evaluate Marginal Sharpening on three such models: SmoLLM3 [4], Ministral 3 [20], and Qwen3 [28]. These models explicitly separate reasoning from the final response, as illustrated in Figure 1. In our implementation, the reasoning trace is the generated prefix up to `</think>`, while the continuation after this delimiter is treated as the answer segment. This provides a benchmark-independent definition of the reasoning–answer boundary.

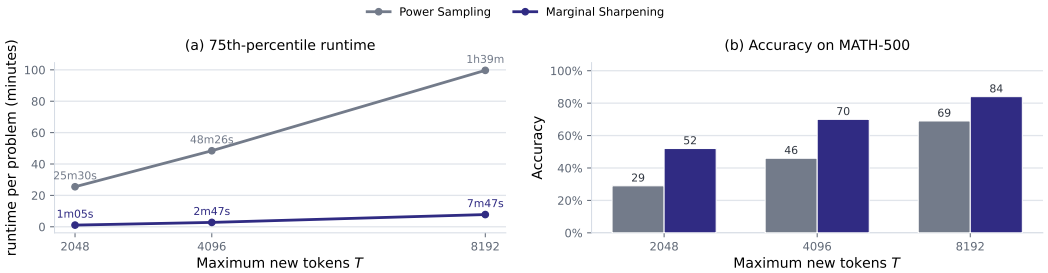


Figure 4: Runtime and accuracy comparison for Qwen3-4B on MATH-500 as T changes.

Power Sampling scales poorly. Although both Power Sampling and Marginal Sharpening require $\mathcal{O}(T)$ token-level model evaluations, their wall-clock scaling differs substantially. Power Sampling [14] repeatedly revisits and resamples blocks of the same sequence, introducing sequential refinement steps. In contrast, for fixed K and P, Marginal Sharpening generates reasoning traces in parallel and requires only a single autoregressive decoding pass. Figure 4 compares accuracy and upper-quartile runtime[2] on MATH-500 with Qwen3-4B as T increases. Marginal Sharpening is consistently faster and more accurate, with the absolute runtime gap widening substantially for longer generations. This becomes particularly important for modern reasoning models, where generation lengths such as $T \geq 16384$ are common. Overall, answer-level sharpening provides a substantially more scalable test-time compute target, achieving stronger performance at lower computational cost.

Performance for Long Reasoning. We further evaluate Marginal Sharpening at $T = 16384$ on reasoning models from three different families. We omit Power Sampling in this regime, as its computational scaling makes it impractical for such long generations. Table 2 summarizes the results. Although the gains are more modest for these already strong models, Marginal Sharpening remains highly effective, achieving the best result on 13 of the 15 model–benchmark combinations. Importantly, the improvements are not confined to a particular task type or model family: the gains observed at shorter generation lengths persist as reasoning traces become substantially longer, across both mathematical reasoning and code generation. Taken together with our previous experiments, these results establish sharpening the answer marginal as an effective test-time compute objective across generation lengths, model families, and task domains.

Table 2: Results for long-context reasoning models.

Model / Method	MATH-500	Minerva	HumanEval+	MBPP+	SxH
SmoLLM3 (3B)					
Base	0.892	0.377	0.623	0.677	0.118
Low Temperature	0.883	0.370	0.549	0.642	0.153
Majority Voting ($k = 32$)	0.940	**0.445**	0.488	0.718	0.149
MS ($K = 8,\ P = 4$)	0.952	0.439	0.594	**0.741**	0.229
MS ($K = 16, P = 2$)	**0.956**	0.441	**0.628**	0.735	**0.242**
Ministral 3 (3B)					
Base	0.654	0.294	0.808	0.716	0.256
Low Temperature	0.704	0.302	0.765	0.687	0.302
Majority Voting ($k = 32$)	0.800	0.404	0.823	0.740	**0.335**
MS ($K = 8,\ P = 4$)	0.824	**0.412**	0.890	0.773	**0.337**
MS ($K = 16, P = 2$)	**0.826**	**0.412**	**0.915**	**0.780**	**0.333**
Qwen3 (4B)					
Base	0.905	0.503	0.887	0.780	0.379
Low Temperature	0.913	0.504	0.891	0.780	0.378
Majority Voting ($k = 32$)	0.942	0.518	0.890	0.780	**0.453**
MS ($K = 8,\ P = 4$)	**0.956**	**0.526**	0.902	0.781	0.433
MS ($K = 16, P = 2$)	0.945	**0.526**	**0.909**	0.781	0.423

