ANNEALED PREDICTOR-CORRECTOR SAMPLING FOR INFERENCE-TIME SCALING OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

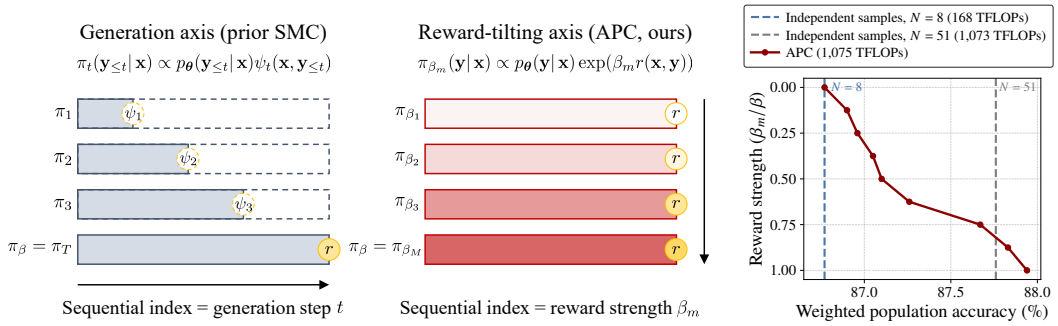
Autoregressive large language models (LLMs) have demonstrated strong performance on various reasoning tasks, and when a reward model is available, inference-time scaling can steer them further toward high-reward responses. Best-of-N sampling draws responses independently from the base model and selects the highest-reward response, but reward information from one candidate is not used to guide the generation of others. Recent Sequential Monte Carlo (SMC) approaches instead guide generation along the token axis, requiring intermediate estimates of partial-response quality that may prematurely discard promising trajectories. We propose the Annealed Predictor-Corrector (APC) sampler, which performs SMC along the reward-tilting axis by constructing a sequence of intermediate distributions with gradually increasing reward strength. APC maintains complete responses as particles and alternates between predictor and corrector steps. The predictor progressively favors higher-reward responses through reweighting, while the corrector refines particles by regenerating response suffixes from the base model and selecting among candidates using a transition that preserves the current intermediate target distribution. Applying this correction along the annealing path enables responses to be progressively refined before the target distribution becomes strongly concentrated. We further introduce an adaptive stopping rule that avoids unnecessary computation once a response stabilizes. Across reasoning benchmarks, APC provides a favorable accuracy-compute trade-off compared with independent sampling and token-axis inference-time scaling baselines.

1 INTRODUCTION

Autoregressive large language models (LLMs) have achieved strong performance across a wide range of reasoning tasks, including mathematical reasoning, code generation, and scientific problem solving (Lewkowycz et al., 2022; Chen et al., 2021; Achiam et al., 2023), largely driven by increases in model size and training data (Kaplan et al., 2020; Hoffmann et al., 2022). Alongside these advances, inference-time scaling has emerged as an important approach for improving response quality by allocating additional computation at test time without further training (Snell et al., 2025; Wu et al., 2025; Muennighoff et al., 2025). When a task provides a reward model or an external evaluator, an important challenge is how to efficiently steer the base model toward high-reward responses. A simple and widely used approach is Best-of-N sampling, which independently generates multiple responses and selects the one with the highest reward. Despite its simplicity, Best-of-N remains a strong inference-time scaling baseline (Wu et al., 2025; Zhang et al., 2025a). However, because candidates are generated independently from the same base distribution, reward assigned to one candidate does not affect how the others are generated. Consequently, scaling inference compute relies on drawing more independent samples, which can become computationally inefficient as the sampling budget increases (Snell et al., 2025).

Recent work has applied particle-based Monte Carlo methods, such as Sequential Monte Carlo (SMC) (Del Moral et al., 2006), to improve inference-time generation in LLMs (Zhao et al., 2024; Feng et al., 2025; Puri et al., 2025). These approaches generally define the sequential index along the autoregressive token or reasoning-step axis and maintain a population of partial trajectories. Intermediate evaluators, such as process reward models (Lightman et al., 2024; Wang et al., 2024), assign larger weights to promising trajectories and guide resampling as generation proceeds. However, intermediate scores can be imperfect proxies for the completion quality (Puri et al., 2025; Zhang et al.,

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



(a) Generation vs. reward-tilting axis (b) Population accuracy along tilt

Figure 1: Motivation for Annealed Predictor-Corrector (APC). (a) Prior SMC methods (Zhao et al., 2024; Puri et al., 2025) progress along the generation axis and operate on partial responses, whereas APC progresses along the reward-tilting axis while maintaining complete responses. (b) Weighted population accuracy of APC along the reward-tilting path on MATH-500, compared with self-normalized importance sampling (SNIS) on independent samples at the same initial sample count ($N = 8$) and at matched compute ($N = 51$). All results are averaged over three random seeds.

2025b; Golowich et al., 2026), potentially causing promising trajectories to be discarded prematurely. Moreover, because the sequence of intermediate targets is defined over progressively extended prefixes, discarded prefixes are typically not revisited. These limitations motivate reconsidering whether the sequential index of SMC should be tied to the generation trajectory itself.

The sequential index in SMC can parameterize a sequence of intermediate target distributions rather than the progression of a generative process (Del Moral et al., 2006). Motivated by this perspective, we construct the SMC sequence along the reward strength rather than the token axis. By maintaining complete responses as particles throughout sampling, we can evaluate rewards directly on complete responses while gradually shifting the population toward higher-reward regions. This population-based refinement is related to evolutionary search (Romera-Paredes et al., 2024; Novikov et al., 2025; Jiang et al., 2026), where candidate solutions are repeatedly evaluated and refined using evaluator feedback. Figure 1 illustrates this motivation by contrasting generation-axis SMC with our reward-tilting construction and showing population accuracy along the reward-tilting path.

Building on this formulation, we propose the Annealed Predictor-Corrector (APC) sampler. APC starts from the base language model (LM) distribution and follows a sequence of intermediate reward-tilted distributions with progressively increasing reward strength. At each stage, the predictor reweights particles using incremental importance weights, while the corrector further explores the current intermediate distribution by retaining a response prefix and regenerating its suffix from the base LM. Because the suffix proposal is given by the autoregressive conditional of the same base model, the corresponding proposal terms cancel, yielding a tractable invariant corrector without an additional proposal model. Importantly, unlike suffix-regeneration methods that operate directly at a fixed final reward strength, APC applies the corrector throughout the annealing path, allowing particles to be refined before the target becomes strongly concentrated on a small set of high-reward responses. Unlike evolutionary search, this refinement is defined by explicit intermediate target distributions and target-preserving transitions. Because particles remain complete responses throughout the path, we can monitor stage-wise sampled outputs and terminate the remaining annealing path once the extracted answer stabilizes. Across reasoning benchmarks, APC achieves a favorable trade-off between performance and computation compared with independent-sampling, search-based, and generation-axis particle methods.

2 Related Work

Population-Based Inference for LLMs A common approach to inference-time scaling is to generate multiple candidate responses and then aggregate or select among them. Self-consistency (Wang et al., 2023) aggregates final answers through majority voting, whereas Best-of-N selects the response with the highest score under a reward model or external evaluator. Despite its simplicity, Best-of-N

remains a strong and widely used baseline for inference-time scaling (Wu et al., 2025; Zhang et al., 2025a), and several works have studied its scaling behavior (Gao et al., 2023; Huang et al., 2025; Sriraman & Block, 2026) or combined reward-based selection with aggregation (Rakhsha et al., 2025; Di et al., 2026). However, these methods generate candidate responses independently before selection or aggregation.

Particle-based and search-based approaches instead introduce interactions among candidates during generation. Twisted SMC (Zhao et al., 2024) performs probabilistic inference over LM sequences using twist functions that estimate future sequence-level potentials, while verifier-guided search and particle-filtering methods (Snell et al., 2025; Beeching et al., 2024; Puri et al., 2025) repeatedly evaluate and select among partial responses. Therefore, these approaches rely on intermediate estimates of partial response quality (Lightman et al., 2024; Wang et al., 2024). Other sampling-based methods instead operate directly on complete sequences using suffix-regeneration MCMC (Faria et al., 2024; Faria & Smith, 2025), typically applying Markov transitions at a fixed reward-tilted target. In contrast, APC couples suffix regeneration with population-based SMC over a sequence of intermediate reward strengths, using predictor reweighting and resampling to move the population along the annealing path.

LLM-Guided Evolutionary Search Another closely related line of work uses LLMs within evolutionary search frameworks. Methods such as FunSearch (Romera-Paredes et al., 2024), ReEvo (Ye et al., 2024), and AlphaEvolve (Novikov et al., 2025) maintain or iteratively refine candidate solutions through evolutionary search using evaluator feedback to guide subsequent generations. These approaches demonstrate the effectiveness of population-level exploration and iterative refinement for program synthesis, combinatorial optimization, and scientific discovery (Romera-Paredes et al., 2024; Ye et al., 2024; Novikov et al., 2025). Recent work has begun to formulate program evolution from a probabilistic perspective using reward-tilted distributions and SMC (Jiang et al., 2026). APC is related to this population-based refinement perspective but differs in defining each stage through an explicit reward-tilted target distribution rather than only evaluator-guided selection and mutation. The suffix-regeneration corrector in APC is constructed to preserve each intermediate target, while predictor reweighting and resampling provide the population-level transition along the annealing path. We further evaluate this setting through direct comparisons with alternative inference-time samplers under matched computational budgets.

