CONTROLLING SHARPENING IN LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM post-training via reinforcement learning has shown consistent improvements in accuracy at the cost of reducing behavioral coverage—a phenomenon known as sharpening that can stall progress on difficult tasks. Repeated-sampling objectives such as $\mathrm{pass@k}$ and $\mathrm{max@k}$ are known to mitigate sharpening, but the mechanism behind it remains poorly understood. Moreover, existing approaches for $\mathrm{@k}$ optimization use dedicated algorithms, and no general framework exists for adapting established policy optimization methods from $\mathrm{@1}$ to $\mathrm{@k}$. We derive the optimal policies for KL-regularized $\mathrm{pass@k}$ and $\mathrm{max@k}$ objectives in closed form. Our explicit solutions reveal that $\mathrm{@k}$ optimization naturally redistributes learning pressure based on prompt difficulty, directing it toward harder problems. Crucially, we show that standard $\mathrm{@1}$ objectives achieve these exact same optimal policies through prompt-adaptive regularization or reward transformation. Ultimately, this yields an optimizer-independent explanation for why $\mathrm{@k}$ objectives reduce sharpening on easy tasks while prioritizing success on harder ones. Building on these insights, we introduce $\mathrm{@k}$ *Adaptive Pressure for Policy Optimization (*KAPPO*)*. Following a one-time reference calibration, KAPPO seamlessly adapts existing $\mathrm{@1}$ methods to $\mathrm{@k}$ via prompt-adaptive regularization or reward transformation, without otherwise altering their training procedures. Experiments with several RL methods on mathematical reasoning and code generation demonstrate that KAPPO provides direct control over sharpening. Crucially, we demonstrate that $\mathrm{@k}$ pressure can be fully realized with single-completion training, matching the performance of multi-completion $\mathrm{@k}$ optimizers at a fraction of the cost.

1 INTRODUCTION

Reinforcement learning (RL) has become a central component of LLM post-training, producing substantial performance improvements in mathematical reasoning, code generation, and other verifiable domains (Shao et al., 2024; Chen et al., 2025a; Yu et al., 2025). Yet, an increase in average performance does not by itself reveal how the model has improved. Pretrained LLMs contain extensive latent capabilities, and RL can affect them through two distinct mechanisms. It may uncover successful behavior that is rarely expressed by the base policy, or it may concentrate probability on solutions that are already readily available. The latter phenomenon, referred to as *sharpening*, improves single-trial success but often reduces the range of problems solvable by the model under repeated trials (Yue et al., 2025). Further capability progress thus depends on effectively controlling sharpening.

Recent studies confirm this exact limitation: RL post-training dynamics are indeed dominated by sharpening (Yue et al., 2025; Song et al., 2025). The impressive single-trial gains frequently diminish or reverse in multi-trial settings, as the generated reasoning paths are confined to a narrow subset of the base model's original distribution. Counteracting this over-concentration is notoriously difficult. For example, simply injecting randomness or diversity at the token level rarely translates into meaningful diversity in high-level reasoning strategies (Walder & Karkhanis, 2025; Chen et al., 2025b).

Multi-trial metrics provide a practical lens for quantifying the extent of sharpening (Yue et al., 2025). For binary rewards, $\mathrm{pass@k}$ measures the probability that at least one of k completions succeeds; for continuous rewards, $\mathrm{max@k}$ measures the expected maximum reward among k completions. Gains driven by sharpening concentrate at small k and diminish as the sampling budget grows, whereas gains that persist at large k indicate broader coverage of successful behavior. While these metrics do not prove that the model has discovered reasoning strategies outside its base distribution, they remain the clearest operational proxies for diverse reasoning. These observations motivate using $\mathrm{@k}$ metrics

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.

not just for evaluation, but also as training objectives to mitigate sharpening (Walder & Karkhanis, 2025). However, the mechanism that makes @k objectives less prone to sharpening remains poorly understood. Moreover, existing approaches rely on dedicated algorithms for @k optimization, precluding a direct transfer of standard @1 reinforcement learning advancements to the @k setting.

We derive the closed-form optimal policies for KL-regularized pass@k and max@k. Our analysis reveals that @k optimization acts as prompt-adaptive pressure: it restricts optimization on tasks the model already solves reliably and redirects pressure toward tasks on which successful behavior is rare. We show that this behavior is achievable with ordinary @1 objectives via prompt-adaptive regularization or reward transformation. Leveraging this connection, we introduce @k *Adaptive Pressure for Policy Optimization (*KAPPO*)*, a framework for turning existing @1 policy-optimization methods into @k optimizers. After a one-time reference calibration, KAPPO introduces prompt-adaptive pressure while preserving all other characteristics of the base method. Our contributions are:

1. We derive closed-form optimal policies for KL-regularized pass@k and max@k objectives.
2. We prove that @k optimization redistributes optimization pressure according to task difficulty via prompt-adaptive regularization or reward transformation, prioritizing hard tasks.
3. We introduce KAPPO, a framework that turns existing @1 methods into @k optimizers. In KAPPO, k directly controls sharpening without requiring online groups of k completions.
4. We instantiate KAPPO across several policy optimization methods, matching the performance of a direct multi-completion @k-optimization baseline at a fraction of its rollouts.

2 Related Work

Exploration mechanisms in RLVR. Beyond the sharpening evidence discussed above, RL has been shown to consistently reduce generation diversity (Kirk et al., 2024), including on problems the model has not yet solved (Song et al., 2025). Application of standard exploration strategies, such as entropy regularization (Mnih et al., 2016; Haarnoja et al., 2018; Ahmed et al., 2019) and noisy reward perturbations (Lu & Van Roy, 2017), has shown limited or sensitive improvements relative to methods designed specifically for multi-attempt reasoning (Walder & Karkhanis, 2025; Chen et al., 2025b). More structured approaches are guided by epistemic uncertainty via policy ensembles (Menet et al., 2026a), or introduce exploration bonuses based on final outcomes (Song et al., 2025), hidden representations (Tuyls et al., 2026; Menet et al., 2026b), or sequence-level diversity (Hu et al., 2026). These methods rely on different proxies for novelty and have been evaluated in heterogeneous settings; no established exploration mechanism for LLM post-training has yet emerged.

Pass@k and max@k optimization. Originally introduced as an evaluation metric for code generation (Chen et al., 2021), pass@k has recently been adopted as a training objective. Walder & Karkhanis (2025) derive unbiased policy-gradient estimators, called PKPO, for binary pass@k and continuous max@k rewards. They show that increasing k during training preserves higher entropy, increases the number of problems solved at least once, and unblocks learning on difficult tasks where pass@1 optimization stalls. max@k objectives are submodular (Krause & Golovin, 2014), and encourage "hedging the bets" given limited expressivity, a principle associated with exploration. Subsequent work studies analytical advantage design (Chen et al., 2025b), token-level mechanisms for preventing probability over-concentration (Peng et al., 2025), improved policy-gradient estimators for order-statistic objectives (Parmas et al., 2026), and a pass@k interpretation of maximum likelihood objectives (Tajwar et al., 2026). These approaches primarily analyze training gradients or construct multi-completion learning signals. We instead derive the corresponding KL-regularized optimal policies and show that the effect of @k optimization can be reproduced through prompt-adaptive @1 objectives. This provides a general route for turning existing @1 methods into @k variants without constructing expensive @k-dependent multi-completion learning signals.

3 Preliminaries

KL-regularized reward optimization. Let $x \sim \mathcal{D}$ denote a prompt, $y \sim \pi(\cdot \mid x)$ a completion, and $R(y, x)$ the corresponding reward. Standard KL-regularized RL optimizes the objective

$$\mathcal{J}(\pi) := \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(\cdot|x)}[R(y,x)] - \beta\,\mathrm{KL}\left(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x)\right)\right], \tag{1}$$

where π_{ref} is a fixed reference policy and $\beta > 0$ controls the strength of regularization. While some recent RLVR methods omit the KL penalty to allow larger policy shifts (Yu et al., 2025), other works suggest that carefully designed KL-regularized objectives enhance both reasoning performance and training stability (Liu et al., 2025). Beyond potential empirical advantages, KL regularization is a prerequisite for non-degenerate optimal policies. Indeed, the optimal policy is given in closed form:

$$\pi^*(y \mid x) = \tfrac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)\exp(R(y,x)/\beta). \tag{2}$$

Methods such as REINFORCE (Williams, 1992), RLOO (Ahmadian et al., 2024), PPO (Schulman et al., 2017), and GRPO (Shao et al., 2024) approximate gradients of Eq. 1. In contrast, methods like DPO (Rafailov et al., 2023), A^*PO (Brantley et al., 2025), and QRPO (Matrenok et al., 2025) directly fit a parameterized policy π_θ to the optimal policy π^* given in Eq. 2. Our analysis and contributions are agnostic to the choice of optimizer, separating @k capabilities from other advancements.

Pass@k and max@k. Let $y_1, \ldots, y_k \overset{\text{i.i.d.}}{\sim} \pi(\cdot \mid x)$. We define $\max@k$ and $\text{pass}@k$ as

$$\max@k(\pi; x) = \mathbb{E}\big[\max_{i=1,\ldots,k} R(y_i, x)\big], \qquad \text{pass}@k(\pi; x) = \mathbb{P}\left[\exists\, i \in \{1,\ldots,k\} : R(y_i, x) = 1\right].$$

Note that for binary rewards $R(y,x) \in \{0,1\}$, $\max@k$ coincides with $\text{pass}@k$. Moreover, for $k = 1$, both expressions recover the ordinary @1 expected reward $\mathbb{E}_{y\sim\pi(\cdot|x)}[R(y,x)]$ used in the standard RL objective $\mathcal{J}$ in Eq. 1. Let $V_k(\pi; x)$ denote either of these @k objectives. Substituting $V_k(\pi; x)$ into $\mathcal{J}$ in place of the @1 reward yields the KL-regularized objective studied in this work:

$$\mathcal{J}_k(\pi) = \mathbb{E}_{x\sim\mathcal{D}}\left[V_k(\pi; x) - \beta\,\mathrm{KL}\left(\pi(\cdot \mid x)\|\pi_{\text{ref}}(\cdot \mid x)\right)\right]. \tag{3}$$

4 @k Optimal Policies and Adaptive Pressure

4.1 Binary Rewards: pass@k

Proofs of all results in this subsection, including boundary cases, are provided in Appendix B. We first consider binary rewards, i.e., we set $V_k(\pi; x) = \text{pass}@k(\pi; x)$ in Eq. 3 for $\mathcal{J}_k$, and derive the closed-form optimal policy which maximizes $\mathcal{J}_k$. It takes the familiar exponential-tilt form.

Proposition 4.1: Closed-form optimal policy for the pass@k objective
Let $R(y,x) \in \{0,1\}$. Then the optimal policy for $\mathcal{J}_k$ is

$$\pi^*(y \mid x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)\exp\left(\frac{R(y,x)}{\beta_{\text{eff}}(x)}\right), \tag{4}$$

$$\text{where } \beta_{\text{eff}}(x) = \frac{\beta}{k(1-p^*(x))^{k-1}} \text{ and } Z(x) = 1 - p_{\text{ref}}(x) + p_{\text{ref}}(x)\exp\left(\frac{1}{\beta_{\text{eff}}(x)}\right). \tag{5}$$

Here, $p_{\text{ref}}(x) = \mathbb{P}_{y\sim\pi_{\text{ref}}(\cdot|x)}[R(y,x)=1]$ and $p^*(x) = \mathbb{P}_{y\sim\pi^*(\cdot|x)}[R(y,x)=1]$ are, respectively, the success probabilities of the reference model and optimal model.

Compared with the optimal policy (Eq. 2) for the single-reward objective (Eq. 1), the constant regularization strength β is replaced by the prompt-dependent effective regularization strength $\beta_{\text{eff}}(x)$. The obtained expression for the optimal policy, however, remains implicit since $p^*(x)$ depends on the unknown optimal policy π^*. Nevertheless, it can be made fully explicit by summing Eq. 4 over successful completions, which yields a scalar equation relating $p^*(x)$ directly to $p_{\text{ref}}(x)$.

Proposition 4.2: Scalar fixed-point characterization
For $0 < p_{\text{ref}}(x) < 1$, the optimal success probability $p^*(x)$ is the unique solution in $(0,1)$ of

$$k(1-p^*(x))^{k-1} = \beta\log\frac{p^*(x)(1-p_{\text{ref}}(x))}{p_{\text{ref}}(x)(1-p^*(x))}. \tag{6}$$

The reference success probability $p_{\text{ref}}(x)$ can be estimated directly from reference-model samples,[1], while $p^*(x)$ is computed by solving Eq. 6. For an error analysis, see Appendix K. Proposition 4.1 suggests two equivalent ways to recover the pass@k optimal policy with single-reward optimization.

Corollary 4.3: pass@k as prompt-adaptive regularization

Optimizing the ordinary single-reward objective in Eq. 1 using the prompt-adaptive regularization strength $\beta_{\text{eff}}(x)$ from Eq. 5 results exactly in the pass@k optimal policy provided in Eq. 4.

This equivalence holds because both objectives decompose over prompts and $\beta_{\text{eff}}(x)$ is constant w.r.t. the completion y for each fixed x. As the optimal policy given in Eq. 4 depends on the reward and KL coefficient only through their ratio, we obtain a complementary reward-side interpretation.

Corollary 4.4: pass@k as prompt-adaptive reward scaling

Optimizing the ordinary single-reward objective $\mathcal{J}$ given in Eq. 1 with constant regularization strength β but applying a prompt-adaptive reward transformation

$$\widetilde{R}_k(y, x) = \frac{\beta}{\beta_{\text{eff}}(x)} R(y, x) = k(1 - p^*(x))^{k-1} R(y, x) \tag{7}$$

results exactly in the pass@k optimal policy provided in Eq. 4.

Two views of adaptive pressure. Corollary 4.3 and Corollary 4.4 show that pass@k optimization can be expressed as ordinary single-reward optimization with either prompt-adaptive regularization or, equivalently, prompt-adaptive reward scaling. Taking $1 - p_{\text{ref}}(x)$ as a simple measure of prompt complexity, easy tasks x have high $p_{\text{ref}}(x)$ and consequently high $p^*(x)$, resulting in stronger effective regularization, while hard tasks receive weaker effective regularization and therefore stronger optimization pressure (see Figure 1a). Equivalently, under the reward-side view, a successful completion receives a smaller reward multiplier on easy tasks and a larger multiplier on difficult ones.

Intuition: reducing sharpening. Consider optimizing the usual single-reward objective (Eq. 1) on a binary-reward task. The reward gain depends only on the increase in success probability, regardless of initial performance—for example, moving from near zero to 0.1 on a hard task is worth as much as moving from 0.9 to 1 on an already-easy task. For an LLM, however, the latter is often much easier: the model already knows how to solve the task and only needs to concentrate probability on known successful behavior. In contrast, increasing a near-zero success probability requires acquiring entirely novel reasoning strategies. The adaptive pressure induced by pass@k objectives counteracts this tendency towards sharpening by imposing stronger effective regularization on already-easy tasks or, equivalently, through the prompt-adaptive reward scaling described above.

4.2 Continuous Rewards: max@k

Next, we consider real-valued rewards, i.e., we set $V_k(\pi; x) = \max@k(\pi; x)$ in Eq. 3 for $\mathcal{J}_k$, and derive the associated closed-form optimal policy which maximizes $\mathcal{J}_k$. For $k > 1$, we obtain:

Proposition 4.5: Closed-form optimal policy for the max@k objective

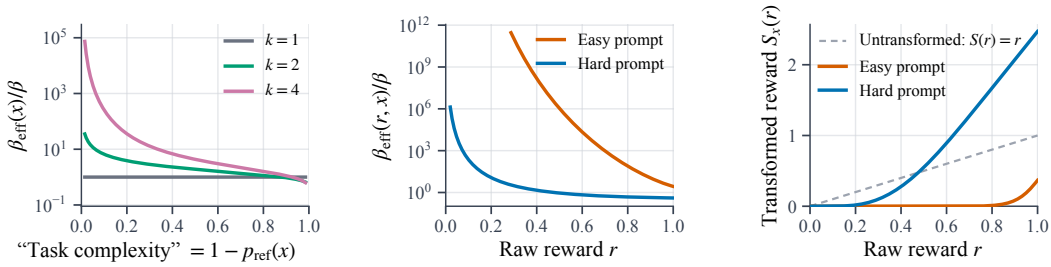
The optimal policy for the max@k objective $\mathcal{J}_k$ under real-valued rewards evaluates to

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{k}{\beta} \int_{-\infty}^{R(y,x)} F_{\pi^*}(r \mid x)^{k-1} \, dr \right), \tag{8}$$

where $F_{\pi^*}(r \mid x) := \mathbb{P}_{y' \sim \pi^*(\cdot|x)}[R(y', x) \leq r]$ is the reward CDF under the optimal policy.