6 Conclusion

We introduced *Marginal Sharpening*, an inference-time method that treats reasoning traces as latent variables and sharpens the answer marginal they induce, rather than the probability of complete trajectories. For integer sharpening strengths, this target admits a multi-trace representation, yielding a simple autoregressive sampler, with convergence guarantees, that samples several traces and produces one answer supported across them. Empirically, marginal sharpening improves over temperature sampling, outperforms sequence-level power sampling on reasoning benchmarks, and is an order of magnitude faster, particularly in long-reasoning settings. These results show that self-consistency can be understood as an answer-level inference principle than as a voting heuristic, and that test-time compute is more effectively spent on the answer belief induced by latent reasoning than on selecting the most likely transcript. Marginal sharpening is only one formalization of this principle: generalizing this principle and developing better samplers are promising directions for future work.

[2]We report the 75th percentile to capture runtime on hard problems.

AI USE STATEMENT

In this work, we used generative AI tools to assist with method implementation, scaling experiments, polishing the manuscript, and expanding details in proofs developed by the authors. We also used them to generate code for scientific figures and diagrams based on experimental results.

We did not use generative AI tools to generate synthetic datasets; develop theoretical models or conceptual frameworks; formulate mathematical claims or provide essential ingredients for their proofs; propose or refine hypotheses; translate text; clean or reformat datasets; analyze qualitative data; or interpret results.

All AI-assisted content was reviewed and verified by the authors. LLM-generated text was independently checked and corrected by all authors, while LLM-generated code was independently verified by two authors. We take full responsibility for the final content of this work.

ETHICS STATEMENT

Marginal sharpening improves the inference-time reasoning ability of existing language models without additional training. This can benefit applications where stronger reasoning is valuable, including mathematical problem solving, code generation, proof assistance, and scientific or technical workflows. By making test-time compute more effective, the method may also reduce the cost of obtaining stronger answers from a fixed model.

Stronger reasoning can benefit applications such as mathematics, coding, proof assistance, and technical workflows, but it can also amplify dual-use risks such as malicious automation or planning. Marginal sharpening does not introduce risks independent of the underlying model, but it can expose stronger capabilities at inference time. Its use should therefore be paired with the same safeguards as other reasoning-enhancement methods.

REPRODUCIBILITY STATEMENT

We provide the complete source code implementing the proposed method and the experiments conducted in the supplementary materials for this submission. This is the simplest and most reliable way to reproduce the results.

To reproduce the results from scratch, we provide a detailed description of the method in the form of pseudocode in Algorithm 1. A list of used datasets and evaluation methods, as well as links to the articles presenting them, can be found in the first paragraph of the Section 5. The hyperparameters used are listed in the same paragraph and in the paragraphs corresponding to each individual experiment. Detailed proofs of the theoretical statements presented in this article can be found in Appendices A and B.

REFERENCES

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka,

Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

[2] S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017. ISSN 08834237, 21688745. URL http://www.jstor.org/stable/26408299.

[3] Seyedarmin Azizi, Erfan Baghaei Potraghloo, Minoo Ahmadi, Souvik Kundu, and Massoud Pedram. Power-SMC: Low-latency sequence-level power sampling for training-free LLM reasoning. *arXiv preprint arXiv:2602.10273*, 2026. URL https://arxiv.org/abs/2602.10273.

[4] Elie Bakouch, Carlos Miguel Patiño, Anton Lozhkov, et al. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

[5] 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. URL https://arxiv.org/abs/2107.03374.

[6] 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.

[7] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*, 2024. URL https://openreview.net/forum?id=LjsjHF7nAN.

[8] Yi Cheng, Xiao Liang, Yeyun Gong, Wen Xiao, Song Wang, Yuji Zhang, Wenjun Hou, Kaishuai Xu, Wenge Liu, Wenjie Li, Jian Jiao, Qi Chen, Peng CHENG, and Wayne Xiong. Integrative decoding: Improving factuality via implicit self-consistency. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gGWYecsK1U.

[9] N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer Series in Statistics. Springer International Publishing, 2020. ISBN 9783030478452. URL https://books.google.fr/books?id=ZZEAEAAAQBAJ.

[10] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

[11] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/abs/2305.14992.