3 Preliminaries

3.1 Problem Formulation: Reward-Tilted LLM Inference

We denote an input prompt by $\mathbf{x} \in \mathcal{X}$ and its response by $\mathbf{y} = (y_1, \cdots, y_T) \in \mathcal{Y}$. An autoregressive LLM $p_{\boldsymbol{\theta}}$ factorizes the response distribution into next-token probabilities as $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(y_t|\mathbf{x}, \mathbf{y}_{<t})$. We assume a reward function $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assigns a scalar reward to each prompt and complete response. Our goal is to generate high-reward responses while remaining close to the distribution induced by the base LLM. To this end, we consider the reward-tilted distribution (Rafailov et al., 2023; Geuter et al., 2026),

$$\pi_\beta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\beta(\mathbf{x})} p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \exp\big(\beta r(\mathbf{x}, \mathbf{y})\big), \tag{1}$$

where $\beta \geq 0$ controls the reward strength, and $Z_\beta(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \exp\big(\beta r(\mathbf{x}, \mathbf{y})\big)$ is the corresponding normalizing constant. When $\beta = 0$, π_β reduces to the base model distribution; larger values of β place more probability mass on higher-reward responses.

For $\beta > 0$, this distribution can equivalently be expressed as the solution to KL-regularized reward maximization, a formulation widely studied in alignment (Rafailov et al., 2023; Khalaf et al., 2025):

$$\pi_\beta = \arg\max_q \left\{ \mathbb{E}_{\mathbf{y} \sim q(\cdot|\mathbf{x})}[r(\mathbf{x}, \mathbf{y})] - \frac{1}{\beta} D_{\mathrm{KL}}\left(q(\cdot|\mathbf{x}) || p_{\boldsymbol{\theta}}(\cdot|\mathbf{x})\right) \right\}. \tag{2}$$

Therefore, π_β defines a target distribution that favors high-reward responses while regularizing deviation from the base model distribution. Our objective is to efficiently generate samples from π_β at inference time without further updating the LLM parameters $\boldsymbol{\theta}$.

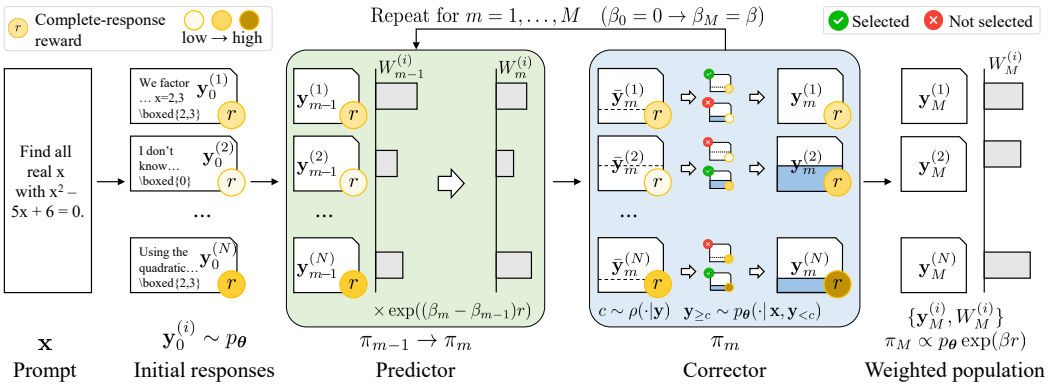


Figure 2: Overview of the Annealed Predictor-Corrector (APC) sampler. APC gradually moves complete-response particles from the base LM $\pi_0 = p_{\boldsymbol{\theta}}$ to the reward-tilted target $\pi_M = \pi_\beta$.

3.2 Sequential Monte Carlo Samplers

Sequential Monte Carlo (SMC) approximates a sequence of target distributions using a population of weighted particles (Del Moral et al., 2006). We consider a sequence of target distributions $\{\pi_m\}_{m=0}^M$ on a common state space $\mathcal{Y}$, where π_0 is an easy-to-sample initial distribution and π_M is the final target. Each target is written as $\pi_m(\mathbf{y}) = \gamma_m(\mathbf{y})/Z_m$, where γ_m denotes an unnormalized target density and Z_m is its normalizing constant, which is generally intractable.

At stage m, SMC approximates π_m using N weighted particles $\{(\mathbf{y}_m^{(i)}, W_m^{(i)})\}_{i=1}^N$ and updates their weights W_m as the target changes from π_{m-1} to π_m. When the forward kernel leaves π_m invariant and the backward kernel is chosen as its π_m-reversal, the incremental importance weight simplifies to $w_m(\mathbf{y}) \propto \gamma_m(\mathbf{y})/\gamma_{m-1}(\mathbf{y})$, depending only on consecutive unnormalized target densities (Del Moral et al., 2006). As the importance weights concentrate on a small number of particles, the particle approximation may suffer from weight degeneracy. A standard diagnostic is the effective sample size (ESS), defined as $\mathrm{ESS}_m = \left(\sum_{i=1}^N \left(W_m^{(i)}\right)^2\right)^{-1}$, which ranges from 1 to N (Kong et al., 1994). In practice, resampling is commonly triggered when the ESS falls below a predefined threshold, replicating high-weight particles while discarding low-weight particles (Del Moral et al., 2006). In SMC samplers, resampling can be followed by a Markov move that leaves the current intermediate target invariant, enabling particles to further explore the current distribution and recover diversity lost during resampling (Gilks & Berzuini, 2001; Del Moral et al., 2006). This invariance distinguishes the move from an arbitrary mutation, since repeated application does not change the target distribution.

Prior particle-based approaches to LLM inference typically define the sequential index along the token or reasoning-step axis (Zhao et al., 2024; Feng et al., 2025; Puri et al., 2025). In this work, we instead use the SMC index m to parameterize a sequence of intermediate distributions connecting the easy-to-sample base LLM distribution $p_{\boldsymbol{\theta}}$ to the reward-tilted target π_β. Specifically, the sequential index is defined by the reward strength rather than token position.

4 Method

We introduce the Annealed Predictor-Corrector (APC) sampler, an inference-time scaling method for sampling from reward-tilted distributions of autoregressive LLMs. Instead of directly targeting a strongly reward-tilted distribution, APC gradually moves a particle population toward the final target by increasing the reward strength. This allows the population to concentrate on higher-reward regions while refining promising responses. To this end, we formulate SMC along the reward-tilting axis, where the sequential index corresponds to reward strength rather than token position.

Figure 2 provides an overview of APC. At each intermediate target, the predictor progressively emphasizes higher-reward responses through incremental importance weighting. Then, the corrector explores and refines the particle population through suffix regeneration while preserving the current intermediate target. Importantly, APC only requires rewards on complete responses and does not require scores for partial responses or an additional proposal model.

4.1 Annealed Predictor-Corrector Sampling

Annealed Target Sequence We construct a sequence of intermediate distributions with progressively increasing reward strength, rather than moving directly from the base LLM distribution to the final reward-tilted target distribution. Specifically, we define an annealing schedule $\{\beta_m\}$ such that $0 = \beta_0 < \beta_1 < \cdots < \beta_M = \beta$, and we define the intermediate target π_m at stage m and its corresponding unnormalized density γ_m as follows:

$$\pi_m(\mathbf{y}|\mathbf{x}) \coloneqq \pi_{\beta_m}(\mathbf{y}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \exp(\beta_m r(\mathbf{x}, \mathbf{y}))/Z_m(\mathbf{x}), \tag{3}$$

$$\gamma_m(\mathbf{y}|\mathbf{x}) \coloneqq p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \exp(\beta_m r(\mathbf{x}, \mathbf{y})). \tag{4}$$

Here, $\pi_0(\mathbf{y}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ is the initial distribution given by the base LLM, and $\pi_M(\mathbf{y}|\mathbf{x}) = \pi_\beta(\mathbf{y}|\mathbf{x})$ is the final reward-tilted target distribution. APC maintains a population of complete-response particles and sequentially moves this population through the intermediate target sequence $\{\pi_m\}_{m=0}^M$.

Predictor At stage m, the predictor updates the particle weights as the target changes from π_{m-1} to π_m. Since these distributions share the same base LLM distribution $p_{\boldsymbol{\theta}}$ and differ only in the reward strength, the incremental importance weight $w_m(\mathbf{y})$ can be expressed as follows:

$$w_m(\mathbf{y}) \coloneqq \frac{\gamma_m(\mathbf{y}|\mathbf{x})}{\gamma_{m-1}(\mathbf{y}|\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \exp(\beta_m r(\mathbf{x}, \mathbf{y}))}{p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \exp(\beta_{m-1} r(\mathbf{x}, \mathbf{y}))} = \exp\big((\beta_m - \beta_{m-1}) r(\mathbf{x}, \mathbf{y})\big). \tag{5}$$

Thus, the predictor updates particle weights solely based on complete-response rewards. Higher-reward responses receive larger relative weights as the reward strength increases, gradually shifting the population toward higher-reward regions.

Let $W_{m-1}^{(i)}$ denote the normalized weight of particle $\mathbf{y}_{m-1}^{(i)}$ at stage $m-1$. At stage m,

$$W_m^{(i)} = \tilde{W}_m^{(i)} \Big/ \textstyle\sum_{j=1}^N \tilde{W}_m^{(j)}, \text{ where } \tilde{W}_m^{(i)} = W_{m-1}^{(i)} \exp\big((\beta_m - \beta_{m-1}) r(\mathbf{x}, \mathbf{y}_{m-1}^{(i)})\big). \tag{6}$$

As in standard SMC methods (Del Moral et al., 2006), we optionally perform resampling when the particle weights become overly concentrated. As described in Section 3.2, we compute the effective sample size, ESS_m, and trigger resampling when it falls below the predefined threshold τ_{ESS}: $\mathrm{ESS}_m \coloneqq \big(\sum_{i=1}^N \big(W_m^{(i)}\big)^2\big)^{-1} < \tau_{\mathrm{ESS}} N$. When this condition holds, we sample the indices

$$A_m^{(i)} \sim \mathrm{Cat}(W_m^{(1)}, \cdots, W_m^{(N)}), \qquad \bar{\mathbf{y}}_m^{(i)} = \mathbf{y}_{m-1}^{(A_m^{(i)})}, \tag{7}$$

and reset all particle weights to $1/N$. When resampling is not triggered, we set $\bar{\mathbf{y}}_m^{(i)} = \mathbf{y}_{m-1}^{(i)}$ and retain the weights $W_m^{(i)}$. We use $\bar{\mathbf{y}}_m^{(i)}$ to denote the particle passed from the predictor to the corrector.