The optimal policy identified above remains implicit because the CDF F_{π^*} on its right-hand side is itself induced by the unknown optimal policy π^*. In contrast to the binary-reward case, computing

[1]The same reference samples can be reused across training epochs and hyperparameter sweeps.



(a) pass@k: effective regularization (b) max@k: effective regularization (c) max@k: reward transformation

Figure 1: Adaptive pressure induced by @k objectives. **(a)** For binary rewards, increasing k strengthens regularization on easy tasks while retaining greater optimization pressure on difficult tasks. **(b–c)** For continuous rewards, we illustrate the same effect at $k = 4$ using synthetic "easy" $\text{Beta}(8, 2)$ and "difficult" $\text{Beta}(2, 8)$ reference reward distributions on $[0, 1]$. The easy task receives stronger effective regularization and smaller transformed rewards than the difficult task. All panels use $\beta = 0.5$.

F_{π^*} is more difficult and requires a sufficiently accurate estimate of the entire reference reward distribution $F_{\pi_{ref}}$. It can, nevertheless, be computed numerically before training; we provide the procedure in Appendix D. All derivations and regularity conditions are explained in Appendix C.

Following the derivation for pass@k, a natural question is whether we can also identify a task-dependent β_{eff} here that reduces @k optimization to @1. A simple substitution shows that:

$$\pi^*(y \mid x) = \tfrac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)\exp\left(\tfrac{R(y,x)}{\beta_{\text{eff}}(R(y,x),x)}\right) \text{ for } \beta_{\text{eff}}(r,x) = \frac{\beta}{k} r \Big/ \int_{-\infty}^{r} F_{\pi^*}(u \mid x)^{k-1}\, du. \quad (9)$$

Note that, to ensure well-definedness, one can assume $R(y, x) > 0$ without loss of generality given (lower) bounded rewards, since the objectives in Eqs. 1 and 3 are invariant to constant reward shifts.

Eq. 9 provides an interpretation analogous to the binary-reward case: max@k replaces constant regularization with a strength that adapts to task complexity. In the max@k case, however, the effective strength also varies with the reward of the individual completion. As an illustrative example, Figure 1b uses two synthetic reference reward distributions on the $[0, 1]$ interval representing an "easy" and a "difficult" task. This completion dependence means that substituting $\beta_{\text{eff}}(R(y, x), x)$ into the ordinary single-reward objective would produce a different optimal policy; hence, the exact adaptive-regularization equivalence established in Corollary 4.3 for binary rewards does not extend to continuous rewards. Still, β_{eff} provides a useful interpretation of how max@k redistributes optimization pressure and could be adapted in practice to design completion-adaptive regularization. Alternatively, we may follow our derivations for pass@k and instead consider adaptive reward transformations to redistribute optimization pressure. We obtain the following objective-level equivalence:

Corollary 4.6: max@k as prompt-adaptive reward transformation
Optimizing the ordinary single-reward objective $\mathcal{J}$ given in Eq. 1 with the transformed reward

$$\widetilde{R}(y, x) = S_x(R(y, x)) \text{ for } S_x(r) = k \int_{-\infty}^{r} F_{\pi^*}(u \mid x)^{k-1}\, du \quad (10)$$

results in the max@k optimal policy derived in Proposition 4.5.

The transformed-reward view shows the same prompt-adaptive effect. In the illustration in Figure 1c, the same raw reward is assigned a larger transformed value on the difficult task than on the easy task, providing a complementary reward-space view of how max@k can be used to control sharpening.

Explicit closed-form optimal policy for win-rate reward. As discussed, the optimal policy in Eq. 8 remains implicit because it depends on the unknown CDF F_{π^*}. A workaround is to consider a different but closely related reward: the reference quantile $F_{\text{ref}}(r \mid x) = \mathbb{P}_{y' \sim \pi_{\text{ref}}(\cdot|x)}[R(y', x) \leq r]$, which preserves reward ordering and represents the win rate against reference-policy completions. Indeed, the max@k optimal policy for these win-rate rewards admits a fully explicit expression.

Proposition 4.7: Explicit optimal policy for win-rate $\max@k$

For a continuous reference reward distribution with CDF $F_{\text{ref}}(r \mid x)$, the objective $\mathcal{J}_k$ with win-rate rewards $F_{\text{ref}}(R(y,x) \mid x)$ admits the following optimal policy:

$$\pi_q^*(y \mid x) = \pi_{\text{ref}}(y \mid x) \exp\left(\frac{S_x^{\text{q}}(R(y,x))}{\beta}\right), \tag{11}$$

where

$$S_x^{\text{q}}(r) = \beta \log\left(C + \frac{1}{\beta}\Psi^{-1}(F_{\text{ref}}(r \mid x))^k\right),\ \Psi(z) = \int_0^z \frac{du}{C + \frac{1}{\beta}u^k},\ \text{and } C \text{ s.t. } \Psi(1) = 1. \tag{12}$$

For a proof, see Appendix F. The scalar C and mapping Ψ depend only on k and β, and can be numerically pre-computed prior to training. The only prompt-dependent quantity in S_x^{q} is $F_{\text{ref}}(r \mid x)$, which, similarly to $p_{\text{ref}}(x)$ for binary-rewards, can be estimated once from a fixed bank of reference-model completions (see Appendix K for an error analysis). Thus, the adaptive-reward principle of Corollary 4.6 admits a directly computable realization for win-rate-based optimal policies.

5 KAPPO: @k Adaptive Pressure for Policy Optimization

We introduce @k *Adaptive Pressure for Policy Optimization* (KAPPO), a framework for adapting existing @1 policy-optimization methods to @k settings. The parameter k can be tuned directly to control sharpening without otherwise changing the optimization algorithm. Given a base optimizer, KAPPO redistributes optimization pressure either through adaptive reward transformation or adaptive regularization, following our derivations in Section 4. The required prompt-dependent transformations and coefficients are estimated once before training at negligible cost and reused across runs; we call this stage *reference calibration* (see Appendix J). We instantiate both interfaces below to demonstrate KAPPO's generality and effectiveness across several popular RLVR optimizers.

5.1 Regularization-Side Adaptation: @k-DPO

The regularization-side interface is particularly useful for methods that do not directly optimize scalar rewards, most prominently DPO (Rafailov et al., 2023). We apply it in the binary-reward setting,[2] using the connection established in Corollary 4.3 to replace DPO's global KL coefficient with the prompt-dependent $\beta_{\text{eff}}(x)$ from Eq. 5. We call the resulting method @k-DPO:

$$\mathcal{L}_{\text{@k-DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{pref}}} \log\sigma\left(\beta_{\text{eff}}(x)\left[\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right]\right). \tag{13}$$

Before training, we perform reference calibration by generating a fixed bank of M reference-model completions for each prompt x and use the associated binary rewards to estimate $p_{\text{ref}}(x)$. We then solve Eq. 6 for $p^*(x)$ and use it to compute $\beta_{\text{eff}}(x)$. Each online update subsequently requires only a pair of completions per prompt. Except for β_{eff}, the DPO training procedure remains unchanged.

5.2 Reward-Side Adaptation: @k-$A^\star$PO and @k-GRPO

To demonstrate the reward-side interface, we consider two complementary instantiations. First, we choose $A^\star$PO (Brantley et al., 2025), a method that uses only a single online completion per prompt. This choice isolates a central implication of our analysis: online tuples of k completions are not necessary to induce @k optimization pressure. Second, we also apply the interface to GRPO (Shao et al., 2024). Following common practice in RLVR, we use GRPO without a KL regularization term (Yu et al., 2025); this additionally tests whether KAPPO remains effective beyond the KL-regularized setting assumed by our theory.

@k-$A^\star$PO. $A^\star$PO optimizes the squared-regression loss

$$\mathcal{L}_{A^\star\text{PO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\text{old}}(\cdot|x)}\left[\left(R(y,x) - \beta\log Z(x) - \beta\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right)^2\right], \tag{14}$$

[2] We note that prompt-adaptive regularization is also possible for continuous rewards, but without the same objective-level equivalence, e.g., by directly adjusting the regularization coefficient of the gradient expression.

with the partition function $Z(x)$ estimated from reference-model completions. Replacing $R(y, x)$ with the prompt-adaptive reward yields @k-$A^\star$PO. We treat both binary and continuous rewards:

For binary rewards, we substitute the adaptive reward $\frac{\beta R(y,x)}{\beta_{\text{eff}}(x)}$ into Eq. 14. We obtain β_{eff} and $Z(x)$ through the same one-time reference calibration procedure as for the binary @k-DPO instantiation.

For continuous rewards, we apply the win-rate-based adaptive reward transformation S_x^{q} from Eq. 12 and set $Z(x) = 1$ since normalization follows automatically (see Proposition 4.7). We conduct a one-time reference calibration to approximate $F_{\text{ref}}(r \mid x)$. As in the binary case, we sample a fixed bank $\{y_j^{\text{ref}}\}_{j=1}^M$ of reference-model completions for each prompt. We then use this bank to construct the approximation $\widehat{F}_{\text{ref}}(r \mid x) = \frac{1}{M}\sum_{j=1}^M \mathbb{I}[R(y_j^{\text{ref}}, x) \le r]$ for each prompt x.

@k-GRPO. Here, we construct @k-GRPO for the continuous-reward setting. For the corresponding binary-reward adaptation as well as additional properties of GRPO instantiations, see Appendix G. Given a group of n completions with rewards $R_i = R(y_i, x)$, we replace the raw rewards by the win-rate-based tranformed rewards $S_x^{\text{q}}(R_i)$ in the GRPO loss:

$$\mathcal{L}_{@\text{k-GRPO}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D},\, y_1,\ldots,y_n\sim\pi_{\text{old}}(\cdot|x)}\big[\tfrac{1}{n}\textstyle\sum_{i=1}^n \frac{1}{|y_i|}\sum_t u_{i,t}^{\text{clip}}(\theta)\big], \tag{15}$$

$$u_{i,t}^{\text{clip}}(\theta) = \min\big(\tfrac{\pi_\theta(y_i^t|y_i^{<t},x)}{\pi_{\text{old}}(y_i^t|y_i^{<t},x)}\widehat{A}_i^k,\ \text{clip}\big(\tfrac{\pi_\theta(y_i^t|y_i^{<t},x)}{\pi_{\text{old}}(y_i^t|y_i^{<t},x)}, 1-\epsilon, 1+\epsilon\big)\widehat{A}_i^k\big),\ \widehat{A}_i^k = \tfrac{S_x^{\text{q}}(R_i) - \frac{1}{n}\sum_{j=1}^n S_x^{\text{q}}(R_j)}{\text{std}_{j=1,\ldots,n}[S_x^{\text{q}}(R_j)]}.$$

@k-GRPO uses the same one-time reference calibration as the continuous @k-$A^\star$PO instantiation. Apart from the reward transformation S_x^{q}, the standard GRPO procedure remains unchanged.

6 Experiments

6.1 Setup

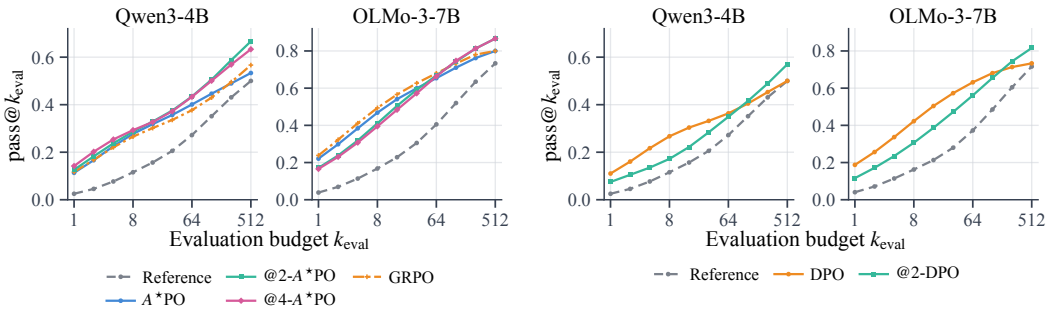
Tasks and scope. We evaluate KAPPO on mathematical reasoning with binary rewards (correctness) and code generation with continuous rewards (test case pass rate). We investigate all instantiations of KAPPO presented in Section 5, but note that KAPPO could be applied to other algorithms.

Model selection. Two principles guide our choice of initial checkpoints. First, we avoid models that have undergone reasoning-focused preference optimization or RL. This reflects the practical setting in which RL for reasoning is applied directly after supervised fine-tuning (SFT), and separates the effect of our training from reasoning behavior acquired via an undocumented prior curriculum. Second, we aim to minimize possible contamination of the training and evaluation datasets. These criteria lead us to `OLMo-3-7B-Instruct-SFT` and `Qwen3-4B-Base`. The former is an SFT-only, pre-RL checkpoint with a fully open training pipeline including explicit data decontamination procedures (Team Olmo et al., 2025). For a second model family, we use the Qwen pre-trained rather than a post-trained checkpoint to limit exposure to undocumented post-training data, although its pretraining corpus cannot be fully audited (Yang et al., 2025). Hyperparameters and checkpoints are selected on a validation split; complete experimental details are provided in Appendix H.

Baselines. As elaborated in Section 5, KAPPO can be applied to a wide range of RL optimizers by adjusting the effective regularization or transforming the reward. Since applying KAPPO with $k = 1$ leaves the underlying @1 optimizer unchanged, we use $k = 1$ as the primary baseline to isolate the effect of adaptive @k pressure. We also compare a KAPPO-adjusted A^*PO algorithm with PKPO (Walder & Karkhanis, 2025), a dedicated pass@k optimization algorithm that requires at least $n \ge k$ new on-policy completions per prompt.

6.2 Binary Rewards: Mathematical Reasoning

Data and evaluation. We design the train–validation–test split to mirror a common model development setting: training data and validation benchmarks are closely coupled during development, while the final test consists of newer and often more difficult problems, appearing after model release. To closely match this setting, we perform RL on the training split of MATH and use its test split for in-distribution validation (Hendrycks et al., 2021). Final evaluation is performed on AIME



(a) $A^\star$PO and @k-$A^\star$PO. (b) DPO and @k-DPO.

Figure 2: $\text{pass@k}_{\text{eval}}$ on AIME 2026 as the sampling budget at test time (k_{eval}) increases. Setting $k > 1$ during training consistently reduces sharpening. In Figure **(a)**, we plot standard GRPO and @k-$A^\star$PO trained with $k \in \{1, 2, 4\}$. In Figure **(b)**, we plot @k-DPO trained with $k \in \{1, 2\}$.

2026 (Mathematical Association of America, 2026), a markedly harder dataset whose problems predominantly exceed MATH's highest difficulty tier. AIME 2026 also postdates both model initialization checkpoints, ruling out direct contamination of our test benchmark. A completion receives reward one if its extracted final answer is correct and zero otherwise, i.e., we consider binary rewards.

Controlling sharpening by adjusting k. Figure 2 plots $\text{pass@k}_{\text{eval}}$ against the number of completions k_{eval}. We examine the improvement over the initial checkpoint as a function of the sampling budget. Gains *exclusively* at small k_{eval} indicate that training has further concentrated probability on already-accessible behavior, whereas gains at large k_{eval} point toward acquisition of novel behavior. Figure 2a considers reward-side adaptation. The $A^\star$PO baseline (pass@1 training) closely tracks GRPO, showing that single-completion optimization performs competitive to multi-completion optimization for pass@1 training. However, the gains achieved for both models diminish as k_{eval} increases. Training with $k > 1$ typically (but not always) retains less of the pass@1 gain achieved by $k = 1$ training, but consistently overtakes the single-reward baselines at larger evaluation budgets. Figure 2b exhibits the same effect for the regularization-side interface. Standard DPO produces gains concentrated at small evaluation budgets, whereas @k-DPO trained with $k = 2$ overtakes it as k_{eval} increases. This pattern holds for both OLMo and Qwen. Together, these results corroborate the effect predicted by our analysis across two base optimizers and both KAPPO interfaces: @k adaptation counteracts excessive sharpening and shifts optimization pressure from already-easy tasks toward difficult tasks where successful behavior remains rare.