[12] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

[13] Xiaotong Ji, Rasul Tutunov, Matthieu Zimmer, and Haitham Bou Ammar. Scalable power sampling: Unlocking efficient, training-free reasoning for LLMs via distribution sharpening. *arXiv preprint arXiv:2601.21590*, 2026. URL https://arxiv.org/abs/2601.21590.

[14] Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. *arXiv preprint arXiv:2510.14901*, 2025. URL https://arxiv.org/abs/2510.14901.

[15] Kimi Team, Angang Du, Bofei Gao, et al. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. URL https://arxiv.org/abs/2501.12599.

[16] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

[17] Hynek Kydlíček. Math-Verify: Math verification library. https://github.com/huggingface/Math-Verify, 2025. Software.

[18] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

[19] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

[20] Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

[21] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

[22] OpenAI. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

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

[24] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

[25] Haonan Wang, Chao Du, Kenji Kawaguchi, and Tianyu Pang. Think in parallel, answer as one: Logit averaging for open-ended reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hvit36Dyzl.

[26] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. URL https://arxiv.org/abs/2203.11171.

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[28] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[29] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://arxiv.org/abs/2305.10601.

[30] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019. URL https://arxiv.org/abs/1809.08887.

[31] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

[32] Tinghui Zhu, Kai Zhang, Jian Xie, and Yu Su. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=S1XnUsqwr7.

A Local Optimality of the Proposal

The choice of proposal is important in sequential importance sampling, since large variability in the incremental weights can quickly lead to weight degeneracy [9]. One common local criterion is to minimize this variance conditional on the information available at the current step. The proposal used in Algorithm 1 has this property, as shown next.

Proposition A.1 (Local optimality of the proposal)**.** *Fix a prompt x, an integer $K \geq 1$, a trace group $z_{1:K}$, and an answer prefix $a_{<t}$ such that $\rho_t(x, z_{1:K}, a_{<t}) > 0$. For any next-token proposal $r_t(\cdot \mid x, z_{1:K}, a_{<t})$ that is positive wherever $\prod_{k=1}^{K} \pi(v \mid x, z_k, a_{<t}) > 0$, define the incremental importance weight*

$$W_t^r = \frac{\prod_{k=1}^{K} \pi(V \mid x, z_k, a_{<t})}{r_t(V \mid x, z_{1:K}, a_{<t})}, \qquad V \sim r_t(\cdot \mid x, z_{1:K}, a_{<t}).$$

Then the proposal used in Algorithm 1,

$$q_t(v \mid x, z_{1:K}, a_{<t}) = \frac{\prod_{k=1}^{K} \pi(v \mid x, z_k, a_{<t})}{\rho_t(x, z_{1:K}, a_{<t})},$$

minimizes the conditional variance of the incremental importance weight:

$$\mathrm{Var}(W_t^q \mid x, z_{1:K}, a_{<t}) = 0.$$

Proof. Fix x, $z_{1:K}$, and $a_{<t}$. All expectations and variances below are conditional on this trace group and prefix. For any admissible proposal r_t, the expected incremental weight is

$$\mathbb{E}[W_t^r] = \sum_{\substack{v \in \mathcal{V}: \\ r_t(v \mid x, z_{1:K}, a_{<t}) > 0}} r_t(v \mid x, z_{1:K}, a_{<t}) \frac{\prod_{k=1}^{K} \pi(v \mid x, z_k, a_{<t})}{r_t(v \mid x, z_{1:K}, a_{<t})} \tag{10}$$

$$= \sum_{v \in \mathcal{V}} \prod_{k=1}^{K} \pi(v \mid x, z_k, a_{<t}) \tag{11}$$

$$= \rho_t(x, z_{1:K}, a_{<t}), \tag{12}$$

where the second equality follows from the support condition. Thus, all admissible proposals have the same expected incremental weight. Under q_t, the incremental weight is constant:

$$W_t^q = \frac{\prod_{k=1}^{K} \pi(V \mid x, z_k, a_{<t})}{q_t(V \mid x, z_{1:K}, a_{<t})} \tag{13}$$

$$= \rho_t(x, z_{1:K}, a_{<t}) \qquad \text{almost surely.} \tag{14}$$

Consequently,

$$\mathrm{Var}(W_t^q) = \mathbb{E}\Big[\big(W_t^q - \rho_t(x, z_{1:K}, a_{<t})\big)^2\Big] = 0.$$

q_t is the minimizer setting the conditional variance of the incremental importance weight to zero. □

This result concerns the current token draw, conditional on the trace group and prefix. The normalizer ρ_t may still vary across particles and along trajectories, so the proposition does not imply that q_t minimizes the variance of the terminal importance weight. Reducing this variance requires look-ahead over future continuations, which is computationally expensive and outside the scope of this work.