Corrector The predictor assigns larger weights to particles with higher rewards as the reward strength increases, but reweighting and resampling alone do not generate new responses. Therefore, we introduce a corrector that explores the response space further while preserving the current target π_m. This predictor-corrector approach originates from numerical continuation methods (Allgower & Georg, 2012) and has also been adopted in sampling for diffusion models (Song et al., 2021).

For a particle $\bar{\mathbf{y}}_m^{(i)}$ entering the corrector, we write $\mathbf{y} = \bar{\mathbf{y}}_m^{(i)}$ and suppress the particle index i. We introduce a cut-selection distribution $\rho(c|\mathbf{y})$ and sample $c \sim \rho(c|\mathbf{y})$, where c is the first position of the regenerated suffix and $\mathbf{y} = (\mathbf{y}_{<c}, \mathbf{y}_{\geq c})$. We augment the intermediate target with c as an auxiliary variable such that marginalizing out c recovers π_m:

$$\tilde{\pi}_m(\mathbf{y}, c|\mathbf{x}) \coloneqq \pi_m(\mathbf{y}|\mathbf{x}) \rho(c|\mathbf{y}). \tag{8}$$

Given the fixed prefix $\mathbf{y}_{<c}$, we generate $K-1$ new suffixes independently from the base LLM. Together with the current response, they form K complete-response candidates:

$$\mathbf{y}_{\geq c}^{(k)} \sim p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}, \mathbf{y}_{<c}), \quad k = 1, \cdots, K-1, \tag{9}$$

$$\mathbf{y}^{(0)} = \mathbf{y}, \qquad \mathbf{y}^{(k)} = (\mathbf{y}_{<c}, \mathbf{y}_{\geq c}^{(k)}), \quad k = 1, \cdots, K-1. \tag{10}$$

Given the shared prefix and cut position, the augmented target for the suffix is given by

$$\tilde{\pi}_m(\mathbf{y}_{\geq c}|\mathbf{x}, \mathbf{y}_{<c}, c) \propto p_{\boldsymbol{\theta}}(\mathbf{y}_{\geq c}|\mathbf{x}, \mathbf{y}_{<c}) \exp(\beta_m r(\mathbf{x}, \mathbf{y})) \rho(c|\mathbf{y}). \tag{11}$$

Because the suffix proposal in Eq. (9) uses the same base distribution, the corresponding terms cancel in the importance ratio. Therefore, the resulting selection weight for candidate $\mathbf{y}^{(k)}$ is

$$\omega_m^{(k)} \propto \exp(\beta_m r(\mathbf{x}, \mathbf{y}^{(k)}))\rho(c|\mathbf{y}^{(k)}). \tag{12}$$

Although all candidates in the pool share the same cut position c, $\rho(\cdot|\mathbf{y}^{(k)})$ is generally candidate-dependent since the regenerated suffix changes the relative position of c from the end of the response.

Then, we sample one candidate according to the normalized weights:

$$J \sim \mathrm{Cat}\big(\omega_m^{(0)} / \textstyle\sum_{l=0}^{K-1} \omega_m^{(l)}, \cdots, \omega_m^{(K-1)} / \textstyle\sum_{l=0}^{K-1} \omega_m^{(l)}\big), \quad \mathbf{y}' = \mathbf{y}^{(J)}. \tag{13}$$

For particle i, we set the corrector output to $\mathbf{y}_m^{(i)} = \mathbf{y}'$.

This corrector can be viewed as an iterated sampling-importance resampling (i-SIR) update of the current response and $K-1$ suffix proposals (Andrieu et al., 2010; Grenioux et al., 2023). Due to the autoregressive factorization of the base LLM, the proposal distribution cancels with the corresponding factor in the augmented target, leaving weights that depend only on complete-response reward and cut-selection probability. As shown in Theorem 1, the corrector preserves the intermediate target π_m.

Theorem 1. *Fix a prompt $\mathbf{x}$ and an annealing stage m. Suppose $\rho(\cdot|\mathbf{y})$ is a distribution over valid cut positions and that a sampled cut remains valid for all suffix proposals constructed from the shared prefix. Let $\mathcal{K}_m(\mathbf{y}' \mid \mathbf{y}; \mathbf{x})$ denote the corrector transition that samples $c \sim \rho(\cdot \mid \mathbf{y})$, generates $K-1$ suffix proposals according to Eq. (9), and selects a candidate using the weights in Eq. (12). Then, $\pi_m(\cdot \mid \mathbf{x})$ is invariant under $\mathcal{K}_m$, i.e., for every $\mathbf{y}' \in \mathcal{Y}$,*

$$\textstyle\sum_{\mathbf{y} \in \mathcal{Y}} \pi_m(\mathbf{y} \mid \mathbf{x}) \mathcal{K}_m(\mathbf{y}' \mid \mathbf{y}; \mathbf{x}) = \pi_m(\mathbf{y}' \mid \mathbf{x}). \tag{14}$$

4.2 Practical Considerations

Cut Position The cut position determines how much of the current response the corrector regenerates. We define $\rho(c|\mathbf{y})$ using a truncated geometric distribution over the number of regenerated trailing units, such as reasoning steps, with a scale hyperparameter λ, where larger values of λ induce more global updates. Additionally, we assign a small point mass to full-response regeneration, which allows the corrector to escape prefixes inherited from the current response.

Adaptive Stopping Since APC maintains complete responses throughout the annealing path, we can obtain a task-level output, such as the final answer in mathematical reasoning, after each annealing stage and monitor its stability across stages. After completing stage m, we sample $I_m \sim \mathrm{Cat}(W_m^{(1)}, ..., W_m^{(N)})$ from the current weighted population and extract the task-specific output a_m from $\mathbf{y}_m^{(I_m)}$. If the outputs from κ consecutive stages are identical and non-empty, i.e., $a_{m-\kappa+1} = \cdots = a_m$, we terminate the run and return $\mathbf{y}_m^{(I_m)}$. The remaining annealing stages and additional corrector updates at the final target are skipped. Therefore, adaptive stopping returns a response from the current stage-m particle approximation rather than continuing to the final target π_β, trading the remainder of the annealing path for reduced inference computation.

Overall Procedure APC initializes the particles by sampling complete responses from the base LLM. At each annealing stage, the predictor reweights the particles toward the next intermediate target and optionally resamples them based on the ESS. Subsequently, the corrector refines the resulting population. After each stage, APC samples a task-level output from the weighted population and applies the adaptive stopping criterion. If the criterion is satisfied, the entire run terminates; otherwise, the predictor-corrector updates continue along the reward-strength schedule until the final target π_β is reached. The complete APC procedure is summarized in Algorithm 1 of Appendix B.

5 Experiments

5.1 Experimental Settings

Setup We evaluate mathematical reasoning on MATH-500 (Lightman et al., 2024) and AMC23 (Yang et al., 2024) using Qwen2.5-Math-7B-Instruct (Yang et al., 2024) with Qwen2.5-Math-PRM-7B (Zhang et al., 2025b), and instruction following on IFEval (Zhou et al., 2023) using

Table 1: Accuracy (%) and inference compute (TFLOPs) on MATH-500 and AMC23 using Qwen2.5-Math-7B-Instruct with Qwen2.5-Math-PRM-7B as the process reward model. For each method, we report the highest accuracy attained over the evaluated compute budgets, together with the corresponding total forward-pass compute in TFLOPs. All results are averaged over three random seeds. The best and second-best accuracies are shown in **bold** and underline, respectively.

	MATH-500		AMC23	
Method	Accuracy ↑	TFLOPs ↓	Accuracy ↑	TFLOPs ↓
Single Sample	80.2	10	54.2	15
Greedy	82.6	10	62.5	15
Selection over N i.i.d. samples				
Best-of-N	87.9	1,347	64.2	1,956
SNIS	87.8	1,347	63.3	1,956
Self-Consistency (Wang et al., 2023)	88.0	644	**73.3**	941
Weighted Majority Voting (Wang et al., 2024)	88.3	1,347	64.2	1,956
Search over partial responses				
Beam Search (Snell et al., 2025)	85.0	1,642	62.5	2,984
DVTS (Beeching et al., 2024)	88.1	1,834	62.5	3,201
Reward-guided samplers				
QAlign (Faria & Smith, 2025)	87.8	956	67.5	1,410
Step-Level SMC (Zhao et al., 2024)	86.7	911	62.5	1,626
Particle Filtering (Puri et al., 2025)	<u>88.6</u>	1,804	70.8	3,703
APC (ours)	**88.8**	682	<u>72.5</u>	253

Llama-3.1-Tulu-3-8B-SFT (Lambert et al., 2025) with Llama-3.1-Tulu-3-8B-RM, following Faria & Smith (2025). We set $\beta = 100$ for mathematical reasoning and $\beta = 20$ for IFEval. For MATH-500 and AMC23, APC uses $N = 8$ and $N = 4$ particles, respectively, with $K = 2$, $M = 8$, and reasoning-step cuts; for IFEval, we use $N = 4$, $K = 2$, $M = 4$, and token-level cuts. All experiments use a linear annealing schedule with $\tau_{\text{ESS}} = 0.5$, and adaptive stopping is applied only to mathematical reasoning. Inference cost is measured as total forward-pass FLOPs per prompt, including both generation and reward-model evaluation. Further details are provided in Appendix C.