Agreement with dedicated multi-completion pass@k optimization. We next verify that a KAPPO adaptation of a single-completion optimizer can match the performance of a state-of-the-art dedicated multi-completion pass@k algorithm. To that end, we compare @k-$A^\star$PO with PKPO for $k \in \{2, 4\}$ using a matched number of prompt batches and the minimum feasible PKPO rollout mul-

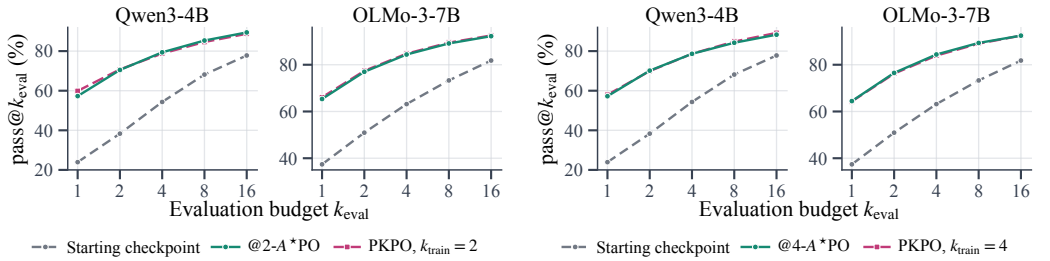


Figure 3: MATH Level 5 validation performance of the starting checkpoints, @k-$A^\star$PO checkpoints, and PKPO checkpoints. @k-$A^\star$PO matches the performance of PKPO at a fraction of the on-policy training rollouts, requiring only a single-time reference calibration at $\leq 5\%$ training time overhead.

tiplicity ($n = k$). Across both models, @k-$A^\star$PO achieves validation performance comparable to PKPO (see Figure 3) while requiring only a fraction of the on-policy rollouts, reducing training cost. This supports our central prediction: @k optimization can be achieved without relying on online k-completion learning signals, but by injecting adaptive pressure into single-completion optimizers.

6.3 Continuous Rewards: Code Generation

Data and evaluation. We follow the same data selection strategy as in the binary-reward experiments: training and validation use closely related data, while final evaluation uses a newer and substantially harder problem source to approximate the distribution shift encountered after model deployment. We conduct our max@k experiments using `Qwen3-4B-Base`. Specifically, we train on the training split of LeetCodeDataset (Xia et al., 2025) and use its test split for validation. The two splits share the same source and task format, with a temporal cutoff on July 2024 (Xia et al., 2025). Each completion receives a continuous reward in $[0, 1]$ equal to the fraction of test cases it passes. For final evaluation, we use the hard subset of LiveCodeBench v6, a substantially harder collection of competitive-programming problems (Jain et al., 2025) released between January and April 2025, limiting the opportunity for their inclusion in Qwen3 pretraining (released April 28th). Nevertheless, since Qwen's pretraining cutoff is not documented, contamination cannot be ruled out completely.

Controlling sharpening by adjusting k. We observe the same qualitative effect as in the binary-reward experiments. Figure 4 compares the initial checkpoint with win-rate-based @k-$A^\star$PO and @k-GRPO trained with $k \in \{1, 2, 4\}$. The $k = 1$ configuration typically provides the largest improvement for max@1, but its advantage contracts as the evaluation budget increases and the starting checkpoint eventually catches up. Training with $k > 1$ retains (slightly) less of the max@1 gain achieved by $k = 1$ training, but overtakes it at larger evaluation budgets, maintaining a greater advantage over the starting checkpoint. Thus, our explicit win-rate-based reward transformation provides analogous control over sharpening for continuous rewards as our binary-reward adaptations.

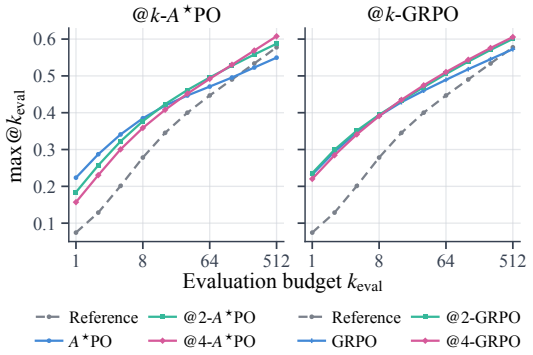


Figure 4: max@k_{eval} on the hard subset of LiveCodeBench v6 for Qwen3-4B. Win-rate-based @k-$A^\star$PO (left) and @k-GRPO (right) mitigate excessive sharpening for $k > 1$.

7 Conclusion

We derive closed-form optimal policies for KL-regularized pass@k and max@k optimization, proving that @k optimization redistributes pressure according to task difficulty. For binary rewards, this pressure can be expressed equivalently through prompt-adaptive reward scaling or regularization strength. For continuous rewards, it appears as an adaptive reward transformation or a completion-dependent regularization strength, with a fully explicit construction for win-rate-based rewards. Leveraging these insights, we introduce KAPPO, a framework that adapts existing @1 policy-optimization methods to @k settings without requiring online multi-completion rollouts. In particular, our @k-$A^\star$PO instantiation requires only one new completion per prompt, yet achieves performance comparable to a SoTA multi-completion baseline. Validated across mathematical reasoning and code generation, @k-$A^\star$PO, @k-DPO, and @k-GRPO successfully control sharpening, confirming that diverse reasoning can be effectively encouraged via prompt-adaptive optimization pressure.

Limitations and future work. Our reference calibration relies on a *static* bank of M completions for every prompt instead of, e.g., optimizing the budget based on the noise transfer of the equations in Propositions 4.2 and 4.7, resulting in noisier estimates for p^* and S_x^{q}. Future work may develop better budget allocations. Second, our continuous-reward experiments optimize the win-rate surrogate S_x^{q} rather than raw-reward max@k, leaving empirical evaluation of numerical F_{π^*} reconstruction (see Appendix D) to future work. Lastly, our win-rate solution assumes a continuous reference reward distribution; its application to discrete rewards with ties is an approximation.

AI USE STATEMENT

We used generative AI tools to check proofs, improve their exposition, edit the manuscript, and assist with literature search and reference checking. AI also supported the implementation of methods, training infrastructure, evaluation, and plotting, following author-specified designs and guidance. For data processing, AI assisted with deduplication and analysis of dataset diversity, task difficulty, and metadata. The central research ideas and essential proof steps were developed by the authors. All experimental decisions, including experimental design, hyperparameters, comparisons, and all interpretations of results were determined solely by the authors. The authors reviewed the AI-assisted code, verified mathematical claims and proofs, checked cited sources and their relevance, and checked figures against experimental outputs. We take responsibility for the final content of this work, including all AI-assisted text, claims, and artifacts.

ETHICS STATEMENT

We improve understanding of pass@k and max@k optimization and introduce a method to adapt existing reinforcement learning methods from @1 to @k. As a result, post-training of frontier models may be accelerated, which can results in all kinds of secondary effects, both economical and societal. Our experiments use only publicly available models and datasets in accordance with their licenses.

REPRODUCIBILITY STATEMENT

All considered datasets and models are publicly available. Instead of proposing a novel RL optimization method with many degrees of freedom to be specified, we identify a minimalist intervention to turn any standard @1 RL method into an @k method. We provide pseudocode for reference calibration with binary and continuous win-rate rewards in Appendix J.1. As a result, all of our experiments can be easily reproduced. For completeness, we describe the exact experimental details in Appendix H. Our entire code base is available in the supplementary materials.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. 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. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 151–160. PMLR, 2019. URL https://proceedings.mlr.press/v97/ahmed19a.html.

Kianté Brantley, Mingyu Chen, Zhaolin Gao, Jason Lee, Wen Sun, Wenhao Zhan, and Xuezhou Zhang. Accelerating RL for LLM reasoning with optimal advantage regression. In *Advances in Neural Information Processing Systems*, volume 38, pp. 151492–151531. Curran Associates, Inc., 2025. doi: 10.52202/085713-5067. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/de8e1aebdb1e8a96970a58a641eac3cc-Abstract-Conference.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders,

Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceReason-Nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025a. doi: 10.48550/arXiv.2505.16400. URL https://arxiv.org/abs/2505.16400.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b. doi: 10.48550/arXiv.2508.10751. URL https://arxiv.org/abs/2508.10751.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.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 Joaquin Vanschoren and Serena Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Zican Hu, Shilin Zhang, Yafu Li, Jianhao (Elliott) Yan, Xuyang Hu, Leyang Cui, Xiaoye Qu, Chunlin Chen, Yu Cheng, and Zhi Wang. Diversity-incentivized exploration for versatile reasoning. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 71175–71205, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/74164872c24945a391473a8da5415843-Paper-Conference.pdf.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In Yisong Yue, Animesh Garg, Nanyun Peng, Fei Sha, and Rose Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/94074dd5a072d28ff75a76dabed43767-Paper-Conference.pdf.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *International Conference on Learning Representations*, pp. 20620–20653, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5a68d05006d5b05dd9463dd9c0219db0-Paper-Conference.pdf.

Andreas Krause and Daniel Golovin. Submodular function maximization. In Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli (eds.), *Tractability: Practical Approaches to Hard Problems*, chapter 3, pp. 71–104. Cambridge University Press, 2014. doi: 10.1017/CBO9781139177801.004. URL https://doi.org/10.1017/CBO9781139177801.004.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17998–18031. Curran Associates, Inc., 2025. doi: 10.52202/085713-0608. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1a22b912945fb7c0bdd079e792b31b6f-Abstract-Conference.html.

Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

Pascal Massart. The tight constant in the Dvoretzky–Kiefer–Wolfowitz inequality. *The Annals of Probability*, 18(3):1269–1283, 1990. doi: 10.1214/aop/1176990746. URL https://doi.org/10.1214/aop/1176990746.

Mathematical Association of America. American invitational mathematics examination (AIME), 2026. URL https://maa.org/maa-invitational-competitions/.

Simon Matrenok, Skander Moalla, and Caglar Gulcehre. Quantile reward policy optimization: Alignment with pointwise regression and exact partition functions. In *Advances in Neural Information Processing Systems*, volume 39, pp. 102693–102750. Curran Associates, Inc., 2025. doi: 10.52202/085713-3434. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/94840c41497ead6a84f493f029eba7fa-Abstract-Conference.html.

Nicolas Menet, Andreas Krause, and Abbas Rahimi. POETS: Uncertainty-aware LLM optimization via compute-efficient policy ensembles. *Advances in Neural Information Processing Systems*, 40, 2026a. URL https://arxiv.org/abs/2605.07775.

Nicolas Menet, Aleksandar Terzic, Michael Hersche, Andreas Krause, and Abbas Rahimi. Thompson sampling via fine-tuning of LLMs. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 68794–68822, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/707a2d58641b2192203b4bf4c532cfe1-Paper-Conference.pdf.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 2016. PMLR. URL https://proceedings.mlr.press/v48/mniha16.html.

Paavo Parmas, Yongmin Kim, Kohsei Matsutani, Shota Takashiro, Soichiro Nishimori, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. OrderGrad: Optimizing beyond the mean with order-statistic policy gradient estimation. *arXiv preprint arXiv:2606.06096*, 2026. doi: 10.48550/arXiv.2606.06096. URL https://arxiv.org/abs/2606.06096.

Ruotian Peng, Yi Ren, Zhouliang Yu, Weiyang Liu, and Yandong Wen. Beyond the sampled token: Preserving candidate support in RLVR. *arXiv preprint arXiv:2510.14807*, 2025. doi: 10.48550/arXiv.2510.14807. URL https://arxiv.org/abs/2510.14807.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Yuda Song, Julia Kempe, and Rémi Munos. Outcome-based exploration for LLM reasoning. *arXiv preprint arXiv:2509.06941*, 2025. doi: 10.48550/arXiv.2509.06941. URL https://arxiv.org/abs/2509.06941.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. In *International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2602.02710. URL https://icml.cc/virtual/2026/poster/65332.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. doi: 10.48550/arXiv.2512.13961. URL https://arxiv.org/abs/2512.13961.

Jens Tuyls, Dylan Foster, Akshay Krishnamurthy, and Jordan Ash. Representation-based exploration for language models: From test-time to post-training. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 120621–120645, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c420aca62da1d668e8e79e0bcae24e6b-Paper-Conference.pdf.

Christian Walder and Deep Tejas Karkhanis. Pass@K policy optimization: Solving harder reinforcement learning problems. In *Advances in Neural Information Processing Systems*, volume 38, pp. 152416–152445. Curran Associates, Inc., 2025. doi: 10.52202/085713-5096. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/df8a1a6306c3bcc89f6078759feadf62-Abstract-Conference.html.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. LeetCodeDataset: A temporal dataset for robust evaluation and efficient training of code LLMs. *arXiv preprint arXiv:2504.14655*, 2025. doi: 10.48550/arXiv.2504.14655. URL https://arxiv.org/abs/2504.14655.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin

Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, volume 38, pp. 57654–57689. Curran Associates, Inc., 2025. doi: 10.52202/085713-1933. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/537d5aa768c2d534016a4d06f87bc8fb-Abstract-Conference.html.

APPENDIX

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 Preliminaries **2**

4 @k Optimal Policies and Adaptive Pressure **3**

4.1 Binary Rewards: pass@k 3

4.2 Continuous Rewards: max@k 4

5 KAPPO: @k Adaptive Pressure for Policy Optimization **6**

5.1 Regularization-Side Adaptation: @k-DPO 6

5.2 Reward-Side Adaptation: @k-$A^\star$PO and @k-GRPO 6

6 Experiments **7**

6.1 Setup 7

6.2 Binary Rewards: Mathematical Reasoning 7

6.3 Continuous Rewards: Code Generation 9

7 Conclusion **9**

A A Variational Optimality Lemma **17**

B Binary-Rewards Derivation **19**

B.1 Optimal Policy 20

B.2 Scalar Equation and Uniqueness 20

B.3 Boundary Cases 21

B.4 Objective-Level Equivalences 21

C Continuous-Rewards Derivation **22**

C.1 Concavity of the Reward Functional 22

C.2 Directional Derivative 23

C.3 Optimal Policy 23

D Numerical Reconstruction of the Raw-Reward Optimal CDF **24**

D.1 Continuous Reference Reward Distribution 24

D.2 Discrete Reconstruction 25

D.3 Empirical Approximation 26

E Reward-Scale Assumptions for Continuous Rewards **26**

F Explicit $\max@k$ Solution for Win-Rate Rewards **27**

G KAPPO Adaptation of GRPO **29**

H Experimental Details **30**

I Compute Resources **31**

J Reference Calibration **31**

J.1 Calibration Algorithms 31

J.2 Boundary Handling 33

K Sensitivity to Finite Reference Calibration **34**

K.1 Binary Rewards 35

K.2 Continuous Rewards 37

L Comparison with Direct Multi-Completion Optimization **39**

A A VARIATIONAL OPTIMALITY LEMMA

We first establish a general result used in the derivations of both optimal policies. Since all objectives considered in this work decompose over prompts, we fix a prompt x throughout this section. Let

$$\mathcal{S}_x = \{y : \pi_{\mathrm{ref}}(y \mid x) > 0\} \tag{16}$$

denote the support of the reference policy. Any policy assigning positive probability outside $\mathcal{S}_x$ has infinite KL divergence, so it is sufficient to optimize over distributions supported on $\mathcal{S}_x$.

Consider an objective of the form

$$\mathcal{J}_x(\pi) = U_x(\pi) - \beta\,\mathrm{KL}\left(\pi(\cdot \mid x) \| \pi_{\mathrm{ref}}(\cdot \mid x)\right), \tag{17}$$

where U_x is a concave reward functional and $\beta > 0$.

For a feasible policy π, consider a perturbation

$$\pi_\varepsilon(y \mid x) = \pi(y \mid x) + \varepsilon h(y \mid x), \tag{18}$$

where

$$\sum_{y \in \mathcal{S}_x} |h(y \mid x)| < \infty, \qquad \sum_{y \in \mathcal{S}_x} h(y \mid x) = 0, \tag{19}$$

and there exists $\varepsilon_0 > 0$ such that $\pi_\varepsilon(y \mid x) \geq 0$ for every $\varepsilon \in [0, \varepsilon_0]$ and $y \in \mathcal{S}_x$. The first condition ensures that the perturbation has finite total variation, the second preserves normalization, and the third ensures that the perturbed policy remains feasible in the direction under consideration.

We assume that the corresponding one-sided directional derivative of U_x exists and can be written as

$$\frac{d}{d\varepsilon} U_x(\pi + \varepsilon h)\Big|_{\varepsilon = 0^+} = \sum_{y \in \mathcal{S}_x} h(y \mid x) A_\pi(y, x) \tag{20}$$

for some score $A_\pi(y, x)$, with all displayed sums finite. The specific score corresponding to each @k objective is derived in the following sections.

We additionally assume that an optimal policy exists and has full support on $\mathcal{S}_x$. Under the regularity conditions used below, the reward change produced by transferring probability ε between two completions is $O(\varepsilon)$. Suppose, however, that $\pi(y \mid x) = 0$ for some $y \in \mathcal{S}_x$ and transfer probability ε to y. The resulting change in the negative KL term contains

$$-\beta\varepsilon \log \varepsilon = \beta\varepsilon \log \frac{1}{\varepsilon}. \tag{21}$$

Although this quantity tends to zero as $\varepsilon \to 0$, its ratio to ε diverges. Therefore, for sufficiently small ε, the KL gain dominates the $O(\varepsilon)$ reward change, and a policy assigning zero probability to part of $\mathcal{S}_x$ cannot be optimal.