B Convergence Result

The result follows from a standard self-normalized importance sampling argument [2]. The key property in our setting is that the proposal yields uniformly bounded importance weights: since each local normalizer satisfies $\rho_t \leq 1$, we have $0 \leq W_p = \prod_t \rho_t \leq 1$. This directly controls the second moment of the weights and leads to the explicit finite-P bound below, accompanied with a detailed proof for completeness.

Proposition B.1 (Finite-particle guarantee). *Fix a prompt x and an integer $K \geq 1$. Let $\mathcal{L}_{K,P}(\cdot \mid x)$ denote the distribution of the final answer returned by Algorithm 1 using $P \geq 1$ independent particles, and define*

$$Z_K(x) = \sum_{a \in \mathcal{A}} m(a \mid x)^K.$$

Then

$$\|\mathcal{L}_{K,P}(\cdot \mid x) - p_K^{\mathrm{marg}}(\cdot \mid x)\|_{\mathrm{TV}} \leq \frac{1 - Z_K(x) + \sqrt{1 - Z_K(x)}}{2PZ_K(x)}.$$

In particular,

$$\mathcal{L}_{K,P}(\cdot \mid x) \xrightarrow[P \to \infty]{\mathrm{TV}} p_K^{\mathrm{marg}}(\cdot \mid x).$$

Moreover, when $K = 1$, $Z_1(x) = 1$, and the algorithm is exact for every $P \geq 1$:

$$\mathcal{L}_{1,P}(\cdot \mid x) = m(\cdot \mid x).$$

Proof. Fix x and K. Let

$$W_p = w\left(z_{1:K}^{(p)}, A^{(p)}\right)$$

denote the importance weight of particle p. By Equation (9),

$$W_p = \frac{\pi_z^{\otimes K}\left(z_{1:K}^{(p)} \mid x\right) \prod_{k=1}^K \pi\left(A^{(p)} \mid x, z_k^{(p)}\right)}{\pi_z^{\otimes K}\left(z_{1:K}^{(p)} \mid x\right) Q_{z_{1:K}^{(p)}}\left(A^{(p)} \mid x\right)} \tag{15}$$

$$= \frac{\prod_{k=1}^K \pi\left(A^{(p)} \mid x, z_k^{(p)}\right)}{Q_{z_{1:K}^{(p)}}\left(A^{(p)} \mid x\right)} = \prod_{t=1}^{|A^{(p)}|} \rho_t\left(x, z_{1:K}^{(p)}, A_{<t}^{(p)}\right). \tag{16}$$

The particles $(A^{(p)}, W_p)$ are independent and identically distributed. Moreover,

$$\mathbb{E}[W_1] = \sum_{a \in \mathcal{A}} \mathbb{E}_{z_{1:K} \sim \pi_z^{\otimes K}(\cdot \mid x)}\left[Q_{z_{1:K}}(a \mid x) w(z_{1:K}, a)\right] \tag{17}$$

$$= \sum_{a \in \mathcal{A}} \mathbb{E}_{z_{1:K} \sim \pi_z^{\otimes K}(\cdot \mid x)}\left[\prod_{k=1}^K \pi(a \mid x, z_k)\right] \tag{18}$$

$$= \sum_{a \in \mathcal{A}} m(a \mid x)^K = Z_K(x). \tag{19}$$

Let $f : \mathcal{A} \to [-1, 1]$ be arbitrary and define

$$\widehat{H}_P(f) = \frac{1}{P} \sum_{p=1}^P W_p f(A^{(p)}), \qquad \widehat{Z}_P = \frac{1}{P} \sum_{p=1}^P W_p.$$

Conditionally on the generated particles, the algorithm selects particle p with probability

$$\frac{W_p}{\sum_{r=1}^P W_r}.$$

Therefore,

$$\mathbb{E}\left[f(a^\star) \mid \{A^{(p)}, W_p\}_{p=1}^P\right] = \frac{\widehat{H}_P(f)}{\widehat{Z}_P},$$

and hence

$$\mathcal{L}_{K,P}(f \mid x) = \mathbb{E}\left[\frac{\widehat{H}_P(f)}{\widehat{Z}_P}\right].$$