Baselines We report Single Sample and Greedy as basic decoding baselines, using one sampled response and greedy decoding, respectively. We compare APC with independent-sampling methods, including Best-of-N, self-normalized importance sampling (SNIS), self-consistency (Wang et al., 2023), and weighted majority voting (Wang et al., 2024); search methods over partial responses, including PRM-guided beam search (Snell et al., 2025) and DVTS (Beeching et al., 2024); and reward-guided samplers, including QAlign (Faria & Smith, 2025), step-level SMC (Zhao et al., 2024), and particle filtering (Puri et al., 2025). On IFEval, we include only methods compatible with complete-response rewards and not requiring a discrete extracted answer.

5.2 Experimental Results

Mathematical Reasoning Figure 3 shows the accuracy-compute trade-off on MATH-500 across hyperparameters and inference budgets. APC lies on the favorable accuracy-compute frontier over a broad range of compute budgets by using reward feedback to refine the existing particle population rather than allocating compute solely to independent sampling. Self-consistency remains competitive at lower budgets, while APC becomes more favorable as additional compute enables repeated reward-guided refinement. Since APC, like the baselines, generates responses from the same base LLM, its advantage primarily comes from a more efficient use of inference compute rather than access to a stronger proposal model. Accordingly, the gap to strong baselines narrows at larger budgets.

Table 1 shows that APC achieves 88.8% accuracy on MATH-500 with 682 TFLOPs, compared to 88.6% at 1,804 TFLOPs for particle filtering and 88.3% at 1,347 TFLOPs for weighted majority voting. APC also outperforms QAlign, a suffix-regeneration baseline, while using less computation, showing that the benefit is not simply due to suffix regeneration itself. On AMC23, APC reaches 72.5% with 253 TFLOPs, while self-consistency reaches a slightly higher 73.3% but requires 941

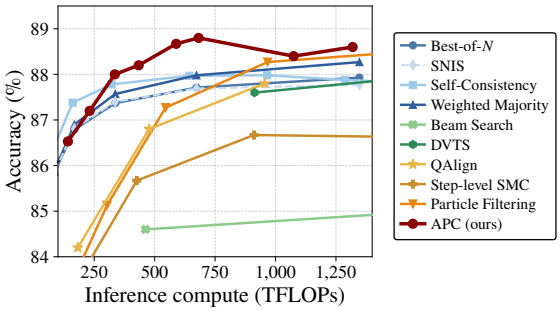


Figure 3: Accuracy-compute trade-off on MATH-500.

Table 2: Accuracy (%) and inference compute (TFLOPs) on IFEval using Llama-3.1-Tulu-3-8B-SFT with Llama-3.1-Tulu-3-8B-RM.

Method	Accuracy ↑	TFLOPs ↓
Single Sample	67.9	4
Greedy	70.8	4
Best-of-N (N=4)	73.3	36
Best-of-N (N=28)	76.0	252
Best-of-N (N=64)	76.2	577
QAlign	73.8	269
APC (ours)	**77.9**	253

Table 3: Ablation study of APC on MATH-500. Results are from a single seed. *# Corr.* denotes the total number of corrector updates, and *Acc.* denotes accuracy (%).

Configuration	# Corr.	Acc.	TFLOPs
No annealing	10	82.2	643
No corrector	0	87.4	166
APC (ours)	10	**88.8**	645

Table 4: Inference-compute breakdown on MATH-500 (three-seed average). *Gen./RM* are generation/reward-model TFLOPs, and *Seq.* denotes sequential calls.

		TFLOPs			
Method	Acc.	Gen.	RM	Total	Seq.
Best-of-N	87.9	643	704	1,347	1.0
QAlign	87.8	337	619	956	7.0
Particle Filtering	88.6	259	1,545	1,804	9.1
APC (ours)	88.8	269	413	682	6.4

TFLOPs. Overall, APC achieves competitive or higher accuracy with substantially lower total inference compute. This efficiency comes at the cost of greater sequential depth than fully parallel independent sampling, which we analyze further in Section 5.3. A qualitative case study of APC on MATH-500 is provided in Table 9 and Figures 5 and 6 of Appendix D.

Instruction Following To evaluate APC beyond mathematical reasoning, particularly where intermediate rewards are difficult to evaluate, Table 2 compares APC with baselines that use complete-response rewards on IFEval to evaluate instruction-following ability. Since SNIS performs similarly to Best-of-N, we report only Best-of-N in the table. We also exclude answer-aggregation methods such as self-consistency, which are not directly applicable to free-form instruction following outputs. APC achieves 77.9% accuracy at 253 TFLOPs, outperforming both Best-of-N and QAlign at comparable computation. It also achieves higher accuracy than Best-of-N with a substantially larger sampling budget. These results show that APC remains effective beyond step-structured reasoning tasks using annealed refinement with a token-level corrector. In particular, APC requires only complete-response rewards, making it applicable when intermediate rewards are difficult to define or evaluate.

5.3 Analysis

Importance of Annealing Table 3 isolates the effect of annealing by comparing APC with *No annealing*, which applies the same 10 corrector updates directly at the final reward strength. Without annealing, accuracy drops to 82.2%, whereas distributing the same number of corrector updates along the annealing path reaches 88.8% with nearly identical computation. This shows that the gain does not simply come from additional regeneration steps, but from applying corrector updates at intermediate reward strengths before the target becomes strongly concentrated.

Role of the Corrector Table 3 also isolates the role of the corrector by comparing APC with *No corrector*, which leaves the particle population unchanged along the annealing path and thus reduces to reward-based reweighting of the initial responses. This configuration achieves 87.4% accuracy, comparable to SNIS with the same number of samples. Together, these ablations show that APC benefits from combining annealing with corrector updates, progressively refining responses as reward strength increases. We further analyze the cut-probability correction in Table 8 of Appendix D. Incorporating the correction term required by Theorem 1 yields a small accuracy improvement.

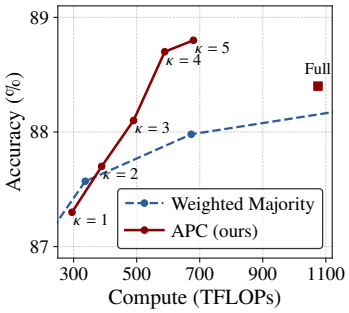


Figure 4: Sensitivity analysis of the stopping hyperparameter κ.

Table 5: Sensitivity analysis of APC to the reward strength β and particle population size N on MATH-500. All results are from the same single seed, with all remaining hyperparameters fixed.

Hyperparameter	Value	Accuracy	TFLOPs
Reward strength β	20	88.6	622
	50	88.8	632
	100	88.8	645
	300	88.8	675
Particle population size N	4	87.0	324
	8	88.8	645
	16	88.6	1,319

Computation Analysis To understand the source of computational efficiency, Table 4 decomposes the total forward-pass FLOPs on MATH-500 into response generation and reward-model evaluation. Although particle filtering and APC incur similar generation costs, particle filtering repeatedly evaluates partial trajectories throughout generation, resulting in substantially higher reward-model cost. In particular, reward evaluation accounts for approximately 86% of its total computation, compared to 61% for APC. Compared to Best-of-N, APC reduces both generation and reward-model evaluation costs by refining a smaller particle population instead of drawing many independent responses. This efficiency comes with increased sequential depth: Best-of-N requires a single generation round, whereas APC requires an average of 6.4 sequential calls. Therefore, APC trades parallelism for substantially lower total inference computation.

Effect of Adaptive Stopping Figure 4 shows the accuracy-compute trade-off as the stopping hyperparameter κ varies. Increasing κ improves accuracy from 87.3% to 88.8%, while running the full annealing path requires substantially more computation without further improving accuracy. Since APC maintains complete responses throughout the annealing path, we can sample a task-level output from the weighted population after each stage and directly monitor its answer. Adaptive stopping terminates the run when the sampled answer remains identical and non-empty for κ consecutive stages. This is related in spirit to self-consistency (Wang et al., 2023), but instead of measuring agreement among independently sampled responses, our criterion measures agreement across successive stage-wise outputs from the evolving APC population. Therefore, adaptive stopping avoids the remaining predictor-corrector updates once the output has stabilized, reducing inference computation.

Sensitivity Analysis Table 5 shows the sensitivity of APC to the reward strength β and particle population size N. APC is relatively insensitive to β, with little change in accuracy as β increases from 20 to 300. This robustness may be attributed to annealing, which gradually increases the reward strength rather than applying the final concentrated target from the beginning. Increasing N from 4 to 8 improves accuracy, suggesting that a sufficiently large particle population helps maintain diverse responses. However, further increasing N to 16 provides no additional improvement despite approximately doubling the inference computation, which indicates diminishing returns once the population sufficiently covers promising responses. Additional analyses of the cut scale λ and the allocation between particles N and corrector candidates K are in Tables 6 and 7 of Appendix D.

6 Conclusion

We propose the Annealed Predictor-Corrector (APC) sampler for reward-guided inference-time scaling of autoregressive LLMs. APC performs SMC along the reward-tilting axis, maintaining complete responses and progressively refining them through predictor and target-preserving corrector updates. Maintaining complete responses also enables adaptive stopping to reduce unnecessary computation. Experiments on mathematical reasoning demonstrate that APC offers a favorable accuracy-compute trade-off compared with independent sampling and existing stepwise SMC-based inference methods. Beyond mathematical reasoning, APC also improves instruction following using only complete-response rewards, demonstrating its utility when intermediate supervision is unavailable or difficult to obtain.

AI USE STATEMENT

In this work, we used generative AI tools for refining theoretical and conceptual frameworks; refining and reviewing mathematical claims; reviewing and modifying mathematical proofs; providing feedback on research methodology and experimental design; implementing methods; assisting with translation; and supporting the interpretation of experimental results. We have not used generative AI tools for generating synthetic data sets; providing critical ingredients for proving mathematical claims; proposing the core research hypotheses; cleaning or reformatting datasets; or supporting qualitative or thematic data analysis.