Lemma A.1 (Variational form of a KL-regularized optimum)**.** *Under the assumptions above, any maximizer π^* of Eq. 17 satisfies*

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{A_{\pi^*}(y, x)}{\beta}\right), \tag{22}$$

where

$$Z(x) = \sum_{y \in \mathcal{S}_x} \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{A_{\pi^*}(y, x)}{\beta}\right). \tag{23}$$

Conversely, any feasible policy satisfying Eq. 22 is the unique maximizer of Eq. 17.

Proof. For compactness, define

$$B_{\pi^*}(y, x) = A_{\pi^*}(y, x) - \beta\left(\log \frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + 1\right). \tag{24}$$

Consider any perturbation h such that $\pi^* + \varepsilon h$ remains feasible for both positive and negative sufficiently small ε. Applying Eq. 20 to h and $-h$ shows that the corresponding two-sided derivative exists. Since $\varepsilon = 0$ is a local maximum of $\mathcal{J}_x(\pi^* + \varepsilon h)$, this derivative must vanish:

$$\frac{d}{d\varepsilon}\mathcal{J}_x(\pi^* + \varepsilon h)\bigg|_{\varepsilon=0} = 0. \tag{25}$$

The directional derivative of the KL term is

$$\frac{d}{d\varepsilon}\mathrm{KL}\left(\pi^* + \varepsilon h \| \pi_{\mathrm{ref}}\right)\bigg|_{\varepsilon=0} = \sum_{y \in \mathcal{S}_x} h(y \mid x)\left(\log \frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + 1\right). \tag{26}$$

Together with Eq. 20, this gives

$$\sum_{y \in \mathcal{S}_x} h(y \mid x) B_{\pi^*}(y, x) = 0. \tag{27}$$

Now consider any two completions $y, z \in \mathcal{S}_x$ and choose

$$h = \mathbf{e}_y - \mathbf{e}_z, \tag{28}$$

where $\mathbf{e}_y$ denotes a unit point mass at y. Since π^* has full support, this perturbation remains feasible for both signs of sufficiently small ε. Substituting it into Eq. 27 yields

$$B_{\pi^*}(y, x) - B_{\pi^*}(z, x) = 0. \tag{29}$$

Because y and z were arbitrary, $B_{\pi^*}(y, x)$ is constant with respect to y on $\mathcal{S}_x$. Hence, there exists a prompt-dependent scalar $c(x)$ such that

$$A_{\pi^*}(y, x) - \beta\left(\log \frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + 1\right) = c(x) \tag{30}$$

for every $y \in \mathcal{S}_x$.

Rearranging Eq. 30 gives

$$\pi^*(y \mid x) = \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{A_{\pi^*}(y, x)}{\beta} - \frac{c(x)}{\beta} - 1\right). \tag{31}$$

The final two terms are independent of y and can therefore be absorbed into the normalization constant. Imposing $\sum_y \pi^*(y \mid x) = 1$ gives Eqs. 22 and 23.

It remains to show that this stationary policy is the unique global maximizer. Let π be any other feasible policy and define

$$h(y \mid x) = \pi(y \mid x) - \pi^*(y \mid x). \tag{32}$$

This perturbation satisfies

$$\sum_y |h(y \mid x)| \le 2, \qquad \sum_y h(y \mid x) = 0, \tag{33}$$

and

$$\pi^* + \varepsilon h = (1 - \varepsilon)\pi^* + \varepsilon \pi \tag{34}$$

is a valid policy for every $\varepsilon \in [0, 1]$.

Define

$$g(\varepsilon) = U_x(\pi^* + \varepsilon h). \tag{35}$$

Because U_x is concave, g is concave on $[0, 1]$. A concave function lies below its supporting line at the origin, and hence

$$g(1) - g(0) \le g'(0^+). \tag{36}$$

Using Eq. 20, this gives

$$U_x(\pi) - U_x(\pi^*) \le \sum_{y \in \mathcal{S}_x} \left(\pi(y \mid x) - \pi^*(y \mid x)\right) A_{\pi^*}(y, x). \tag{37}$$

The KL difference admits the exact decomposition

$$\begin{aligned}&\mathrm{KL}\left(\pi\|\pi_{\mathrm{ref}}\right)-\mathrm{KL}\left(\pi^*\|\pi_{\mathrm{ref}}\right)\\&\quad=\sum_{y\in\mathcal{S}_x}\left(\pi(y\mid x)-\pi^*(y\mid x)\right)\left(\log\frac{\pi^*(y\mid x)}{\pi_{\mathrm{ref}}(y\mid x)}+1\right)+\mathrm{KL}\left(\pi\|\pi^*\right).\end{aligned}\tag{38}$$

To verify this identity, expand

$$\log\frac{\pi(y\mid x)}{\pi_{\mathrm{ref}}(y\mid x)}=\log\frac{\pi(y\mid x)}{\pi^*(y\mid x)}+\log\frac{\pi^*(y\mid x)}{\pi_{\mathrm{ref}}(y\mid x)}\tag{39}$$

inside the first KL term and use $\sum_y(\pi(y\mid x)-\pi^*(y\mid x))=0$.

Combining Eqs. 37 and 38 yields

$$\begin{aligned}\mathcal{J}_x(\pi)-\mathcal{J}_x(\pi^*)\le&\sum_{y\in\mathcal{S}_x}\left(\pi(y\mid x)-\pi^*(y\mid x)\right)B_{\pi^*}(y,x)\\&-\beta\,\mathrm{KL}\left(\pi\|\pi^*\right).\end{aligned}\tag{40}$$

By Eq. 30, $B_{\pi^*}(y,x)=c(x)$ for every y. Since both policies have total mass one,

$$\sum_{y\in\mathcal{S}_x}\left(\pi(y\mid x)-\pi^*(y\mid x)\right)B_{\pi^*}(y,x)=c(x)\sum_{y\in\mathcal{S}_x}\left(\pi(y\mid x)-\pi^*(y\mid x)\right)=0.\tag{41}$$

Consequently,

$$\mathcal{J}_x(\pi)-\mathcal{J}_x(\pi^*)\le-\beta\,\mathrm{KL}\left(\pi\|\pi^*\right)\le 0.\tag{42}$$

Equality holds only if $\pi=\pi^*$, proving that π^* is the unique global maximizer. □

The score $A_\pi(y,x)$ is defined only up to an additive term that depends on x but not on y. Such a term vanishes in Eq. 20 because every permitted perturbation has zero total mass, and it is absorbed into $Z(x)$ in Eq. 22. The following sections derive the relevant score for the $\mathrm{pass@k}$ and $\mathrm{max@k}$ reward functionals.

B Binary-Rewards Derivation

We now apply Lemma A.1 to the $\mathrm{pass@k}$ objective. As before, the objective decomposes over prompts, so we fix a prompt x. For any policy π, define its success probability

$$p_\pi(x)=\mathbb{P}_{y\sim\pi(\cdot|x)}\left[R(y,x)=1\right]=\sum_y\pi(y\mid x)R(y,x).\tag{43}$$

For binary rewards, the probability that at least one of k independent completions succeeds is

$$U_x(\pi)=1-\left(1-p_\pi(x)\right)^k.\tag{44}$$

The prompt-wise KL-regularized objective is therefore

$$\mathcal{J}_{k,x}(\pi)=1-\left(1-p_\pi(x)\right)^k-\beta\,\mathrm{KL}\left(\pi(\cdot\mid x)\|\pi_{\mathrm{ref}}(\cdot\mid x)\right).\tag{45}$$

The mapping $p\mapsto 1-(1-p)^k$ is concave on $[0,1]$ for $k\ge 1$: it is affine for $k=1$, while for $k\ge 2$,

$$\frac{d^2}{dp^2}\left[1-(1-p)^k\right]=-k(k-1)(1-p)^{k-2}\le 0.\tag{46}$$

Since $p_\pi(x)$ is linear in π, the reward functional $U_x(\pi)$ is concave. The assumptions of Lemma A.1 therefore apply.

B.1 Optimal Policy

Consider a perturbation $\pi_\varepsilon = \pi + \varepsilon h$ satisfying the conditions of Appendix A. Its success probability is

$$p_{\pi_\varepsilon}(x) = p_\pi(x) + \varepsilon \sum_y h(y \mid x) R(y, x). \tag{47}$$

Consequently, the directional derivative of the reward functional is

$$\frac{d}{d\varepsilon} U_x(\pi + \varepsilon h)\Big|_{\varepsilon=0^+} = k \left(1 - p_\pi(x)\right)^{k-1} \sum_y h(y \mid x) R(y, x). \tag{48}$$

Thus, the score in Eq. 20 is

$$A_\pi(y, x) = k \left(1 - p_\pi(x)\right)^{k-1} R(y, x). \tag{49}$$

Applying Lemma A.1 gives

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{k}{\beta} \left(1 - p^*(x)\right)^{k-1} R(y, x)\right), \tag{50}$$

where

$$p^*(x) = p_{\pi^*}(x). \tag{51}$$

The solution is initially implicit because its right-hand side depends on the success probability of the unknown optimal policy.

Define

$$p_{\text{ref}}(x) = \mathbb{P}_{y \sim \pi_{\text{ref}}(\cdot \mid x)} \left[R(y, x) = 1\right]. \tag{52}$$

Since the reward is binary, the reference policy assigns total mass $1 - p_{\text{ref}}(x)$ to unsuccessful completions and total mass $p_{\text{ref}}(x)$ to successful completions. Normalizing Eq. 50 therefore gives

$$Z(x) = 1 - p_{\text{ref}}(x) + p_{\text{ref}}(x) \exp\left(\frac{k}{\beta} \left(1 - p^*(x)\right)^{k-1}\right). \tag{53}$$

Introducing

$$\beta_{\text{eff}}(x) = \frac{\beta}{k \left(1 - p^*(x)\right)^{k-1}}, \tag{54}$$

we recover the form reported in Eq. 4:

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{R(y, x)}{\beta_{\text{eff}}(x)}\right). \tag{55}$$

B.2 Scalar Equation and Uniqueness

It remains to determine $p^*(x)$. Summing Eq. 50 over successful completions gives

$$p^*(x) = \frac{p_{\text{ref}}(x) \exp\left(\frac{k}{\beta}(1 - p^*(x))^{k-1}\right)}{1 - p_{\text{ref}}(x) + p_{\text{ref}}(x) \exp\left(\frac{k}{\beta}(1 - p^*(x))^{k-1}\right)}. \tag{56}$$

Taking the odds ratio and then the logarithm yields

$$k \left(1 - p^*(x)\right)^{k-1} = \beta \log \frac{p^*(x) \left(1 - p_{\text{ref}}(x)\right)}{p_{\text{ref}}(x) \left(1 - p^*(x)\right)}, \tag{57}$$

which is Eq. 6 in the main text.

To establish uniqueness, fix $q = p_{\text{ref}}(x) \in (0, 1)$ and define, for $p \in (0, 1)$,

$$\Phi_q(p) = \beta \log \frac{p(1 - q)}{q(1 - p)} - k(1 - p)^{k-1}. \tag{58}$$

Equation 57 is equivalent to $\Phi_q(p) = 0$. The function satisfies

$$\lim_{p\to 0^+} \Phi_q(p) = -\infty, \qquad \lim_{p\to 1^-} \Phi_q(p) = +\infty, \tag{59}$$

and

$$\Phi_q'(p) = \frac{\beta}{p(1-p)} + k(k-1)(1-p)^{k-2} > 0. \tag{60}$$

Hence, Φ_q is continuous and strictly increasing from $-\infty$ to $+\infty$, so it has exactly one root in $(0, 1)$. Moreover,

$$\Phi_q(q) = -k(1-q)^{k-1} < 0, \tag{61}$$

which implies that the unique solution satisfies

$$p^*(x) > p_{\mathrm{ref}}(x). \tag{62}$$

For $k = 1$, Eq. 57 reduces to

$$1 = \beta \log \frac{p^*(x)\,(1 - p_{\mathrm{ref}}(x))}{p_{\mathrm{ref}}(x)\,(1 - p^*(x))}, \tag{63}$$

whose solution is

$$p^*(x) = \frac{p_{\mathrm{ref}}(x)e^{1/\beta}}{1 - p_{\mathrm{ref}}(x) + p_{\mathrm{ref}}(x)e^{1/\beta}}. \tag{64}$$

In this case, $\beta_{\mathrm{eff}}(x) = \beta$, and Eq. 50 reduces to the standard KL-regularized single-reward optimal policy.

B.3 Boundary Cases

The scalar equation above assumes $0 < p_{\mathrm{ref}}(x) < 1$. The two boundary cases follow directly from the support constraint imposed by the KL term.

If $p_{\mathrm{ref}}(x) = 0$, the reference policy assigns no probability to successful completions. Every policy with finite KL divergence is therefore also supported only on unsuccessful completions, so

$$p^*(x) = 0, \qquad \pi^*(\cdot \mid x) = \pi_{\mathrm{ref}}(\cdot \mid x). \tag{65}$$

The reward term is identically zero over the feasible set, and the reference policy uniquely minimizes the KL term.

If $p_{\mathrm{ref}}(x) = 1$, every completion in the support of the reference policy is successful. Hence, every finite-KL policy has success probability one and the $\mathrm{pass@k}$ reward is constant:

$$p^*(x) = 1, \qquad \pi^*(\cdot \mid x) = \pi_{\mathrm{ref}}(\cdot \mid x). \tag{66}$$

For $k > 1$, this corresponds to $\beta_{\mathrm{eff}}(x) = +\infty$: once the task is solved with probability one, the $\mathrm{pass@k}$ reward provides no further optimization pressure. For $k = 1$, the effective regularization remains β.

B.4 Objective-Level Equivalences

We finally prove the two objective-level interpretations stated in the main text.

Adaptive regularization. For each prompt x, consider the ordinary single-reward objective with the prompt-dependent KL coefficient $\beta_{\mathrm{eff}}(x)$. Its unique optimal policy is

$$\pi^*_{\mathrm{reg}}(y \mid x) = \frac{1}{Z_{\mathrm{reg}}(x)} \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{R(y, x)}{\beta_{\mathrm{eff}}(x)}\right). \tag{67}$$

This is exactly the $\mathrm{pass@k}$ optimal policy derived in Eq. 50. Since the objective decomposes over prompts and $\beta_{\mathrm{eff}}(x)$ is constant with respect to y for each fixed x, this proves Corollary 4.3.

Adaptive reward scaling. Now retain the constant KL coefficient β and replace the original binary reward with

$$\widetilde{R}_k(y,x) = \frac{\beta}{\beta_{\text{eff}}(x)} R(y,x) = k\,(1-p^*(x))^{k-1}\,R(y,x). \tag{68}$$

The corresponding single-reward optimal policy is

$$\pi^*_{\text{rew}}(y \mid x) = \frac{1}{Z_{\text{rew}}(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{\widetilde{R}_k(y,x)}{\beta}\right) \tag{69}$$

$$= \frac{1}{Z_{\text{rew}}(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{R(y,x)}{\beta_{\text{eff}}(x)}\right). \tag{70}$$

This again coincides exactly with the $\mathrm{pass@k}$ optimal policy, proving Corollary 4.4.

C Continuous-Rewards Derivation

We now derive the optimal policy for the $\max@k$ objective with real-valued rewards. As before, the objective decomposes over prompts, so we fix a prompt x. For the derivation, it is sufficient to assume that

$$R(y,x) \in [a_x, b_x] \tag{71}$$

for every y in the support of the reference policy, where the endpoints may depend on x. A common reward range across prompts is not required for the optimal-policy derivation; it becomes relevant only for the cross-prompt interpretation discussed in Appendix E.

Boundedness guarantees that the relevant expectations and directional derivatives are finite and permits differentiation under the expectation. The result extends to unbounded rewards whenever the corresponding expectations, integrals, and directional derivatives are finite and the required interchange of differentiation and expectation is valid. We also assume that the prompt-wise optimum is attained. For a finite completion space, existence follows from compactness of the probability simplex. For a countably infinite space, boundedness of the reward implies that every maximizing sequence has uniformly bounded KL divergence from π_{ref}. The corresponding KL sublevel sets are tight, preventing probability mass from escaping, and lower semicontinuity of KL then ensures that the supremum is attained.

No continuity assumption on the reward distribution is required in this section. In particular, the reward CDF may contain atoms.