On the other hand, the importance-weight identity gives

$$\mathbb{E}[\widehat{H}_P(f)] = \mathbb{E}[W_1 f(A^{(1)})] \tag{20}$$
$$= \sum_{a\in\mathcal{A}} f(a) m(a \mid x)^K \tag{21}$$
$$= Z_K(x)\, p_K^{\mathrm{marg}}(f \mid x). \tag{22}$$

Consequently,

$$\mathcal{L}_{K,P}(f \mid x) - p_K^{\mathrm{marg}}(f \mid x) = \mathbb{E}\left[\frac{\widehat{H}_P(f)}{\widehat{Z}_P} - \frac{\widehat{H}_P(f)}{Z_K(x)}\right] \tag{23}$$
$$= \mathbb{E}\left[\widehat{H}_P(f)\left(\frac{1}{\widehat{Z}_P} - \frac{1}{Z_K(x)}\right)\right]. \tag{24}$$

For any $y > 0$ and $Z > 0$,

$$\frac{1}{y} - \frac{1}{Z} = \frac{Z-y}{Z^2} + \frac{(Z-y)^2}{Z^2 y}.$$

Applying this identity with $y = \widehat{Z}_P$ and $Z = Z_K(x)$ gives

$$\mathcal{L}_{K,P}(f \mid x) - p_K^{\mathrm{marg}}(f \mid x) = \frac{1}{Z_K(x)^2}\mathbb{E}\left[\widehat{H}_P(f)\left(Z_K(x) - \widehat{Z}_P\right)\right] \tag{25}$$
$$+ \frac{1}{Z_K(x)^2}\mathbb{E}\left[\widehat{H}_P(f)\frac{\left(Z_K(x) - \widehat{Z}_P\right)^2}{\widehat{Z}_P}\right]. \tag{26}$$

We first bound the linear term. Expanding the empirical averages,

$$\mathbb{E}\left[\widehat{H}_P(f)\left(Z_K(x) - \widehat{Z}_P\right)\right] \tag{27}$$
$$= \frac{1}{P^2}\sum_{p=1}^{P}\sum_{r=1}^{P}\mathbb{E}\left[W_p f(A^{(p)})\left(Z_K(x) - W_r\right)\right]. \tag{28}$$

For $p \neq r$, independence gives

$$\mathbb{E}\left[W_p f(A^{(p)})\left(Z_K(x) - W_r\right)\right] \tag{29}$$
$$= \mathbb{E}\left[W_p f(A^{(p)})\right]\left(Z_K(x) - \mathbb{E}[W_r]\right) = 0. \tag{30}$$

Only the P diagonal terms remain, so

$$\mathbb{E}\left[\widehat{H}_P(f)\left(Z_K(x) - \widehat{Z}_P\right)\right] = \frac{1}{P}\mathbb{E}\left[W_1 f(A^{(1)})\left(Z_K(x) - W_1\right)\right].$$

Since $|f| \le 1$, Cauchy–Schwarz yields

$$\left|\mathbb{E}\left[\widehat{H}_P(f)\left(Z_K(x) - \widehat{Z}_P\right)\right]\right| \tag{31}$$
$$\le \frac{1}{P}\mathbb{E}\left[W_1\left|Z_K(x) - W_1\right|\right] \tag{32}$$
$$\le \frac{1}{P}\sqrt{\mathbb{E}[W_1^2]\ \mathrm{Var}(W_1)}. \tag{33}$$

We next bound the quadratic term. Since $W_p \ge 0$ and $|f| \le 1$,

$$\left|\widehat{H}_P(f)\right| \le \frac{1}{P}\sum_{p=1}^{P} W_p = \widehat{Z}_P.$$

It follows that

$$\left|\mathbb{E}\left[\widehat{H}_P(f)\frac{\left(Z_K(x)-\widehat{Z}_P\right)^2}{\widehat{Z}_P}\right]\right| \tag{34}$$

$$\leq \mathbb{E}\left[\left(Z_K(x)-\widehat{Z}_P\right)^2\right] \tag{35}$$

$$= \mathrm{Var}(\widehat{Z}_P) \tag{36}$$

$$= \frac{\mathrm{Var}(W_1)}{P}. \tag{37}$$

Combining Equations (26), (33), and (37), we obtain

$$|\mathcal{L}_{K,P}(f \mid x) - p_K^{\mathrm{marg}}(f \mid x)| \leq \frac{\mathrm{Var}(W_1) + \sqrt{\mathbb{E}[W_1^2]\,\mathrm{Var}(W_1)}}{PZ_K(x)^2}.$$