Additionally, we used generative AI tools for creating and editing software code, including code used to refine scientific figures; summarizing and analyzing existing literature; identifying relevant literature; and editing the manuscript to improve readability.

We have reviewed all AI-assisted work. Mathematical claims and derivations suggested or refined with AI tools were independently checked and revised by the authors. AI-assisted proofs were reviewed and revised by the authors. AI-assisted implementations were inspected and tested for correctness, and suggestions regarding methodology or experimental design were evaluated by the authors. AI-assisted analysis of experimental results was reviewed by the authors, who determined the final interpretation and conclusions. The scientific content and design of figures were determined by the authors, with AI tools used to assist with their implementation. Relevant literature identified with AI tools was checked against the original sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Eugene L Allgower and Kurt Georg. *Numerical continuation methods: an introduction*. Springer Science & Business Media, 2012.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3): 269–342, 2010.

Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models, 2024. URL https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute.

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.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 06 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x. URL https://doi.org/10.1111/j.1467-9868.2006.00553.x.

Qiwei Di, Kaixuan Ji, Xuheng Li, Heyang Zhao, and Quanquan Gu. Best-of-majority: Minimax-optimal strategy for pass@k inference scaling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a6CVQpjbXq.

Gonçalo Faria and Noah A Smith. Sample, don't search: Rethinking test-time alignment for language models. *arXiv preprint arXiv:2504.03790*, 2025.

Gonçalo Faria, Sweta Agrawal, António Farinhas, Ricardo Rei, José G. C. de Souza, and Andre Martins. QUEST: Quality-aware metropolis-hastings sampling for machine translation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=dLnduWGTB4.

Shengyu Feng, Xiang Kong, Shuang Ma, Aonan Zhang, Dong Yin, Chong Wang, Ruoming Pang, and Yiming Yang. Step-by-step reasoning for math problems via twisted sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ze4aPP0tIn.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pp. 10835–10866. PMLR, 2023.

Jonathan Geuter, Youssef Mroueh, and David Alvarez-Melis. Guided speculative inference for efficient test-time alignment of LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=miNzDqDENd.

Walter R. Gilks and Carlo Berzuini. Following a moving target-monte carlo inference for dynamic bayesian models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63 (1):127–146, 2001. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/2680638.

Noah Golowich, Fan Chen, Dhruv Rohatgi, Raghav Singhal, Carles Domingo-Enrich, Dylan J Foster, and Akshay Krishnamurthy. Reject, resample, repeat: Understanding parallel reasoning in language model inference. *arXiv preprint arXiv:2603.07887*, 2026.

Louis Grenioux, Alain Oliviero Durmus, Eric Moulines, and Marylou Gabrié. On sampling with approximate transport maps. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11698–11733. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/grenioux23a.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. 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=iBBcRUlOAPR.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is best-of-n the best of them? Coverage, scaling, and optimality in inference-time alignment. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 25075–25126. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/huang25c.html.

Jiachen Jiang, Huminhao Zhu, and Zhihui Zhu. Smcevolve: Principled scientific discovery via sequential monte carlo evolution. *arXiv preprint arXiv:2605.15308*, 2026.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio Calmon. Inference-time reward hacking in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=hSX7Dd8dxy.

Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994. ISSN 01621459, 1537274X. URL http://www.jstor.org/stable/2291224.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh 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 Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Isha Puri, Shivchander Sudalairaj, Guangxuan Xu, Abhishek Bhandwaldar, Kai Xu, and Akash Srivastava. Rollout roulette: A probabilistic inference approach to inference-time scaling of LLMs using particle-based monte carlo methods. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=qPQUrjiA0q.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Amin Rakhsha, Kanika Madan, Tianyu Zhang, Amir-massoud Farahmand, and Amir Khasahmadi. Majority of the bests: Improving best-of-n via bootstrapping. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 37844–37877. Curran Associates, Inc., 2025. doi: 10.52202/085713-1268. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/36556567e8437f137da23047309155dd-Paper-Conference.pdf.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Ved Sriraman and Adam Block. Revisiting the (sub)optimality of best-of-n for inference-time alignment. In Steve Hanneke and Tor Lattimore (eds.), *Proceedings of Thirty Ninth Conference on Learning Theory*, volume 336 of *Proceedings of Machine Learning Research*, pp. 5980–6028. PMLR, 29 Jun–03 Jul 2026. URL https://proceedings.mlr.press/v336/sriraman26a.html.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 55824–55845, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/8c3caae2f725c8e2a55ecd600563d172-Paper-Conference.pdf.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=Ccwp4tFEtE.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential Monte Carlo. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60704–60748. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhao24c.html.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Proof of Theorem 1

Theorem 1. *Fix a prompt* $\mathbf{x}$ *and an annealing stage* m*. Suppose* $\rho(\cdot|\mathbf{y})$ *is a distribution over valid cut positions and that a sampled cut remains valid for all suffix proposals constructed from the shared prefix. Let* $\mathcal{K}_m(\mathbf{y}' \mid \mathbf{y}; \mathbf{x})$ *denote the corrector transition that samples* $c \sim \rho(\cdot \mid \mathbf{y})$*, generates* $K-1$ *suffix proposals according to Eq. (9), and selects a candidate using the weights in Eq. (12). Then,* $\pi_m(\cdot \mid \mathbf{x})$ *is invariant under* $\mathcal{K}_m$*, i.e., for every* $\mathbf{y}' \in \mathcal{Y}$*,*

$$\textstyle\sum_{\mathbf{y}\in\mathcal{Y}} \pi_m(\mathbf{y} \mid \mathbf{x})\mathcal{K}_m(\mathbf{y}' \mid \mathbf{y}; \mathbf{x}) = \pi_m(\mathbf{y}' \mid \mathbf{x}). \tag{14}$$

Proof. First, consider the augmented target in Eq. (8). By definition, marginalizing out c gives

$$\sum_c \tilde{\pi}_m(\mathbf{y}, c|\mathbf{x}) = \pi_m(\mathbf{y}|\mathbf{x}) \sum_c \rho(c|\mathbf{y}) = \pi_m(\mathbf{y}|\mathbf{x}). \tag{15}$$

Therefore, it suffices to show that the corrector leaves the augmented distribution $\tilde{\pi}_m$ invariant, since the marginal distribution of $\mathbf{y}$ under $\tilde{\pi}_m$ is π_m.

Fix a cut c and a prefix $\mathbf{y}_{<c}$. For notational simplicity, define

$$q_c(\mathbf{y}_{\geq c}) \coloneqq p_{\boldsymbol{\theta}}(\mathbf{y}_{\geq c}|\mathbf{x}, \mathbf{y}_{<c}), \tag{16}$$

$$G_{m,c}(\mathbf{y}) \coloneqq \exp(\beta_m r(\mathbf{x}, \mathbf{y}))\rho(c|\mathbf{y}), \tag{17}$$

where $\mathbf{y} = (\mathbf{y}_{<c}, \mathbf{y}_{\geq c})$. By the assumption, the sampled cut c remains valid for every suffix proposal constructed from the fixed prefix $\mathbf{y}_{<c}$. Then, by Eq. (11), the conditional augmented target of the suffix can be written as

$$\mu_{m,c}(\mathbf{y}_{\geq c}) \coloneqq \tilde{\pi}_m(\mathbf{y}_{\geq c}|\mathbf{x}, \mathbf{y}_{<c}, c) = q_c(\mathbf{y}_{\geq c})G_{m,c}(\mathbf{y})/Z_{m,c}, \tag{18}$$

where $Z_{m,c}$ is the corresponding normalizing constant.

Let $\mathcal{P}_{m,c}(\mathbf{y}'|\mathbf{y}; \mathbf{x})$ denote the corrector transition with the cut c fixed. Consider two distinct complete responses $\mathbf{y} = (\mathbf{y}_{<c}, \mathbf{y}_{\geq c})$ and $\mathbf{y}' = (\mathbf{y}_{<c}, \mathbf{y}'_{\geq c})$, which share the same prefix. The transition from $\mathbf{y}_{\geq c}$ to $\mathbf{y}'_{\geq c}$ occurs when one of $K-1$ proposals generates $\mathbf{y}'_{\geq c}$ and that candidate is selected. Since the proposal positions are exchangeable, $\mathcal{P}_{m,c}$ can be derived as follows:

$$\begin{aligned}\mathcal{P}_{m,c}(\mathbf{y}'|\mathbf{y}; \mathbf{x}) &= (K-1) \times q_c(\mathbf{y}'_{\geq c}) \\ &\quad \times G_{m,c}(\mathbf{y}')\mathbb{E}_{\mathbf{y}^{(2)}_{\geq c},\ldots,\mathbf{y}^{(K-1)}_{\geq c}\sim q_c}\left[\frac{1}{G_{m,c}(\mathbf{y}) + G_{m,c}(\mathbf{y}') + \sum_{l=2}^{K-1} G_{m,c}(\mathbf{y}^{(l)})}\right],\end{aligned} \tag{19}$$

where $\mathbf{y}^{(l)} = (\mathbf{y}_{<c}, \mathbf{y}^{(l)}_{\geq c})$. Multiplying by $\mu_{m,c}(\mathbf{y}_{\geq c})$ gives

$$\begin{aligned}\mu_{m,c}(\mathbf{y}_{\geq c})\mathcal{P}_{m,c}(\mathbf{y}'|\mathbf{y}; \mathbf{x}) =& \frac{K-1}{Z_{m,c}} q_c(\mathbf{y}_{\geq c})G_{m,c}(\mathbf{y})q_c(\mathbf{y}'_{\geq c})G_{m,c}(\mathbf{y}') \\ &\times \mathbb{E}_{\mathbf{y}^{(2)}_{\geq c},\ldots\mathbf{y}^{(K-1)}_{\geq c}\sim q_c}\left[\frac{1}{G_{m,c}(\mathbf{y}) + G_{m,c}(\mathbf{y}') + \sum_{l=2}^{K-1} G_{m,c}(\mathbf{y}^{(l)})}\right].\end{aligned} \tag{20}$$

The right-hand side of Eq. (20) is symmetric with respect to $\mathbf{y}$ and $\mathbf{y}'$. Therefore,

$$\mu_{m,c}(\mathbf{y}_{\geq c})\mathcal{P}_{m,c}(\mathbf{y}'|\mathbf{y}; \mathbf{x}) = \mu_{m,c}(\mathbf{y}'_{\geq c})\mathcal{P}_{m,c}(\mathbf{y}|\mathbf{y}'; \mathbf{x}). \tag{21}$$

Thus, $\mathcal{P}_{m,c}$ satisfies detailed balance with respect to $\mu_{m,c}$ and hence leaves $\mu_{m,c}$ invariant.