For a policy π, define

$$U_{k,x}(\pi) = \mathbb{E}_{y_1,\ldots,y_k \overset{\text{i.i.d.}}{\sim} \pi(\cdot|x)} \left[\max_{i=1,\ldots,k} R(y_i,x)\right]. \tag{72}$$

The prompt-wise KL-regularized objective is

$$\mathcal{J}_{k,x}(\pi) = U_{k,x}(\pi) - \beta\,\mathrm{KL}\left(\pi(\cdot \mid x) \| \pi_{\text{ref}}(\cdot \mid x)\right). \tag{73}$$

C.1 Concavity of the Reward Functional

Let

$$F_\pi(r \mid x) = \mathbb{P}_{y\sim\pi(\cdot|x)}\left[R(y,x) \le r\right] \tag{74}$$

denote the reward CDF under π. Since the maximum of k independent rewards is bounded below by a_x, the tail-integral identity gives

$$U_{k,x}(\pi) = a_x + \int_{a_x}^{b_x} \left[1 - F_\pi(r \mid x)^k\right] dr. \tag{75}$$

For each fixed r, $F_\pi(r \mid x)$ is linear in π, while the mapping

$$u \longmapsto 1 - u^k \tag{76}$$

is concave on $[0,1]$. Integration preserves concavity, so $U_{k,x}(\pi)$ is a concave functional of π. Combined with the strict concavity of the negative KL term, this also implies that Eq. 73 has at most one maximizer.

C.2 Directional Derivative

Consider a perturbation

$$\pi_\varepsilon = \pi + \varepsilon h \tag{77}$$

satisfying the conditions of Appendix A. Expanding the product measure in $U_{k,x}(\pi_\varepsilon)$ and differentiating at $\varepsilon = 0$ gives one term for each of the k sampled completions. By symmetry,

$$\frac{d}{d\varepsilon} U_{k,x}(\pi + \varepsilon h)\Big|_{\varepsilon=0^+} = k \sum_y h(y \mid x) \mathbb{E}_{y_2,\ldots,y_k \overset{\text{i.i.d.}}{\sim} \pi(\cdot|x)} \left[\max\left\{R(y,x), R(y_2,x), \ldots, R(y_k,x)\right\}\right]. \tag{78}$$

Thus, one valid score in Eq. 20 is

$$\Gamma_\pi(y,x) = k\, \mathbb{E}_{y_2,\ldots,y_k \overset{\text{i.i.d.}}{\sim} \pi(\cdot|x)} \left[\max\left\{R(y,x), R(y_2,x), \ldots, R(y_k,x)\right\}\right]. \tag{79}$$

To isolate the part that depends on y, let

$$M_{k-1} = \max_{i=2,\ldots,k} R(y_i, x). \tag{80}$$

Using

$$\max\{a,b\} = b + (a-b)_+, \qquad (u)_+ = \max\{u, 0\}, \tag{81}$$

we obtain

$$\Gamma_\pi(y,x) = k\, \mathbb{E}_\pi\left[M_{k-1}\right] + k\, \mathbb{E}_\pi\left[\left(R(y,x) - M_{k-1}\right)_+\right]. \tag{82}$$

The first term depends on π and x but not on the candidate completion y. It can therefore be omitted from the score by the additive-constant invariance established after Lemma A.1.

For any real number a and random variable M,

$$(a - M)_+ = \int_{-\infty}^{a} \mathbb{I}[M \le r]\, dr. \tag{83}$$

Taking expectations and applying Tonelli's theorem gives

$$\mathbb{E}\left[(a - M)_+\right] = \int_{-\infty}^{a} \mathbb{P}[M \le r]\, dr. \tag{84}$$

This identity holds for arbitrary reward distributions and does not require their CDFs to be continuous.

Because M_{k-1} is the maximum of $k-1$ independent rewards sampled from π,

$$\mathbb{P}[M_{k-1} \le r] = F_\pi(r \mid x)^{k-1}. \tag{85}$$

Substituting this into Eq. 84, with $a = R(y,x)$, gives

$$\mathbb{E}_\pi\left[\left(R(y,x) - M_{k-1}\right)_+\right] = \int_{-\infty}^{R(y,x)} F_\pi(r \mid x)^{k-1}\, dr. \tag{86}$$

We may therefore choose the completion-dependent score

$$A_\pi(y,x) = k \int_{-\infty}^{R(y,x)} F_\pi(r \mid x)^{k-1}\, dr. \tag{87}$$

C.3 Optimal Policy

Applying Lemma A.1 with the score in Eq. 87 gives

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{k}{\beta} \int_{-\infty}^{R(y,x)} F_{\pi^*}(r \mid x)^{k-1}\, dr\right), \tag{88}$$

where

$$Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{k}{\beta} \int_{-\infty}^{R(y,x)} F_{\pi^*}(r \mid x)^{k-1}\, dr\right). \tag{89}$$

This is Eq. 8 from the main text.

The characterization remains implicit because the CDF F_{π^*} appearing on its right-hand side is itself induced by the unknown optimal policy. Nevertheless, concavity of $U_{k,x}$ and strict concavity of the negative KL term ensure that any policy satisfying Eq. 88 is the unique global maximizer. A numerical procedure for reconstructing F_{π^*} from the reference reward distribution is provided in Appendix D.

D Numerical Reconstruction of the Raw-Reward Optimal CDF

The optimal policy in Eq. 88 depends on the unknown reward CDF F_{π^*}. In this section, we show how this CDF can be reconstructed numerically from the reference reward distribution. As in the previous derivations, we fix a prompt x and suppress its dependence where doing so does not cause ambiguity.

Define the optimal-policy score

$$H(r) = k \int_{-\infty}^{r} F_{\pi^*}(u \mid x)^{k-1}\, du \tag{90}$$

and let

$$c(x) = \frac{1}{Z(x)}. \tag{91}$$

Because the likelihood ratio in Eq. 88 depends on a completion only through its reward, the reward distributions under the optimal and reference policies satisfy

$$dP_{\pi^*}(r \mid x) = c(x) \exp\left(\frac{H(r)}{\beta}\right) dP_{\mathrm{ref}}(r \mid x). \tag{92}$$

Thus, once the scalar $c(x)$ is known, the optimal reward distribution can be reconstructed progressively from lower to higher rewards. The normalization condition determines the unique value of $c(x)$.

D.1 Continuous Reference Reward Distribution

Suppose first that the reference reward distribution has density $f_{\mathrm{ref}}(r \mid x)$ on a bounded interval $[a_x, b_x]$. Let $f_{\pi^*}(r \mid x)$ denote the corresponding optimal density. Equation 92 gives

$$f_{\pi^*}(r \mid x) = c(x) f_{\mathrm{ref}}(r \mid x) \exp\left(\frac{H(r)}{\beta}\right). \tag{93}$$

Since

$$\frac{d}{dr} F_{\pi^*}(r \mid x) = f_{\pi^*}(r \mid x), \qquad \frac{d}{dr} H(r) = k F_{\pi^*}(r \mid x)^{k-1}, \tag{94}$$

the unknown functions satisfy the coupled system

$$\begin{aligned} \frac{d}{dr} F_{\pi^*}(r \mid x) &= c(x) f_{\mathrm{ref}}(r \mid x) \exp\left(\frac{H(r)}{\beta}\right), \\ \frac{d}{dr} H(r) &= k F_{\pi^*}(r \mid x)^{k-1}, \end{aligned} \tag{95}$$

with initial conditions

$$F_{\pi^*}(a_x \mid x) = 0, \qquad H(a_x) = 0. \tag{96}$$

For a candidate value of $c(x)$, the system can be integrated forward from a_x to b_x. The correct value is the one satisfying the terminal normalization condition

$$F_{\pi^*}(b_x \mid x) = 1. \tag{97}$$

This gives a one-dimensional shooting problem. The terminal value $F_{\pi^*}(b_x \mid x)$ increases monotonically with $c(x)$, so the normalizing value can be found using bisection or another scalar root-finding method.

D.2 Discrete Reconstruction

In practice, the reference reward distribution is estimated from a finite bank of reference-model completions and is therefore discrete. Let

$$r_1 < r_2 < \cdots < r_m \tag{98}$$

be its distinct observed reward values, and define

$$q_j = \mathbb{P}_{\pi_{\mathrm{ref}}}\left[R(y,x) = r_j\right], \qquad j = 1, \ldots, m. \tag{99}$$

Let

$$p_j = \mathbb{P}_{\pi^*}\left[R(y,x) = r_j\right], \qquad P_j = \sum_{\ell=1}^{j} p_\ell \tag{100}$$

denote the optimal probability mass at reward r_j and the corresponding cumulative probability.

Define

$$H_j = H(r_j) = k \int_{-\infty}^{r_j} F_{\pi^*}(u \mid x)^{k-1}\, du. \tag{101}$$

Since the optimal CDF is constant between consecutive reward values,

$$H_1 = 0, \qquad H_j = H_{j-1} + k(r_j - r_{j-1})P_{j-1}^{k-1}, \quad j = 2, \ldots, m. \tag{102}$$

Grouping Eq. 88 by reward value gives

$$p_j = c(x) q_j \exp\left(\frac{H_j}{\beta}\right). \tag{103}$$

Equations 102 and 103 yield a forward recursion for any candidate $c = c(x)$:

$$\begin{aligned}
H_1(c) &= 0, \\
p_1(c) &= cq_1, \\
P_1(c) &= p_1(c), \\
H_j(c) &= H_{j-1}(c) + k(r_j - r_{j-1})P_{j-1}(c)^{k-1}, \\
p_j(c) &= cq_j \exp\left(\frac{H_j(c)}{\beta}\right), \\
P_j(c) &= P_{j-1}(c) + p_j(c), \qquad j = 2, \ldots, m.
\end{aligned} \tag{104}$$

The required normalization is

$$P_m(c) = 1. \tag{105}$$

Each quantity in the recursion is continuous and strictly increasing in c. Moreover,

$$P_m(0) = 0, \tag{106}$$

while, for $q_1 > 0$,

$$P_m\left(\frac{1}{q_1}\right) \geq p_1\left(\frac{1}{q_1}\right) = 1, \tag{107}$$

with strict inequality whenever $m > 1$. Therefore, Eq. 105 has a unique solution in

$$0 < c \leq \frac{1}{q_1}, \tag{108}$$

which can be found efficiently by bisection. Equivalently, one may treat the optimal probability of the lowest reward, $p_1 = cq_1$, as the scalar search variable.

Once c has been determined, the recursion recovers all optimal masses p_j, cumulative probabilities P_j, and scores H_j. For $r \in [r_j, r_{j+1})$, the reconstructed CDF and score are

$$F_{\pi^*}(r \mid x) = P_j, \qquad H(r) = H_j + k(r - r_j)P_j^{k-1}. \tag{109}$$

Thus, the score is linearly interpolated between adjacent reward levels, with slope determined by the reconstructed optimal CDF.

D.3 Empirical Approximation

Given M reference-model completions, the discrete reference masses are estimated as

$$\widehat{q}_j = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}\left[R(y_i^{\mathrm{ref}}, x) = r_j\right]. \tag{110}$$

Substituting these empirical masses into Eq. 104 produces an approximation of F_{π^*} and the raw-reward transformation H. The reconstruction must be performed once for each prompt and each pair (k, β).

Unlike the binary-reward case, where the reference distribution is summarized by the single scalar $p_{\mathrm{ref}}(x)$, the continuous case requires estimating the complete reference reward distribution. Errors in its estimated masses and reward locations propagate through the forward recursion and the subsequent exponential reweighting. This motivates the explicit reference-win-rate construction derived in Appendix F.

E Reward-Scale Assumptions for Continuous Rewards

The derivation in Appendix C requires only prompt-wise bounded rewards, or suitable integrability conditions in the unbounded case. The stronger shared-range assumption used in the main text is needed only for interpreting the resulting adaptation across prompts.

Shift invariance. Both objectives studied in this work are invariant, at the level of their optimizer, to prompt-dependent constant reward shifts. For

$$R_c(y, x) = R(y, x) + c(x), \tag{111}$$

we have

$$\mathbb{E}_{y\sim\pi(\cdot|x)}[R_c(y, x)] = \mathbb{E}_{y\sim\pi(\cdot|x)}[R(y, x)] + c(x) \tag{112}$$

and

$$\mathbb{E}\left[\max_{i=1,\dots,k} R_c(y_i, x)\right] = \mathbb{E}\left[\max_{i=1,\dots,k} R(y_i, x)\right] + c(x). \tag{113}$$

The shift therefore adds a policy-independent constant to Eqs. 1 and 3, leaving their optimal policies unchanged. Consequently, if rewards are bounded below by $r_{\min}$, we may replace R with $R - r_{\min}$ and work with nonnegative rewards without loss of generality.

Nonnegative rewards and effective regularization. Suppose that the reward support for prompt x has lower endpoint a_x. Since $F_{\pi^*}(u \mid x) = 0$ below a_x, the exponent in Eq. 8 can be written, for $r > a_x$, as

$$\frac{k}{\beta} \int_{a_x}^{r} F_{\pi^*}(u \mid x)^{k-1}\, du = \frac{r - a_x}{\beta_{\mathrm{eff}}(r, x)}, \tag{114}$$

where

$$\beta_{\mathrm{eff}}(r, x) = \frac{\beta(r - a_x)}{k \displaystyle\int_{a_x}^{r} F_{\pi^*}(u \mid x)^{k-1}\, du}. \tag{115}$$

Shifting the lower endpoint to zero gives the simpler expression in Eq. 9. The nonnegative convention therefore provides a natural origin relative to which reward magnitude and effective regularization can be compared. Without such an origin, a definition based directly on r/β_{eff} would depend on an arbitrary reward shift and could become singular or negative.

This convention is not required for the transformed-reward interpretation. More generally, for any anchor a,

$$S_{x,a}(r) = k \int_{a}^{r} F_{\pi^*}(u \mid x)^{k-1}\, du. \tag{116}$$

Changing a adds the same constant to $S_{x,a}(r)$ for every completion of a fixed prompt. This constant is absorbed into the partition function and therefore does not affect the optimal policy.

Common reward range. Our cross-prompt interpretation treats the location of the reference reward distribution — and, informally, its mean

$$\mu_{\mathrm{ref}}(x) = \mathbb{E}_{y\sim\pi_{\mathrm{ref}}(\cdot|x)}[R(y,x)] \tag{117}$$

— as an indicator of task difficulty: prompts receiving lower rewards under the reference policy are regarded as more difficult. This interpretation is meaningful only when rewards can be compared across prompts. In particular, a reward value r_1 must represent a better outcome than $r_2 < r_1$ independently of which prompts produced the two values.

We therefore assume a shared, semantically meaningful fixed range,

$$R(y,x) \in [r_{\min}, r_{\max}] \qquad \text{for every } x. \tag{118}$$

Without common anchors, the average reward cannot define prompt difficulty consistently. Indeed, because the objectives are invariant to prompt-dependent reward shifts, one could change the mean reward of any prompt arbitrarily without changing either its optimization problem or its optimal policy. A difference between two means would then reflect only a difference in reward coordinates rather than a difference in task difficulty. Prompt-dependent rescalings create the same comparability problem and additionally alter the balance between reward and KL regularization unless β is rescaled accordingly. A common fixed range rules out these ambiguities, as in normalized verifier scores and test-case pass rates.

The mean provides an intuitive summary of difficulty but does not, by itself, guarantee a pointwise ordering of the adaptive transformations. A sufficient condition for the ordering illustrated in Figures 1b and 1c can be stated in terms of the optimal reward CDFs. Consider an easy prompt x_{e} and a difficult prompt x_{h} on a common range $[0, r_{\max}]$ such that

$$F_{\pi^*}(r \mid x_{\mathrm{e}}) \le F_{\pi^*}(r \mid x_{\mathrm{h}}) \qquad \text{for all } r \in [0, r_{\max}]. \tag{119}$$

Then, for every common raw reward value,

$$S_{x_{\mathrm{e}}}(r) \le S_{x_{\mathrm{h}}}(r), \qquad \beta_{\mathrm{eff}}(r, x_{\mathrm{e}}) \ge \beta_{\mathrm{eff}}(r, x_{\mathrm{h}}). \tag{120}$$

Thus, the easy prompt receives a smaller transformed reward and stronger effective regularization. This condition formalizes the representative easy–difficult comparison used in the main text; it is not a universal definition of task difficulty. If the reward CDFs cross, the corresponding transformation and effective-regularization curves may also cross, and no global ordering is implied.

F Explicit max@k Solution for Win-Rate Rewards

We now derive the explicit solution in Eqs. 11–12. Fix a prompt x and suppress its dependence where unambiguous. Let

$$F_{\mathrm{ref}}(r \mid x) = \mathbb{P}_{y'\sim\pi_{\mathrm{ref}}(\cdot|x)}[R(y',x) \le r] \tag{121}$$

denote the reference reward CDF, and define the reference quantile, or win rate, of a completion y as

$$Q(y) = F_{\mathrm{ref}}(R(y,x) \mid x). \tag{122}$$

We initially assume that the reference reward distribution is atomless, so the probability integral transform gives

$$Q(y) \sim \mathrm{Unif}[0,1], \qquad y \sim \pi_{\mathrm{ref}}(\cdot \mid x). \tag{123}$$

Let

$$G(q) = \mathbb{P}_{y\sim\pi^*_q(\cdot|x)}[Q(y) \le q] \tag{124}$$

be the quantile CDF under the optimal policy, and let $g(q) = G'(q)$ denote its density. Applying the continuous-reward optimality condition from Appendix C with $Q(y)$ as the reward gives

$$g(q) = \frac{1}{Z(x)} \exp\left(\frac{k}{\beta}\int_0^q G(u)^{k-1}\,du\right). \tag{125}$$

Here the policy ratio depends only on $Q(y)$. Since the reference quantile density is uniform, this ratio is precisely the optimal quantile density g.