Since

$$\mathbb{E}[W_1^2] = \mathrm{Var}(W_1) + Z_K(x)^2,$$

this can equivalently be written as

$$|\mathcal{L}_{K,P}(f \mid x) - p_K^{\mathrm{marg}}(f \mid x)| \leq \frac{\mathrm{Var}(W_1) + \sqrt{\mathrm{Var}(W_1)\,(\mathrm{Var}(W_1) + Z_K(x)^2)}}{PZ_K(x)^2}.$$

Using the dual representation of total variation,

$$\|\mu - \nu\|_{\mathrm{TV}} = \frac{1}{2}\sup_{\|f\|_\infty \leq 1} |\mu(f) - \nu(f)|,$$

we obtain the intermediate bound

$$\|\mathcal{L}_{K,P}(\cdot \mid x) - p_K^{\mathrm{marg}}(\cdot \mid x)\|_{\mathrm{TV}} \leq \frac{\mathrm{Var}(W_1) + \sqrt{\mathrm{Var}(W_1)\,(\mathrm{Var}(W_1) + Z_K(x)^2)}}{2PZ_K(x)^2}. \tag{38}$$

It remains to express this bound only through $Z_K(x)$. At every token,

$$\rho_t = \sum_{u\in\mathcal{V}}\prod_{k=1}^{K}\pi(u \mid x, z_k, a_{<t}) \leq \sum_{u\in\mathcal{V}}\pi(u \mid x, z_1, a_{<t}) = 1.$$

Therefore,

$$0 \leq W_1 = \prod_t \rho_t \leq 1.$$

It follows that $W_1^2 \leq W_1$, and hence

$$\mathbb{E}[W_1^2] \leq \mathbb{E}[W_1] = Z_K(x).$$

Consequently,

$$\mathrm{Var}(W_1) = \mathbb{E}[W_1^2] - Z_K(x)^2 \tag{39}$$

$$\leq Z_K(x) - Z_K(x)^2 \tag{40}$$

$$= Z_K(x)\big(1 - Z_K(x)\big). \tag{41}$$

Moreover,

$$\sqrt{\mathbb{E}[W_1^2]\,\mathrm{Var}(W_1)} \leq \sqrt{Z_K(x)^2\big(1 - Z_K(x)\big)} \tag{42}$$

$$= Z_K(x)\sqrt{1 - Z_K(x)}. \tag{43}$$

Substituting these inequalities into Equation (38) gives

$$\|\mathcal{L}_{K,P}(\cdot \mid x) - p_K^{\mathrm{marg}}(\cdot \mid x)\|_{\mathrm{TV}} \leq \frac{1 - Z_K(x) + \sqrt{1 - Z_K(x)}}{2PZ_K(x)}.$$

Finally, when $K = 1$,

$$Z_1(x) = \sum_{a \in \mathcal{A}} m(a \mid x) = 1.$$

Equivalently, every local normalizer satisfies

$$\rho_t = \sum_{u \in \mathcal{V}} \pi(u \mid x, z, a_{<t}) = 1,$$

so all particle weights are equal to one. The bound is therefore zero, and

$$\mathcal{L}_{1,P}(\cdot \mid x) = m(\cdot \mid x)$$

for every $P \geq 1$. □

Statistical Advantage of Sharpening the Marginal. The convergence bound depends on the answer-level normalizer $Z_K(x)$, which aggregates probability mass across reasoning traces supporting the same answer. For the same sharpening strength $K \geq 1$, this normalizer is at least as large as its full-sequence counterpart:

$$Z_K(x) = \sum_{a \in \mathcal{A}} \left(\sum_{z \in \mathcal{Z}} \pi(z, a \mid x) \right)^K \geq \sum_{a \in \mathcal{A}} \sum_{z \in \mathcal{Z}} \pi(z, a \mid x)^K = Z_K^{\text{joint}}(x).$$

These constants have a simple interpretation: $Z_K(x)$ is the probability that K independent completions share the same answer, whereas $Z_K^{\text{joint}}(x)$ requires their complete reasoning–answer trajectories to coincide. Agreement at the answer level is less restrictive, particularly when many long and diverse reasoning traces support the same answer. Since the bound $(1 - Z + \sqrt{1 - Z})/(2PZ)$ decreases with Z, marginal sharpening yields a no-larger normalization-based error bound than joint sharpening under the same bounded-weight importance-sampling assumptions. This comparison establishes an advantage in the worst-case guarantee, rather than a universal improvement in sampling efficiency: the actual error and runtime also depend on the proposal and sampling algorithm.