For a fixed prefix and cut, the augmented target factorizes as

$$\tilde{\pi}_m(\mathbf{y}, c|\mathbf{x}) = \tilde{\pi}_m(\mathbf{y}_{<c}, c|\mathbf{x})\tilde{\pi}_m(\mathbf{y}_{\geq c}|\mathbf{x}, \mathbf{y}_{<c}, c). \tag{22}$$

The corrector leaves $\mathbf{y}_{<c}$ and c unchanged and applies $\mathcal{P}_{m,c}$ only to the suffix. Since $\mathcal{P}_{m,c}$ preserves the conditional distribution of the suffix, the corrector leaves the augmented target $\tilde{\pi}_m(\mathbf{y}, c|\mathbf{x})$ invariant.

Finally, by definition,

$$\mathcal{K}_m(\mathbf{y}'|\mathbf{y}; \mathbf{x}) = \sum_c \rho(c|\mathbf{y})\mathcal{P}_{m,c}(\mathbf{y}'|\mathbf{y}; \mathbf{x}). \tag{23}$$

Therefore,

$$\sum_{\mathbf{y}\in\mathcal{Y}} \pi_m(\mathbf{y}|\mathbf{x})\mathcal{K}_m(\mathbf{y}'|\mathbf{y}; \mathbf{x}) = \sum_{\mathbf{y},c} \tilde{\pi}_m(\mathbf{y}, c|\mathbf{x})\mathcal{P}_{m,c}(\mathbf{y}'|\mathbf{y}; \mathbf{x}) = \sum_c \tilde{\pi}_m(\mathbf{y}', c|\mathbf{x}) = \pi_m(\mathbf{y}'|\mathbf{x}). \tag{24}$$

□

B ALGORITHM

Algorithm 1 summarizes the full APC procedure, including the predictor update, ESS-based resampling, corrector refinement, and adaptive stopping. The algorithm is presented for a single corrector update per annealing stage, with repeated corrector updates applied analogously when used.

Algorithm 1 Annealed Predictor-Corrector (APC) Sampling

Input: Base LLM $p_{\boldsymbol{\theta}}$, reward function r, prompt $\mathbf{x}$, annealing schedule $\{\beta_m\}_{m=0}^{M}$, number of particles N, number of corrector candidates K, cut distribution $\rho(c|\mathbf{y})$, ESS threshold τ_{ESS}, stopping hyperparameter κ

Sample initial particles $\mathbf{y}_0^{(i)} \sim p_{\boldsymbol{\theta}}(\cdot|\mathbf{x})$ for all i
Set $W_0^{(i)} \leftarrow 1/N$ for all i
Initialize the stage-output history $\mathcal{H} \leftarrow ()$
for $m = 1$ **to** M **do**
  ▷ **Predictor**
  $\tilde{W}_m^{(i)} \leftarrow W_{m-1}^{(i)} \exp((\beta_m - \beta_{m-1}) r(\mathbf{x}, \mathbf{y}_{m-1}^{(i)}))$ for all i
  $W_m^{(i)} \leftarrow \tilde{W}_m^{(i)} / \sum_{j=1}^{N} \tilde{W}_m^{(j)}$ for all i
  $\text{ESS}_m \leftarrow \left(\sum_{i=1}^{N} \left(W_m^{(i)}\right)^2\right)^{-1}$
  // Resampling
  if $\text{ESS}_m < \tau_{\text{ESS}} N$ **then**
    Sample $A_m^{(i)} \sim \text{Cat}(W_m^{(1)}, ..., W_m^{(N)})$ for all i
    $\bar{\mathbf{y}}_m^{(i)} \leftarrow \mathbf{y}_{m-1}^{(A_m^{(i)})}$ for all i
    $W_m^{(i)} \leftarrow 1/N$ for all i
  else
    $\bar{\mathbf{y}}_m^{(i)} \leftarrow \mathbf{y}_{m-1}^{(i)}$ for all i
  end if
  $\mathbf{y}_m^{(i)} \leftarrow \bar{\mathbf{y}}_m^{(i)}$ for all i
  ▷ **Corrector**
  Sample $c_m^{(i)} \sim \rho(\cdot \mid \mathbf{y}_m^{(i)})$ for all i
  Sample suffix proposals $\mathbf{y}_{m,\geq c_m^{(i)}}^{(i,k)} \sim p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}, \mathbf{y}_{m,<c_m^{(i)}}^{(i)}), k = 1, ..., K-1$ for all i
  $\mathbf{y}_m^{(i,0)} \leftarrow \mathbf{y}_m^{(i)}, \mathbf{y}_m^{(i,k)} \leftarrow \left(\mathbf{y}_{m,<c_m^{(i)}}^{(i)}, \mathbf{y}_{m,\geq c_m^{(i)}}^{(i,k)}\right), k = 1, ..., K-1$ for all i
  $\omega_m^{(i,k)} \leftarrow \exp(\beta_m r(\mathbf{x}, \mathbf{y}_m^{(i,k)})) \rho(c_m^{(i)} | \mathbf{y}_m^{(i,k)}), k = 0, ..., K-1$ for all i
  Sample $J_m^{(i)} \sim \text{Cat}\left(\frac{\omega_m^{(i,0)}}{\sum_{k=0}^{K-1} \omega_m^{(i,k)}}, ..., \frac{\omega_m^{(i,K-1)}}{\sum_{k=0}^{K-1} \omega_m^{(i,k)}}\right)$ for all i
  $\mathbf{y}_m^{(i)} \leftarrow \mathbf{y}_m^{(i,J_m^{(i)})}$ for all i
  // Adaptive stopping
  Sample $I_m \sim \text{Cat}(W_m^{(1)}, \ldots, W_m^{(N)})$
  Extract a_m from $\mathbf{y}_m^{(I_m)}$ and append a_m to $\mathcal{H}$
  if $m \geq \kappa$ and $a_{m-\kappa+1} = \cdots = a_m \neq \varnothing$ **then**
    return $\mathbf{y}_m^{(I_m)}$
  end if
end for
Sample $I \sim \text{Cat}(W_M^{(1)}, \ldots, W_M^{(N)})$

Output: $\mathbf{y}_M^{(I)}$

C ADDITIONAL EXPERIMENTAL SETTINGS

C.1 MATHEMATICAL REASONING

Models We use Qwen2.5-Math-7B-Instruct (Yang et al., 2024) as the base LLM and Qwen2.5-Math-PRM-7B (Zhang et al., 2025b) as the reward model. The reward model assigns a score to

each reasoning step. Following the recommended step-level scoring setup of Zhang et al. (2025b), we segment each response into reasoning steps using blank lines. Let $s_t \in (0, 1)$ denote the score assigned to step t. We define the complete-response reward as $r(\mathbf{x}, \mathbf{y}) = \sum_t \log \max(s_t, 10^{-4})$, corresponding to multiplicative aggregation of the step scores in log space. For all mathematical reasoning experiments, we use temperature 0.7, top-p 1.0, a maximum generation length of $2,048$ tokens, and a final reward strength $\beta = 100$.

Benchmarks and Evaluation We evaluate on MATH-500 (Lightman et al., 2024) and AMC23 (Yang et al., 2024). MATH-500 contains 500 challenging problems sampled from the MATH benchmark (Hendrycks et al., 2021), and AMC23 consists of 40 problems from the 2023 AMC 12 competitions. A response is considered correct if its final "`\boxed{}`" expression is equivalent to the reference answer under the answer checker. Unfinished responses and responses without a valid final answer are counted as incorrect.

APC Configuration For MATH-500, APC uses $N = 8$ particles, and for AMC23 we use $N = 4$. We use $K = 2$ candidates per corrector update and $M = 8$ annealing stages with the linear reward-strength schedule, i.e., $\beta_m = (m/M)\beta, m = 0, ..., M$. We apply one corrector update at each annealing stage, followed by two additional updates at the final target. We use ESS-based resampling with $\tau_{\text{ESS}} = 0.5$. For MATH-500, the main configuration uses $\lambda = 3$ and $\kappa = 5$; for AMC23, we use $\lambda = 3$ and $\kappa = 2$. If the run reaches the end of the schedule, the final response is selected by resampling according to the final particle weights. Cuts are placed at reasoning-step boundaries. The number of trailing reasoning steps to regenerate follows a truncated geometric distribution with mean λ, with an additional probability $\rho_{\text{full}} = 0.1$ of regenerating the complete response. For adaptive stopping, after each annealing stage we sample one response according to the current particle weights and extract its answer. If the sampled answers remain identical and non-empty for κ consecutive annealing stages, the entire APC run terminates and returns the most recently sampled response. The initial population is not included in this stopping history. Adaptive stopping skips all remaining annealing stages and additional corrector updates at the final target. Unless otherwise stated, the MATH-500 analyses use the single seed with $\lambda = 1$, $\rho_{\text{full}} = 0.1$, and the full annealing path without adaptive stopping. Additional experiments vary λ, the particle population, and the adaptive stopping parameters as reported in Section 5.3 and Appendix D.