Differentiating Eq. 125 yields

$$\frac{g'(q)}{g(q)} = \frac{k}{\beta} G(q)^{k-1}. \tag{126}$$

Using $G'(q) = g(q)$ and treating g as a function of G, we obtain

$$\frac{dg}{dG} = \frac{g'(q)}{G'(q)} = \frac{k}{\beta} G^{k-1}. \tag{127}$$

Integration therefore gives

$$g(q) = C + \frac{1}{\beta} G(q)^k, \tag{128}$$

where $C = g(0) > 0$. Since $G' = g$, the optimal quantile CDF satisfies

$$G'(q) = C + \frac{1}{\beta} G(q)^k, \qquad G(0) = 0, \qquad G(1) = 1. \tag{129}$$

Separating variables gives

$$q = \int_0^{G(q)} \frac{du}{C + \frac{1}{\beta} u^k} \equiv \Psi(G(q)). \tag{130}$$

The terminal condition $G(1) = 1$ requires

$$1 = \int_0^1 \frac{du}{C + \frac{1}{\beta} u^k}. \tag{131}$$

The right-hand side is continuous and strictly decreasing in C, diverges as $C \downarrow 0$, and converges to zero as $C \to \infty$. Hence Eq. 131 has a unique solution $C > 0$. For this value of C, Ψ is a strictly increasing bijection from $[0, 1]$ to $[0, 1]$, and therefore

$$G = \Psi^{-1}. \tag{132}$$

Substituting the resulting density Eq. 128 back into the policy ratio gives

$$\pi_q^*(y \mid x) = \pi_{\mathrm{ref}}(y \mid x) \left(C + \frac{1}{\beta} G(F_{\mathrm{ref}}(R(y, x) \mid x))^k \right). \tag{133}$$

Equivalently, defining

$$S_x^{\mathrm{q}}(r) = \beta \log \left(C + \frac{1}{\beta} G(F_{\mathrm{ref}}(r \mid x))^k \right) \tag{134}$$

yields the transformed-reward representation

$$\pi_q^*(y \mid x) = \pi_{\mathrm{ref}}(y \mid x) \exp \left(\frac{S_x^{\mathrm{q}}(R(y, x))}{\beta} \right). \tag{135}$$

Normalization is already incorporated because

$$\mathbb{E}_{y \sim \pi_{\mathrm{ref}}(\cdot|x)} \left[\exp \left(\frac{S_x^{\mathrm{q}}(R(y, x))}{\beta} \right) \right] = \int_0^1 g(q)\, dq = G(1) - G(0) = 1. \tag{136}$$

Thus, the partition function in this representation is exactly one. Moreover, C, Ψ, and G depend only on k and β, while all prompt dependence enters through the reference CDF F_{ref}.

For $k = 1$, the construction reduces to the standard exponential tilt:

$$g(q) = \frac{\exp(q/\beta)}{\displaystyle\int_0^1 \exp(u/\beta)\, du}. \tag{137}$$

Ties and discrete rewards. In practical settings such as test-case pass-rate rewards, the reference reward distribution is discrete and contains ties. The deterministic win rate $Q(y) = F_{\mathrm{ref}}(R(y, x) \mid x)$ is then not exactly uniform under the reference policy, so the continuous solution provides an approximation to the optimal policy for this win-rate objective. We retain this deterministic convention in our experiments and evaluate the resulting adaptation empirically. The identity $Z(x) = 1$ likewise holds for the true CDF of a continuous reference reward distribution, but need not hold for discrete rewards or finite-bank CDF estimates. Our implementation retains $Z(x) = 1$ as part of the same continuous-reference approximation.

G KAPPO ADAPTATION OF GRPO

GRPO constructs its learning signal by centering and standardizing rewards within each prompt's completion group. This section explains how that normalization interacts with the adaptive reward transformations derived in Section 4. For a group of rewards $R_1, \dots, R_n$, write $\bar{R} = \frac{1}{n}\sum_i R_i$ and $s_R = \mathrm{std}_i(R_i)$. For clarity, we omit the numerical denominator stabilizer and write the expressions below for nonzero-variance groups; constant-reward groups receive zero advantages.

Cancellation of binary reward scaling. For binary rewards, Corollary 4.4 gives the transformed reward $\widetilde{R}_i = a_k(x)R_i$, where

$$a_k(x) = \frac{\beta}{\beta_{\text{eff}}(x)} = k(1 - p^*(x))^{k-1}. \tag{138}$$

Since this multiplier is shared by all completions of the same prompt, it scales both the centered rewards and their standard deviation. For $a_k(x) > 0$, the resulting advantages satisfy

$$\frac{\widetilde{R}_i - \overline{\widetilde{R}}}{\mathrm{std}_j(\widetilde{R}_j)} = \frac{a_k(x)(R_i - \bar{R})}{a_k(x)s_R} = \frac{R_i - \bar{R}}{s_R}. \tag{139}$$

Thus, substituting the transformed binary rewards before group standardization cancels the intended prompt-dependent scaling.

Preserving the binary adaptation. The core issue here is that standardized GRPO does not generally estimate the policy gradient of the underlying expected-reward objective. Dividing each group by its empirical reward standard deviation reweights each prompt's update by a factor that depends on the sampled completions and the current policy; consequently, the expected GRPO update generally differs from the policy gradient of the original objective. For the same reason, KAPPO's adaptive reward scaling is discarded, as shown in Eq. 139. Therefore, the failure of ordinary reward substitution in this case arises precisely because standardized GRPO does not follow the objective to which that connection applies.

Accordingly, we treat the KAPPO adaptation of standardized GRPO as an empirical extension of our theoretical result. To retain the derived redistribution of pressure while preserving GRPO's normalization, we apply the adaptive multiplier after standardization:

$$\widehat{A}_i^k = a_k(x)\frac{R_i - \bar{R}}{s_R}. \tag{140}$$

Since the clipped GRPO surrogate is positively homogeneous in the advantage, this construction multiplies each prompt's contribution to the GRPO loss and gradient by exactly $a_k(x)$. It therefore preserves the theoretically derived cross-prompt pressure redistribution within GRPO's existing learning signal, without claiming an exact objective-level equivalence. This provides a natural reward-side adaptation for GRPO without KL regularization.

Two alternative implementations recover the theoretically grounded interfaces more directly. Removing standard-deviation normalization while retaining group centering allows the ordinary KAPPO reward scaling to survive. Alternatively, for KL-regularized GRPO, the regularization-side interface can be applied through the adaptive coefficient $\beta_{\text{eff}}(x)$, as established in Corollary 4.3.

Continuous reward transformations. For continuous rewards, group standardization does not cause the same cancellation. The transformation S_x^{q} is nonlinear for $k > 1$, whereas centering and standardization eliminate only affine changes in reward. The transformed rewards are therefore affected by GRPO's normalization but are not generally discarded by it. This is consistent with the purpose of the adaptation: KAPPO supplies the @k reward transformation, while GRPO retains its defining group-relative reweighting of the resulting learning signal.

For the win-rate-based transformation used in our experiments, the interaction is particularly clear. At the beginning of training, when the current policy is close to the reference policy, the population mean and standard deviation of the transformed rewards are approximately the same for every prompt. GRPO normalization is therefore trivial with respect to prompt adaptivity: it applies the

same affine transformation across prompts and does not alter the redistribution of optimization pressure induced by KAPPO.

Indeed, write

$$Q_x(Y) = F_{\mathrm{ref}}(R(Y,x) \mid x), \qquad S_x^{\mathrm{q}}(R(Y,x)) = T_{k,\beta}(Q_x(Y)), \tag{141}$$

where $T_{k,\beta}$ is shared across prompts. Under the continuous-reference assumption from Appendix F, at initialization

$$Q_x(Y) \sim \mathrm{Unif}[0,1], \qquad Y \sim \pi_{\mathrm{ref}}(\cdot \mid x). \tag{142}$$

Hence,

$$\mu_{k,\beta} = \mathbb{E}_{U\sim\mathrm{Unif}[0,1]}[T_{k,\beta}(U)], \qquad \sigma^2_{k,\beta} = \mathrm{Var}_{U\sim\mathrm{Unif}[0,1]}[T_{k,\beta}(U)] \tag{143}$$

do not depend on the prompt, and GRPO applies the common normalization

$$S_x^{\mathrm{q}}(r) \longmapsto \frac{S_x^{\mathrm{q}}(r) - \mu_{k,\beta}}{\sigma_{k,\beta}}. \tag{144}$$

Thus, the prompt-adaptive structure of S_x^{q} is preserved up to a common shift and scale.

As training proceeds, the current policy's win-rate distribution can evolve differently across prompts, so the corresponding means and standard deviations become prompt dependent. GRPO normalization then becomes nontrivial and interacts with the KAPPO transformation, introducing GRPO's usual prompt-dependent reweighting on top of the adaptive pressure supplied by KAPPO. Since S_x^{q} is nonlinear, this normalization affects but does not cancel the @k transformation.

H Experimental Details

General setup. We perform full-parameter training in bfloat16 using AdamW with zero weight decay. We use a cosine learning-rate schedule with a 3% warmup, a maximum prompt and response length of 4096 tokens, and sample training completions with temperature 1 and top-$p = 1$. Hyperparameters and intermediate checkpoints are selected using the corresponding validation sets. All KAPPO adaptations use $M = 30$ reference completions per prompt, generated once before training.

Mathematical reasoning. We train for at most 1000 steps. For @k-$A^\star$PO, we search learning rates in $\{5\times10^{-6}, 10^{-5}, 2\times10^{-5}\}$ and β in $\{2.5\times10^{-4}, 5\times10^{-4}, 10^{-3}\}$. For PKPO, we search learning rates in $\{10^{-6}, 5\times10^{-6}, 10^{-5}\}$ and use the minimum rollout multiplicity $n = k$. GRPO uses $n = 8$ completions per prompt and learning rates in $\{5\times10^{-6}, 10^{-5}, 2\times10^{-5}\}$. It uses standard group-relative advantage normalization without an additional KL penalty.

For standard DPO, we search learning rates in $\{5\times10^{-6}, 10^{-5}, 2\times10^{-5}\}$ and β in $\{0.01, 0.02, 0.05\}$. For @k-DPO, we use the same learning-rate grid and search the base coefficient used to construct $\beta_{\mathrm{eff}}(x)$ in $\{10^{-4}, 2\times10^{-4}, 5\times10^{-4}\}$. Both variants construct a preference pair by sampling eight completions and selecting the highest- and lowest-reward candidates; ties receive zero loss weight. We use 64 prompts per DPO step, yielding 128 selected completions after pair construction, and disable log-probability length normalization. For @k-DPO, we add a failure pseudo-count of $\alpha = 0.5$ and cap $\beta_{\mathrm{eff}}(x)$ at 2; the precise boundary handling is described in Appendix J.2. The other mathematical-reasoning experiments use 128 prompts per training step.

Code generation. We train Qwen3-4B for at most 500 steps using 128 prompts per step. For win-rate @k-$A^\star$PO, we search learning rates in $\{10^{-5}, 2\times10^{-5}\}$ and β in $\{5\times10^{-4}, 10^{-3}, 5\times10^{-3}\}$. For PKPO and GRPO, we search learning rates in $\{10^{-6}, 5\times10^{-6}, 10^{-5}\}$. PKPO uses $n = k$, whereas GRPO uses $n = 8$. During training, rewards are computed using up to 64 held-out test cases per problem; validation and final evaluation use the complete available test suite.

For win-rate @k-GRPO, we search learning rates in $\{5\times10^{-6}, 10^{-5}\}$ and the reward-transformation parameter β in $\{5\times10^{-4}, 10^{-3}, 5\times10^{-3}\}$. As in standard GRPO, we use $n = 8$ completions per prompt and no KL penalty.

I COMPUTE RESOURCES

All training runs used 32 NVIDIA GH200 GPUs. Both mathematical reasoning and code-generation runs took approximately 10–12 hours, corresponding to 320–384 GPU-hours per run. The overall computational cost of the study, including hyperparameter sweeps, exploratory runs, and evaluation exceeded 100,000 GPU-hours.

J REFERENCE CALIBRATION

KAPPO requires prompt-level statistics of the reference policy. We estimate them once before training from a fixed bank of M reference completions per prompt. Since the reference policy is fixed, the entire bank can be generated as an inference-only job using high-throughput engines such as vLLM or SGLang. Generation can be continuously batched and performed asynchronously, without backward passes, optimizer updates, parameter offloading, or repeated weight synchronization between the actor and generation workers. The resulting rewards are stored and reused across all training runs that share the same starting model and dataset.

For binary rewards, calibration estimates

$$\widehat{p}_{\mathrm{ref}}(x) = \frac{1}{M}\sum_{j=1}^{M} \mathbb{I}[R(y_j^{\mathrm{ref}}, x) = 1]. \tag{145}$$

For each choice of k and β, we then solve Eq. 6 for $p^*(x)$ and compute $\beta_{\mathrm{eff}}(x)$ and $Z(x)$ (defined in Eq. 5). The @k-DPO instantiation uses $\beta_{\mathrm{eff}}(x)$, whereas binary @k-$A^\star$PO additionally uses $Z(x)$ and the reward multiplier $\beta/\beta_{\mathrm{eff}}(x)$.

For continuous rewards, the same bank defines the empirical reference CDF

$$\widehat{F}_{\mathrm{ref}}(r \mid x) = \frac{1}{M}\sum_{j=1}^{M} \mathbb{I}[R(y_j^{\mathrm{ref}}, x) \le r]. \tag{146}$$

It is substituted into the explicit transformation S_x^{q} in Eq. 12. The remaining quantities C and Ψ depend only on k and β and are precomputed once. Both continuous @k-$A^\star$PO and @k-GRPO therefore reuse exactly the same calibrated reference rewards; changing k, β, the learning rate, or the random seed requires no additional reference generation.

We use $M = 30$ throughout our experiments. For MATH, this corresponds to $30 \times 7{,}500 = 225{,}000$ reference completions per model. Generating the complete bank took approximately 0.5 hours in our setup, whereas a single 1,000-step training run took approximately 10–12 hours. Thus, even if charged entirely to one run, calibration contributes only

$$\frac{0.5}{\text{10–12}} \approx 4.2\%\text{–}5.0\% \tag{147}$$

of its wall-clock time. More importantly, this is a one-time cost shared by every run using the same model and dataset. Amortized over N runs, its relative contribution is only $(4.2\%\text{–}5.0\%)/N$; over six runs, for example, it adds about five minutes per run, or less than 1%. The bank is typically reused across substantially more configurations, making the practical cost of reference calibration negligible relative to training.

J.1 CALIBRATION ALGORITHMS

Algorithms 1 and 2 summarize reference calibration for binary and continuous win-rate rewards, respectively. Both start from the same reusable reference bank: for each training prompt $x \in \mathcal{X}$, draw $y_{x,1}^{\mathrm{ref}}, \ldots, y_{x,M}^{\mathrm{ref}} \overset{\text{i.i.d.}}{\sim} \pi_{\mathrm{ref}}(\cdot \mid x)$ and store their rewards $r_{x,j}^{\mathrm{ref}} = R(y_{x,j}^{\mathrm{ref}}, x)$. This sampling is performed only once per reference model and dataset. Changing k or β requires only recomputing the numerical tables below, not generating additional completions.