C Additional Discussions and Experiments

C.1 Fixed-Budget Allocation of Test-Time Compute

Figure 5 extends the fixed-budget analysis to all three standard models. We compare five allocations satisfying $KP = 32$, ranging from $(K, P) = (2, 16)$ to $(K, P) = (32, 1)$. Every allocation improves over temperature sampling across all model–benchmark pairs, showing that Marginal Sharpening is robust to how the budget is divided between sharpening strength and particle approximation. No single allocation dominates across all tasks. On the Qwen2.5 models, mathematical benchmarks generally benefit from larger K, with the best results often obtained around $K = 16$ or $K = 32$. Coding benchmarks tend to favor more balanced allocations, typically around $K = 4$ or $K = 8$, although the differences are small. In particular, once $K \geq 4$, performance generally varies by only a few accuracy points across allocations. These results support the tradeoff identified in Proposition 3.1: increasing K defines a sharper target, while increasing P improves its approximation. In practice, a moderate value of K appears sufficient, after which the remaining budget can be allocated to increasing P.

C.2 Power Sampling: Runtime and Pathological Behavior

Runtime. Figure 6 compares the mean runtime of Power Sampling with four Marginal Sharpening allocations satisfying $KP = 32$. Power Sampling is slower in every model–benchmark combination. The gap ranges from $2.89\times$ to $3.74\times$ on MATH-500, from $2.62\times$ to $2.88\times$ on Minerva, and from $2.21\times$ to $6.32\times$ on HumanEval+, with an especially large difference on MBPP+: Power Sampling is $6.30\times$ slower for Phi-3.5 and respectively $35.70\times$ and $46.72\times$ slower for Qwen2.5 7B/14B.

This gap follows from the different computational structures of the methods. Power Sampling repeatedly revisits and resamples blocks of the same generation, creating sequential refinement steps. Marginal Sharpening instead generates its reasoning traces in parallel and performs a single autoregressive pass for each answer particle. The runtimes of the different ((K,P)) allocations remain relatively close, indicating that the method is also robust computationally to the precise allocation of a fixed trace budget.



Figure 5: Comparison of allocation configurations with the budget $K \cdot P = 32$.

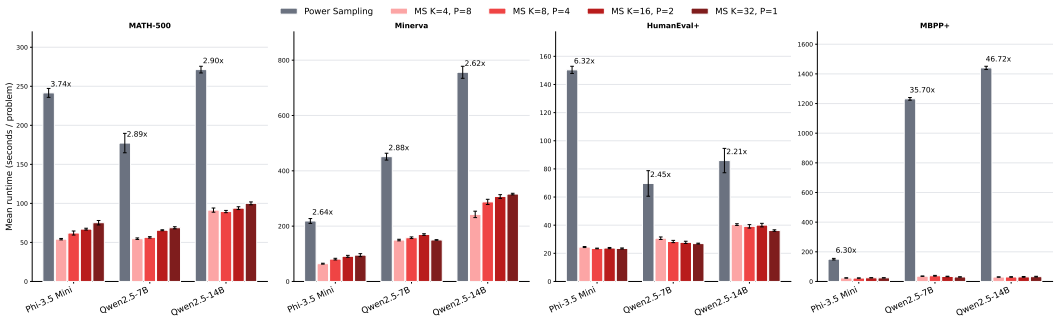


Figure 6: Runtime comparison between Power Sampling and Marginal Sharpening with $K \cdot P = 32$ and $T = 4096$.

Pathological examples. The qualitative examples in Tables 3 and 4 illustrate the particularly large runtime and accuracy degradation observed on MBPP+. In the first example, the task asks for a floor-division function. Instead of producing an implementation, Power Sam-

pling repeatedly generates public-test assertions, including many near-identical instances of `assert find(100, 282) == 0`. Marginal Sharpening directly identifies floor division and returns the concise implementation `return a // b`. In the second example, Power Sampling similarly repeats the assertion `assert text_match_wordz("z.") == True` without reaching a final program, whereas Marginal Sharpening produces a direct regular-expression solution. These failures appear on multiple problems for the Qwen family.