C.2 Instruction Following

Models For instruction following, we use Llama-3.1-Tulu-3-8B-SFT (Lambert et al., 2025) as the base LLM and Llama-3.1-Tulu-3-8B-RM as the reward model, following Faria & Smith (2025). The reward model is an outcome reward model that assigns a scalar score to the complete response. We use temperature 1.0, top-p 1.0, and a maximum response length of 800 generated tokens.

Benchmarks and Evaluation We evaluate on IFEval (Zhou et al., 2023), which contains 541 prompts with one or more automatically verifiable instructions covering constraints on length, lexical content, formatting, and structure. We use the official IFEval instruction checkers. A prompt is counted as correct only if the generated response satisfies all associated instructions under the loose evaluation criterion of Zhou et al. (2023). Evaluation is performed directly on the complete generated response, and responses reaching the token limit are evaluated as generated.

APC Configuration We set $\beta = 20$ and use $N = 4$ particles, $K = 2$ candidates, and $M = 4$ annealing stages with a linear schedule. We apply one corrector update at each annealing stage, followed by four additional updates at the final target. The ESS threshold is $\tau_{\text{ESS}} = 0.5$. The final APC output is the highest-reward particle at the final stage, and adaptive stopping is not used on IFEval.

Since instruction-following responses do not naturally provide the reasoning-step boundaries used in the mathematical reasoning experiments, we define the cut distribution over token positions. Let b denote the number of trailing tokens to regenerate. We sample b from an equal-weight mixture of truncated geometric distributions with means 32, 128, and 512, and assign $\rho_{\text{full}} = 0.2$ to full-response regeneration.

C.3 BASELINES

Within each task, all baselines use the same base model, reward model, decoding parameters, and response-length limit as APC. We compare against independent-sampling methods (Wang et al., 2023; 2024), PRM-guided beam search (Snell et al., 2025), DVTS (Beeching et al., 2024), QAlign (Faria & Smith, 2025), step-level SMC (Zhao et al., 2024), and particle filtering (Puri et al., 2025).

Single Sample and Greedy. Single Sample reports the performance of one response sampled using the task-specific decoding configuration, while Greedy uses a single greedy decode.

Best-of-N. Best-of-N samples N independent complete responses and returns the highest-reward response. We evaluate $N \in \{1, 2, 4, 8, 16, 32, 64\}$ and additionally use compute-matched values of N when needed for comparison.

Self-Normalized Importance Sampling (SNIS). SNIS uses the same sampled responses as Best-of-N and selects a response according to $\bar{w}_i \propto \exp(\beta r_i)$. We use the same reward strength as APC. On IFEval, SNIS performs similarly to Best-of-N, so we omit it from the main table.

Self-Consistency. Self-consistency (Wang et al., 2023) returns the most frequent extracted final answer among the sampled responses. We evaluate $N \in \{1, 2, 4, 8, 16, 32, 64, 96, 128\}$.

Weighted Majority Voting. Weighted majority voting (Wang et al., 2024) sums $\bar{W}_i \propto \exp(\beta r_i)$ over responses sharing the same extracted answer and returns the answer with the largest total weight. We use the same sampled responses as Best-of-N and use the same reward strength as APC.

Beam Search. Following the search framework of Beeching et al. (2024), our implementation corresponds to the single-subtree case of DVTS. At each iteration, n candidate next steps are sampled from the current partial response, and each candidate is evaluated after a one-step greedy lookahead with temperature 0. The highest-scoring candidate is selected, and only its first generated step is committed. For scoring, step separators are collapsed so that the entire resulting prefix is treated as a single step by the PRM, using the same prefix-scoring rule as particle filtering. We evaluate $n \in \{4, 16\}$.

DVTS. Following Beeching et al. (2024), the search is divided into $n/4$ independent subtrees of width 4. Within each subtree, candidate next steps are evaluated after a one-step greedy lookahead, and only the first step of the highest-scoring candidate is committed. As above, the resulting prefix is scored by the PRM after collapsing step separators, so that the complete prefix is treated as a single step. Generation is limited to 40 reasoning steps, and a subtree terminates once it produces a boxed answer. We evaluate $n \in \{8, 16, 32\}$.

QAlign. Following Faria & Smith (2025), QAlign performs Metropolis-Hastings updates by uniformly sampling a cut position from the current response and regenerating the suffix using the base model. A proposal $\mathbf{y}'$ from the current response $\mathbf{y}$ is accepted with probability $\min\{1, \exp(\beta(r' - r))\frac{|\mathbf{y}|}{|\mathbf{y}'|}\}$, where the length ratio corrects for the uniform cut proposal. For mathematical reasoning, we use the same reward strength β as APC and return the majority answer over visited states. We evaluate $n \in \{1, 4, 8\}$ independent chains; the $n = 1$ configuration uses 10 transitions, while $n \in \{4, 8\}$ uses six transitions. For IFEval, we follow the original QAlign reward tilt, $\exp(r/0.5)$, corresponding to $\beta = 2$ in our notation. Because IFEval has no discrete extracted answer, we use minimum Bayes risk (MBR) selection with ROUGE-1 similarity over all visited states. We run four parallel chains for 12 transitions and pool their visited states. In Table 2, we report the result after eight transitions, whose inference cost is comparable to that of APC.

Step-Level SMC. Following Zhao et al. (2024), we maintain n particles and extend each particle by one reasoning step per iteration, for at most 60 steps. If s_t denotes the prefix-level process score after the new step, the incremental weight is s_t^{β}, such that the accumulated weight corresponds to $\exp(\beta r)$ under our reward definition. Systematic resampling is triggered when $\mathrm{ESS} < n/2$. The final response is selected by resampling according to the final particle weights, using the same output rule as APC. Our implementation corresponds to the untwisted case of Zhao et al. (2024). We evaluate $n \in \{2, 4, 8, 16\}$.

Particle Filtering. Following the released implementation of Puri et al. (2025), each of the n particles is extended by one reasoning step at each iteration. Generation is limited to 40 reasoning steps. Each resulting prefix is evaluated by the PRM: step separators are collapsed and the entire prefix is treated

Table 6: Sensitivity analysis of APC to the cut scale λ on MATH-500. All results are from the same single seed, with all remaining hyperparameters fixed.

Value	Accuracy	TFLOPs
1	88.8	645
2	89.0	927
3	89.6	1,060

Table 7: Allocation of inference compute between the particle population and corrector candidates of APC on MATH-500 using Qwen2.5-Math-7B-Instruct. Results are from a single seed.

Particles N	Candidates K	Accuracy	TFLOPs
8	2	**88.8**	645
4	4	87.6	615
2	8	85.2	**578**

as a single step. Particle weights are computed as $\mathrm{softmax}(\mathrm{logit}(p_t)/\tau)$, where p_t is the resulting PRM probability. Then, we perform multinomial resampling at every iteration. We use $\tau = 1$ for $n \leq 8$ and $\tau = 2$ for $n \geq 16$. The final output is the highest-scoring response among the final particles. We evaluate $n \in \{2, 4, 8, 16, 32, 64\}$.

C.4 Implementation Details

Each MATH-500 experiment is run on a single NVIDIA B200 GPU. For the other benchmarks, including AMC23 and IFEval, each experiment is run on a single NVIDIA A100 GPU. Our implementation builds on the codebase of Guided Speculative Inference (Geuter et al., 2026).[1] We also use its mathematical answer grader, which is adapted from PRM800K (Lightman et al., 2024),[2] with answer normalization following the MATH benchmark implementation (Hendrycks et al., 2021). We implement APC and reimplement all baselines within the same inference framework. Particle filtering (Puri et al., 2025) follows the authors' released code,[3] PRM-guided beam search (Snell et al., 2025) and DVTS (Beeching et al., 2024) follow the Search and Learn codebase,[4] and QAlign (Faria & Smith, 2025) follows its official implementation.[5] This shared framework ensures consistent model execution and inference-cost measurement across methods. For IFEval, we use the official instruction checker of Zhou et al. (2023),[6] using the implementation distributed with the QAlign codebase.

We measure inference cost as the total forward-pass FLOPs per prompt. Following the $2PT$ convention (Kaplan et al., 2020), a Transformer forward processing T tokens with P parameters is approximated by $2PT$ FLOPs. Reported computation includes both base-model generation and all reward-model evaluations. For generation, we count only newly generated tokens; prefill of the prompt and of any retained response prefix is not charged, since these prefixes are cached by vLLM. For each reward-model evaluation, we count both the prompt and response tokens processed by the reward model. For methods that use multiple generation or reward-model calls, the corresponding costs are accumulated over all calls. In Table 4, *Seq.* denotes the average number of sequential generation calls, treating calls for different particles or candidates within the same generation round as parallel.

D Additional Experimental Results

Sensitivity Analysis of Cut Scale λ Table 6 examines the effect of the cut scale λ, which controls how globally the corrector regenerates a response. Larger λ increases inference computation because a larger portion of the response is regenerated.

Particle Population vs. Corrector Candidates Table 7 studies how inference computation is allocated between the particle population N and the number of corrector candidates K. Note that these

[1] https://github.com/j-geuter/GSI
[2] https://github.com/openai/prm800k
[3] https://github.com/probabilistic-inference-scaling/probabilistic-inference-scaling
[4] https://github.com/huggingface/search-and-learn
[5] https://github.com/goncalorafaria/qalign
[6] https://github.com/google-research/google-research/tree/master/instruction_following_eval

Table 8: Effect of the cut-probability correction on MATH-500 using Qwen2.5-Math-7B-Instruct, with $\lambda = 3$. Results are from a single seed.

Cut correction	Accuracy	TFLOPs
✗	89.0	990
✓	89.6	1,060

Table 9: Evolution of the particle population in one APC run on MATH-500. Each cell reports the extracted answer at each stage, with its complete-response reward r below. Green and red cells indicate correct and incorrect final answers, respectively. — indicates that no valid final answer is extracted. A superscript $*$ indicates that the regenerated candidate is accepted by the corrector. *Correct-answer weight* is the total weight assigned to particles with the correct answer. *ESS* is computed after predictor reweighting, and ✓ indicates that resampling is performed at the stage.