Algorithm 1 Reference calibration for binary rewards

Require: Reference rewards $\{r^{\mathrm{ref}}_{x,j}\}$, bank size M, training parameter $k \geq 1$, $\beta > 0$, method $m \in \{A^\star\mathrm{PO}, \mathrm{DPO}\}$

$\alpha \leftarrow 0$ for $A^\star$PO, or $\alpha \leftarrow 0.5$ for DPO
for $j = 0, \ldots, M$ **do**
  $q_j \leftarrow j/(M+\alpha)$
  if $q_j = 0$ **then**
    $p_j \leftarrow 0$; $a_j \leftarrow k$; $\ell_j \leftarrow 0$
  else if $q_j = 1$ **then**
    $p_j \leftarrow 1$
    $a_j \leftarrow 1$ if $k = 1$, otherwise $a_j \leftarrow 0$
    $\ell_j \leftarrow a_j$
  else
    Solve Eq. 6 with $p_{\mathrm{ref}} = q_j$ for p_j
    $a_j \leftarrow k(1-p_j)^{k-1}$
    $\ell_j \leftarrow \beta \log(1 - q_j + q_j \exp(a_j/\beta))$
  end if
  if $m = \mathrm{DPO}$ **then**
    $b_j \leftarrow \min\{\beta/a_j,\, 2\}$
  end if
end for
for each prompt $x \in \mathcal{X}$ **do**
  $N(x) \leftarrow \sum_{j=1}^{M} \mathbb{I}[r^{\mathrm{ref}}_{x,j} = 1]$
  $\widehat{p}^*(x) \leftarrow p_{N(x)}$
  if $m = A^\star\mathrm{PO}$ **then**
    $\widehat{a}_k(x) \leftarrow a_{N(x)}$; $\ell(x) \leftarrow \ell_{N(x)}$
  else
    $\widehat{\beta}_{\mathrm{eff}}(x) \leftarrow b_{N(x)}$
  end if
end for
return $\{\widehat{a}_k(x), \ell(x)\}_{x\in\mathcal{X}}$ for $A^\star$PO, or $\{\widehat{\beta}_{\mathrm{eff}}(x)\}_{x\in\mathcal{X}}$ for DPO

Binary rewards. The calibration depends on each prompt only through its reference success count $N(x)$. We therefore precompute the required quantities for all $M+1$ possible counts. Algorithm 1 includes the method-specific boundary conventions used in our experiments: no smoothing for $A^\star$PO, and a failure pseudo-count with a coefficient cap for DPO. We store $\ell(x) = \beta \log \widehat{Z}(x)$ rather than $\widehat{Z}(x)$.

During training, binary @k-$A^\star$PO uses the regression target $\widehat{a}_k(x)R(y,x) - \ell(x)$, while @k-DPO uses $\widehat{\beta}_{\mathrm{eff}}(x)$ in its preference loss. Neither requires further calibration during training. The fixed-point solve and partition computation are implemented in numerically stable coordinates, as described in Appendix J.2.

Continuous win-rate rewards. An empirical CDF constructed from M reference rewards takes values only in $\{0, 1/M, \ldots, 1\}$. Consequently, the entire win-rate-based transformation can be precomputed on this grid. The resulting table is shared by all prompts and by both continuous @k-$A^\star$PO and @k-GRPO.

For an online completion with reward r, TRANSFORMREWARD returns $\widehat{S}^{\mathrm{q}}_x(r) = T_{M\widehat{F}_{\mathrm{ref}}(r|x)}$ using only comparisons with stored rewards and a table lookup. No numerical integration or root finding is performed during training. The inclusive comparison preserves our convention for ties and includes both empirical-CDF endpoints without smoothing.

Continuous @k-$A^\star$PO uses this value as its regression target, retaining $Z(x) = 1$ under the continuous-reference approximation discussed in Appendix F. @k-GRPO uses the same transformed rewards before its usual group normalization. The parameter β determines the transformation in both cases; using it for calibration does not require adding a KL penalty to GRPO.

Algorithm 2 Reference calibration for continuous win-rate rewards

Require: Reference rewards $\{r^{\mathrm{ref}}_{x,j}\}$, bank size M, training parameter $k \geq 1$, $\beta > 0$
if $k = 1$ **then**
  $\log C \leftarrow -\log \beta - \beta^{-1} - \log(1 - e^{-1/\beta})$
  for $j = 0, \ldots, M$ **do**
    $T_j \leftarrow j/M + \beta \log C$
  end for
else
  Solve $\displaystyle\int_0^1 \frac{du}{C + u^k/\beta} = 1$ for $C > 0$
  Define $\displaystyle\Psi(z) = \int_0^z \frac{du}{C + u^k/\beta}$
  $T_0 \leftarrow \beta \log C$; $T_M \leftarrow \beta \log(C + 1/\beta)$
  for $j = 1, \ldots, M - 1$ **do**
    Solve $\Psi(z_j) = j/M$ for $z_j \in (0, 1)$
    $T_j \leftarrow \beta \log(C + z_j^k/\beta)$
  end for
end if
return Stored reference rewards and the shared table $\{T_j\}_{j=0}^{M}$

function TRANSFORMREWARD(x, r)
  $j \leftarrow \sum_{i=1}^{M} \mathbb{I}[r^{\mathrm{ref}}_{x,i} \leq r]$
  return T_j
end function

J.2 BOUNDARY HANDLING

A finite reference bank can contain no successes or only successes even when the true success probability lies strictly between zero and one. The implementation therefore needs explicit conventions for these empirical endpoints, distinct from the population boundary cases in Appendix B. Let $N(x)$ denote the number of successful completions among the M reference samples.

Binary @k-$A^\star$PO. We use the unsmoothed estimate $\widehat{p}_{\mathrm{ref}}(x) = N(x)/M$. Writing $a_k(x) = \beta/\beta_{\mathrm{eff}}(x)$, the endpoint values for $k > 1$ are

$$\big(\widehat{p}^*(x), \widehat{a}_k(x), \beta \log \widehat{Z}(x)\big) = \begin{cases} (0, k, 0), & N(x) = 0, \\ (1, 0, 0), & N(x) = M. \end{cases} \tag{148}$$

We compute the regression target $\widehat{a}_k(x)R(y, x) - \beta \log \widehat{Z}(x)$ directly, avoiding an explicit infinite $\beta_{\mathrm{eff}}(x)$ at $N(x) = M$. Thus, a zero-success bank gives target $kR(y, x)$, whereas an all-success bank gives target zero for $k > 1$. For $k = 1$, the multiplier remains one; the targets at $N(x) = 0$ and $N(x) = M$ are $R(y, x)$ and $R(y, x) - 1$, respectively. No probability smoothing is used for this method.

Binary @k-DPO. Unlike the regression target, the DPO loss requires a finite coefficient. We therefore add a failure pseudo-count and solve the fixed-point equation using $\widehat{p}_{\mathrm{ref}}(x) = N(x)/(M + \alpha)$ with $\alpha = 0.5$. This leaves the zero-success endpoint unchanged but moves the all-success estimate below one. The formula applies to every count, not only to all-success banks. We then use

$$\widehat{\beta}_{\mathrm{eff}}(x) = \min\left\{ \frac{\beta}{k(1 - \widehat{p}^*(x))^{k-1}},\ 2 \right\}. \tag{149}$$

Smoothing prevents the infinite coefficient at an empirical success probability of one, while the cap limits very large coefficients near this boundary.

Win-rate endpoints and ties. For continuous @k-$A^\star$PO and @k-GRPO, we use the inclusive empirical CDF without smoothing or clipping its values away from zero and one. The transformation

remains finite at both endpoints:

$$S_x^{\mathrm{q}}(r) = \begin{cases} \beta \log C, & \widehat{F}_{\mathrm{ref}}(r \mid x) = 0, \\ \beta \log(C + 1/\beta), & \widehat{F}_{\mathrm{ref}}(r \mid x) = 1. \end{cases} \tag{150}$$

Rewards below the smallest reference reward receive the first value; rewards at or above the largest receive the second. Ties retain the inclusive-CDF convention, including for a constant-reward reference bank. For $A^\star$PO, we retain $Z(x) = 1$ under the continuous-reference approximation discussed in Appendix F.

Numerical stability. For interior binary estimates, we solve the scalar equation in log-odds coordinates and compute a_k directly from that solution, avoiding loss of precision in $1 - p^*$ near one. Partition terms and win-rate transformations are evaluated in the log domain, with a separate closed-form computation for $k = 1$, to avoid overflow at small β. These numerical evaluations do not introduce additional probability smoothing.

Degenerate online groups. In @k-GRPO, we divide centered transformed rewards by their group standard deviation plus 10^{-6}; constant-reward groups therefore have zero advantages. For both DPO variants, selected pairs with equal rewards receive zero loss weight.

K Sensitivity to Finite Reference Calibration

Reference calibration computes the quantities and mappings required by KAPPO before training. For binary rewards, we first estimate $p_{\mathrm{ref}}(x)$ from a fixed bank of reference completions, then solve Eq. 6 for $p^*(x)$ and compute the adaptive reward multiplier $a_k(x)$ or KL coefficient $\beta_{\mathrm{eff}}(x)$. For continuous rewards, we estimate the reference CDF $F_{\mathrm{ref}}(r \mid x)$ and substitute it into the win-rate-based transformation $S_x^{\mathrm{q}}(r)$. This section analyzes how finite-sample noise in the estimates of p_{ref} and F_{ref} propagates to these final quantities. We distinguish *noise*, measured by the standard deviation across independently drawn reference banks, from *bias*, measured by the difference between the mean estimate and the exact value obtained from the true reference distribution. The figures use our experimental bank size $M = 30$ and an illustrative coefficient $\beta = 0.1$.

Exact finite-sample calculation. Fix a prompt and let q denote the true reference quantity being estimated: $q = p_{\mathrm{ref}}(x)$ for binary rewards, or $q = F_{\mathrm{ref}}(r \mid x)$ at a fixed reward threshold r for continuous rewards. Each reference completion then produces a binary observation—success in the first case and the event $R(y, x) \le r$ in the second. Hence, the number N of positive observations in a bank of M completions satisfies $N \sim \mathrm{Binomial}(M, q)$, and the empirical estimate is $\widehat{q} = N/M$.

Let $\theta(q)$ denote any exact downstream quantity computed from q, such as p^*, a_k, β_{eff}, or $S_x^{\mathrm{q}}(r)$. Let $\widehat{\theta}_j$ be the value produced by the implemented calibration rule when $N = j$, including any smoothing or clipping. Since $\Pr(N = j) = w_j(q) := \binom{M}{j} q^j (1-q)^{M-j}$, its exact finite-bank mean, noise, and bias are

$$\mu_\theta(q) = \sum_{j=0}^{M} w_j(q) \widehat{\theta}_j, \qquad \sigma_\theta(q) = \left(\sum_{j=0}^{M} w_j(q) [\widehat{\theta}_j - \mu_\theta(q)]^2 \right)^{1/2}, \qquad b_\theta(q) = \mu_\theta(q) - \theta(q). \tag{151}$$

We report absolute noise and bias for p^* and a_k. Because the scale of β_{eff} varies substantially, for this quantity we define relative noise and relative bias as $\sigma_\theta(q)/\theta(q)$ and $b_\theta(q)/\theta(q)$, respectively, using the exact coefficient in the denominator. Equation 151 sums over all $M + 1$ possible reference-bank outcomes, without Monte Carlo simulation or a local approximation. The reported errors therefore arise solely from estimating the reference distribution with a finite bank, rather than from the numerical solvers used after estimation.

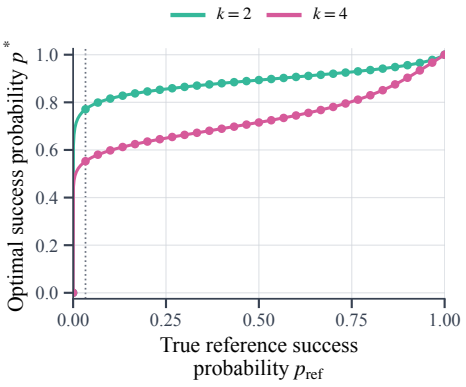


Figure 5: Exact optimal success probability as a function of the true reference success probability, for $\beta = 0.1$. Markers show the attainable empirical estimates j/M for $M = 30$; the dotted line marks the first positive estimate $1/M$. The map is continuous, but its steep rise near zero is compressed on the linear horizontal scale.

K.1 Binary Rewards

Suppressing the prompt argument, set $q = p_{\text{ref}}(x)$ and let $g(q)$ be the optimal success probability determined by Eq. 6. The exact maps applied during reference calibration are

$$p^* = g(q), \qquad a_k(q) = k[1 - g(q)]^{k-1}, \qquad \beta_{\text{eff}}(q) = \frac{\beta}{a_k(q)}, \tag{152}$$

where a_k is the adaptive reward multiplier. For $k > 1$, the boundary values are $g(0) = 0$, $g(1) = 1$, $a_k(0) = k$, and $a_k(1) = 0$.

Sensitive regions. Figure 5 shows that g rises sharply near zero and changes more gradually in the interior. A finite bank may therefore fail to resolve a narrow region over which the inferred p^* changes substantially. For example, at $k = 4$, $g(0) = 0$ whereas $g(1/30) \approx 0.553$, so observing one success instead of none produces a large change in $\widehat{p}^*$. The probability of observing no successes is $(1 - q)^M$, which for $M = 30$ is approximately 0.740 when $q = 0.01$ and 0.042 when $q = 0.1$.

Estimators used in the analysis. For p^* and a_k, we use the unsmoothed estimate $\widehat{q} = N/M$ employed in our binary @k-$A^\star$PO runs, giving $\widehat{p}^* = g(\widehat{q})$ and $\widehat{a}_k = a_k(\widehat{q})$. Thus, each horizontal-axis value in Figure 6 fixes the true p_{ref}, while the plotted noise describes how the inferred quantities vary across independently drawn reference banks. For β_{eff}, we use the one-sided smoothing and cap employed by @k-DPO:

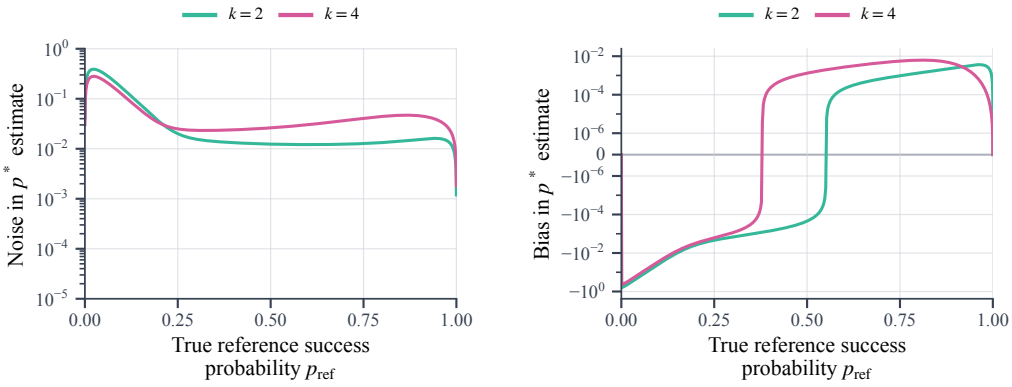
$$\widehat{q}_{\text{D}} = \frac{N}{M + 0.5}, \qquad \widehat{\beta}_{\text{eff}} = \min\left\{\frac{\beta}{a_k(\widehat{q}_{\text{D}})}, 2\right\}. \tag{153}$$

The smoothing keeps $\widehat{q}_{\text{D}} < 1$ while preserving $\widehat{q}_{\text{D}} = 0$ for a zero-success bank. Without smoothing or clipping, $N = M$ would produce an infinite coefficient for $k > 1$, making its finite-bank moments undefined. We measure bias relative to the exact, unclipped $\beta_{\text{eff}}(q)$, so the reported bias includes the effect of these implementation safeguards.

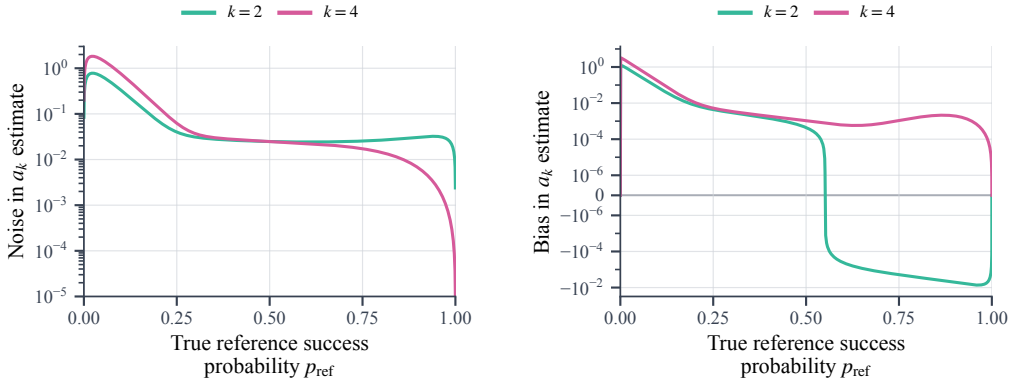
Consistency and convergence rates. Fix k, $\beta > 0$, and the true reference success probability q. With independent reference completions, $\widehat{q} = N/M$ converges almost surely to q. Since g and a_k are continuous, the unsmoothed estimates $\widehat{p}^*$ and $\widehat{a}_k$ are therefore strongly consistent. For fixed $q \in (0, 1)$, these maps are smooth, and a Taylor expansion gives, for $\theta \in \{g, a_k\}$,

$$\sigma_\theta(q) = |\theta'(q)|\sqrt{\frac{q(1-q)}{M}} + o(M^{-1/2}), \qquad b_\theta(q) = \frac{q(1-q)}{2M}\theta''(q) + o(M^{-1}). \tag{154}$$

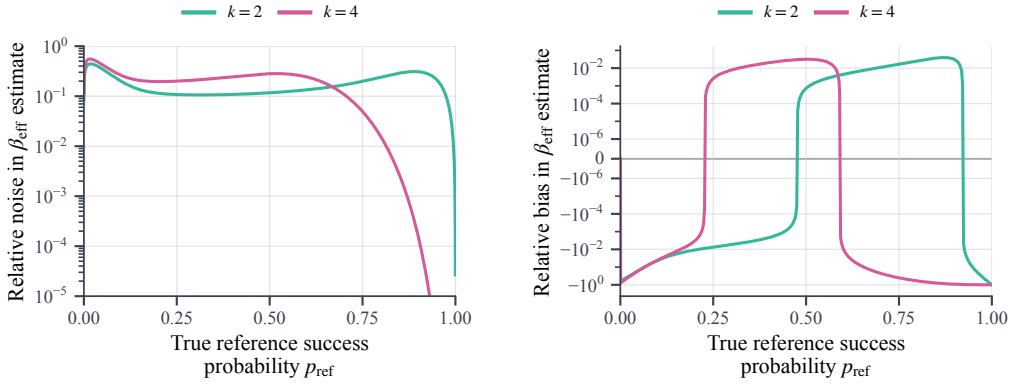
Thus, sampling noise is $O(M^{-1/2})$ and bias is $O(M^{-1})$. At $q = 0$ and $q = 1$, both unsmoothed estimates are exact for every M. The asymptotic constants depend on the local slope and curvature



(a) Optimal success probability p^*.