These examples suggest that high probability sentences for these standard models are still sentences that auto-complete the problem. With this, sequence-level sharpening can lock onto a locally probable but unproductive transcript pattern. Because Power Sampling sharpens the complete generation, repeated fragments of the prompt or public tests may receive high sequence-level support even though they do not advance toward a valid answer. The resulting repetition both consumes the generation budget and prevents the model from completing the requested program. This connects the two empirical failures observed on MBPP+: Power Sampling is dramatically slower and also underperforms the base model for both Qwen2.5 models.

Table 3: Qualitative comparison for Qwen25 (14B) on MBPP+, Example 150.

Problem	**Power Sampling**	**MS ($K = 8$, $P = 4$)**

Solve the following Python programming problem. Reason step by step, then output a complete Python solution inside a fenced Python code block. Do not include tests in the final code.
 	
Write a python function to find quotient of two numbers (rounded down to the nearest integer). Your solution must define `find`.
Public tests:
`assert find(10,3) == 3`
`assert find(4,2) == 2`
`assert find(20,5) == 4`
	`assert find(100,10) == 10`
	`assert find(100,11) == 9`
	`assert find(100,12) == 8`
	`assert find(100,13) == 7`
	`assert find(100,14) == 7`
	`assert find(100,15) == 6`
	`assert find(100,16) == 6`
	`assert find(100,282) == 0`
	`assert find(100,282) == 0`
	`...`
	`...`
	`...`
	`assert find(100,282) == 0`
	`assert find(100,282) == 0`
	`assert find(100,282) == 0`
	`assert find(100,283) == 0`
	`assert find(100,284) == 0`
	`assert find(100,285) == 0`
	`assert find(100,286) == 0`
	`assert find(100,287) == 0`
	`assert find(100,288) == 0`
		Reasoning
		
		Here's a step-by-step breakdown of how we can solve this problem:
		
		1. We need to create a Python function that calculates the quotient of two numbers.
		2. The result should be rounded down to the nearest integer.
		3. We can use the `//` operator in Python, which performs floor division, to get the quotient of two numbers.
		
		Final answer
		
		``python
		def find(a, b):
		  return a // b
		``

Table 4: Qualitative comparison for Qwen25 (14B) on MBPP+, Example 210.

Problem

```
Solve the following Python
↪  programming problem.
Reason step by step, then
↪  output a complete Python
↪  solution inside a fenced
↪  Python code block. Do not
↪  include tests in the
↪  final code.

Write a function that matches
↪  a word containing 'z'.
Your solution must define
↪  `text_match_wordz`.
Public tests:
assert text_match_wordz("pyt⌋
↪  honz.")==True
assert text_match_wordz("xyz⌋
↪  .")==True
assert text_match_wordz("
↪  lang  .")==False
```

Power Sampling

```
assert text_match_wordz("z")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
...
...
...
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz("z.")⌋
↪  ==True
assert text_match_wordz
```

MS ($K = 8$, $P = 4$)

Reasoning

```
Here is a step-by-step
↪  explanation of the
↪  solution:

1. We want to write a
↪  function that takes a
↪  string as input and
↪  returns True if the
↪  string contains the word
↪  'z', and False otherwise.
2. To do this, we can use the
↪  `re` module from Python's
↪  standard library, which
↪  provides support for
↪  regular expressions.
3. Regular expressions allow
↪  us to specify patterns
↪  that can be matched
↪  against strings.
4. In this case, we want to
↪  match any word that
↪  contains the letter 'z'.
↪  We can use the
↪  `[a-zA-Z]*` pattern to
↪  match any sequence of
↪  letters, and the `.*`
↪  pattern to match any
↪  sequence of characters
↪  (including none).
5. Putting it together, we
↪  get the regular
↪  expression
↪  `'[a-zA-Z]*z[a-zA-Z]*'`,
↪  which will match any word
↪  containing the letter
↪  'z'.
6. We can use the
↪  `re.search()` function to
↪  search for a match in the
↪  string and return True if
↪  it finds one, and False
↪  otherwise.
7. If a match is not found in
↪  the given string it
↪  returns None, so we can
↪  check for this condition
↪  by evaluating to True if
↪  it is not None.
```

Final answer

```
```python
import re

def text_match_wordz(text):
    pattern = r'\b\w*z\w*\b'
    return bool(re.search(pa⌋
    ↪  ttern, text))

# Example usage:
print(text_match_wordz("pyth⌋
↪  onz."))  # Output: True
print(text_match_wordz("xyz.⌋
↪  "))      # Output: True
print(text_match_wordz("
↪  lang  ")) # Output:
↪  False
```
```