Stage m	0	1	2	3	4	5	6	7	8
Particle 1	—	676*	306*	306	306	406	406	406*	406*
	−20.03	−2.55	−1.53	−1.53	−1.53	−0.63	−0.63	−0.62	−0.62
Particle 2	—	676*	306	306	306*	406	406*	406	406*
	−85.90	−2.55	−1.53	−1.53	−1.53	−0.63	−0.60	−0.60	−0.60
Particle 3	586	676	306	306	406*	406	406*	406	406
	−9.93	−2.55	−1.53	−1.53	−0.63	−0.63	−0.63	−0.63	−0.63
Particle 4	—	676*	306	306	306	406*	406*	406	406
	−51.19	−2.55	−1.53	−1.53	−1.53	−0.62	−0.62	−0.62	−0.62
Particle 5	—	676*	306*	406*	406	406	406	406	406
	−51.02	−2.55	−1.56	−1.42	−1.42	−0.63	−0.63	−0.63	−0.63
Particle 6	—	676*	306*	306	306*	406	406*	406	406*
	−74.33	−2.58	−1.53	−1.53	−1.53	−0.63	−0.63	−0.63	−0.63
Particle 7	676	306*	306	306*	306*	406	406	406	406
	−2.55	−1.53	−1.53	−1.53	−1.53	−0.63	−0.63	−0.63	−0.63
Particle 8	—	676	306	306	306	406	406*	406	406
	−73.48	−2.55	−1.53	−1.53	−1.53	−0.63	−0.64	−0.64	−0.64
Correct-answer weight	0.00	0.00	0.00	0.09	0.39	1.00	1.00	1.00	1.00
ESS		1.00	1.00	7.91	6.38	1.00	8.00	7.88	7.48
Resampling		✓	✓			✓			

configurations use 8, 6, and 5 annealing stages, respectively, to maintain similar compute budgets. Under similar compute budgets, reducing N while increasing K degrades accuracy, suggesting that maintaining a sufficiently diverse particle population is more beneficial than generating additional local candidates per particle.

Effect of Cut-Probability Correction Table 8 examines the cut-probability correction required by Theorem 1. Applying the correction yields a slightly improved accuracy. This result supports the practical use of the invariant corrector derived in our analysis, and we apply the correction in all experiments.

Example Generation Path We visualize one APC run on MATH-500 in Table 9 and Figures 5 and 6 to illustrate how the predictor and corrector interact over the particle population. We use the following problem: "*Let $P(x)$ be a quadratic polynomial with real coefficients satisfying $x^2 - 2x + 2 \le P(x) \le 2x^2 - 4x + 3$ for all real numbers x, and suppose $P(11) = 181$. Find $P(16)$.*" The correct answer is 406. The predictor concentrates weight on higher-reward complete responses and resamples them, while the corrector introduces new responses through regeneration.

Problem **Answer:** 406

Let $P(x)$ be a quadratic polynomial with real coefficients satisfying $x^2 - 2x + 2 \le P(x) \le 2x^2 - 4x + 3$ for all real numbers x, and suppose $P(11) = 181$. Find $P(16)$.

Particle 1: $r = -20.03$**, no final answer**

[...] Let's consider the polynomial $Q(x) = P(x) - (x^2 - 2x + 2)$. [...]

$$0 \le Q(x) \le x^2 - 2x + 1 = (x - 1)^2.$$

[...]

$$k = \frac{462}{384} = \frac{231}{192} = \frac{77}{64}.$$

[...] $P(16) = [...] = \frac{14867}{64} = 232.328125$.

[The response degenerates into unrelated tokens.]

Particle 2: $r = -85.90$**, no final answer**

[...] To find a specific form of $P(x)$, consider the midpoint of the given inequalities: [...]
We hypothesize that $P(x)$ might be of the form:

$$P(x) = k\left(x^2 - 2x + 1\right) + \frac{5}{2} = k(x - 1)^2 + \frac{5}{2}.$$

However, this form does not satisfy the given inequalities for all x. [...]

[The response degenerates into unrelated tokens.]

Particle 3: $r = -9.93$**, answer** 586 **(wrong)**

[...] $a + b + c = 1$ [...]
We also evaluate the given inequalities at $x = 0$: [...]

$$2 \le c \le 3$$

We now have: $c = 2$ [...]

$$P(16) = -254$$

Therefore, the value of $P(16)$ is: $\boxed{586}$

Particle 4: $r = -51.19$**, no final answer**

[...] This implies that the polynomial $P(x)$ must be exactly the average of $x^2 - 2x + 2$ and $2x^2 - 4x + 3$. [...]

$$P(11) = [...] = \frac{302}{2} = 151.$$

However, this does not satisfy $P(11) = 181$. [...]

[The response degenerates into unrelated tokens.]

Particle 5: $r = -51.02$**, no final answer**

[...] For this quadratic to be non-negative for all x, its discriminant must be less than or equal to zero: [...]

$$k = \frac{262}{100} = 2.62$$

Since k must be between 0 and 1, we

[The response degenerates into unrelated tokens.]

Particle 6: $r = -74.33$**, no final answer**

[...] the discriminants of both quadratic expressions must be less than or equal to zero. [...]

$$k = \frac{57}{21} = \frac{19}{7}$$

Step 7: Verify k is within the range $[0, 1]$
Since $\frac{19}{7} \approx 2.714$, there must be an error

[The response degenerates into unrelated tokens.]

Particle 7: $r = -2.55$**, answer** 676 **(wrong)**

[...] However, a simpler approach is to assume that $P(x)$ is a linear combination of the two given polynomials. Let:

$$P(x) = \alpha(x^2 - 2x + 2) + (1 - \alpha)(2x^2 - 4x + 3).$$

[...]
Thus, the value of $P(16)$ is: $\boxed{676}$.

Particle 8: $r = -73.48$**, no final answer**

[...] To find a, b, and c, we start by considering the midpoint of the two given polynomials: [...]
Thus, we assume:

$$P(x) = \frac{3x^2 - 4x + 5}{2}$$

[...] $P(11) = [...] = \frac{324}{2} = 162$
This does not match the given $P(11) = 181$. [...]

[The response degenerates into unrelated tokens.]

Figure 5: Initial particle population at stage 0 for the APC run in Table 9, with the problem and its correct answer shown at the top. The eight responses are sampled independently from the base LLM before annealing. Each box shows a selected portion of one complete response, with omitted text denoted by "[...]". None of the initial particles produces the correct answer, 406. Particles 3 and 7 produce incorrect final answers, while the remaining particles do not produce final answers. Gray italic text denotes our annotation.

Stage 1: Full regeneration

Current response $r = -2.55$

[...] However, a simpler approach is to assume that $P(x)$ is a linear combination of the two given polynomials. Let:

$$P(x) = \alpha(x^2 - 2x + 2) + (1 - \alpha)(2x^2 - 4x + 3).$$

[...]

Thus, the value of $P(16)$ is: $\boxed{676}$.

Regenerated response $r = -1.53$

Given that $P(x)$ is a quadratic polynomial with real coefficients satisfying the inequalities [...], we need to determine $P(16)$.
First, let's express $P(x)$ in the general form:

$$P(x) = ax^2 + bx + c.$$

[...]
This function $Q(x)$ must satisfy:

$$0 \le Q(x) \le x^2 - 2x + 1 = (x - 1)^2.$$

[...]

$$P(16) = (16 - 1)^2 + 81 = 15^2 + 81 = 225 + 81 = 306.$$

Therefore, the value of $P(16)$ is: $\boxed{306}$

Stage 4: Partial regeneration

Retained prefix

Given that $P(x)$ is a quadratic polynomial with real coefficients satisfying the inequalities [...], we need to determine $P(16)$.
First, let's express $P(x)$ in the general form: $P(x) = ax^2 + bx + c$.
We know that $P(x)$ must lie between the two given quadratic polynomials:

$$x^2 - 2x + 2 \le ax^2 + bx + c \le 2x^2 - 4x + 3.$$

To find the specific form of $P(x)$, consider the function:

$$Q(x) = P(x) - (x^2 - 2x + 2).$$

This function $Q(x)$ must satisfy:

$$0 \le Q(x) \le x^2 - 2x + 1 = (x - 1)^2.$$

Current suffix $r = -1.53$

Since $(x - 1)^2 \ge 0$ and is zero only at $x = 1$, $Q(x)$ must also be zero at $x = 1$. Therefore, $P(x)$ must have a double root at $x = 1$, which implies:

$$P(x) = (x - 1)^2 + k.$$

[...]

$$P(16) = (16 - 1)^2 + 81 = 15^2 + 81 = 225 + 81 = 306.$$

Therefore, the value of $P(16)$ is: $\boxed{306}$

Regenerated suffix $r = -0.63$

[...]
Therefore, we can write:

$$Q(x) = k(x - 1)^2 \quad \text{for some constant } k.$$

Substituting $Q(x)$ back into the expression for $P(x)$, we get:

$$P(x) = (x^2 - 2x + 2) + k(x - 1)^2.$$

[...]

$$k = \frac{80}{100} = \frac{4}{5}$$

[...]

$$P(16) = \frac{2030}{5}, \quad P(16) = 406.$$

Thus, the value of $P(16)$ is: $\boxed{406}$

Figure 6: Example corrector updates from the APC run in Table 9. At Stage 1, full regeneration changes the extracted answer from 676 to 306. At Stage 4, partial regeneration retains the prefix of the current response and regenerates its suffix, producing the correct answer 406. Gray, red, and blue boxes indicate the retained prefix, current text, and regenerated text, respectively, with omitted text denoted by "[...]".