(b) Adaptive reward multiplier a_k.



(c) Adaptive KL coefficient β_{eff} (relative errors).

Figure 6: Exact finite-bank noise (left) and signed bias (right) for $M = 30$ and $\beta = 0.1$, plotted against the true reference success probability. Panels (a) and (b) use $\widehat{q} = N/M$; panel (c) uses Eq. 153. Coefficient errors are divided by the exact $\beta_{\text{eff}}(q)$; the endpoint $q = 1$, where this coefficient diverges, is omitted. Noise uses logarithmic axes, while signed bias uses symmetric-logarithmic axes that are linear within $[-10^{-6}, 10^{-6}]$. Zero noise at deterministic endpoints is omitted from the logarithmic axes.

of the calibration maps, which can be large near sensitive regions. Consequently, these rates describe the large-bank limit but need not accurately predict the errors at $M = 30$; the figures instead use the exact finite-bank calculation in Eq. 151.

Effect of coefficient stabilization. The failure pseudo-count used by @k-DPO vanishes asymptotically: $\widehat{q}_{\mathrm{D}} = N/(M+0.5)$ converges almost surely to q, and its bias is $-q/(2M+1) = O(M^{-1})$. The fixed coefficient cap, however, does not vanish. For $q \in (0,1)$,

$$\widehat{\beta}_{\mathrm{eff}} \xrightarrow[M\to\infty]{\text{a.s.}} \min\{\beta_{\mathrm{eff}}(q), 2\}. \tag{155}$$

When $\beta_{\mathrm{eff}}(q) < 2$, the coefficient remains consistent for its exact value, with $O(M^{-1/2})$ noise and $O(M^{-1})$ bias; smoothing changes the leading bias coefficient but not its order. When $\beta_{\mathrm{eff}}(q) > 2$, sampling noise vanishes while bias relative to the exact coefficient converges to $2 - \beta_{\mathrm{eff}}(q)$. For a fixed q strictly in this capped region, deviations from the cap become exponentially unlikely as M grows. At the clipping threshold $\beta_{\mathrm{eff}}(q) = 2$ for $k > 1$, the estimator is still consistent, but clipping introduces a kink and its downward bias is generally of order $M^{-1/2}$ rather than M^{-1}. Dividing by the fixed, finite $\beta_{\mathrm{eff}}(q)$ preserves these rates for the relative errors.

Implications. For p^* and a_k, finite-bank error is concentrated near the low-success boundary and becomes much smaller in the interior. At $k = 4$ and $p_{\mathrm{ref}} = 0.1$, $\widehat{p}^*$ has standard deviation 0.123 and bias -0.028; at $p_{\mathrm{ref}} = 0.5$, these decrease to 0.026 and 0.001. The same boundary events affect the adaptive multiplier in the opposite direction: a bank containing no successes yields $\widehat{p}^* = 0$ and hence the maximal multiplier $\widehat{a}_k = k$, producing an upward bias in the pressure assigned to prompts with small but nonzero success probability. Increasing M reduces both noise and bias because the probability $(1 - p_{\mathrm{ref}})^M$ of such unresolved zero-success banks decays exponentially. Additional reference completions are therefore most valuable for hard prompts for which Mp_{ref} is small. At the exact boundary $p_{\mathrm{ref}} = 0$, there is no estimation error: every bank contains zero successes and the maximal multiplier is the correct value.

The effective KL coefficient has a different limitation near $p_{\mathrm{ref}} = 1$. Because the exact β_{eff} diverges while the implemented estimator is capped, its sampling noise can be small simply because nearly all estimates have saturated at the cap, even though its downward bias relative to the exact coefficient is large. Increasing M reduces finite-bank noise but cannot remove this cap-induced bias; doing so requires changing the cap or the stabilization rule. Thus, larger reference banks address the main uncertainty on low-success prompts, whereas accuracy on extremely easy prompts is ultimately limited by the chosen coefficient stabilization.

K.2 Continuous Rewards

For continuous rewards, reference calibration estimates the CDF value at each reward threshold and substitutes it into the win-rate-based transformation. At a fixed prompt and raw reward r, write Eq. 12 as

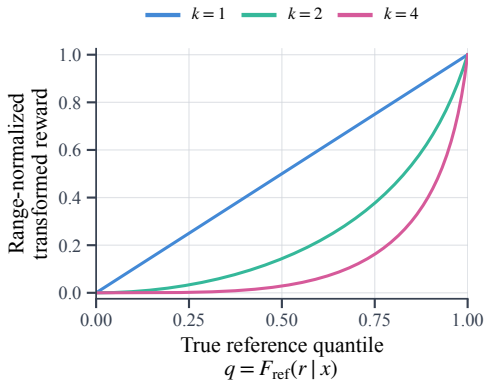
$$S_x^{\mathrm{q}}(r) = T(q), \qquad q = F_{\mathrm{ref}}(r \mid x), \qquad T(q) = \beta \log\left(C + \frac{G(q)^k}{\beta}\right), \quad G = \Psi^{-1}. \tag{156}$$

Here, N counts the reference completions whose rewards do not exceed r, so $N \sim \mathrm{Binomial}(M, q)$ and $\widehat{q} = N/M$. The estimated transformed reward is therefore $T(\widehat{q})$, and its exact noise and bias follow from Eq. 151 by setting $\theta(q) = T(q)$ and $\widehat{\theta}_j = T(j/M)$. This pointwise calculation requires no model of the full raw-reward distribution. To compare different k, we divide the plotted noise and bias by the transformation range $\Delta_T = T(1) - T(0)$.

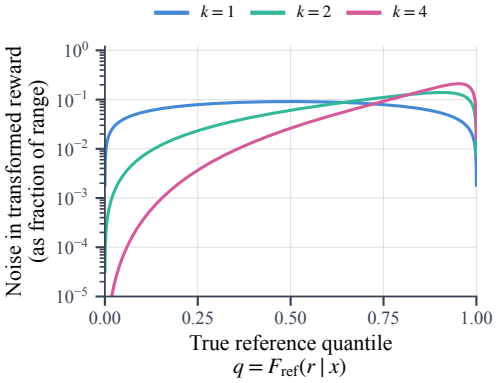
Where quantile error matters. For $k > 1$, the sensitivity of the transformation follows from $G'(q) = C + G(q)^k/\beta$:

$$T'(q) = kG(q)^{k-1}, \qquad T''(q) = k(k-1)G(q)^{k-2}\left(C + \frac{G(q)^k}{\beta}\right) \geq 0. \tag{157}$$

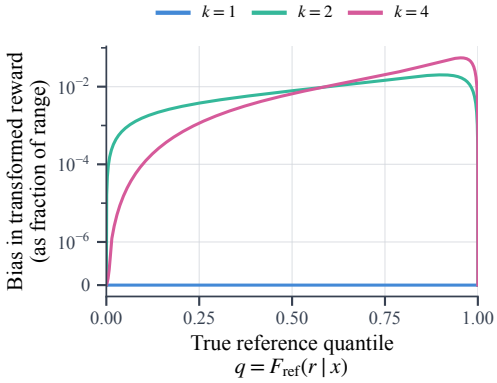
In particular, $0 \leq T'(q) \leq k$, so $|T(\widehat{q}) - T(q)| \leq k|\widehat{q} - q|$. Sensitivity increases toward the upper reference quantiles, while convexity implies $\mathbb{E}[T(\widehat{q})] \geq T(q)$: although $\widehat{q}$ is unbiased, the resulting transformed reward is generally upward-biased. For $k = 1$, T is affine, so the bias is zero and the standard deviation is exactly $\sqrt{q(1-q)/M}$.



(a) Win-rate-based reward transformation.



(b) Noise in the transformed reward.



(c) Bias in the transformed reward.

Figure 7: Win-rate transformation and exact finite-bank noise and bias for $M = 30$ and $\beta = 0.1$. To compare different k, values are normalized by $\Delta_T = T(1) - T(0)$; this normalization is only for display, and the transformation panel additionally subtracts $T(0)$. The horizontal axis is the true reference quantile of a fixed raw reward. Noise and bias use the same axis conventions as Figure 6.

Consistency and convergence rates. For fixed k and β, almost-sure convergence of the empirical CDF and continuity of T imply that the estimated transformation $\widehat{S}_x^{\mathrm{q}}(r) := T(\widehat{F}_{\mathrm{ref}}(r \mid x))$ is strongly consistent for $S_x^{\mathrm{q}}(r)$. Moreover, using the derivatives in Eq. 157, we obtain finite-sample bounds on the pointwise noise and bias:

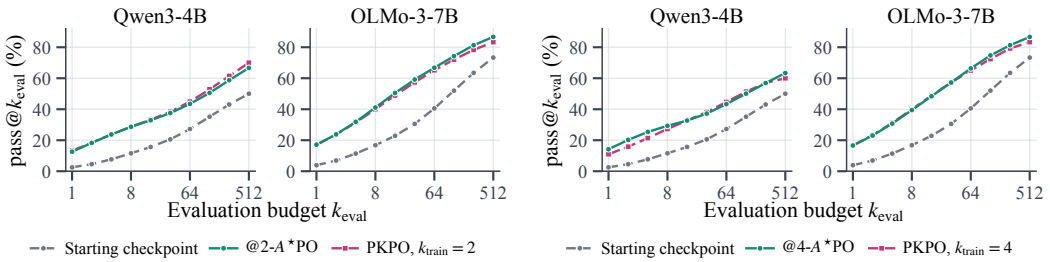
$$\sigma_T(q) \le k\sqrt{\frac{q(1-q)}{M}}, \qquad 0 \le b_T(q) \le \frac{k(k-1)(C+\beta^{-1})q(1-q)}{2M}. \tag{158}$$

The noise bound follows from the Lipschitz property of T and $\mathrm{Var}(\widehat{q}) = q(1-q)/M$. The bias bound follows from a second-order Taylor expansion, $\mathbb{E}[\widehat{q} - q] = 0$, and $0 \le T''(q) \le k(k-1)(C+\beta^{-1})$. Thus, noise is $O(M^{-1/2})$ and bias is $O(M^{-1})$, with bounds valid for every M. At fixed interior quantiles, the sharper asymptotic expressions in Eq. 154 also apply with $\theta = T$. At $q = 0$ and $q = 1$, both errors are zero.

Consistency also holds for the entire reward transformation, uniformly over reward thresholds. For each fixed prompt and $\delta \in (0, 1)$, the Dvoretzky–Kiefer–Wolfowitz–Massart inequality (Massart, 1990) and the k-Lipschitz property of T imply

$$\Pr\left(\sup_r \left|\widehat{S}_x^{\mathrm{q}}(r) - S_x^{\mathrm{q}}(r)\right| > k\sqrt{\frac{\log(2/\delta)}{2M}}\right) \le \delta. \tag{159}$$

Hence, a growing reference bank consistently estimates the whole mapping. These guarantees concern estimation of $T(F_{\mathrm{ref}}(r \mid x))$ and hold for both continuous and discrete reference reward distributions.



(a) $k_{\text{train}} = 2$. (b) $k_{\text{train}} = 4$.

Figure 8: AIME 2026 test performance of the starting checkpoints and the selected @k-$A^\star$PO and PKPO checkpoints across evaluation sampling budgets. Each subfigure reports Qwen3-4B (left) and OLMo-3-7B (right).

Implications. The win-rate transformation has the opposite sensitive region: both noise and bias are small at low and intermediate quantiles but increase toward the upper tail, where T is steepest. At $k = 4$, the standard deviation and bias of the transformed reward are 2.62% and 0.66% of its range at $q = 0.5$, but increase to 18.27% and 4.43% at $q = 0.9$. The bias is positive because the convex transformation amplifies upward empirical-CDF errors more than downward ones. Near the upper boundary, the main finite-sample event is assigning the maximal empirical quantile to a reward whose true quantile is below one; for example, at $q = 0.99$, this occurs with probability $\Pr(\widehat{q} = 1) = q^{30} \approx 0.740$.

Increasing M reduces both effects by concentrating $\widehat{q}$ around q and decreasing the probability q^M of assigning a premature maximal quantile. Additional reference completions are therefore most useful when accurate upper-tail ranks are important, particularly where $M(1 - q)$ is small. Since the same reference bank is reused throughout training, its error is a fixed distortion of the estimated transformation rather than fresh noise at every update. Overall, $M = 30$ provides substantially better resolution away from the sensitive boundaries, while larger banks can be targeted to hard binary prompts or to applications requiring more accurate upper-tail quantiles. For continuous rewards, this analysis isolates uncertainty from estimating the reference CDF; the separate approximation introduced by tied or discrete rewards is discussed in Appendix F.

L Comparison with Direct Multi-Completion Optimization

Checkpoint selection. For each model, method, and training value $k_{\text{train}} \in \{2, 4\}$, we select the checkpoint that maximizes $\text{pass@k}_{\text{train}}$ on the MATH Level 5 validation problems. Figure 3 reports the complete validation curves of the selected checkpoints, and Table 1 reports the corresponding numerical values.

At the evaluation budget matching the training objective, @k-$A^\star$PO is within 0.5 percentage points of PKPO in every comparison. The two methods also produce similar performance across the remaining evaluation budgets. This supports the central prediction of our analysis: adaptive pressure applied through an ordinary single-completion optimizer can reproduce direct multi-completion pass@k optimization.

Figure 8 extends this comparison to AIME 2026. Across both model families and training values of k, the two methods achieve comparable performance across evaluation budgets, showing that their agreement extends beyond the in-distribution validation set to harder, unseen problems.

Rollout-compute comparison. With a prompt batch of 128 and 1,000 training steps, @k-$A^\star$PO generates $128{,}000$ on-policy completions per run. PKPO generates n completions for every prompt and therefore requires $128{,}000n$ completions. At its minimum rollout multiplicity $n = k$, this gives $256{,}000$ completions for $k = 2$ and $512{,}000$ for $k = 4$. Thus, @k-$A^\star$PO reduces online rollout generation by 50% and 75%, respectively; larger PKPO groups increase this difference proportionally.

Table 1: MATH Level 5 validation performance (in %) of the starting checkpoints and selected @k-$A^\star$PO and PKPO checkpoints, corresponding to Figure 3.

Model	Method	pass@1	pass@2	pass@4	pass@8	pass@16
OLMo-3-7B	Starting checkpoint	37.38	50.94	63.16	73.29	81.80
	@2-$A^\star$PO	65.34	76.98	84.38	89.11	92.22
	PKPO ($k_{\text{train}} = 2$)	66.08	77.38	84.73	89.46	92.52
	@4-$A^\star$PO	64.46	76.51	84.39	89.33	92.45
	PKPO ($k_{\text{train}} = 4$)	64.31	76.18	83.92	89.04	92.45
Qwen3-4B	Starting checkpoint	23.93	38.32	54.32	68.16	77.79
	@2-$A^\star$PO	57.27	70.48	79.39	85.34	89.43
	PKPO ($k_{\text{train}} = 2$)	59.88	70.77	78.77	84.59	88.75
	@4-$A^\star$PO	57.22	70.14	78.63	84.22	88.29
	PKPO ($k_{\text{train}} = 4$)	57.94	69.91	78.65	84.81	89.20

@k-$A^\star$PO additionally requires one-time reference calibration, whose implementation and cost are detailed in Appendix J. In our setup, generating the complete reference bank took approximately half an hour, compared with 10–12 hours for one training run, and the same bank is reused across all configurations of a given model and dataset. Even when amortized over only six runs, calibration therefore adds approximately five minutes per run, or less than 1% of training time. Its contribution is consequently small relative to the savings from removing PKPO's k-dependent online rollout multiplicity.